Ansys 2024 R1 DEVELOPMENT

Take A Leap Of Certainty™

©2024 ANSYS, Inc. or its affiliated companies
Unauthorized use, distribution, or duplication is prohibited.

Speos User's Guide

Ansys, Inc.
Southpointe
2600 ANSYS Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
(T) 724-746-3304
(F) 724-514-9494

Release 2024 R1
January 2024

Ansys, Inc. and ANSYS Europe, Ltd. are UL registered ISO 9001:2015 companies

Contents

1: Welcome!....14
2: Ansys Product Improvement Program....15
3: Speos Software Overview....19
3.1. Graphical User Interface....19
3.2. Launching Speos Using Command Line....21
3.3. Setting Speos Preferences....22
3.4. Geometry Modeler....25
3.5. Lightweight/Heavyweight Import....26
3.5.1. Lightweight/Heavyweight Import Overview....26
3.5.2. Switching a Body from Lightweight to Heavyweight....27
3.5.3. Lightweight/Heavyweight Import Troubleshooting....28
3.6. Using the Feature Contextual Menu....29
3.7. Useful Commands and Design Tools....30
3.8. Extensions and Units....37
3.9. Beta Features....39
3.10. Speos Files Analysis....40
3.10.1. Speos Files Analysis Overview....40
3.10.2. Using the Speos File Analysis....41
3.11. Presets....43
3.11.1. Presets Overview....43
3.11.2. Customizing the Preset Repository....44
3.11.3. Creating a Preset....45
3.11.4. Exporting a Speos Object to a Preset....46
3.11.5. Setting a Preset as Default....46
3.11.6. Creating a Speos Object from a Default Preset....47
3.11.7. Applying a Preset to an Existing Speos Object....47
3.11.8. Accessing the Quick Preset Menu....48
4: Imports....49
4.1. CAD File Import (CATIA, NX, CREO, Solidworks)....49
4.1.1. CAD File Import Overview....49
4.1.2. Configuring the SpaceClaim Reader....49
4.1.3. CATIA File Import Limitations....50
4.2. SCDOC File Import....51
4.3. IGES and STEP Files Import....51
4.4. STL Files Import....52
4.5. CATIA V6 Import....53
4.6. Geometry Update Tool....53
4.6.1. Geometry Update Overview....53
4.6.2. Importing an External CAD Part....54
4.6.3. Updating an External CAD Part....55

DRAFT

4.7. Incorrect Imports - Solutions....56
5: Materials....57
5.1. Optical Properties....57
5.1.1. Optical Properties Overview....57
5.1.2. Non-Homogeneous Material....58
5.1.2.1. Understanding Non-Homogeneous Material....58
5.1.2.2. Graded Material File....61
5.1.2.3. List Of Methods....63
5.1.3. Surface State Plugin....66
5.1.3.1. Surface State Plugin Examples....66
5.1.3.2. Creating and Testing a Surface State Plugin in Windows....66
5.1.3.3. Creating and Testing a Surface State Plugin in Linux....67
5.1.4. Optical Properties Creation....68
5.1.4.1. Creating Optical Properties....68
5.1.4.2. Creating Face Optical Properties....70
5.1.5. VOP on Surface....73
5.1.5.1. VOP on Surface Overview....73
5.1.5.2. Creating a VOP on Surface....73
5.1.6. Optical Properties Management....74
5.1.6.1. Creating a Material Library....74
5.1.6.2. Opening a Material Library....75
5.1.6.3. Applying a Material from a Material Library....76
5.1.6.4. Applying a Material from the Tree....77
5.1.6.5. Replacing a Material on Geometries....78
5.1.6.6. Converting Face Optical Properties....78
5.1.7. Locate Material Tool....78
5.1.7.1. Understanding the Locate Material Tool....78
5.1.7.2. Using the Locate Material Tool....80
5.1.7.3. Applying a Visualization Color to a Material....82
5.1.7.4. Defining the Visualization Options....82
5.2. Texture Mapping....83
5.2.1. Texture Mapping Overview....83
5.2.2. Understanding UV Mapping....84
5.2.3. Understanding Texture Mapping....85
5.2.4. Texture Mapping Process Overview....87
5.2.5. Texture Mapping Preview....89
5.2.6. Creating a Texture Mapping....92
5.2.6.1. Creating the UV Mapping....93
5.2.6.2. Applying Textures....97
5.2.7. Activating the Texture Mapping Preview....100
5.2.8. Texture Normalization....102
5.2.8.1. Understanding Texture Normalization....102
5.2.8.2. Setting the Texture Normalization....103
5.3. Polarization....103

DRAFT

5.3.1. Understanding Polarization....103
5.3.2. Creating a Polarizer....104
6: Local Meshing....106
6.1. Understanding Meshing Properties....106
6.2. Creating a Local Meshing....108
7: Sources....110
7.1. Sources Overview....110
7.2. Introduction to Optics....111
7.3. Sources Creation....114
7.3.1. Creating an Interactive Source....114
7.3.2. Creating a Ray File Source....115
7.3.3. Light Field Source....118
7.3.3.1. Light Field Overview....118
7.3.3.2. Understanding the Parameters of a Light Field Source....119
7.3.3.3. Creating a Light Field Source....119
7.3.4. Surface Source....121
7.3.4.1. Understanding the Parameters of a Surface Source....121
7.3.4.2. Creating a Surface Source....126
7.3.5. Display Source....130
7.3.5.1. Understanding the Parameters of a Display Source....130
7.3.5.2. Creating a Display Source....134
7.3.6. Creating a Luminaire Source....137
7.3.7. Thermic Source....139
7.3.7.1. Creating a Thermic Source....139
7.3.7.2. Creating a Thermic Source using a Temperature Field File....141
7.3.8. Ambient Sources....144
7.3.8.1. Environment Source....144
7.3.8.2. Creating a Uniform Ambient Source....150
7.3.8.3. Creating a CIE Standard General Sky Ambient Source....151
7.3.8.4. Creating a CIE Standard Overcast Sky Ambient Source....153
7.3.8.5. Natural Light Ambient Source....154
7.3.8.6. U.S. Standard Atmosphere 1976 Source....159
8: Sensors....163
8.1. Sensors Overview....163
8.2. Generic Sensor Parameters and Tools....164
8.2.1. Integration Angle....164
8.2.1.1. Integration Angle Overview....164
8.2.1.2. Direct Simulation with Integration Angle VS Inverse Simulation....165
8.2.1.3. Setting an Integration Angle....166
8.2.1.4. Influence Factors....167
8.2.2. Automatic Framing....169
8.3. Sensor Creation....171
8.3.1. Irradiance Sensor....171
8.3.1.1. Creating an Irradiance Sensor....171

DRAFT

8.3.1.2. Understanding Integration Types....177
8.3.1.3. Understanding the Incident Angles Layer Type....183
8.3.2. Creating a Radiance Sensor....186
8.3.3. Intensity Sensor....191
8.3.3.1. Understanding the Parameters of an Intensity Sensor....191
8.3.3.2. Creating an Intensity Sensor....197
8.3.3.3. Creating a Polar Intensity Sensor....202
8.3.4. Human Eye Sensor....205
8.3.4.1. Understanding the Parameters of a Human Eye Sensor....205
8.3.4.2. Creating a Human Eye Sensor....206
8.3.5. Creating a 3D Irradiance Sensor....209
8.3.6. Creating an Immersive Sensor....212
8.3.7. Creating a 3D Energy Density Sensor....214
8.3.8. Creating an Observer Sensor....216
8.3.9. Light Field Sensor....220
8.3.9.1. Light Field Overview....220
8.3.9.2. Understanding the Parameters of a Light Field Sensor....221
8.3.9.3. Creating a Light Field Sensor....221
8.3.10. LiDAR Sensor....223
8.3.10.1. LiDAR Sensor Models....224
8.3.10.2. LiDAR Sensor Parameters....226
8.3.10.3. Generating a Scanning Sequence File....243
8.3.10.4. Creating a LiDAR Sensor....246
8.3.11. Camera Sensor....258
8.3.11.1. Camera Sensor General View....258
8.3.11.2. Camera Sensor Parameters....259
8.3.11.3. Camera Sensor Creation....281
8.3.12. Physical Camera....288
8.3.12.1. Physical Camera Overview....288
8.3.12.2. Understanding the Physical Camera Parameters....291
8.3.12.3. Aiming Area....294
8.3.12.4. Creating a Physical Camera....295
8.3.12.5. Creating an *.OPTSequence File for a Physical Camera Sensor....297
8.3.13. Creating a Geometric Rotating LiDAR Sensor....298
8.3.14. Creating a Light Expert Group....300
8.4. Stray Light Analysis....300
8.4.1. Stray Light Analysis Overview....300
8.4.2. Understanding the Sequence Detection Tool....301
8.4.3. Making Stray Light Analysis....303
8.4.4. Analyzing Stray Light....304
9: Components....306
9.1. Speos Light Box....306
9.1.1. Speos Light Box Overview....306
9.1.2. Understanding Speos Light Box Parameters....307

DRAFT

9.1.2.1. Trajectory File....309
9.1.2.2. Trajectory Script Example....312
9.1.3. Exporting a Speos Light Box....316
9.1.4. Importing a Speos Light Box....318
9.2. Optical Design Exchange....320
9.2.1. Optical Design Exchange Overview....320
9.2.2. Understanding What Can Be Exported from Ansys Zemax OpticStudio....322
9.2.3. Understanding the Optical Design Exchange Parameters....325
9.2.4. Importing an Optical Design System Created in Ansys Zemax OpticStudio....328
9.2.5. Modifying Imported Geometries....329
9.2.6. Specific Simulation Options....332
9.2.6.1. Setting the Specific Simulation Options....332
9.2.6.2. Understanding the Specific Simulation Options for an Optical Design Exchange....333
9.3. 3D Texture....335
9.3.1. 3D Texture Overview....335
9.3.2. Boolean Operation....336
9.3.3. Mapping File....338
9.3.4. Scale Factors....340
9.3.5. Creating a 3D Texture....342
9.3.6. Mapping....344
9.3.6.1. Understanding Mapping....344
9.3.6.2. Creating a Rectangular Mapping....346
9.3.6.3. Creating a Circular Mapping....348
9.3.6.4. Creating a Hexagonal Mapping....350
9.3.6.5. Variable Pitches Mapping....352
9.3.6.6. Using Mapping Files....356
9.4. Creating a Speos Pattern....358
10: Simulations....362
10.1. Simulations Overview....362
10.2. Simulation Management....363
10.2.1. Simulation Compatibility....363
10.2.2. GPU Simulation Limitations....367
10.2.3. GPU/CPU Differences....369
10.2.4. Computing Simulations....371
10.2.5. Speos Live Preview....373
10.2.5.1. Understanding the Live Preview....373
10.2.5.2. Running a Live Preview....374
10.2.5.3. Live Preview Parameters....375
10.2.5.4. Preference Settings....377
10.2.6. Interactive Live Preview....378
10.2.6.1. Interactive Live Preview Overview....379
10.2.6.2. Using the Interactive Live Preview....380
10.2.6.3. Using the Interactive Live Preview with the Timeline Parameter....381

DRAFT

10.2.6.4. Parameters Compatibility with Interactive Live Preview....381
10.2.7. Exporting a Simulation....384
10.2.8. Linked Exporting a Simulation....384
10.2.9. Speos Core....385
10.2.9.1. Speos Core Overview....385
10.2.9.2. Running a Simulation Using a Local Update....386
10.2.9.3. Network Update....387
10.3. Interactive Simulation....391
10.3.1. Creating an Interactive Simulation....391
10.3.2. Adjusting Interactive Simulation Settings....392
10.3.3. Camera Projected Grid Parameters....396
10.4. Direct Simulation....398
10.4.1. Creating a Direct Simulation....398
10.4.2. Adjusting Direct Simulation Settings....401
10.5. Inverse Simulation....405
10.5.1. Creating an Inverse Simulation....405
10.5.2. Adjusting Inverse Simulation Settings....409
10.5.3. Calculation Properties....413
10.5.3.1. Monte Carlo Calculation Properties....413
10.5.3.2. Deterministic Calculation Properties....417
10.6. Understanding Advanced Simulation Settings....429
10.6.1. Meshing Properties....429
10.6.2. Tangent Bodies Management....432
10.6.3. Smart Engine....434
10.6.4. Dispersion....436
10.6.5. Weight....438
10.7. LiDAR....441
10.7.1. Understanding LiDAR Simulation....441
10.7.2. Understanding LiDAR Simulation with Timeline....443
10.7.3. Creating a LiDAR Simulation....446
10.7.4. Understanding LiDAR Simulation Results....449
10.7.4.1. Fields of View....449
10.7.4.2. Map of Depth....452
10.7.4.3. Raw Time of Flight....452
10.7.4.4. LiDAR Projected Grid Parameters....461
10.8. Geometric Rotating LiDAR Simulation....462
10.8.1. Understanding Rotating LiDAR Simulation....462
10.8.2. Creating a Geometric Rotating LiDAR Simulation....464
10.9. Light Expert....465
10.9.1. Understanding the Light Expert....465
10.9.2. Understanding the Light Expert Parameters....467
10.9.3. Performing a Single-Sensor Light Expert Analysis....467
10.9.4. Performing a Multi-Sensors Light Expert Analysis....468
10.10. Propagation Errors....469

DRAFT

10.10.1. Propagation Errors Overview....469
10.10.2. Understanding Propagation Errors....472
10.10.3. Performing a Propagation Errors Analysis....473
11: Results....475
11.1. Reading the HTML Report....475
11.2. Visualizing Results....477
11.3. Light Path Finder Results....481
11.3.1. Understanding the Light Path Finder Parameters....481
11.3.2. Visualizing an Interactive Simulation LPF Result....482
11.3.3. Visualizing an Inverse or Direct Simulation LPF Result....484
11.3.4. Visualizing a LP3 Result....485
11.3.5. Visualizing a LPF Result for Multi-Sensors Analysis....487
11.3.6. Light Path Finder Advanced Analysis....489
11.3.6.1. Understanding the Light Path Finder Advanced Analysis....489
11.3.6.2. List Of Methods....489
11.4. Export as Geometry....492
11.5. Export Projected Grid as Geometry....493
11.6. Isolating a Simulation Result....494
12: Speos Labs Documentation Links....495
13: Optical Part Design....497
13.1. Optical Part Design Overview....497
13.2. Migration Warnings and Differences between Versions....498
13.3. Parabolic Surface....499
13.3.1. Parabolic Surface Overview....499
13.3.2. Understanding Parabolic Surface Parameters....500
13.3.3. Creating a Parabolic Surface....501
13.4. Optical Surface....502
13.4.1. Optical Surface Overview....502
13.4.2. Creating an Optical Surface....503
13.4.3. Managing Groups and Elements....506
13.4.4. Managing Elements from an Excel File....508
13.4.5. Understanding the Excel File....509
13.4.6. Optical Surface Parameters....511
13.4.6.1. Source Types....511
13.4.6.2. Support....512
13.4.6.3. Target....513
13.4.6.4. Style....514
13.4.6.5. Manufacturing....520
13.4.6.6. Beams....521
13.4.6.7. Support....535
13.4.7. Display Properties....537
13.4.7.1. Understanding Display Properties....537
13.4.7.2. Adjusting Display Properties....540
13.4.8. Interactive Preview....542

DRAFT

13.4.8.1. Understanding the Interactive Preview....542
13.4.8.2. Displaying the Interactive Preview Parameters....544
13.5. Optical Lens....545
13.5.1. Optical Lens Overview....545
13.5.2. Creating an Optical Lens....546
13.5.3. Managing Groups and Elements....549
13.5.4. Managing Elements from an Excel File....551
13.5.5. Understanding the Excel File....553
13.5.6. Optical Lens Parameters....555
13.5.6.1. Source Types....555
13.5.6.2. Support....556
13.5.6.3. Target....557
13.5.6.4. Style....558
13.5.6.5. Manufacturing....567
13.5.6.6. Beams....568
13.5.7. Display Properties....575
13.5.7.1. Understanding Display Properties....575
13.5.7.2. Adjusting Display Properties....578
13.5.8. Interactive Preview....580
13.5.8.1. Understanding the Interactive Preview....580
13.5.8.2. Displaying the Interactive Preview Parameters....583
13.6. Light Guide....584
13.6.1. Light Guide Overview....584
13.6.2. Creating a Light Guide....585
13.6.3. Defining the Light Guide Prisms....586
13.6.4. Defining the Manufacturing Parameters....588
13.6.5. Light Guide Parameters....590
13.6.5.1. Light Guide Body Parameters....590
13.6.5.2. Light Guide Prism Parameters....596
13.6.5.3. Manufacturing Parameters....600
13.7. TIR Lens....602
13.7.1. TIR Lens Overview....602
13.7.2. Understanding the Parameters of a TIR Lens....603
13.7.3. Creating a TIR Lens....606
13.8. Projection Lens....607
13.8.1. Projection Lens Overview....608
13.8.2. Understanding Projection Lens Parameters....608
13.8.3. Creating a Projection Lens....613
13.8.4. Defining the Back Face and Front Face....615
13.8.4.1. Defining a Plan Face....615
13.8.4.2. Defining an Aspherical Face....616
13.8.4.3. Defining an Automatic Face....617
13.8.4.4. Defining a Zernike Face....618
13.9. Poly Ellipsoidal Reflector....619

DRAFT

13.9.1. Poly Ellipsoidal Reflector Overview....619
13.9.2. Understanding Parameters of a Poly Ellipsoidal Reflector....620
13.9.3. Creating a Poly Ellipsoidal Reflector....623
13.10. Freeform Lens....625
13.10.1. Freeform Lens Overview....625
13.10.2. Understanding the Freeform Lens Parameters....626
13.10.3. Creating a Freeform Lens....630
13.10.4. Creating a Freeform Lens Based on an Irradiance Target....631
13.10.5. Creating a Freeform Lens Based on an Intensity Target....633
13.11. Micro Optical Stripes....635
13.11.1. Micro Optical Stripes Overview....635
13.11.2. Understanding the Micro Optical Stripes Parameters....636
13.11.3. Creating Micro Optical Stripes....642
13.11.4. Extracting Tooling Path....645
13.11.5. Exporting As CSV File....645
13.12. Post Processing....645
13.12.1. Post Processed Optical Part Design Geometry Modification....645
13.12.1.1. Modifying a Post Processed Optical Part Design Geometry....646
13.12.1.2. Modifying a Post Processed Optical Part Design Geometry with Post Processing....646
14: Head Up Display....648
14.1. Head Up Display Overview....648
14.2. Design....649
14.2.1. HUD System Overview....649
14.2.2. Understanding the HUD Optical Design Parameters....650
14.2.2.1. General....650
14.2.2.2. Eyebox....650
14.2.2.3. Target Image....651
14.2.2.4. Projector....652
14.2.2.5. Manufacturing....653
14.2.2.6. Advanced Parameters....654
14.2.3. Defining a HUD System with HUD Optical Design....657
14.2.4. CNC Export (Surface Export File)....661
14.3. Analysis....661
14.3.1. HUD Optical Analysis....661
14.3.1.1. Setting the HUD Optical Analysis....661
14.3.1.2. Exporting a HUD Optical Analysis Simulation....675
14.3.1.3. Speos Plugin....675
14.3.1.4. Speos Plugin Examples....712
14.3.2. Results....718
14.3.2.1. Eyebox....718
14.3.2.2. Target Image....719
14.3.2.3. Optical Axis....719
14.3.2.4. Best Focus Virtual Image....720

DRAFT

14.3.2.5. Tangential Virtual Image....722
14.3.2.6. Sagittal Virtual Image....722
14.3.2.7. Best Focus Spot....723
14.3.2.8. Tangential Spot....723
14.3.2.9. Sagittal Spot....724
14.3.2.10. Astigmatism....724
14.3.2.11. Static Distortion....724
14.3.2.12. Dynamic Distortion....725
14.3.2.13. Optical Volume....725
14.3.2.14. Pixel Image....725
14.3.2.15. Ghost Image Optical Axis....726
14.3.2.16. Ghost Image....726
14.3.2.17. PGU....726
14.3.2.18. Warping....727
14.3.2.19. Visualizing a Speos360 Result....727
14.3.3. HOA Tests APIs....728
14.3.3.1. Test APIs....728
14.3.3.2. Image Warping APIs....753
14.3.3.3. Debugging a Plugin....765
15: Optimization....768
15.1. Optimization with Ansys Workbench....768
15.1.1. Speos in Ansys Workbench....768
15.1.2. Creating a Speos system in Ansys Workbench....770
15.1.3. Linking Static Structural Solution to Speos Geometry....774
15.1.4. Optimization Tools....775
15.1.4.1. Speos Parameters' Variation....775
15.1.4.2. Speos Direct Optimization....779
15.2. Optimization with Speos....783
15.2.1. Optimization Overview....784
15.2.2. Understanding Optimization Parameters....785
15.2.3. Creating an Optimization....787
15.2.3.1. Defining the Random Search Optimization....788
15.2.3.2. Defining the Optimization Plugin....788
15.2.3.3. Defining the Design of Experiment....789
15.2.4. Adding and Defining Variables....790
15.2.4.1. Adding and Defining Simulation Variables....790
15.2.4.2. Adding and Defining Design Variables....791
15.2.4.3. Adding and Defining Document Variables....792
15.2.5. Adding and Defining Targets....793
15.2.6. Running the Optimization....794
15.2.7. Reading the HTML Report (Random Search)....794
15.2.8. Optimization Plugin....796
15.2.8.1. Optimization Plugin Overview....796
15.2.8.2. Creating a Project in Visual Studio....797

DRAFT

15.2.8.3. Creating an Optimization Plugin....799
15.2.8.4. Compiling the Project with Visual Studio....810
15.2.8.5. Configuring the XML Optimizer Plugin Configuration File....810
15.3. Optimization with optiSLang....812
15.3.1. Speos in optiSLang....812
15.3.2. Defining the Speos Parameters to Use in optiSLang....813
16: Automation....815
16.1. What is Automation....815
16.2. Methodology....815
16.3. Creating a Script....816
16.4. Forbidden Values Management....818
16.5. Methods....818
16.5.1. Generic Methods....819
16.5.1.1. Common Methods....819
16.5.1.2. Set and Get Methods....819
16.5.1.3. List (Enum) Parameters....820
16.5.1.4. Geometry Selection Methods....820
16.5.1.5. ConvertToScriptVersion....821
16.5.2. Specific Methods....821
16.5.2.1. Light Expert Methods....821
16.5.2.2. Simulation Methods....822
16.5.2.3. Simulation Settings....822
16.5.2.4. Command Methods....824
16.5.2.5. SourceRayFile Methods....825
16.5.2.6. LightGuide Methods....825
16.5.2.7. ControlPointConfiguration Methods....826
16.5.2.8. ProjectionLens Methods....827
16.5.2.9. ControlPlane Methods....827
16.5.2.10. PERAngularSection Methods....828
16.5.2.11. EyeboxConfiguration Methods....828
16.5.2.12. HUDOD Advanced Parameters Methods....829
16.5.2.13. CADUpdate Methods....829
16.5.3. Speos Core Methods....830
16.5.3.1. OpenFile....830
16.5.3.2. RunSimulation....831
16.5.3.3. ShowWindow....831
16.5.3.4. Speos Core Command Lines....832
17: Block Recording Tool....833
17.1. Understanding the Block Recording Tool....833
17.2. Configuring the Environment for the Block Recording Tool....835
17.3. Non-Compatible Speos Features and Actions with the Recording Tool....836
17.4. Troubleshooting: Smart Variable with Wrong Selection....837
18: Speos Sensor System Exporter....839
18.1. Speos Sensor System Exporter Overview....839

DRAFT

18.2. YAML Files Description....840
18.2.1. YAML File Code Introduction....841
18.2.2. YAML Input Parameters File....842
18.2.2.1. YAML Input Parameters File Template....842
18.2.2.2. Debug Options....843
18.2.2.3. Working Modes....844
18.2.2.4. General Remarks....849
18.2.3. YAML Sensor Properties File....849
18.2.3.1. YAML Sensor Properties File Overview....849
18.2.3.2. YAML Sensor Properties File Template....849
18.2.3.3. EMVA Standard Version....852
18.2.3.4. References....852
18.2.3.5. Operating Conditions....852
18.2.3.6. Properties....853
18.2.3.7. Pre-Processing....854
18.2.3.8. Lumerical Data....855
18.2.3.9. EMVA Data....855
18.2.3.10. Development....858
19: Troubleshooting....863
19.1. Known Issues....863
19.1.1. Materials....863
19.1.2. Sources....863
19.1.3. Sensors....863
19.1.4. Components....864
19.1.5. Simulation....864
19.1.6. Optical Part Design....865
19.1.7. Head-Up Display....866
19.1.8. Results....866
19.1.9. Automation....867
19.1.10. Miscellaneous....867
19.2. Error Messages....869
19.2.1. Not enough Speos HPC Licenses....869
19.2.2. Proportional to Body size STEP and SAG parameters are not respected....870
19.2.3. Surface Extrapolated....871
19.2.4. Invalid Support: Offset Support Is Not Possible....871
20.1.1. Copyright and Trademark Information....872

DRAFT

1: Welcome!

This document provides you with conceptual information and detailed procedures to get the best out of Speos.

Speos lets you design and optimize light and systems. Validate the ergonomics of your product and take a virtual picture of it to review designs collaboratively.

Refer to the **Release Note** to see what's new in the latest version.

Main Features:

- Optical Properties Optical properties define how light rays interact with geometries.
- Components Components can be used for data exchange between suppliers and customers. It is compatible with multi-CAD platform where Ansys software are integrated.
- Sources Sources are light sources propagating rays in an optical system.
- Sensors Sensors integrate rays coming from the source to analyze the optical result in the optical system.
- Simulations Simulations give life to the optical system to generate the results, by propagating rays between sources and sensors.
- Optical Part Design Optical Part Design provides geometrical modeling capabilities dedicated to optical and lighting systems.
- Head Up Display Head Up Display is a system that allows you to present data on a transparent display, usually a windshield, without having to look away from the initial viewpoint.
- Optimization Optimization helps find the best solution for your optical system according to an expected result and parameters to be varied.
- Automation Automation allows you to control and automate actions in Speos with routines created thanks to the provided APIs.

DRAFT

2: Ansys Product Improvement Program

This product is covered by the Ansys Product Improvement Program, which enables Ansys, Inc., to collect and analyze *anonymous* usage data reported by our software without affecting your work or product performance. Analyzing product usage data helps us to understand customer usage trends and patterns, interests, and quality or performance issues. The data enable us to develop or enhance product features that better address your needs.

How to Participate

The program is voluntary. To participate, select **Yes** when the Product Improvement Program dialog appears. Only then will collection of data for this product begin.

How the Program Works

After you agree to participate, the product collects anonymous usage data during each session. When you end the session, the collected data is sent to a secure server accessible only to authorized Ansys employees. After Ansys receives the data, various statistical measures such as distributions, counts, means, medians, modes, etc., are used to understand and analyze the data.

Data We Collect

The data we collect under the Ansys Product Improvement Program are limited. The types and amounts of collected data vary from product to product. Typically, the data fall into the categories listed here:

Hardware: Information about the hardware on which the product is running, such as the:

- brand and type of CPU
- number of processors available
- amount of memory available
- brand and type of graphics card

System: Configuration information about the system the product is running on, such as the:

- operating system and version
- country code
- time zone
- language used
- values of environment variables used by the product

Session: Characteristics of the session, such as the:

- interactive or batch setting
- time duration
- total CPU time used
- product license and license settings being used
- product version and build identifiers
- command line options used
- number of processors used
- amount of memory used
- errors and warnings issued

Session Actions: Counts of certain user actions during a session, such as the number of:

DRAFT

- project saves
- restarts
- meshing, solving, postprocessing, etc., actions
- times the Help system is used
- times wizards are used
- toolbar selections

Model: Statistics of the model used in the simulation, such as the:

- number and types of entities used, such as nodes, elements, cells, surfaces, primitives, etc.
- number of material types, loading types, boundary conditions, species, etc.
- number and types of coordinate systems used
- system of units used
- dimensionality (1-D, 2-D, 3-D)

Analysis: Characteristics of the analysis, such as the:

- physics types used
- linear and nonlinear behaviors
- time and frequency domains (static, steady-state, transient, modal, harmonic, etc.)
- analysis options used

Solution: Characteristics of the solution performed, including:

- the choice of solvers and solver options
- the solution controls used, such as convergence criteria, precision settings, and tuning options
- solver statistics such as the number of equations, number of load steps, number of design points, etc.

Specialty: Special options or features used, such as:

- user-provided plug-ins and routines
- coupling of analyses with other Ansys products

Data We Do Not Collect

The Product Improvement Program does *not* collect any information that can identify you personally, your company, or your intellectual property. This includes, but is not limited to:

- names, addresses, or usernames
- file names, part names, or other user-supplied labels
- geometry- or design-specific inputs, such as coordinate values or locations, thicknesses, or other dimensional values
- actual values of material properties, loadings, or any other real-valued user-supplied data

In addition to collecting only anonymous data, we make no record of where we collect data from. We therefore cannot associate collected data with any specific customer, company, or location.

Opting Out of the Program

You may *stop* your participation in the program any time you wish. To do so:

1. Select **File** > **Speos Options**.
2. Select the **Light Simulation** tab and click **Ansys Product Improvement Program**.

   A dialog appears and asks if you want to continue participating in the program.

3. Select **No** and then click **OK**.

   Data will no longer be collected or sent.

DRAFT

The Ansys, Inc., Privacy Policy

All Ansys products are covered by the Ansys, Inc., Privacy Policy.

Frequently Asked Questions

1. *Am I required to participate in this program?*

   No, your participation is voluntary. We encourage you to participate, however, as it helps us create products that will better meet your future needs.

2. *Am I automatically enrolled in this program?*

   No. You are not enrolled unless you explicitly agree to participate.

3. *Does participating in this program put my intellectual property at risk of being collected or discovered by Ansys?*

   No. We do not collect any project-specific, company-specific, or model-specific information.

4. *Can I stop participating even after I agree to participate?*

   Yes, you can stop participating at any time. To do so:

   a. Select **File** > **Speos Options**.
   b. Select the **Light Simulation** tab and click **Ansys Product Improvement Program**.

      A dialog appears and asks if you want to continue participating in the program.

   c. Select **No** and then click **OK**.

      Data will no longer be collected or sent.

5. *Will participation in the program slow the performance of the product?*

   No, the data collection does not affect the product performance in any significant way. The amount of data collected is very small.

6. *How frequently is data collected and sent to Ansys servers?*

   The data is collected during each use session of the product. The collected data is sent to a secure server once per session, when you exit the product.

7. *Is this program available in all Ansys products?*

   Not at this time, although we are adding it to more of our products at each release. The program is available in a product only if this *Ansys Product Improvement Program* description appears in the product documentation, as it does here for this product.

8. *If I enroll in the program for this product, am I automatically enrolled in the program for the other Ansys products I use on the same machine?*

   Yes. Your enrollment choice applies to all Ansys products you use on the same machine. Similarly, if you end your enrollment in the program for one product, you end your enrollment for all Ansys products on that machine.

9. *How is enrollment in the Product Improvement Program determined if I use Ansys products in a cluster?*

   In a cluster configuration, the Product Improvement Program enrollment is determined by the host machine setting.

10. *Can I easily opt out of the Product Improvement Program for all clients in my network installation?*

    Yes. Perform the following steps on the file server:

    a. Navigate to the installation directory: `[Drive:]\v241\commonfiles\globalsettings`

    b. Open the file **ANSYSProductImprovementProgram.txt**.

DRAFT

c. Change the value from "on" to "off" and save the file.

DRAFT

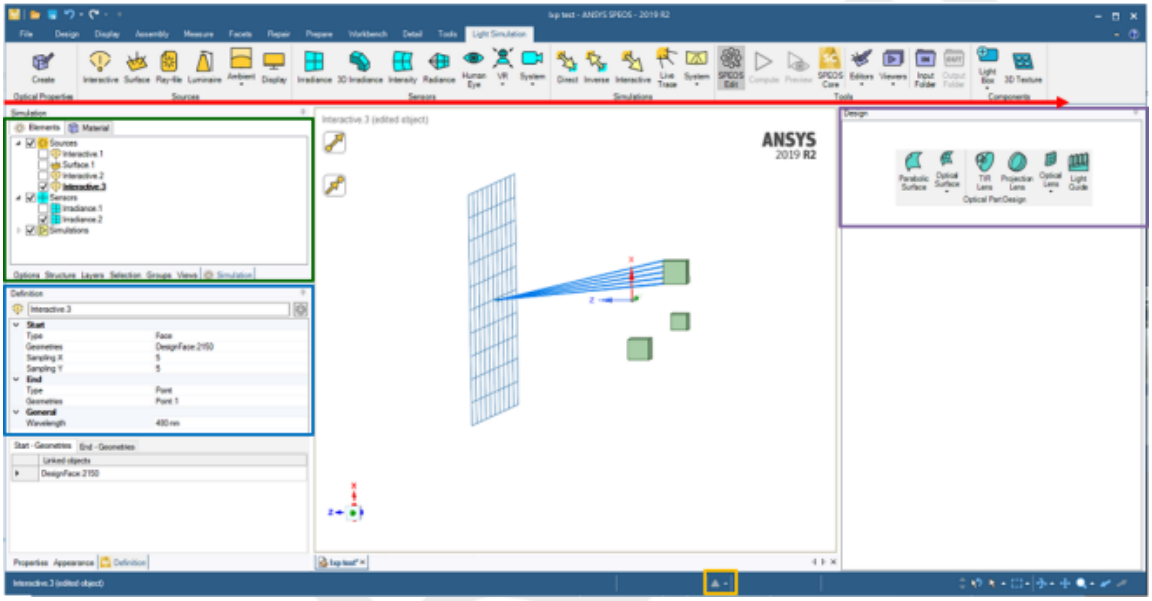
3: Speos Software Overview

This section presents an overview of Speos software (interface, settings, preferences, navigation and tools).

3.1. Graphical User Interface

This page gives a general overview of Speos interface and helps to better understand the working environment.

General Interface

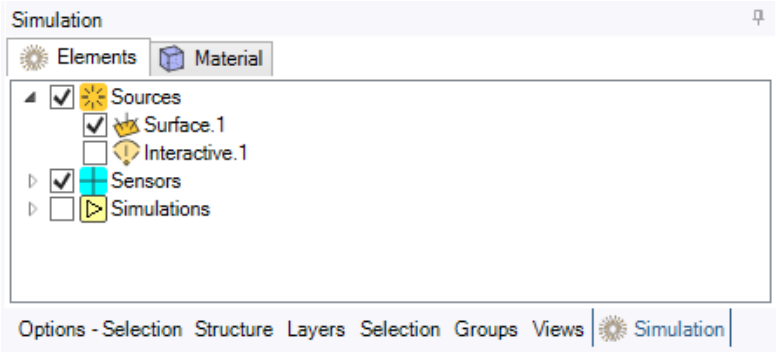


- Speos Features: Work flow based layout.
- Simulation panel (green section): All Speos features are stored here except Optical Part Design features.
- Definition panel (blue section): Displays the parameters of the active Speos feature.
- Design panel (purple section): Displays Optical Part Design features and is only displayed when a feature is created. The Optical Part Design menu is stored in the Design tab.
- Status Information (yellow section): Warnings, errors and notifications are available here.

Interface Highlights

- From the **Simulation** panel, you can visualize the features' state and control their visibility in the 3D view.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



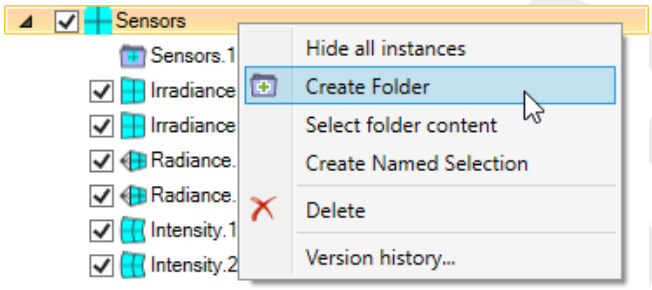
A feature is:

- bolded and underlined when it is in edition. The feature is then considered as active.
- in error when an exclamation point appears after the feature's name.

> **Note:** A feature appears in error if it is not defined or has not been correctly defined.

- hidden in the 3D view when the check box is clear.

- From the **Simulation** panel, you can create unlimited folders and sub-folders under each Speos object family and organize objects as want.



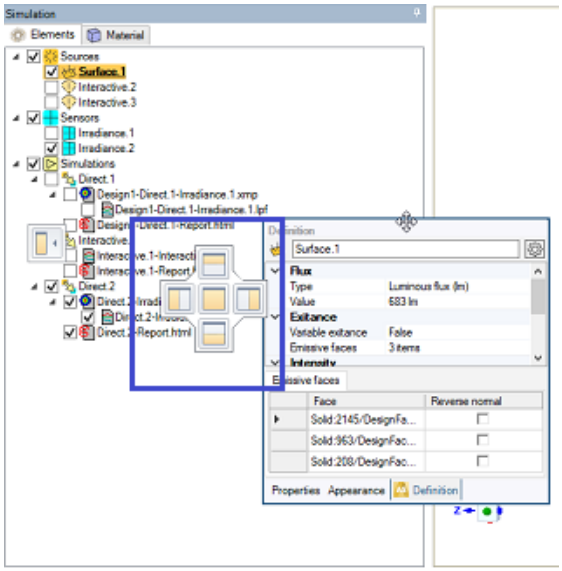
> **Tip:** You can drag and drop Speos objects in and out of the folders created and to reorganise them inside a folder (except for Materials).

- The **Definition** panel is dedicated to Speos features.
- The **Design** panel only appears when an optical design feature is created.

> **Tip:** All panels of the interface can be moved. Simply drag a panel away from a dockable location to drop it where you want to place it.
>
> To restore the original layout at any time, go to **File** > **Speos Options** and in **Appearance**, click **Reset Docking Layout.**

> **Note:** Modifying the Speos tree panels position is not kept when reopening Speos after opening a SpaceClaim session without Speos.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



Related tasks

Using the Feature Contextual Menu on page 29
This page lists all the operations that can be performed from the features' contextual menu.

Related information

Useful Commands and Design Tools on page 30
This page describes Speos selection behavior and provides some useful commands, shortcuts and design tools that are frequently used during feature definition.

3.2. Launching Speos Using Command Line

The following page helps you launch Speos using command lines.

Launching Speos using command lines is useful in case of task automation, to launch scripts with Speos in headless mode.

Command Lines

Command Line	Description
`AnsysSpeosLauncher.exe`	Starts Speos with: • default SpaceClaim.exe • default environment variables for resources
`AnsysSpeosLauncher.exe "file.scdocx"`	Starts Speos and the file defined with: • default SpaceClaim.exe • default environment variables for resources
`AnsysSpeosLauncher.exe "file.scdocx" /[optionname]=[value]`	Starts Speos and the file defined with: • default SpaceClaim.exe • default environment variables for resources • Spaceclaim arguments defined.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Note: All SpaceClaim arguments are supported. For the full arguments list, refer to the following **page** in the SpaceClaim documentation.

3.3. Setting Speos Preferences

This page shows how to set the Speos preferences and general options.

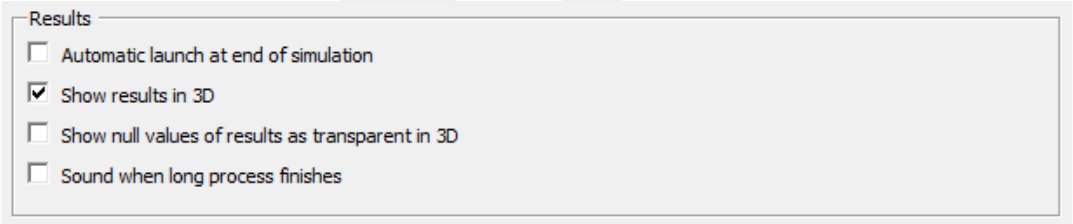
To set Speos preferences:

1. Click **File** > **Speos options**
2. Click the **Light Simulation** section.
3. In **Precision of parameters**, define how many decimals to allow for certain parameters:



   - Define how many decimals need to be available to define the length of rays.
   - Define how many decimals need to be available to define angular values.

4. In **Results**, define the behavior of the simulation results:



   - Check **Automatic launch at end of simulation** if you want the simulations' results to be automatically opened with their associated viewer at the end of the simulation.
   - Deselect **Draw results in 3D** if you do not want the results to be displayed in the 3D view at the end of the simulation.
   - Select **Show null values of results as transparent in 3D** if you do not want to display the black part of a 2D irradiance result in the 3D view.

     This parameter is only a visualization parameter and does not modify the result.

   - Check **Sound when long process finishes** to be warned when a feature has finished its computation.

     Note: This options only concerns certain processes: Direct and Inverse simulations, 3D Textures, Optical Surface, Optical Lens and Light Guide.

5. In **Simulations**, adjust simulation behavior:



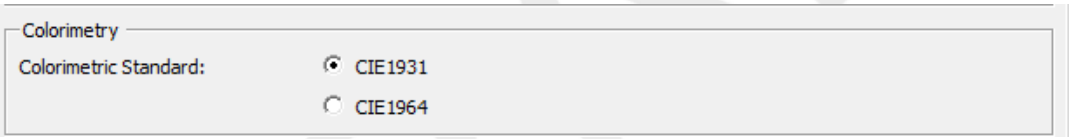
- Define the **Number of thread** to use for direct and inverse simulations.
- Check **VR Sensor Memory Management** to activate the memory management.

> **Note:** Disabling this option can greatly improve simulation performance, but in that case, make sure that you have enough memory on your computer to generate Speos360 files.
>
> You must have more than 6 GB available for immersive sensor and more than X GB available for an observer sensor (X corresponding to the number of positions defined in the sensor).

- Check **Automatic "Save All" before running a simulation** if you want to trigger a backup of the project before the simulation is launched.

> **Note:** This option does not apply to interactive simulations because they are automatically updated.

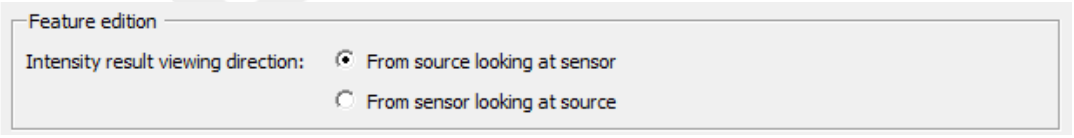
6. In **Colorimetry**, select the default **Colorimetric Standard** to be used for all simulations.



- **CIE 1931**: 2 degrees CIE Standard Colorimetric Observer Data. Only one fovea, which covers about a 2-degree angle of vision, was used during the experiments leading to the 1931 standard observer.
- **CIE 1964**: 10 degrees CIE Standard Colorimetric Observer Data. The 1964 standard observer was based on color-matching experiments using a 10-degree area on the retina.

> **Note:** For The CIE 1964, the luminous level is not correct whatever the unit. Only use the CIE 1964 for color display and color analysis with colorimetric and spectral map.

7. In **Feature Edition**, select the intensity result viewing direction to use for the sensors:



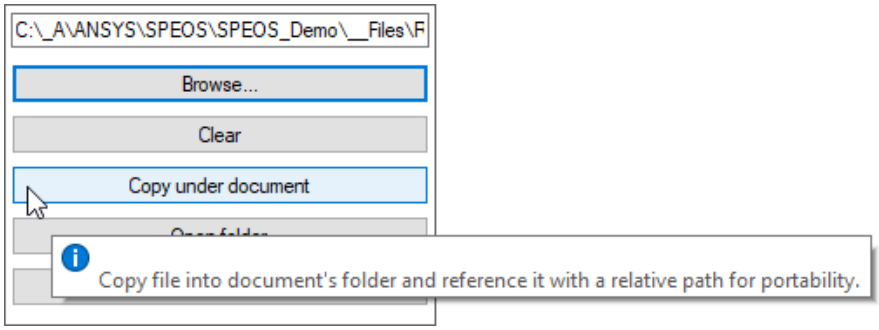
- Select **From source looking at sensor** to position the observer point from where light is emitted.
- Select **From sensor looking at source** to position the observer in the opposite of light direction.

8. In **File Management**, check **Automatically copy selected files under document's folder** to embed any input file selected outside the SPEOS Input Files directory in the current project.

   This option ensures the project's portability as it copies any imported file back into the **SPEOS Input Files** directory.

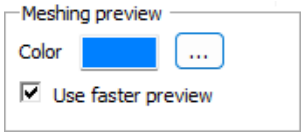
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

If you do not check the option to automatically copy the files, you can still click **Copy under document** to manually perform the copy to the Speos Input Files directory of the current project.



> **Note:** Both options do not apply to *.SPEOSLightBox files, 3D Textures, and CAD assemblies imported using the Geometry Update tool.

9. In **Data import/export**, deactivate **Import/Export geometries without interoperator healing** if you do not want to apply, during import or export, the healing operations that you can find in the Repair tab.
10. In **Preset**, if you want to customize the preset repository, check **Use custom folder path** and browse one.
    For more information, refer to Customizing the Preset Repository on page 44.
11. In **Meshing preview**, you can:



- check **Use faster preview** in order to display the Meshing preview faster in the 3D view.

  Use faster preview option only applies to the Meshing preview option.

  > **Note: Use faster preview** can have some minor rendering issues on its visualization. This does not affect the meshing used for simulation.

- select a specific meshing preview **Color** for a better visibility of the meshing and objects in the 3D view.

  The color applies both to the Meshing preview (Use faster preview) option and the classic meshing.

12. In the **GPU** tab, you can:

- define the GPUs to use to run the simulations with the GPU Compute option.

  > **Note:** If you select multiple GPUs, simulations will consume the sum of the equivalent cores per GPU.
  >
  > If you select no GPU, Speos will automatically select the most powerful available.

- **Allow XMP generation from Simulation Preview** in order to export as XMP or as picture the current live preview result when running the Live Preview.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

> **Note:** When **Allow XMP generation from Simulation Preview** is activated, the Live Preview consumes the sum of equivalent core per GPU and uses all selected GPUs (rather than just the most powerful of the list).

13. From the **Warnings tab**, check or clear the check boxes to activate or deactivate specific warnings.

Speos preferences are set.

Related tasks

Using the Feature Contextual Menu on page 29
This page lists all the operations that can be performed from the features' contextual menu.

Related information

Useful Commands and Design Tools on page 30
This page describes Speos selection behavior and provides some useful commands, shortcuts and design tools that are frequently used during feature definition.

3.4. Geometry Modeler

The geometry modeler manages how geometries are created and stored in SpaceClaim.

As of version 2024 R1, Speos relies on the Parasolid modeler. The ACIS modeler has been removed.

The conversion from a *.scdoc to a *.scdocx still works.

> **Note:** Projects saved with the Parasolid modeler BETA in version 2021 R2 might not be supported in version 2022 R1 and future releases.

Modeler Limitations

Bounding Box Dimension

The bounding box dimension is limited to [-500m ; 500m] in every direction for a component. You can have several components in a project as long as each component box respects the size limit.

- If you import a project .scdoc into a *.scdocx or import a CAD file, make sure each component of the project is smaller than the bounding box dimensions.
- If you directly create a project, make sure to create objects that are smaller than the bounding box dimensions.

Meshing

In case of a thin body, make sure to apply a fixed **Meshing sag mode** and a **Meshing sag value** smaller than the thickness of the body. Otherwise you may generate incorrect results.

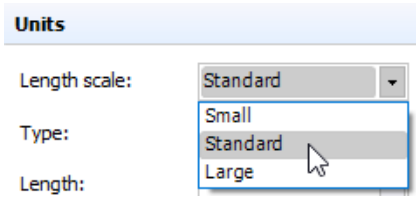
For more information on Meshing, refer to Meshing Properties.

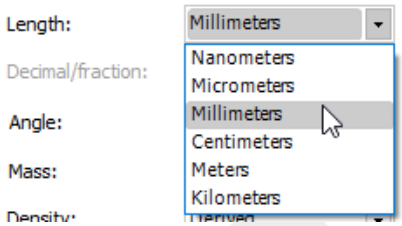
DRAFT

Scale Formats

Only use the **Standard** Length scale is supported by Speos. The **Standard** Length scale corresponds to the **Millimeters**, **Centimeters**, and **Meters** Lengths.

If you set a length smaller than Millimeters via the SpaceClaim Options, only the SpaceClaim environment will be set to this scale and not the Speos one.





3.5. Lightweight/Heavyweight Import

Lightweight and Heavyweight correspond to the information load of imported data from CAD files.

3.5.1. Lightweight/Heavyweight Import Overview

Importing CAD files (CATIA, IGES, STEP) import them in Lightweight. Lightweight is a format that loads a lighter level of detail of data than Heavyweight format.

When you import CAD files (CATIA, IGES, STEP), they are imported in Lightweight. That means, the files are loaded with a lighter level of detail of data than a full load (Heavyweight), which reduces the import time.

The Heavyweight format corresponds to the full dataset of an imported element and is need to edit the element.

Characteristics

- The Lightweight import uses the SpaceClaim Reader.
- Bodies imported in Lightweight are not editable.

  To edit a Lightweight body, you have to switch it from Lightweight to Heavyweight using **Toggle to Heavyweight**, which will load all the data of the body.

  > **Note:** You cannot switch from Heavyweight to Lightweight.

- When you launch a simulation, Lightweight geometries are converted to Heavyweight geometries and the SpaceClaim mesh is applied during the initialization of the simulation, while keeping the visualization of the geometries in the 3D view in Lightweight. This prevents then a loss of performance. Due to the conversion from Lightweight to Heavyweight, the initialization can take more time than usual.

  > **Note:** In case of a large project with multiple simulations, to avoid the conversion from Lightweight to Heavyweight of the geometries during the initialization of each simulation, we recommend you to toggle to heavyweight the geometries and apply the meshing manually. This will save you the conversion time during the initialization.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- You can **Save as** a project containing Lightweight bodies. The project will be lighter than a project saved with Heavyweight bodies
- You cannot **Save as** CATPart Lightweight bodies as they are not editable. Only Heavyweight bodies can be saved as CATPart.
- You cannot export axis systems into *.CATPart files.

  You may have to create manually your axis systems using point and lines if you want to reproduce them.

- A named selection created out of geometries imported with the **Geometries Update** tool may not detect geometry modifications applied in the external CAD part after updating this external CAD part. Therefore, the modified geometry of the named selection is not updated in SpaceClaim.

> **Warning:** Do not confuse the Lightweight which applies to bodies with the Lightweight function from SpaceClaim which applies to the root component.

Lightweight Import Support

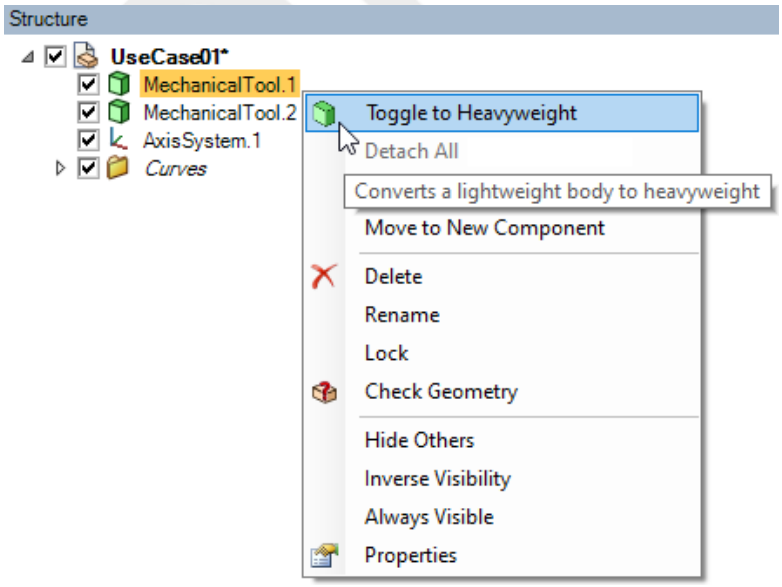
File Type	Extension	File Version	Reader
CATIA V5	*.CATPart ; *.CATProduct	Up to V5_6R2023	SpaceClaim Reader
IGES	*.IGES ; *.IGS	5.1, 5.2, 5.3	SpaceClaim Reader
STEP	*.STEP ; *.STP	AP 203 E1/E2, AP 214, AP 242	SpaceClaim Reader

3.5.2. Switching a Body from Lightweight to Heavyweight

The following procedure shows how to switch a body from lightweight to heavyweight in order to edit a body.

To switch a body from Lightweight to Heavyweight:

1. In the **Structure** tree, right-click a body.
2. Click **Toggle to Heavyweight**.



The body is loaded in heavyweight and now you can edit it.

3.5.3. Lightweight/Heavyweight Import Troubleshooting

The Troubleshooting page provides you with different solutions in case you encounter errors after a CAD File import.

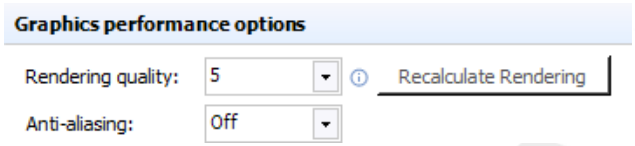
> **Note:** The solution numbering does mean the solutions have to be followed in order.

Solution 1: Approximation Measuring CAD Files Imported in Lightweight

After importing CAD files into Speos in Lightweight, if you measure an imported element, the measure value is approximated compared to the original CAD measure value, and depends on the Rendering quality option (SpaceClaim option).

Try modifying the Rendering quality option (SpaceClaim option):

1. In **Speos Options** > **Popular** > **Graphics performance options**, modify the **Rendering quality** option.



For more information on the Rendering quality option, refer to the **SpaceClaim documentation**.

Solution 2: Errors with Lightweight Bodies

After importing a CAD project or after running a simulation with Lightweight bodies raising an error, we recommend you to:

1. Right-click each geometry and select **Check geometry** to find potential issues.

   If some issues are found:

   a. Switch the geometry from Lightweight to Heavyweight if not yet done.
   b. Select **Check geometry** again.

      If there are still errors, delete the faulty faces.

   c. Use the **Stitch** and **Inexact Edges** tools from the **Repair** tab.
   d. Solidify the geometry with the **Missing Faces** tool from the **Repair** tab.

2. Check the meshing and the Geometrical Distance Tolerance simulation options.

   Make sure that the Geometrical Distance Tolerance is much larger than the meshing tolerance: **GDT >> sag(Body1) + sag(Body2)**

3. Check the meshing using **Preview Meshing** in a simulation or using a **Local Meshing**.

Solution 3: Errors with Imported CATIA File

A CATIA file imported into Speos may result in error. Errors may appear according to different scenarios. For instance:

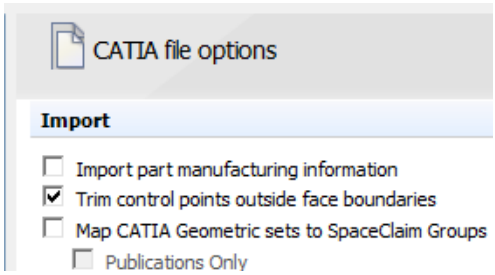
- When switching a body from Lightweight to Heavyweight
- When running a HUD Optical Analysis using an imported CATIA file
- An incorrect preview of a meshing

DRAFT

- etc.

If above solution 2 has not corrected the issue, according to the configuration, you can try the following:

1. In **Speos Options** > **File Options** > **CATIA**, activate or deactivate the option **Trim control point outside face boundaries**.



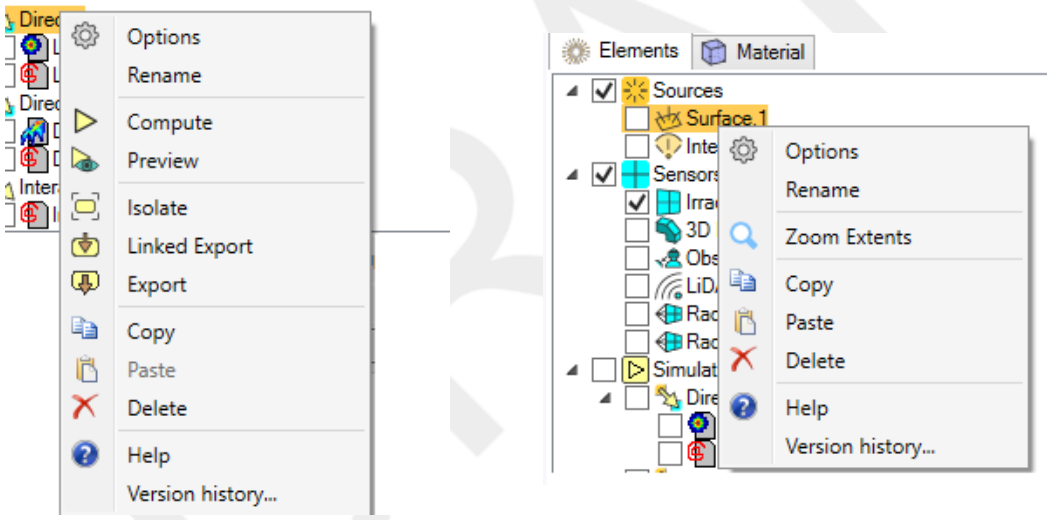
2. Try importing the CATIA file again.

3.6. Using the Feature Contextual Menu

This page lists all the operations that can be performed from the features' contextual menu.

The feature's contextual menus offer a defined set of actions. The actions available in the context menu depend on the feature state and type.

You can access the menu by right-clicking a feature in the **Simulation** panel.



The actions available from this menu are the following:

- **Rename**, **Delete**, **Copy** and **Paste** any feature.

> **Note:** Any source, sensor or simulation can be copied. When you create a feature's copy, you inherit from its definition.
>
> To delete a feature, make sure this feature has no dependencies with other features (a source selected in a simulation for example), otherwise the feature will not be deleted from the Tree.

- **Zoom Extents** allows you to reframe on the feature in the 3D view.

- Automatic framing allows you to reframe the 3D view on a sensor's point of view.
- **Options** allows you to access the optional or advanced settings of a feature.

> **Note:** Advanced simulation settings are only available from the simulation feature contextual menu.

- **Compute** allows you to launch a simulation.
- **Automatic Compute** allows you to switch from a manual to an automatic mode. When this mode is activated, the simulation is automatically updated when a parameter or an input of the simulation is edited.
- Speos HPC Compute allows you to submit a Speos HPC job or an Ansys Cloud job.
- Preview allows you to launch a simulation using the progressive rendering.

  This option is only available with NVIDIA GPU and only displays when the simulation is compatible with progressive rendering.

- Isolate allows you to isolate a simulation result and creates a link in the feature tree to access the folder of the isolated results.
- Linked Export allows you to isolate and export simulation results while keeping a link between the project and the simulation.
- Export allows you to fully export a simulation.
- Export as Geometry allows you to convert rays into construction lines.

Related information

Graphical User Interface on page 19
This page gives a general overview of Speos interface and helps to better understand the working environment.

Useful Commands and Design Tools on page 30
This page describes Speos selection behavior and provides some useful commands, shortcuts and design tools that are frequently used during feature definition.

Computing Simulations on page 371
This page describes the different ways to run a simulation inSpeos.

3.7. Useful Commands and Design Tools

This page describes Speos selection behavior and provides some useful commands, shortcuts and design tools that are frequently used during feature definition.

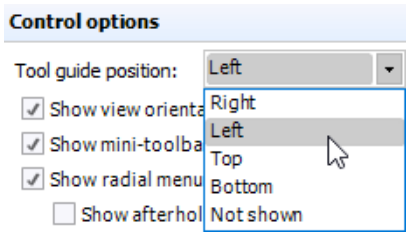
3D view Icons

It may appear sometimes that the icons in the 3D view disappear due to a wrong move.

To display them again:

1. Select **File** > **Speos Options**.
2. Select the **Popular** tab.
3. In the **Control Options** section, set **Tool guide position** to another value than **Not shown**.

DRAFT



Primary Selection Behavior

Standard commands are available and compatible with geometry, source and sensor selection.

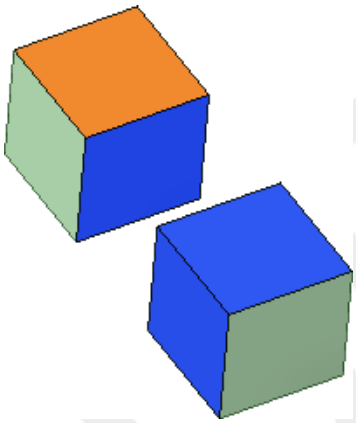
- CTRL and shift commands are available for quick multi-selections.
- CTRL+A selects all bodies present in the scene.
- Box selection selects the set of highlighted bodies.
- ESC key allows you to exit/cancel a selection.

Each selection must be validated for the objects to be correctly imported in the Speos objects' list.

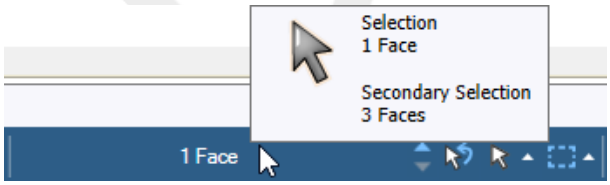
Secondary Selection Behavior

The Secondary Selection allows you to preserve the selection when switching through Speos tools (whereas the Primary selection is currently modified as soon as an object is edited with a selection command):

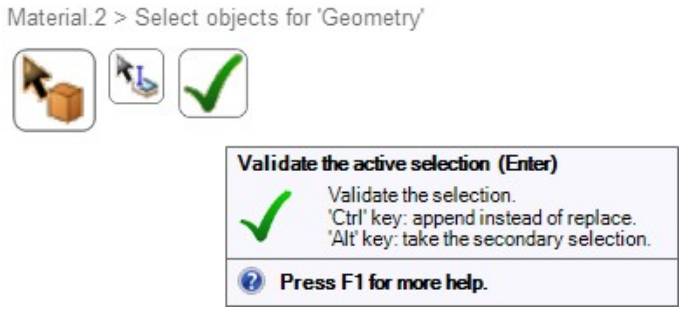
- Hold ALT key to make a secondary selection (appearing in blue).
- Hold CTRL+ALT to make a secondary multiple selection.



- To check the status of primary and secondary selection, mouse-over the bottom tool bar as follows:



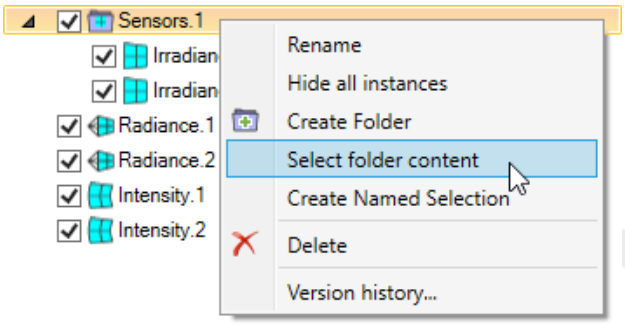
As the Primary Selection, you can use the Secondary Selection with the Speos commands to fill selections such as the Validation command : Hold the ALT key and click **Validate** .



Simulation Panel Selection Behavior

In the **Simulation** Panel, you can create folders and sub-folders under each Speos object family. The **Select folder content** command allows you to select only the direct content of the folder.

To select the content of a folder, right-click the folder and click **Select folder content**.



Disable Preselection

The **Disable preselection** command removes the preselection (temporary highlight) of 3d view elements in order to gain performance on the interface reactivity.

To disable the preselection, right-click anywhere in the 3D view, and check **Disable preselection**.

Revert Selection

In case you lose your selection or want to go back to a previous selection, use the **Revert Selection** command  (bottom right of SpaceClaim interface) to restore the state of your previous or lost selection.

Grouping

Groups or **Named Selections** allow you to better organize your project and save time during element selection. They also allow you to save memory when working with data separation by layer as groups represent one layer in the simulation result.

The grouping function is available for any Speos object (sources, sensors, geometries, OPD face groups etc.).

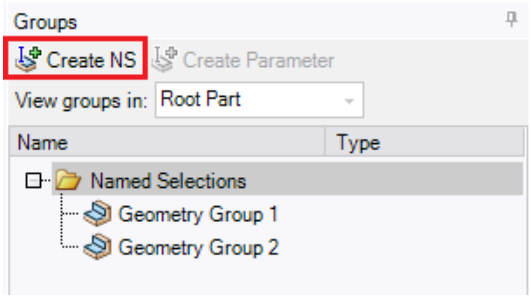
> **Note:** A group should contain a unique set of items. We recommend grouping the same type of objects to ease group management.

DRAFT

Important: You cannot select a component to embed its geometries into a Named Selection. You must select directly the geometries to add them into the Named Selection.

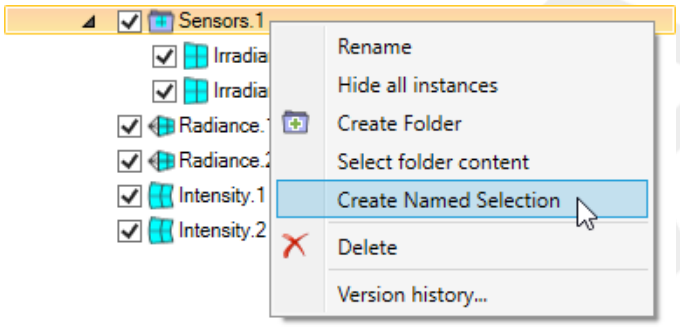
Grouping using Create NS

1. To create **Named Selections**, select objects in the 3D view, from the **Structure** or from the **Simulation** panel.
2. In the **Groups** panel click **Create NS**.



Grouping using the Contextual Menu

From the **Simulation** panel, right-click a node, folder, or sub-folder, and select **Create Named Selection**.



A named selection of the folder content is created in the Groups panel with the name of the folder.

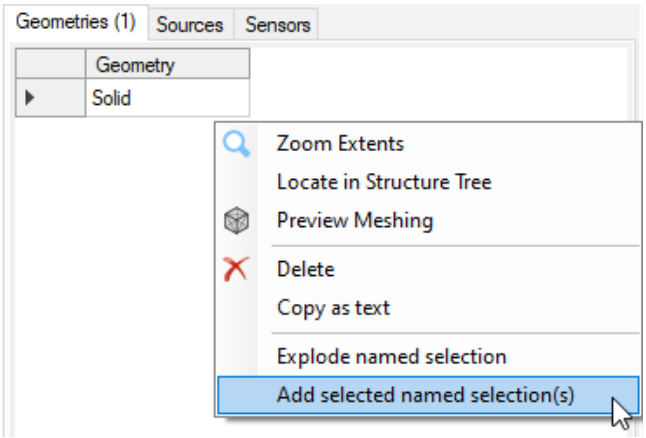
Note: If you modify the content of the folder, the named selection group created from the folder is not automatically updated. You have to recreate the named selection group.

After Grouping

Once created, a named selection can be:

- Selected for a Speos feature (a simulation for example) by either using the 3D view selection tools or the contextual menu of the definition panel.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



- Identified from the groups or the simulation panel at any time.

  In the **Groups** panel, just hover over a named selection to see the Speos objects it contains being highlighted in the **Simulation** panel. Inversely, from the **Simulation** panel, right-click a Speos object and click **Show the named selection(...)** to highlight its associated group in the **Groups** panel.

  > **Note:** Hidden geometries belonging to a group are not highlighted in the tree when selecting the group.

- Exploded back into a set of separate objects.

  When a group is exploded from the **Groups** panel, each object that was in the group is placed in its own group. The new group names are based on the parent group.

  When a group is exploded from the **Definition** panel, the group is replaced by the set of objects it was containing.

  > **Note:** For more information on grouping, consult the **SpaceClaim documentation**.

Speos Useful Commands

Several shortcuts and commands are available in Speos to ease navigation and feature definition.

The following list is non-exhaustive but describes the most frequently used commands and shortcuts.

- The Tab key allows you to switch lines in the definition panel.
- The up and down arrow keys allow you to flip through the choices of a drop-down list without unrolling it.
- F4 key (or **Speos Edit** button from the ribbon) allows you to enter/exit a feature edition mode.
- ESC key allows you to exit a feature edition mode.
- CRTL+N allows you to create a new design.
- CTRL+O allows you to open a project.
- CTRL+G allows you to create a new **Named Selection** containing the current set of selected (highlighted) objects.

Design Tools

When designing the optical system, you might need to create points or axis systems to place the geometries or features in the scene.

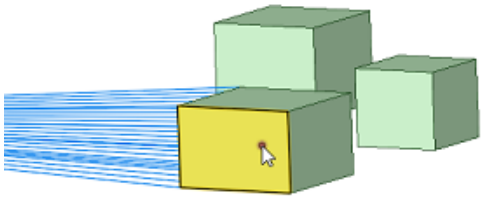
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Points

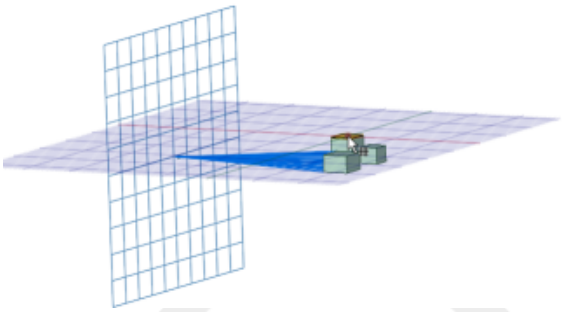
You can use the point creation tool (available from the **Design** tab) to place points in the scene.

To place the point on the right plane/object, you can sketch on a plane or directly in 3D mode:

- To sketch on a plane, place your cursor on a line, edge or surface to automatically change the plane axis.
- Press D to switch to 3D mode and compute the point on a geometry.



3D mode



Plan mode

Axis Systems

When working with Speos features, it might be useful to create and compute axis systems on specific points of interest.

Axis systems allow to better visualize the position and orientation of pieces of a system. It can also be used during feature definition to select directions and origins.

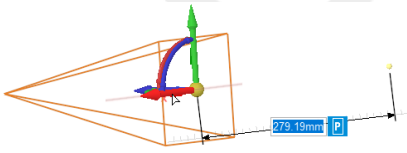
Use the **origin** creation tool to compute a point and its associated axis system (available from the **Design** tab).

Move

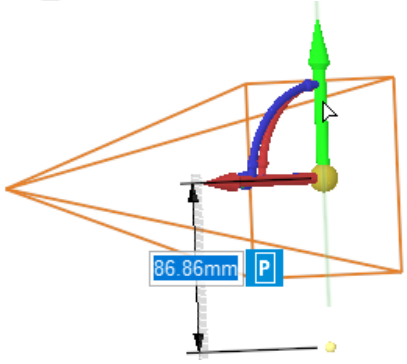
With the **Move** option (available from the **Design** tab), you can move and rotate sensors on any of their axes.

You can also move geometries, even when a source is defined on it.

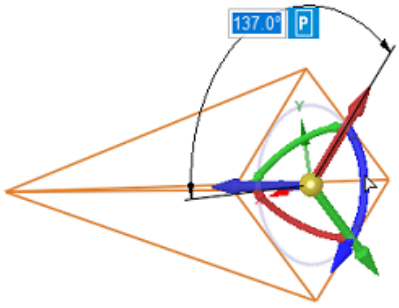
If you want more information, see **Move option from SpaceClaim documentation.**



Move on X Axis



Move on Y Axis



Rotation

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

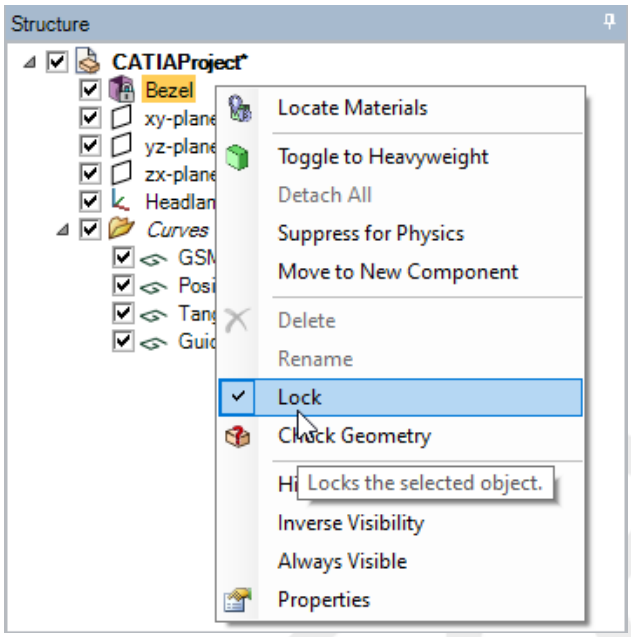
Locking and Unlocking Objects

You can lock the different objects (Surface, Body, Point, Line, Axis System, Plane) available in the Structure tree so they cannot be changed therefore preventing unintended modifications.

Locking Objects

1. In the **Structure** tree, right-click an object.
2. Select **Lock**.

   The **Lock** command is checked.



If an object has sub-components, sub-components are also locked.

Unlocking Objects

1. In the **Structure** tree, right-click a locked object.
2. Select **Lock**.

   The **Lock** command is unchecked.

If an object has sub-components, sub-components are also unlocked.

Related tasks

Using the Feature Contextual Menu on page 29
This page lists all the operations that can be performed from the features' contextual menu.

Related information

Graphical User Interface on page 19
This page gives a general overview of Speos interface and helps to better understand the working environment.

DRAFT

3.8. Extensions and Units

This page lists the extensions and units used in Speos.

Extensions

A system includes different kinds of specific files (spectrum, ray file, material, etc.).

Speos Input Files

SPEOS input files contain all input data as surfaces, materials and spectra, created by the user or downloaded from the Ansys Optical Library specific to the project.

Features	SPEOS Input Files	Extensions	
Optical Properties	Ambient Material File	.material	
	Simple Scattering File	.simplescattering	
	Surface Optical Properties Files	.scattering .brdf .bsdf .bsdf180 .coated .mirror .doe .fluorescent	.grating .retroreflecting .anisotropic .polarizer .anisotropicbsdf .unpolished
	Photon Map File	.pm	

DRAFT

Features	SPEOS Input Files	Extensions
Sources	Spectrum File / Transmittance File	.spectrum
	Intensity Files	.ies .ldt
	HDRI File	.hdr .exr
	Image Files	.bmp .rgb .jpg .tiff .png
	Ray Files	.ray .tm25ray
	Temperature Field File	.OPTTemperatureField
	Surface Optical Properties Files	.scattering .grating .brdf .retroreflecting .anisotropic .bsdf .polarizer .anisotropicbsdf .bsdf180 .unpolished .coated .mirror .doe .fluorescent
	Photon Map File	.pm
Components	3D Texture Mapping	.OPT3DMapping
	Speos Light Box	.SPEOSLightBox
Sensors	Text File	.txt
	Template File	.xmp
	Distortion File	.OPTDistortion File

Speos Output Files

SPEOS output files are automatically created by the software after the simulation is run. They contain result files from simulations.

DRAFT

Feature	SPEOS Output Files	Extensions
Result Files	XMP File	.xmp
	XM3 File	.xm3
	Volume Map File	.vmp
	XMP File (Spectral Irradiance Map)	.Irradiance.xmp
	Light Path Finder Files	.lpf .lp3
	Ray Files	.ray .tm25ray
	Intensity Files	.ies .ltd
	Projected Grid File	.OPTProjectedGrid
	Speos360 File	.speos360
	Photon Map File	.pm
	Speos Light Box	.SPEOSLightBox
Simulation Reports	HTML File	.html

Vocabulary for Photometry and Radiometry Units

Language	Photometry			Radiometry		
	Lx	Cd	L (Cd/m^2)	E (W/m^2)	I (W/sr)	L (W/(sr*m^2))
English	Illuminance	Luminous Intensity	Luminance	Irradiance	Radiant Intensity	Radiance
French	Eclairement	Intensité	Luminance	Eclairement	Intensité	Luminance
German	Beleuchtungsstärke	Lichtstärke	Leuchtdichte	Bestrahlung	Strahlstärke	Strahldichte
Italian	Illuminamento	Intensità	Luminanza	Illuminamento	Intensità	Luminanza
Japanese	照度	光度	輝度	放射照度	放射光度	放射輝度
Chinese	照度	强度	亮度	辐射照度	辐射强度	辐射亮度

3.9. Beta Features

Speos latest release might contain several features in beta mode. This page describes the concept of beta feature and indicates how to enable the access to these features from Speos interface.

What is a Beta Feature?

Beta features are features that are still undergoing development and testing.

DRAFT

The primary purpose of "beta" labeling is to allow an early access to specific features that are close to being finalized.

Should I opt in or out?

Beta features' code and development are, for the most part, complete. Therefore they are operational and can be used/tested.

Keep in mind that these features are not 100% stable and that you could experience some issues like:

- Incorrect values
- Bugs, glitches or visual artifacts

How do I enable these Features?

> **Important:** When the beta option is activated in a previous session, if you open a project in a session with the beta option deactivated, you must reactive the beta option to access beta parameters and beta features.

From Speos interface:

1. Click **File** > **Speos Options**.
2. From the left panel, click **Advanced** and check **Enable beta features**.

3.10. Speos Files Analysis

The Speos Files Analysis tool lists all files that are used as input reference by the Speos objects of the active document.

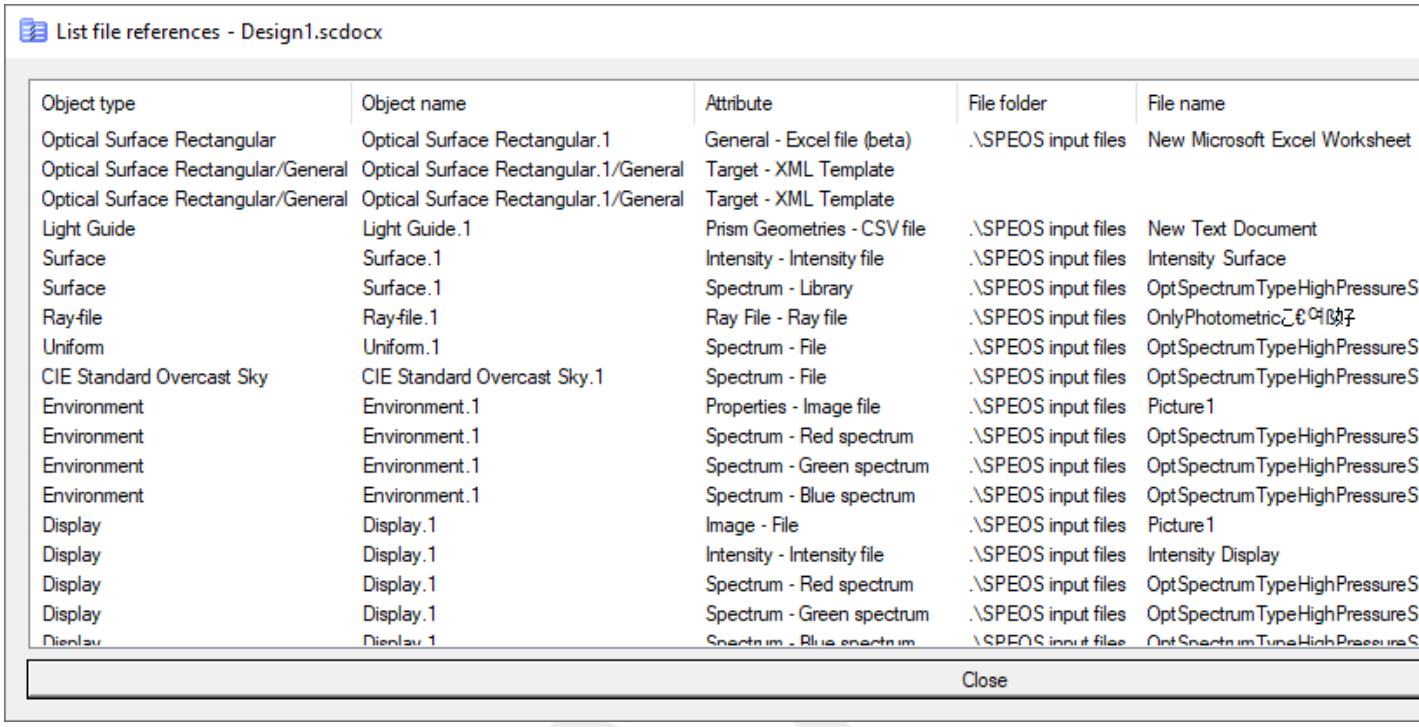
3.10.1. Speos Files Analysis Overview

The following page presents you the goal and the interface of the Speos Files Analysis tool.

When opening a Speos project, some project files may be stored with an absolute path, and Speos cannot find them. The Speos files analysis tool identifies the incorrect or missing file paths, and helps you modify them in order to point to the right directory.

DRAFT

Speos File Analysis Interface



- **Object type** corresponds to the type of the Speos feature.
- **Object name** corresponds to the name you gave in the definition of the Speos feature.
- **Attribute** corresponds to the name of the parameters section and parameter field from the Speos feature definition.
- **File folder** corresponds to the folder where the file is stored.
- **File name** corresponds to the name of the file (without extension).
- **File extension** corresponds to the extension of the file selected.
- **Status** indicates if the file is found or not by the Speos Files Analysis tool.
  - **Found** (path provided, points to an existing file)
  - **Not Found** (path provided, but file not found)
  - **Missing** (blank value, but mandatory)
  - **Not used** (blank value, but not mandatory)
- **Last write** indicates the last time the file has been modified.

3.10.2. Using the Speos File Analysis

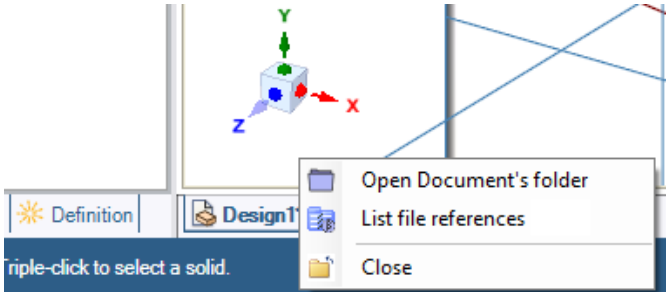
The following procedure helps you open and use the Speos File Analysis tool to quickly see potential incorrect or missing file paths.

To use the Speos Files Analysis tool:

You must have opened a project.

1. In Speos, right-click the project tab.

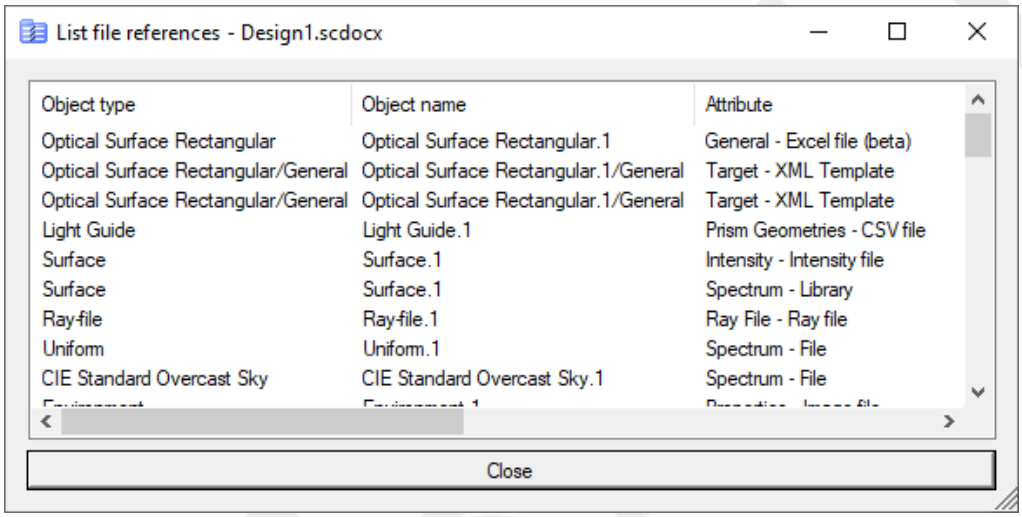
DRAFT



2. Select **List file references**.

> **Note:** You can click each column header to sort them

The Speos Files Analysis tool opens and lists all the input references of the project.



3. According to your needs on a file, you can right-click a file reference line and:

> **Important:** The Speos Files Analysis tool only takes action on the features used in the project. It does not replace/erase/delete files on the FileSystem.

- **Refresh**

  Refresh action allows to update the file reference line selected if a modification has been done on it (file replacement in the definition, filename modification, etc.).

- **Copy selected path(s)**

  Copy select path(s) action allows you to multi-select and copy several lines of the list and paste them in a text editor or else of your choosing.

  This is particularly useful if you want to export the list of all project dependencies, to create a report, to make impact management, a script processing, etc.

- **Replace file path with**

  Replace file path with action replaces the file path (meaning the string of characters "C:\...\...`file.ext") in the reference of the object. The Speos object points to another file.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

No file has been replaced in the FileSystem.

- **Replace folder path with**

  Replace folder path with action replaces the folder path (meaning the string of characters "C:\...\foldername") in the reference of the object.

  > **Note:** When performing this action, make sure that the file referenced in the object (or a file with the same name and extension) is present in the folder that will replace the previous one. Otherwise an error will be raised.

- **Clear**

  Clear action erases the field filled by the input reference in the Speos feature. The Speos feature has no longer input reference for this field.

  No file has been erased on the FileSystem.

  > **Note:** If the **Copy input files** option is activated in the Speos options, the file of the new file path will be copied in the Speos input files folder.

4. Click **Close** when you are done with the Speos Files Analysis tool.

3.11. Presets

Presets allow to create predefined sets of parameters and apply them to new or existing Speos objects.

3.11.1. Presets Overview

A Preset is an XML file *.preset that defines the configuration of a given Speos object type (or only for a subset of its attributes).

As you may rely on the same objects and simulation settings through all of the Speos users of your company, Presets allow you to speed up the creation of Speos objects, and maintain the coherence and continuity along your different projects.

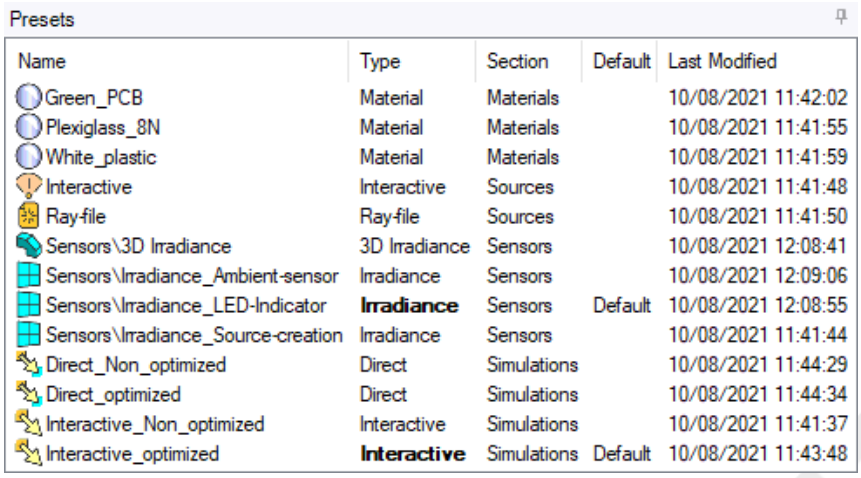
With a Preset you can:

- create a new object in the state of the Preset
- apply a Preset on an existing object of the same type to modify its values
- define Preset as default to create all new objects of the same type with the values of the default Preset.

Presets can be shared with Speos users so that everyone can work on a common basis.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Preset Management



The Preset panel provides you with a list of all created Presets contained in the default or custom Preset repository. From this panel you can manage your Presets.

Presets can be organized in sub-folder in the Preset repository. The sub-folder hierarchy appears as a prefix in the **Name** column of the panel.

The default Presets repository is `C:\ProgramData\ANSYS\v2XX\Optical Products`

> **Note:** The Presets repository (default or custom) must only contain preset files.

Default Preset

Presets can be defined as default which means that the Preset defined as default will apply its values to every new object of the underlying type.

Only one Preset can be set as default for a given object type. Thus, when setting a Preset as default, the previous Default Preset is unset.

The new created object:

- has the same values as defined in the Default Preset.
- is named upon the Preset file name instead of the standard object type name, and the object name is incremented with an index suffix.

  Example: if the Preset file is named "My_Custom_Source.preset", the new object will be named "My_Custom_Source.1".

> **Note:** When the Preset that is renamed is set as default, the internal link between the object type and the associated Preset file path is updated as well.

3.11.2. Customizing the Preset Repository

The following procedure helps you customize the preset repository.

DRAFT

To customize the preset repository:

The Presets repository must be set at least in Read mode, and must only contain preset files.

1. Click **File** > **Speos options**.
2. Click the **Light Simulation** section.
3. In the **General** tab, in the **Presets** section, check **Use custom folder path**.

> **Note:** Make sure the custom path exists.



4. Click ... to browse and select the custom folder path.
5. Click **OK** to validate.

> **Note:** All previously created preset are not moved automatically. You have to transfer them manually in the custom presets repository.

All new presets are now created in the custom presets repository.

3.11.3. Creating a Preset

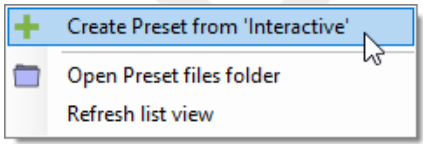
The following procedure helps you create a Preset from an existing Speos object.

To create a preset:

1. From the **Light Simulation** tab, click **Presets**  .
   The Presets panel appears.
2. In the **Speos Simulation** tree, select the object to create the preset from.



3. In the **Presets** panel, right-click anywhere and click **Create Preset from 'Select Speos Object'**.



> **Tip:** You can drag and drop the Speos object directly in the Presets panel to create the preset.

The new created preset inherits the values of the selected Speos object.

Now you can set this preset as default or apply it to a Speos object in the Simulation tree.

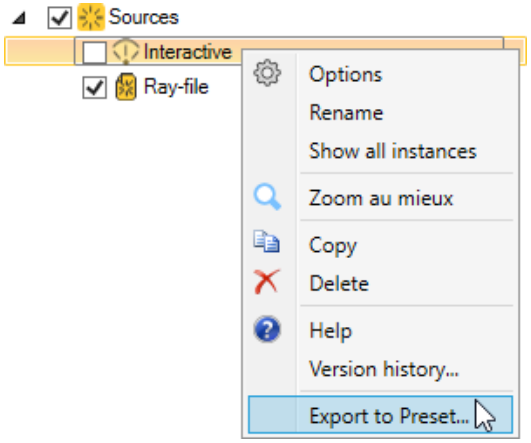
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

3.11.4. Exporting a Speos Object to a Preset

The following procedure helps you create a Preset from an existing Speos object by exporting its values.

To export a Speos object to a preset:

1. In the **Speos Simulation** tree, select the object to export the preset from.
2. Right-click the object and select **Export to Preset**.



The Explorer window opens.

3. In the Explorer window, name the *.preset file.
4. Click **Save**.

The new preset is created and appears in the Presets panel. It inherits the values of the selected Speos object.

Now you can set this preset as default or apply it to a Speos object in the Simulation tree.

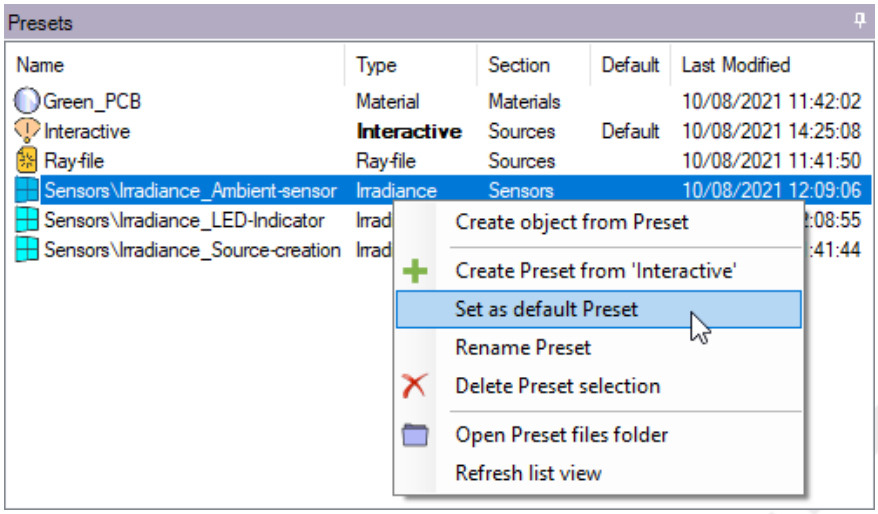
3.11.5. Setting a Preset as Default

The following procedure helps you define a preset as default in order to apply its values to every new object of the underlying type.

To set a preset as default:

1. From the **Light Simulation** tab, click **Presets**  .
   The Presets panel appears.
2. Right-click the preset to set as default and click **Set as default Preset**.

DRAFT



Note: Only one Preset can be set as default for a given object type. Thus, when setting a Preset as default, the previous Default Preset is unset.

Tip: You can change the default preset directly from the Quick Preset menu.

The Preset is set as default and appears in bold with the Default mention in the Preset panel.

Now you can create a Speos object from the default preset.

3.11.6. Creating a Speos Object from a Default Preset

The following procedure helps you create a Speos object from a default preset.

To create an object from a default preset:

A preset of the same type as the object must have been set as default.

From the **Light Simulation** tab, click the Speos feature.

Tip: You can change the default preset directly from the Quick Preset menu.

The Speos object is created in the Simulation tree and inherits the values of the default preset.

3.11.7. Applying a Preset to an Existing Speos Object

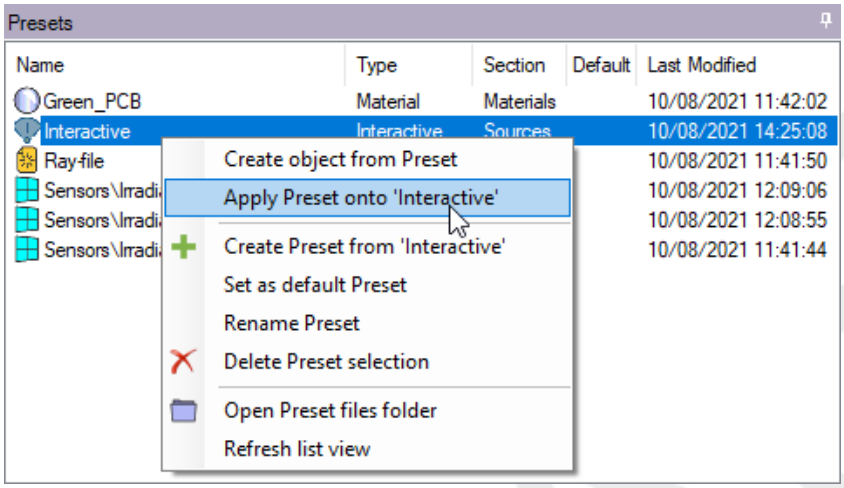
The following procedure helps you apply a preset to an existing Speos object.

To apply a preset to an existing Speos object:

A preset of the same type as the object must have been created.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

1. From the **Light Simulation** tab, click **Presets**  .
   The Presets panel appears.
2. In the **Speos Simulation** tree, select the object on which to apply the preset.
   

3. In the **Presets** panel, right-click the preset you want to apply on the Speos object and click **Apply Preset onto 'Select Speos Object'**.
   

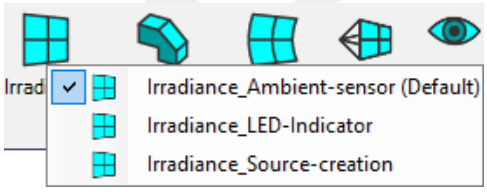
> **Tip:** You can drag and drop the preset directly on the Speos object to apply the preset.

The Speos object inherits the values of the applied preset.

3.11.8. Accessing the Quick Preset Menu

The following procedure helps you access created presets directly from the Light Simulation tab.

To access the Quick Preset menu:

1. In the **Light Simulation** tab, press **[Shift + Left-Click]** on the Speos feature.
   The Quick Preset menu opens for the selected feature.
   

2. From the **Quick Preset** menu:
   - Click a preset (default or not) to create an object.
   - Check a preset to set it as default.

4: Imports

Speos allows you to import and update all kinds of geometries, from CAD geometries or projects to mesh (*.stl) files.

4.1. CAD File Import (CATIA, NX, CREO, Solidworks)

The following section helps you set how to import different CAD files and provides you with information to manage potential import limitations.

4.1.1. CAD File Import Overview

The following page provides you with the import types to import external CAD files according to your needs.

The external CAD files that can be imported into Speos are CATIA V5, NX, CREO, Solidworks files.

SpaceClaim Reader

> **Note:** The SpaceClaim Reader is the importer activated by default. In case it is deactivated, refer to Configure the SpaceClaim Reader to activate it.

The **SpaceClaim Reader** allows you to import CAD files (CATIA V5, NX, CREO, Solidworks) using the SpaceClaim importer.

In case of CATIA files, the files are imported in lightweight. The other CAD files are imported in heavyweight.

For more information on the SpaceClaim Reader, refer to the **SpaceClaim User Guide**.

Workbench Plug-in/Reader

The **Workbench Plug-in/Reader** corresponds to the importer when the SpaceClaim Reader is deactivated.

The **Workbench Plug-in/Reader** is mandatory to use the Block Recording tool in Speos context.

If you want to work with the Block Recording, refer to the page Configuring the Environment for the Block Recording Tool.

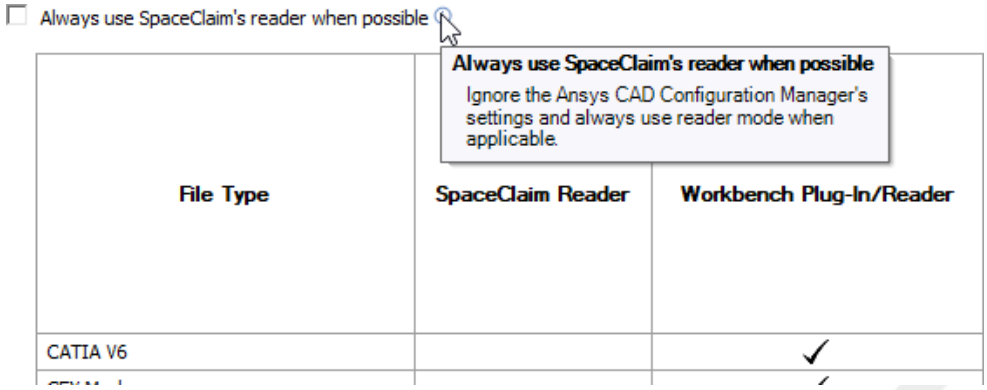
4.1.2. Configuring the SpaceClaim Reader

Activating the SpaceClaim Reader option allows you to import external CAD files into Speos.

To activate the SpaceClaim Reader:

1. In Speos, click **File** > **Speos Options**
2. Expand the **File Options** tab, and select **Workbench**.
3. Activate the option **Always use SpaceClaim's reader when possible** if not already done.

DRAFT



The option **Always use SpaceClaim's reader when possible** is activated by default.

CAD files are ready to be imported.

4.1.3. CATIA File Import Limitations

The following page lists the limitations related to the import of CATIA files into Speos.

CATIA Object Names Conservation

After importing a CATIA file into Speos, the object names (default or custom) from CATIA are kept. However, the geometrical set names from CATIA are lost, which means geometries are renamed.

Non-Supported CATIA Disjoint bodies

SpaceClaim does not support disjoint bodies.

If you switch from Lightweight to Heavyweight a geometry imported from CATIA and originally created in CATIA with disjoint bodies, SpaceClaim creates as many geometries as existing CATIA disjoint bodies. As a consequence, Speos links are only kept on one geometry, and other created geometries from each disjoint body do not have Speos links.

> **Tip:** Instead of importing a CATIA geometry made of disjoint bodies, split the disjoint bodies in different geometries in CATIA, and import each geometry individually.

Non-Supported CATIA non-manifold geometries

Non-manifold geometries created in CATIA are not compatible in Speos after importing the CATIA Project.

Import of Hidden Geometries from CATIA

Importing a *.CATPart file containing hidden geometries in Speos does not have any impact on the import time.

Publication Names VS Geometric Sets in *.CATPart

Known Issue: Speos does not support *.CATpart file import containing "Publication names".

Workaround: Make sure to use "Geometric sets" instead of "Publication names" in your *.CATpart file for the import to interpret it as "Named selections".

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

4.2. SCDOC File Import

The following page provides you with information to help you manage the import of a *.scdoc file into Speos.

Conversion Format

When importing a *.scdoc file (ACIS save format) into Speos, the *.scdoc file will be converted in *.scdocx file (Parasolid save format) upon next save.

> **Note:** As the extension changes from ACIS to Parasolid, the next save of the file will prompt you to save as.

Conversion Time

When converting a *.scdoc file to a *.scdocx file, the conversion can take time. Once converted, the file will open normally, without taking time.

Axes Orientation

The orientation of axes (either support axes or edges used as axes) of the geometries or Speos features may have changed during the project conversion from a *.scdoc file to a *.scdocx file.

We recommend you to check your axes prior to running any simulations.

Optical Part Design

After converting a Light Guide from a *.scdoc file to a *.scdocx file, Light Guide faces used in the Geometries selection of another Speos feature may have changed.

We recommend you to check the geometries selection of your Speos features using light guide faces.

Workbench Project

As a project cannot mix *.scdoc data and *.scdocx data, a Workbench project created with *.scdoc data should be recreated after the *.scdoc file conversion to *.scdocx file.

1. Import the *.scdoc file.
2. Save the converted file.

   The file is saved as into the *.scdocx.
3. From the saved *.scdocx file, recreate the Workbench project.

4.3. IGES and STEP Files Import

The following page provides you with information to help you manage the import of IGES and STEP files into Speos.

IGES files and STEP files can be imported into Speos. They are always imported in lightweight, whatever the importer activated.

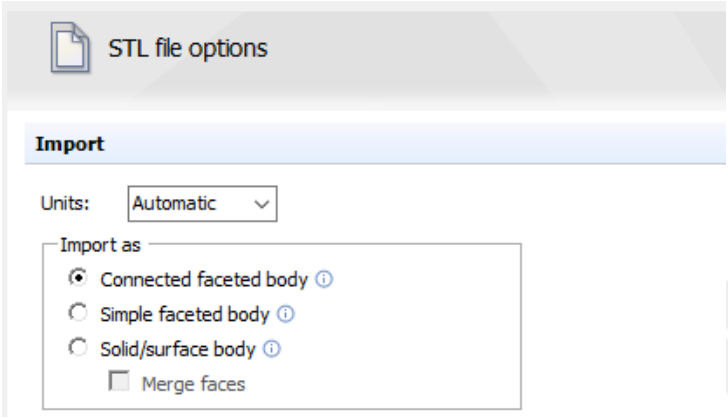
DRAFT

4.4. STL Files Import

This page describes the different options available to import mesh (*.stl) files in Speos and the specificities of each of them. Once imported, mesh (*.stl) files can be selected for Speos simulation as any other CAD geometry.

Three import options are available to import STL files in Speos. However, we recommend choosing between only two of these options.

To access these options, click File > Speos options > File Options > STL.



- **Connected faceted body** is the default and recommended option to use. It allows you to import the mesh as a closed faceted body.

> **Note:** With this option, you cannot visualize the Speos meshing as the meshing used for simulation is directly inherited from the geometry itself.

- **Solid/surface body** can be used if the first import option failed to work. This option allows you to import the mesh as a surface body. The surface body then must be converted into a closed faceted surface by using the **Convert** option.



Solid/surface body



*Conversion tool (**Facets** tab)*



Closed faceted surface

Related information

Geometry Update Tool on page 53

This page introduces the **Geometry Update** tool which allows you to import external CAD files from a CAD software into Speos and update geometries into the CAD software while maintaining their relationship to Speos features.

4.5. CATIA V6 Import

The following page provides you with information to help you manage the import of CATIA V6 file into Speos.

When importing a project from CATIA V6, only surfaces and bodies are imported. Speos does not support points, lines and curves import.

> **CAUTION:** CATIA V6 project can be imported, but are not compatible with the **Geometry Update** - **Update External CAD Part** option.
>
> Bypass: Export the CATIA V6 file as a CATIA V5 file. Then, you can update the external CAD part using the **Update External CAD Part** option.

4.6. Geometry Update Tool

This page introduces the **Geometry Update** tool which allows you to import external CAD files from a CAD software into Speos and update geometries into the CAD software while maintaining their relationship to Speos features.

Related information

STL Files Import on page 52
This page describes the different options available to import mesh (*.stl) files in Speos and the specificities of each of them. Once imported, mesh (*.stl) files can be selected for Speos simulation as any other CAD geometry.

4.6.1. Geometry Update Overview

The **Geometry Update** tool allows you to update your design while keeping its link to Speos dependencies.

Geometrical models used for optical simulation often need to be adjusted throughout the design process, to meet manufacturing requirements for example.

The **Geometry Update** tool allows you to update your design while keeping its link to Speos dependencies. The modified design or parts of it can be re-imported into Speos as many times as you need without having to re-apply Speos features to it.

This tool offers a design and testing flexibility as it takes better consideration of the standard design process.

General Workflow

> **Important:** Make sure that the SpaceClaim Reader is activated.

1. The Design team creates geometries in a CAD software.
2. The Speos team uses the Geometry Update tool to import the geometries from the CAD software to Speos.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Files that can be imported: CATIA V5 files (*.CATPart, *.CATProduct), CREO Parametric files (*.prt, *.xpr, *.asm, *.xas), NX files (*.prt), SolidWorks files (*.sldprt, *.sldasm).

3. The Speos team defines the optical properties thanks to the Speos Light Simulation features.
4. The Design team modifies the geometry in the CAD software.
5. The Speos team uses Geometry Update tool to update the project to work on the latest data.

   When geometries are updated, the link is maintained between Speos features and the newly imported geometries. Materials are still applied on the geometries, sources are adjusted to the faces of the new geometries etc.

Geometry Update Tool

The **Geometry Update** tool is accessible from the **Assembly** tab.

- **Import External CAD Part**  allows you to import a new external part. The part is by default imported in the active part of the document.
- **Select all Imported Parts**  allows you to select specific parts you want to replace with another model.
- **Update External CAD Part**  allows you to select the newer version of a part previously imported in Speos.

Limitations

A named selection created out of geometries imported with the **Geometries Update** tool may not detect geometry modifications applied in the external CAD part after updating this external CAD part. Therefore, the modified geometry of the named selection is not updated in SpaceClaim.

4.6.2. Importing an External CAD Part

The following procedure helps you import an external CAD part from a CAD software into Speos.

To import an external CAD part:

1. In Speos, in the **Assembly** tab, click **Geometry Update** 

.
2. In the 3D view, select **Import External CAD Part**  .

   You can import CATIA V5 files (*.CATPart, *.CATProduct), CREO Parametric files (*.prt, *.xpr, *.asm, *.xas), NX files (*.prt), SolidWorks files (*.sldprt, *.sldasm).
3. Choose the CAD part and click **Open**.
   - If the file you want to import is located under `.\Speos Input files\`, then the **Last known file source** will be stored with a relative path.
   - Only for *.CATPart file: if the file you want to import is not located under `.\Speos Input files\`, then the file is copied in `.\Speos Input files\` only if the option **Automatically copy selected files under document's folder** is activated in the Speos preferences.
   - If the file you want to import is an assembly (*.CATProduct, *.asm, *.prt, *.sldasm) and not located in `.\Speos Input files\`, then the file is not copied in `.\Speos Input files\`, and the file path remains absolute.

   To check the **Last known file source**, right-click the imported component and select **Properties**.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

The CAD part is imported into Speos.

Optical Properties and Speos features can now be applied and created to define the optical system.

4.6.3. Updating an External CAD Part

The following procedure helps you update an external CAD part while keeping its link to Speos dependencies.

> **Note:** You cannot use the Geometry Update tool if you have imported your external CAD part use the Open file command.

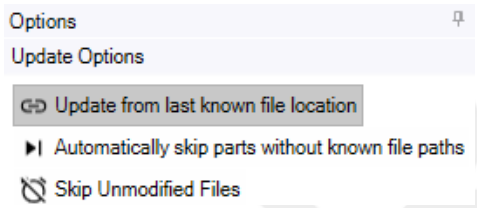
To update an external CAD part:

1. In Speos, in the **Assembly** tab, click **Geometry Update** 

.

> **Tip:** The import/replace option is also accessible with a right-click the part you want to update from the structure tree.

2. In the **Options** tab, in the **Update Options** section, you can define the behavior of the updates:



- **Update from last known file location** directly takes the last known file path used for the import/update.
- **Automatically skip parts without known file paths** skips the update of the parts whose file is not found.
- **Skip Unmodified Files** skips the files for which the reference file's modified date has not change since the last import.

> **Important: Skip Unmodified Files** is activated by default. Unmodified parts are skipped only if the CAD project and Structure trees are identical.

3. In the 3D view or in the Structure tree, select the parts you want to update or click  to **Select all Imported Parts**.

4. In the 3D view, select **Update External CAD Part**  .

Geometries are updated and the link is maintained between Speos features and the newly imported geometries. Materials are still applied on the geometries, sources are adjusted to the faces of the new geometries etc.

DRAFT

4.7. Incorrect Imports - Solutions

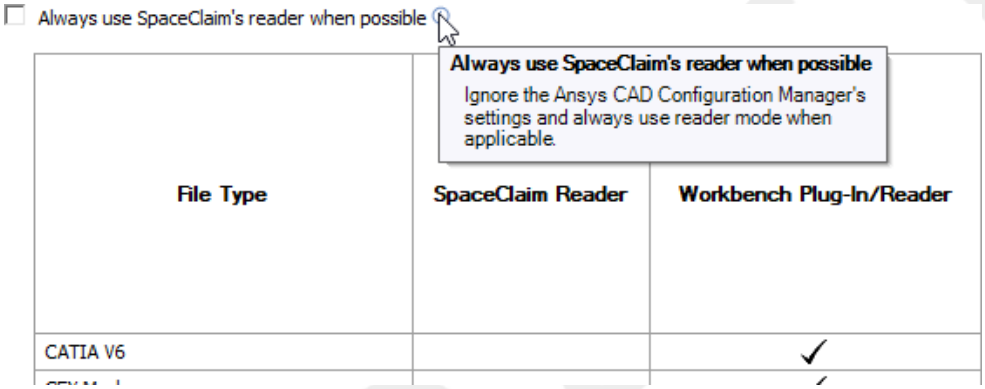
The following page provides you with possible solutions in case you encounter an incorrect import of your CAD file into Speos.

Solution 1: Correcting errors in CAD

In case you imported a CAD file using **Add File** option and find some errors, try to repair the errors in the CAD software, then re-import the CAD file again.

Solution 2: Deactivating Always use SpaceClaim's reader when possible

1. In Speos, click **File** > **Speos Options**
2. Expand the **File Options** tab, and select **Workbench**.
3. Deactivate the option **Always use SpaceClaim's reader when possible**.



4. Import the file again.

Solution 3:Export/Import Optical Part Design (OPD) Geometry in *.CATPart

Exporting an OPD Geometry in *.CATPart file then re-importing the *.CATPart file may deteriorate the original geometry. You may face then with a geometry that could differ from the original one.

To avoid such issues, export your OPD geometry in *.STEP file.

DRAFT

5: Materials

Materials are the entry point of optical properties and texture creation/application. The following section describes both processes.

5.1. Optical Properties

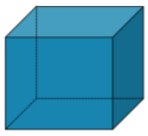
Optical Properties allow to determine how light rays behave and interact with the materials applied on geometries.

5.1.1. Optical Properties Overview

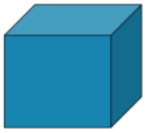
Optical Properties define how light rays interact with geometries in the CAD. Objects have volume and surface optical properties.

Types of Optical Properties

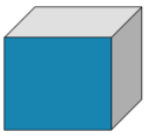
- **Volume Optical Properties (VOP)** define the behavior of light rays when they are propagated in a body. You can set VOP from the interface or build more complex materials with the **User Material Editor** .
- **Surface Optical Properties (SOP)** define the behavior of light rays when they hit the surface of a body. You can set SOP from the interface or build more complex materials with Surface Optical Property Editors like the **Simple Scattering Surface Editor** or the **Advanced Scattering Surface Editor** .
- **Face Optical Properties (FOP)** are specific Surface Optical Properties that define the behavior of light rays when they hit certain face(s) of a body.



VOP



SOP



FOP

To define Optical Properties, you can also:

- Use BSDF or BRDF files obtained from OMS2 and OMS4 measurements (only available for SOP or FOP).
- Download materials from the Ansys Optical Library.

Priority rule

Optical Properties are applied on geometries through a multilayer system. The last layer always prevails during simulation.

A face optical property always overwrites the surface optical property of an object.

DRAFT

Related tasks

Creating Optical Properties on page 68
Creating Volume Optical Properties (VOP) and Surface Optical Properties (SOP) on solids allows you to determine the light rays' behavior when they hit or are propagated in the geometry.

Creating Face Optical Properties on page 70
Creating Face Optical Properties (FOP) allows you to isolate certain faces of a geometry to assign specific optical properties to these faces.

5.1.2. Non-Homogeneous Material

Non-homogeneous material is a material to apply to a body for which the refractive index and the absorption can vary in space.

5.1.2.1. Understanding Non-Homogeneous Material

A non-homogeneous material is the association between a graded material file and a solid body using a coordinate system.

The goal is to define the index of refraction and/or absorption for specific locations in the material volume.

Graded Material

The Graded Material file describes the spectral variations of refractive index and/or absorption regarding the position in space.

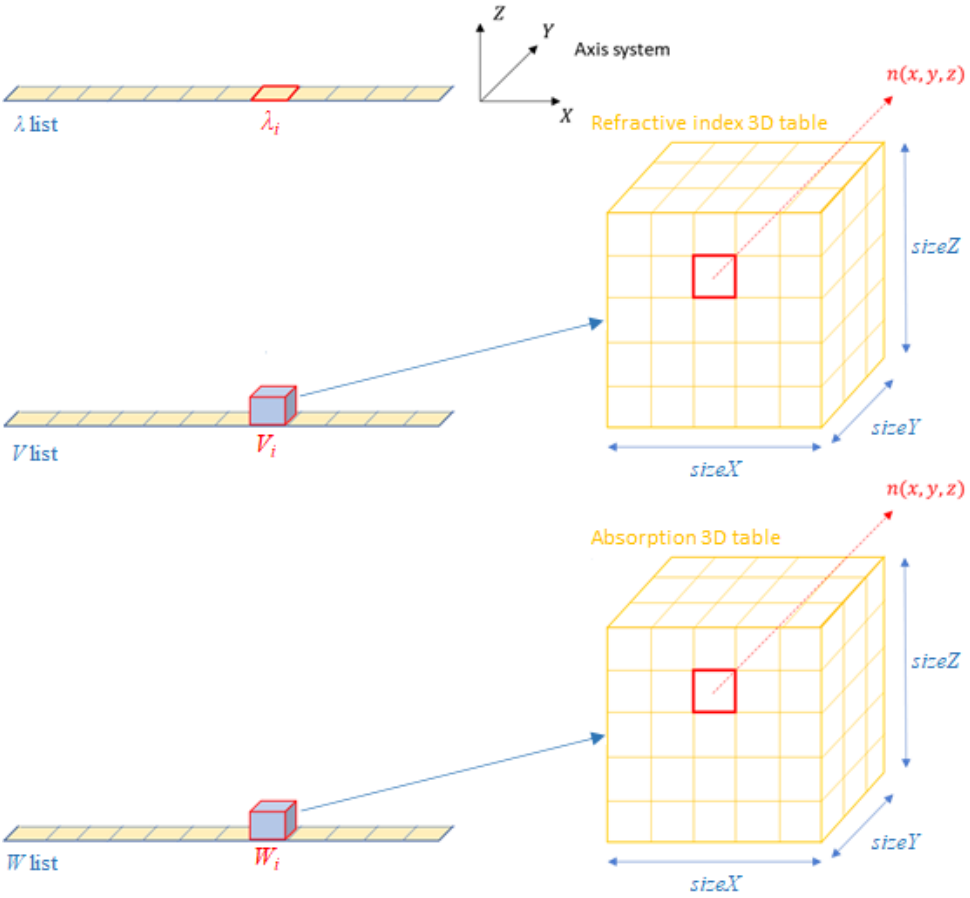
> **Note:** The data for the refractive index variation and the absorption variation generally come from Fluent.

As each wavelength propagates differently in a medium according to the refractive index and/or the absorption, you need for each wavelength:

- a 3D table representing the refractive index variation in space (V list)
- a 3D table representing the absorption variation in space (W list)

Each n in a table represents a 3D zone with its own refractive index. So, a wavelength propagates according to the refractive index of the 3D zone n. The same goes for absorption.

DRAFT

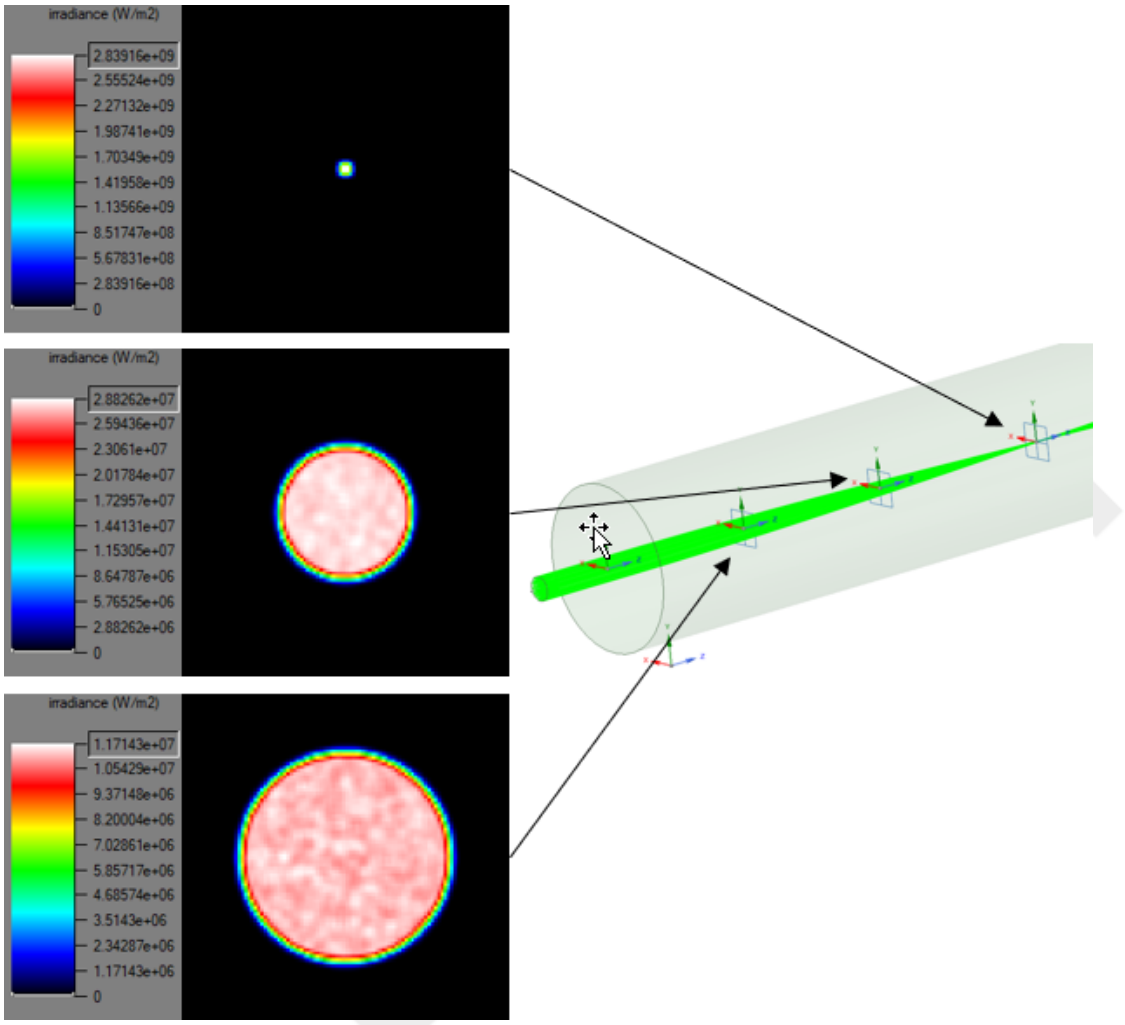


Non-Homogeneous Material Result Example

Figure 1. Interactive simulation showing the propagation inside a graded material



Figure 2. Direct simulation showing the propagation result at different location in the graded material

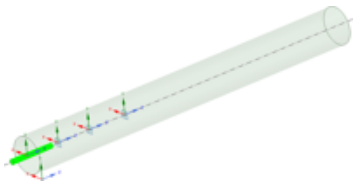


Maximum Number of Surface Interactions

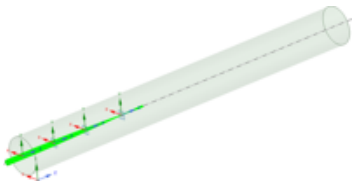
Maximum number of surface interactions has a direct impact on the length of the propagation.

The minimum length of one 3D zone is considered for the propagation: 5% of this length is used to calculate the propagation step. It means that the status of the ray is checked every 5% of the minimum length of one 3D zone and its direction is changed according to the refractive index variation.

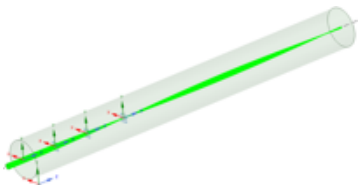
When the 3D table representing the variation has a high resolution, the **Maximum number of surface interactions** option must be high (otherwise rays are stopped) and it has a direct impact on the simulation time.



Maximum number of surface interactions = 100



Maximum number of surface interactions = 500



Maximum number of surface interactions = 5000

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

5.1.2.2. Graded Material File

This page describes the basic steps to create a graded material file through the example of a gradient refractive index material, and provides the python script to dot it.

Graded material file relies on filling a data model describing the spectral variation of refractive index and/or absorption.

Library Script Files

GradedIndexMaterial

Speos provides you with a python script file speos_GradedIndexMaterial.py library which includes the GradedIndexMaterial generic class to:

- access the content of an existing material (openFile, parseFile)
- create the data model and save a new file (createDataModel, SaveFile)
- GradientIndexMaterial class is an example that fills the data model with a gradient index refractive index variation.

Download the script file .speos_GradedIndexMaterial.zip

GradientRefractiveIndexTable

Speos provides you with a python script file speos_GradientRefractiveIndexTable.py library which includes the GradientRefractiveIndexTable class to:

- generate the table with the gradient refractive index variation (GetGradiantRefractiveIndexTable)
- get the refractive indexes with their position (GetRefractiveIndex)

Download the script file .speos_GradientRefractiveIndexTable.zip

General Workflow

To create a graded material file:

1. You need the refractive index variation and/or the absorption variation that directly comes from Fluent. To be processed, the variation must be set as a list.
2. Once you have your list of refractive index and/or absorption, create the python script to generate the graded material thanks to the GradedIndexMaterial class and the GradientIndexMaterial class if you want a gradient material.

> **Note:** When defining the sampling of refractive index or absorption, the value must be strictly greater than 1.

Graded Material File Example

Speos provides you with a python script sample file GradientRefractiveIndexMaterial_example.py. In this example, the graded material is created with the following characteristics:

- 3 wavelengths:468, 532 and 643nm
- Refractive index and absorption do not vary spectrally
- Refractive index follows a gradient refractive index distribution
- Absorption is null

This example covers a specific type of graded material file: gradient material.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Gradient material is a graded material whose variation is constant. Thus, the script uses the GradientIndexMaterial class and the GradientRefractiveIndexTable class to generate the graded material file.

Download the example file .GradientRefractiveIndexMaterial_example.zip

Workflow Example

1. Define the dimensions of the material and the sampling.

```
# Dimensions
xSize = 1 # size along X direction in mm
ySize = 1 # size along Y direction in mm
zSize = 10 # size along Z direction in mm
xSampling = int(xSize / 0.1)
ySampling = int(ySize / 0.1)
zSampling = int(zSize / 0.1)
```

2. Create the absorption table.
   - A list is used in python and corresponds to a "first level" list describing the absorption variation along Z.
   - Each sample along Z contains a "second level" list describing the absorption variation along Y.
   - Each sample along Y contains a "third level" list describing the absorption variation along X.

   In the example, since the absorption is null, the minimum number of samples (2) to define the table is used.

```
absorptionTable = [[[0.0, 0.0], [0.0, 0.0]], [[0.0, 0.0], [0.0, 0.0]]]
```

3. For a gradient material, to create the refractive index table, use the dedicated GradientRefractiveIndexTable class from speos_GradientRefractiveIndexTable.py.

   Note: The GradientRefractiveIndexTable class permits you to create a TableMulti_double_3 table.

   - As you can see in the GetGradientRefractiveIndexTable function, the table must be sized:

```
dimensions = IllumineCore.Extent_uint_3()
dimensions.Set(0, nb_x)  # sampling value along X direction
dimensions.Set(1, nb_y)  # sampling value along Y direction
dimensions.Set(2, nb_z)  # sampling value along Z direction

tableMulti3.Allocate(dimensions)  # allocate memory
```

   - Then, you loop to populate the table:
     - along Z
     - for each Z, you loop along Y
     - for each Y, you loop along X

```
# create position extent
pos.Set(0, ix)
pos.Set(1, iy)
pos.Set(2, iz)
tableMulti3.Set(pos, self.GetRefractiveIndex(xPosition, yPosition))
```

4. Since the refractive index and absorption do not vary with the wavelengths, duplicate the different tables:

```
spectrumTable = [468.0, 532.0, 643.0]
spectralRefractiveIndexTable = [refractiveIndexTable, refractiveIndexTable,
refractiveIndexTable]
spectralAbsorptionTable = [absorptionTable[:], absorptionTable[:],
absorptionTable[:]]
```

5. Use GradedIndexMaterial class to create the data model and save the new file:

```
materialTest = speos_GradedIndexMaterial.GradedIndexMaterial()
materialTest.createDataModel(refractiveIndexDimensions, absorptionDimensions,
 spectrumTable, spectralRefractiveIndexTable, spectralAbsorptionTable)
materialTest.saveFile(filepath)
```

5.1.2.3. List Of Methods

This page describes the methods that should be used to access the data stored in the *.grademateriall file used to define a non-homogeneous material.

As the *.gradedmaterial is a binary file, you must use specific methods to access its content and retrieve its data. Make sure to use the **3.9 version** of IronPython or Python language to write your scripts.

Basic Methods

Name	Description	Syntax
OpenFile	Open the file and fill the data model	GradedIndexMaterialFile.OpenFile(Optis::IO::Path) • Optis::IO::Path : path and filename Should end by .gradedmaterial
SaveFile	Save the file from the data model	object.SaveFile(BSTR bstrFileName) as Boolean • Object: Graded Material File • bstrFileName: path and filename Should end by .gradedmaterial
GetDataModel	Get the graded index material file data model	Optis::DM::DGradedIndexMaterialFile=GetDataModel()
SetDataModel	Set the graded index material file data model	GradedIndexMaterialFile.SetDataModel(Optis::DM::DGradedIndexMaterialFile)

Refractive Index

Name	Description	Syntax
GetRefIndSizeX	Get the dimension in mm along X direction for refractive index data	double GradedMaterialFile.GetRefIndSizeX()

DRAFT

Name	Description	Syntax
GetRefIndSizeY	Get the dimension in mm along Y direction for refractive index data	double GradedMaterialFile.GetRefIndSizeY()
GetRefIndSizeZ	Get the dimension in mm along Z direction for refractive index data	double GradedMaterialFile.GetRefIndSizeZ()
GetRefIndSamplingX	Get the sampling value along X direction for refractive index data	uint GradedMaterialFile.GetRefIndSamplingX()
GetRefIndSamplingY	Get the sampling value along Y direction for refractive index data	uint GradedMaterialFile.GetRefIndSamplingY()
GetRefIndSamplingZ	Get the sampling value along Z direction for refractive index data	uint GradedMaterialFile.GetRefIndSamplingZ()
GetRefIndNbSamples	Return the number of samples in the file for refractive index data	uint GradedMaterialFile.GetRefIndNbSamples()
GetRefIndTable	Get the refractive index table of a specific sample for refractive index data	Optis::Table<double, 3>=GetRefIndTable(uiSample) • Optis::Table<double, 3>: refractive index table • uiSample: index of the sample
GetRefInd	Get the refractive index value for a given sample	double GradedMaterialFile.GetRefInd(uiSample, uiX, uiY, uiZ) • uiSample: index of the sample • uiX uiY uiZ: index table along X, Y, Z directions
GetRefIndWavelength	Get the wavelength of a specific sample for refractive index data	double GradedMaterialFile.GetRefIndWavelength(uiSample) • uiSample: index of the sample
GetRefIndWavelengthsList	Get the list of wavelengths for refractive index data	Optis::Vector<double>=GradedMaterialFile.GetRefIndWavelengthList() • Optis::Vector<double>: list of wavelengths

DRAFT

Absorption

Name	Description	Syntax
GetAbsorptionSizeX	Get the dimension in mm along X direction for absorption data	double GradedMaterialFile.GetAbsorptionSizeX()
GetAbsorptionSizeY	Get the dimension in mm along Y direction for absorption data	double GradedMaterialFile.GetAbsorptionSizeY()
GetAbsorptionSizeZ	Get the dimension in mm along Z direction for absorption data	double GradedMaterialFile.GetAbsorptionSizeZ()
GetAbsorptionSamplingX	Get the sampling value along X direction for absorption data	uint GradedMaterialFile.GetAbsorptionSamplingX()
GetAbsorptionSamplingY	Get the sampling value along Y direction for absorption data	uint GradedMaterialFile.GetAbsorptionSamplingY()
GetAbsorptionSamplingZ	Get the sampling value along Z direction for absorption data	uint GradedMaterialFile.GetAbsorptionSamplingZ()
GetAbsorptionNbSamples	Return the number of samples in the file for absorption data	uint GradedMaterialFile.GetAbsorptionNbSamples()
GetAbsorptionTable	Get the absorption table of a specific sample for absorption data	Optis::Table<double, 3>=GetAbsorptionTable(uiSample) • Optis::Table<double, 3>: refractive index table • uiSample: index of the sample
GetAbsorption	Get the absorption value for a given sample	double GradedMaterialFile.GetAbsorption(uiSample, uiX, uiY, uiZ) • uiSample: index of the sample • uiX uiY uiZ: index table along X, Y, Z directions
GetAbsorptionWavelength	Get the wavelength of a specific sample for absorption data	double GradedMaterialFile.GetAbsorptionWavelength(uiSample) • uiSample: index of the sample
GetAbsorptionWavelengthsList	Get the list of wavelengths for absorption data	Optis::Vector<double>=GradedMaterialFile.GetAbsorptionWavelengthList() • Optis::Vector<double>: list of wavelengths

DRAFT

5.1.3. Surface State Plugin

The Surface State Plugin feature allows you to create a custom made *.sop plug-in and use it as a surface optical properties in Speos.

5.1.3.1. Surface State Plugin Examples

Speos provides you with two complete examples of Surface State Plugin to help you understand how to create and test your own plugin.

Download the file.Surface_State_Plugin_Examples.zip

> **Important:** The version v2 is in BETA mode for the current release (filenames with v2 suffix).

The provided sample codes have been tested and validated on:

- Windows 10 with Visual Studio 2019
- CentOS 7 with Gcc 8.2.

> **Note:** We tried to be compiler agnostic so it should work as well with other compilers.

C++ Example

The first example in the `example-plugin` folder is developed in C++ and represents a surface with lambertian reflexion and transmission.

In the **example-plugin.cpp** file, you can find useful comments to help you create your plugin and to understand how Speos simulations use it.

Python Example

The second example in the `python-plugin` folder is developed in C/C++/Python. It is a proof-of-concept to explain how to implement a bridge between C and Python to be able to develop the surface state plugin in Python.

> **Note:** Python multithreading is highly impacted by the GlobalInterpreterLock which can reduce drastically the scalability of the Speos simulations. But the great advantage of this Python-based plugin is that you do not need to rebuild the plugin when prototyping.

Test Application

Speos provides in the `test` folder an application that mimics the way Speos load the plugin and deliver statistics about the plugin.

5.1.3.2. Creating and Testing a Surface State Plugin in Windows

The following procedure helps you create, build and test your surface state plugin in Windows.

To create, build and test a Surface State Plugin:

Download and Install Visual Studio 2019.

1. Create your plugin with the help of the examples provided.

> **Note:** When creating the plugin, make sure to use the **speos-plugin.h**. This is the header corresponding to the plugin interface. It defines the structures and functions that must be exported by the surface state shared library, and that are necessary for the plugin and Speos to communicate.

2. Build the plugin and the test in Visual Studio:
   a) Right-click **File** > **Open** > **CMake**.
   b) Right-click **Project** > **Configure**.
   c) Right-click **Build** > **Build All**.
   d) Right-click **Build** > **Install**.
3. After building, test your plugin: right-click **Test** > **Run Ctests**.
   For more information, refer to CMake Projects in Visual Studio.

A *.sop file has been compiled. This file is a zip file containing a *.dll file. You can use this *.sop file as input in a simulation to be run with Windows only.

Once you created and tested your plugin, you can use it to create Optical Properties, Face Optical Properties, or Thermic Sources.

5.1.3.3. Creating and Testing a Surface State Plugin in Linux

The following procedure helps you create, build and test your surface state plugin in Linux.

To create, build and test a Surface State Plugin:

Install CMake, Ninja, Gcc.

1. Create your plugin with the help of the examples provided.

> **Note:** When creating the plugin, make sure to use the **speos-plugin.h**. This is the header corresponding to the plugin interface. It defines the structures and functions that must be exported by the surface state shared library, and that are necessary for the plugin and Speos to communicate.

2. Run `cmake -GNinja -DCMAKE_INSTALL_PREFIX=<installation_dir> <source_dir>`
3. Run `ninja`
4. Run `ninja install`
5. Run `ctest`

A *.sop file has been compiled. This file is a zip file containing a *.so file. You can use this *.sop as input in a simulation to be run with Linux only.

Once you created and tested your plugin, you can use it to create Optical Properties, Face Optical Properties, or Thermic Sources.

DRAFT

5.1.4. Optical Properties Creation

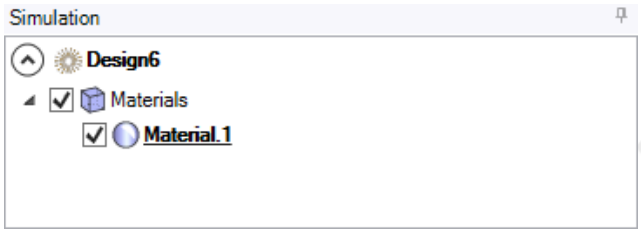
Optical Properties can be applied to solids or faces.

5.1.4.1. Creating Optical Properties

Creating Volume Optical Properties (VOP) and Surface Optical Properties (SOP) on solids allows you to determine the light rays' behavior when they hit or are propagated in the geometry.

To create Optical Properties:

1. From the **Light Simulation** tab, click **Material** .
   A new material is created in the Simulation panel.



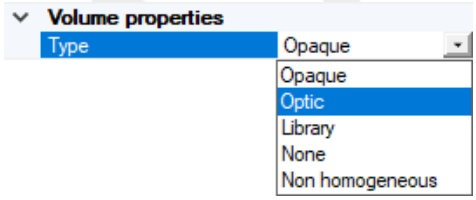
2. If needed, rename the material.
3. In the 3D view, click , select the geometries on which to apply optical properties and click to validate.

> **Tip:** Right-click a Material and click **Select associated geometry** to highlight in the 3D view the geometries on which the material is applied.

   The selection appears in **Geometries** as linked objects.
4. Leave the **Use texture** option to **False** to apply only volume and surface optical properties to the current geometry.

> **Note:** For more information about texture application, see Texture Mapping.

5. If you want to modify the current **Volume properties**:



   - Select **Opaque** for a non transparent material.
   - Select **Optic** for a transparent colorless material without bulk scattering.
     - Edit the Index and Absorption coefficient values.
     - Define Constringence as true or false and edit its values if needed.
   - Select **Library** and click **Browse** to select and load a *.material file.

     If you want to modify the *.material file, click **Open file** to open the **User Material Editor.**

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- Select **None** to define the material without volume optical properties.

  This allows you to address cases like filters.

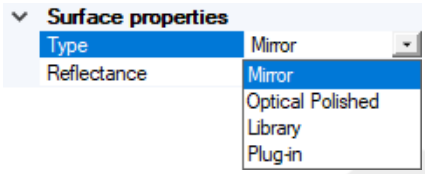
  > **Note:** Defining the Volume properties to **None** means that the body is considered as a surface and not a volume. Otherwise, if you do not set the Volume properties to None for a surface body, this may generate incorrect results.

- Select **Non-homogeneous** to create a material with non-homogeneous refractive index.
  a. In **File**, browse and select the *.gradedmaterial file that describes the spectral variations of refractive index and absorption with the respect to position in space.
  b. Define the coordinate system to align the (x, y, z) variation of the refractive index with the solid body.

  To make sure the graded material and the geometry match, a bounding box is displayed in the 3D view during the edition of the material.

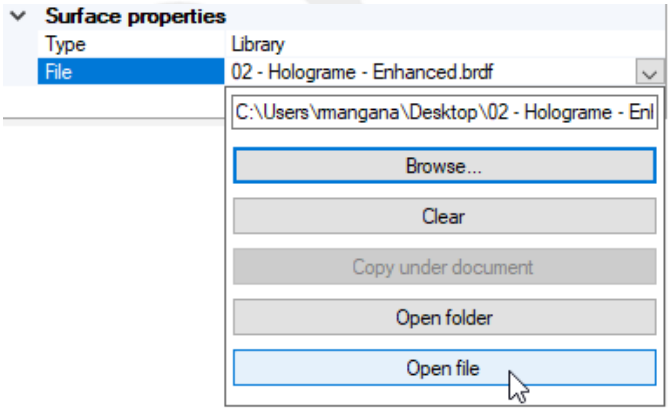
  > **Note:** For more information, refer to Non-Homogeneous Material.

6. If you want to modify the current **Surface properties**:



- Select **Mirror** for a perfect specular surface and adjust the **Reflectance** if needed.
- Select **Optical polished** for a transparent or perfectly polished material (glass, plastic).
- Select **Library** and click **Browse** to select and load a SOP file.

  If you want to modify the SOP file, click **Open file** to open the **Surface Optical Property Editor.**

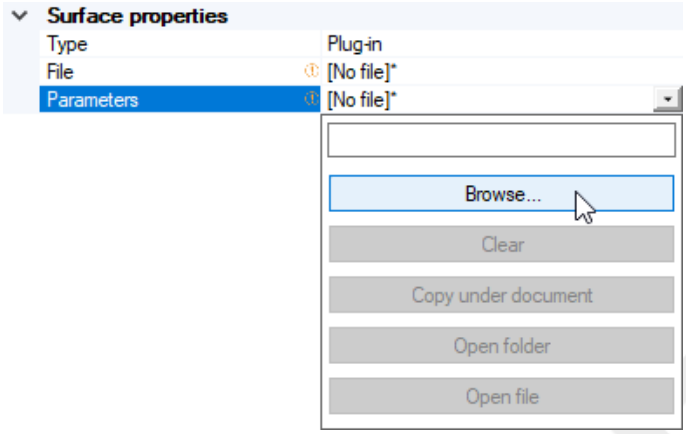


> **Tip:** To define a surface which is polarized, select a **.polarizer** file instead of a .coated file as coated surfaces are isotropic.

DRAFT

- Select **Plug-in**, and click **Browse** to select a custom made *.sop plug-in as **File** and the **Parameters** file for the plug-in.

  > **Note:** Make sure each plug-in created has a different GUID.



  For more information, refer to Surface State Plugin.

The optical properties are now created on the solid(s). You can edit these properties at any time.

Related concepts

Optical Properties Overview on page 57
Optical Properties define how light rays interact with geometries in the CAD. Objects have volume and surface optical properties.

Related tasks

Creating Face Optical Properties on page 70
Creating Face Optical Properties (FOP) allows you to isolate certain faces of a geometry to assign specific optical properties to these faces.

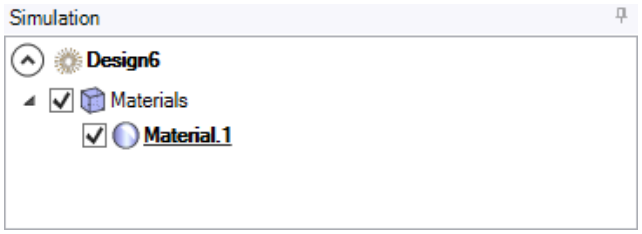
5.1.4.2. Creating Face Optical Properties

Creating Face Optical Properties (FOP) allows you to isolate certain faces of a geometry to assign specific optical properties to these faces.

To create Face Optical Properties:

1. From the **Light Simulation** tab, click **Material**  .

   A new material is created in the Simulation panel.



2. If needed, rename the material.
3. In **General**, from the **Type** drop-down list, select **Face properties**.
4. Leave the **Use texture** option to **False** to apply only surface optical properties to the selected face(s).

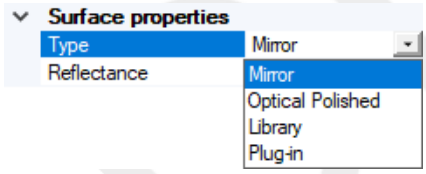
> **Note:** For more information about texture application, see Texture Mapping.

5. In the 3D view, click , select the faces on which to apply optical properties and click to validate.
   You can apply FOP to faceted geometries:
   - In case you want to apply the FOP on a full faceted geometry, select the faceted geometry in the Structure tree.
   - In case you want to apply the FOP on a region of a faceted geometry, create a Named Selection of facets, then select the Named Selection containing the facets.

> **Tip:** Right-click a Material and click **Select associated geometry** to highlight in the 3D view the geometries on which the material is applied.

> **Note:** The selection of faces from an imported *.obj file is not compatible with the Face Optical Properties.

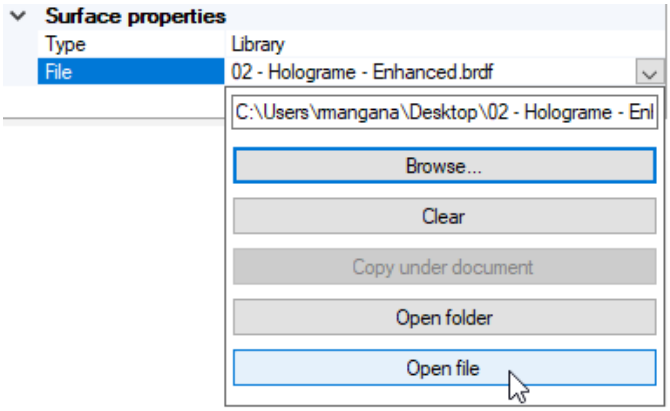
   The selection appears in **Geometries** as linkedobjects.
6. If you want to modify the current **Surface properties**:



   - Select **Mirror** for a perfect specular surface and adjust the **Reflectance** if needed.
   - Select **Optical polished** for a transparent or perfectly polished material (glass, plastic).
   - Select **Library** and click **Browse** to select and load a SOP file.

     If you want to modify the SOP file, click **Open file** to open the **Surface Optical Property Editor**.

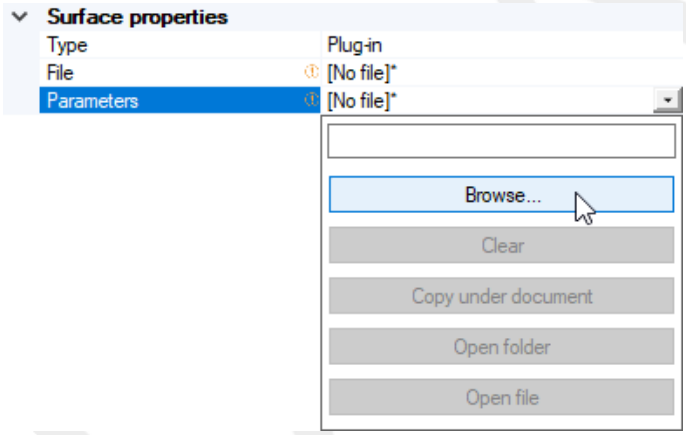
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



> **Tip:** To define a surface which is polarized, select a **.polarizer** file instead of a .coated file as coated surfaces are isotropic.

- Select **Plug-in**, and click **Browse** to select a custom made *.sop plug-in as **File** and the **Parameters** file for the plug-in.

> **Note:** Make sure each plug-in created has a different GUID.



For more information, refer to Surface State Plugin.

The optical properties are now created on the face(s). You can edit these properties at any time.

Related concepts

Optical Properties Overview on page 57
Optical Properties define how light rays interact with geometries in the CAD. Objects have volume and surface optical properties.

Related tasks

Creating Optical Properties on page 68
Creating Volume Optical Properties (VOP) and Surface Optical Properties (SOP) on solids allows you to determine the light rays' behavior when they hit or are propagated in the geometry.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

5.1.5. VOP on Surface

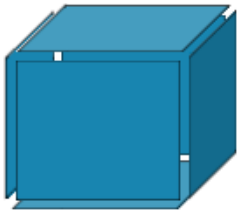
VOP on Surface allows Speos to simulate non-closed bodies with volume properties.

5.1.5.1. VOP on Surface Overview

VOP on Surface allows Speos to simulate non-closed bodies with volume properties.

A non-closed body is a group of surfaces with a same VOP/SOP material applied that are not physically joined.

Figure 3. Non-closed body example



The simulation will consider the surfaces of the non-closed body as joined during the run. Therefore, the VOP on Surface will be automatically applied to the non-closed body which will take into account the volume property.

To consider a surfaces as VOP on Surface, create a group of these surfaces.

This allows you to address principally cases like surface-based windshield.

5.1.5.2. Creating a VOP on Surface

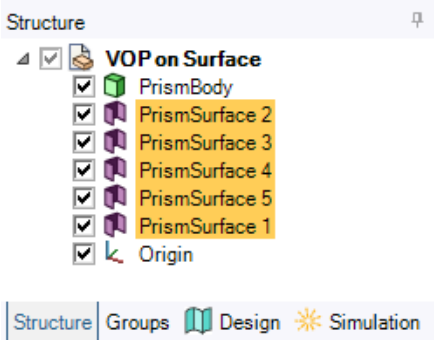
This procedure helps you define a non-closed body in a simulation to be considered as a VOP on Surface.

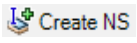
To create a VOP on Surface:

Make sure that the surfaces you want to consider as a VOP on Surface have the same material applied and that material has volume property (Optic or Library).

> **Note:** If a Face Optical Properties material (FOP) is applied on a part of a surface used for the VOP on Surface, the FOP still applies on the part of the resulting body.

1. In the **Structure** tree, select several geometries.



2. In the **Groups** tree, click **Create NS**  to create a group composed of the selected geometries.
3. In the Speos **Simulation** tree, open a Simulation Definition.
4. In the **3D view**, click  , then click  to select the created group.
5. In the **Groups** tree, select the group to consider for the VOP on Surface.
6. In the **3D view**, click **Validate**  .

The group is included in the Simulation Definition and will be considered as a VOP on Surface during the simulation run.

5.1.6. Optical Properties Management

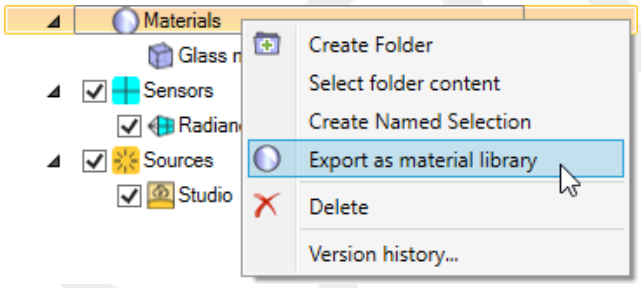
This section helps you manage and share materials in a same project or between projects.

5.1.6.1. Creating a Material Library

Creating a material library consists in exporting from one to all materials present in an opened project in order to share it with other projects.

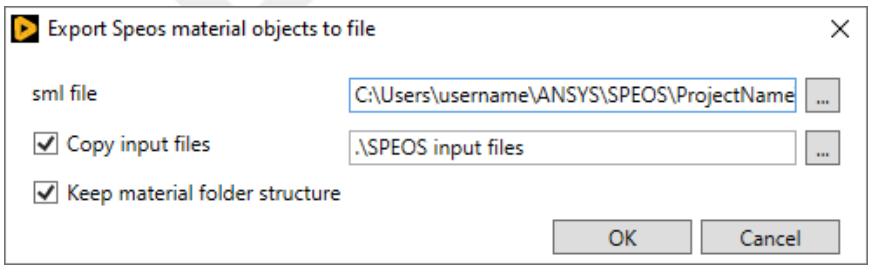
To create a Material Library:

1. In the Speos Simulation tree, if you want to:
   - export all material from the project, right-click **Material**.
   - export some materials from the project, multi-select the materials you want, then right-click a material.
2. Click **Export as material library**.



The **Export Speos material objects to file** window appears.

3. In the **sml file** field, click  and save the material library *.sml file in a dedicated directory.



4. If you want to save all related inputs in a dedicated folder at the location of the library or at a custom location:
   a) Check **Copy input files**

b) In the **Copy input files** field, click ... and define a directory.

> **Note:** Disabling **Copy input files** keeps the relative path to input files as described in their respective material definitions.

5. If you want to keep the structure of the Material node of the Speos tree, check **Keep material folder structure**.

> **Warning:** **Keep material folder structure** is an option available from the version 2023 R1. A *.sml file created in 2023 R1 or after cannot be opened in a release prior to 2023 R1. However, *.sml files created before 2023 R1 can still be opened in 2023 R1 or subsequent versions.

6. Click **OK**.

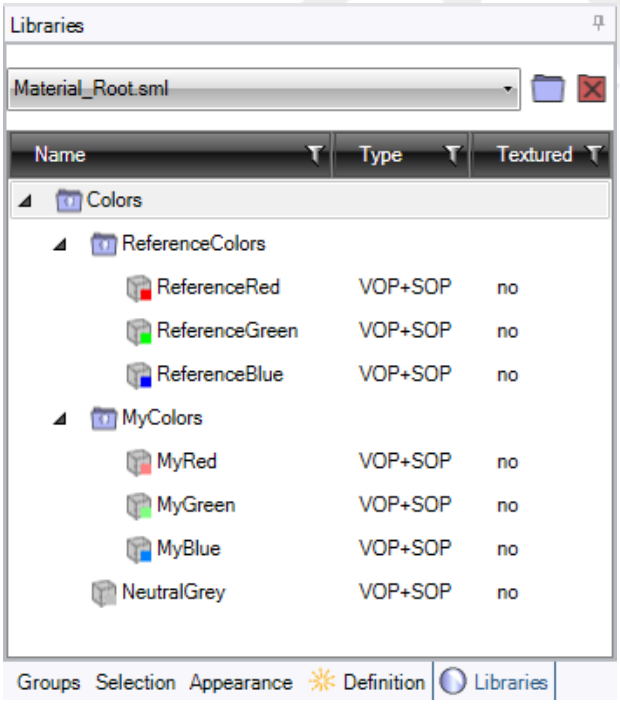
The Material Library is saved and can now be opened in the interface via the libraries tab.

5.1.6.2. Opening a Material Library

Opening a Material Library allows you to directly display a list of materials in the Speos interface.

To open a material library:

1. In the **Light Simulation** tab, click **Material Libraries** to show (or hide) the Libraries panel.
2. In the Speos interface, from the **Libraries** tab, click **Open a library file**.



3. Select a material library *.sml file and click **Open**.

The materials included in the library file appear in the Libraries panel.

> **Note:** You can open several library that will appear in the drop-down list, and switch from one library to another.

5.1.6.3. Applying a Material from a Material Library

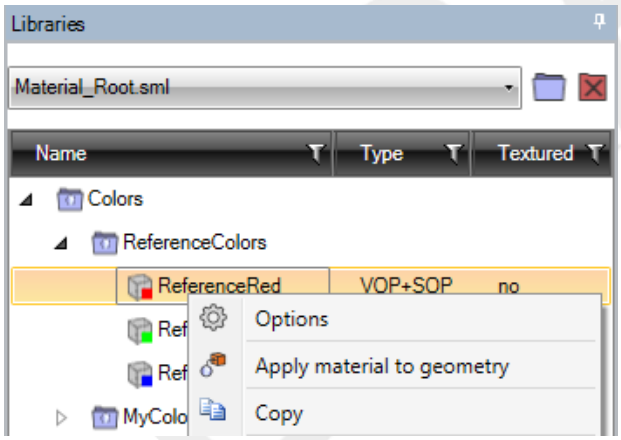
This procedure helps you apply a Volume/Surface Optical Properties material (VOP/SOP) on bodies or Face Optical Properties material (FOP) on faces from an opened material library.

To apply a material on a geometry:

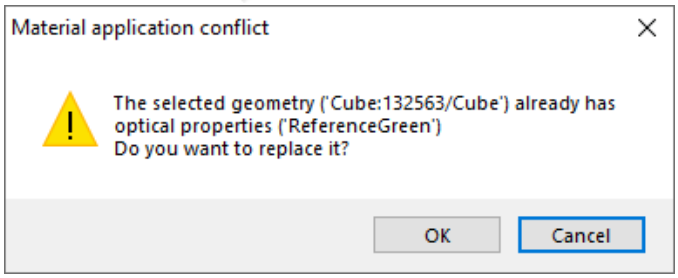
A material library must have been opened.

> **Tip:** Right-click a Material and click **Select associated geometry** to highlight in the 3D view the geometries on which the material is applied.

1. In the **3D view** or in the **Structure** tree:
   - if you want to apply a Volume/Surface Optical Properties material, select one or more bodies.
   - if you want to apply a Face Optical Properties material, select one or more faces.
2. In the **Libraries** panel, from the opened material library, right-click a corresponding material (VOP/SOP or FOP) and click **Apply material to geometry**.



3. If a material is already applied on the element(s) on which you want to apply the selected material, a prompt message asks you if you want to replace it. Click **OK** to replace the material.



A new material is created under the Materials list in the Simulation tree and associated to the bodies or faces selected.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Note: You can also drag and drop a material in the tree. This material is not associated to a geometry, and its tree icon is greyed / .

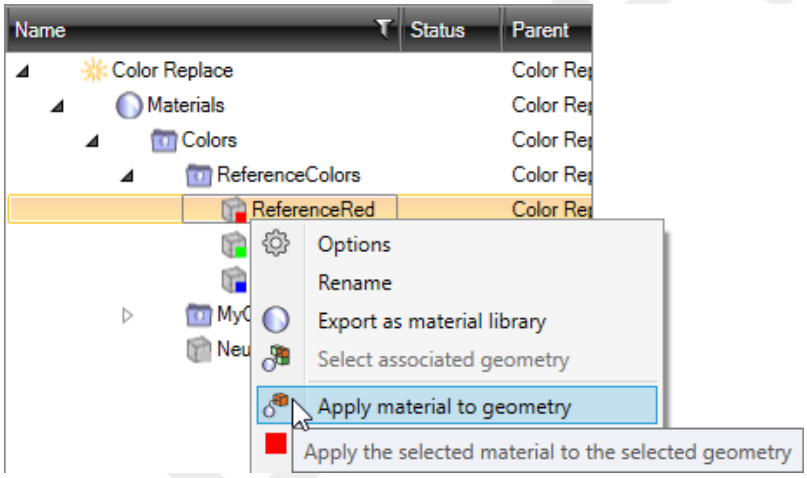
5.1.6.4. Applying a Material from the Tree

This procedure helps you apply a Volume/Surface Optical Properties material (VOP/SOP) on bodies or Face Optical Properties material (FOP) on faces directly from the tree.

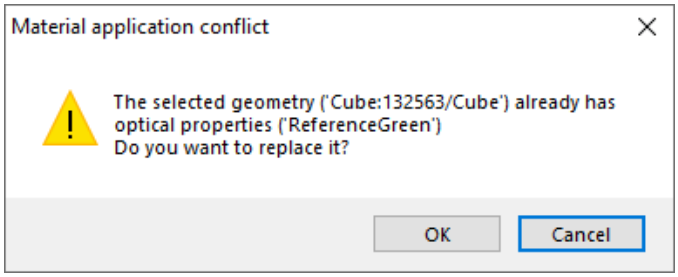
To apply a material on a geometry:

Tip: Right-click a Material and click **Select associated geometry** to highlight in the 3D view the geometries on which the material is applied.

1. In the **3D view** or in the **Structure** tree:
   - if you want to apply a Volume/Surface Optical Properties material, select one or more bodies.
   - if you want to apply a Face Optical Properties material, select one or more faces.
2. In the Speos **Tree**, use **[CTRL + left-click]** to select the material to apply, then right-click the material.
   [CTRL + left-click] on the material avoids losing the selection of the element(s) on which to apply the material.
3. Click **Apply material to geometry**.



4. If a material is already applied on the element(s) on which you want to apply the selected material, a prompt message asks you if you want to replace it. Click **OK** to replace the material.



The material is applied to the bodies or faces selected.

5.1.6.5. Replacing a Material on Geometries

This procedure helps you replace a material applied on all its faces or bodies by another material.

To replace a material:

> **Tip:** Right-click a Material and click **Select associated geometry** to highlight in the 3D view the geometries on which the material is applied.

1. Select a material from the **Library** panel or the **Material** list in the **Simulation** tree.
2. Drag the material X and drop it on a material Y that you want to replace.

The geometries that were using the material Y now use the material X. The material X retrieves the geometries in its definition, and the material Y becomes unused.

5.1.6.6. Converting Face Optical Properties

This procedure helps you convert an existing Face Optical Properties material (FOP) into a new FOP based on the material properties of a selected Volume/Surface Optical Properties material (VOP/SOP) while keeping the faces of the existing FOP.

To convert a FOP:

> **Tip:** Right-click a Material and click **Select associated geometry** to highlight in the 3D view the geometries on which the material is applied.

1. Select a SOP/VOP material from the Library panel or the Material list in the Simulation tree.
2. Drag the SOP/VOP material on the FOP material to convert.
   The **Create new FOP** window appears.
3. Click **OK** to create the new FOP.

The new FOP is created based on the same material properties as the SOP/VOP material used, and is assigned to the faces on which the converted FOP was applied. The converted FOP becomes unused.

5.1.7. Locate Material Tool

The Locate Material tool allows you to analyze the scene and identify on which geometries the materials are applied and on which geometries they are missing.

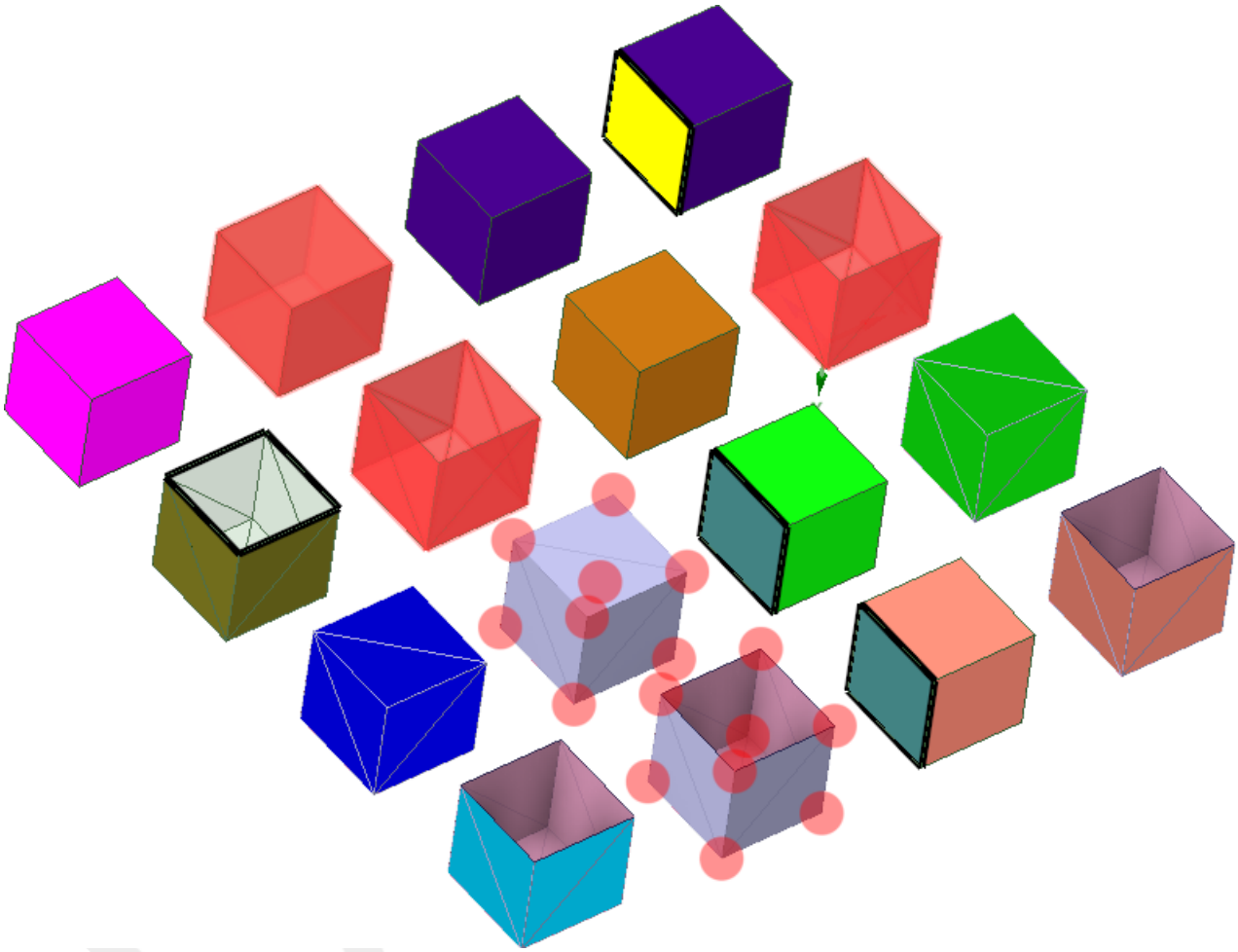
5.1.7.1. Understanding the Locate Material Tool

With the Locate Material tool you can quickly distinguish which material is applied on which geometry and which geometries do not have any material applied.

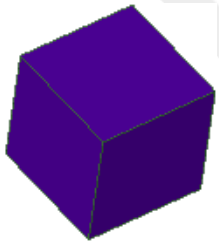
DRAFT

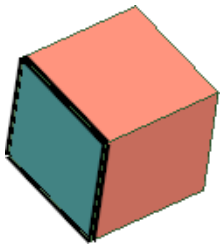
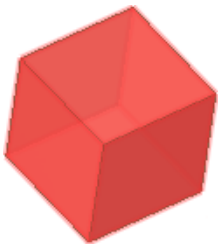
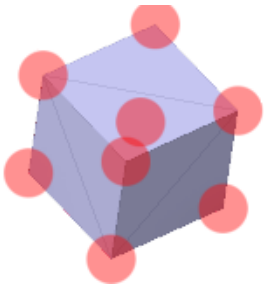
Important: The visualization of the material tool and the selection of geometries are independent and should not be mixed up. When you highlight materials, geometries are not selected. Specific options are dedicated to the selection of geometries.

Figure 4. Material highlighted on geometries and geometries with no material highlighted



Material Highlight

	Materials applied on bodies appear highlighted in the 3D view according to the color visualization selected.

	Material applied on faces appear highlighted with black edges in the 3D view according to the color visualization selected.
	Geometries (Surface or Body) with no material appear highlighted in red.
	Faceted geometries with no material appear with red circles highlighted on their vertices.

Geometry Selection

Besides highlighting the materials, the Locate Material tool allows you to quickly select the geometries on which materials are applied, or select the geometries on which no materials are applied.

5.1.7.2. Using the Locate Material Tool

The following page helps you use the different options of the Locate Material tool.

To use the Locate Material tool:

1. In the **Light Simulation** tab, click  to open the **Locate Material** tool.
2. In the **3D view**, click **Search for applied material**  to activate the Locate Material tool.

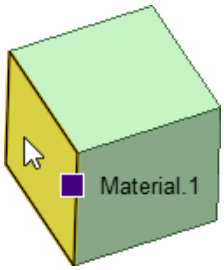
   According to the Visualization options defined, either all materials applied on geometries are highlighted in the scene and/or geometries with no material are highlighted, or nothing is highlighted.

   For more information on the color code, refer to Understanding the Locate Material Tool.
3. Once activated, you can:
   - Select a material in the tree to highlight the geometries on which the material is applied.
   - Click a geometry in the **3D view** or in the **Structure** tree to highlight all geometries using the same material.

DRAFT

> **Tip:** If you want to highlight the materials applied on a geometry without using/opening the Locate Material tool, right-click a geometry and click **Locate Material**.

- Mouse over a geometry to display the corresponding material applied.



- Double-click a geometry to select its material in the Speos tree.
- Triple-click a geometry in the **3D view** to open the definition of the material.

  This action exits the Locate Material tool. Closing the definition mode resumes the Local Material tool.

- Select one or several materials in the Speos tree and click  to select their associated geometries.

  The associated geometries are selected and highlighted in the **3D view** and in the **Structure** tree.

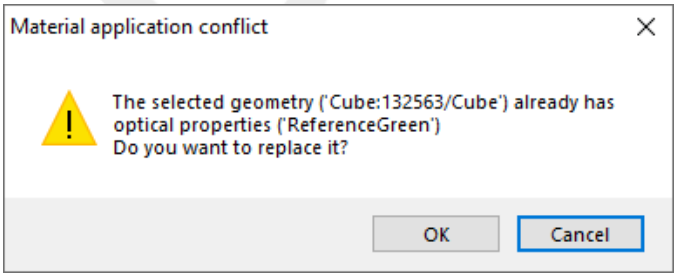
> **Tip:** If you want to select geometries without using/opening the Locate Material tool, right-click a material (or a selection of materials) and click **Select associated geometries**.

- Click  to select the geometries without material.

> **Tip:** Create a Named Selection to group these geometries and apply adequate materials.

- Select a geometry, then use **[CTRL + left-click]** on a material in the tree, then click **Apply material to geometry**  in the **3D view**.

  If a material is already applied on the element(s) on which you want to apply the selected material, a prompt message asks you if you want to replace it. Click **OK** to replace the material.



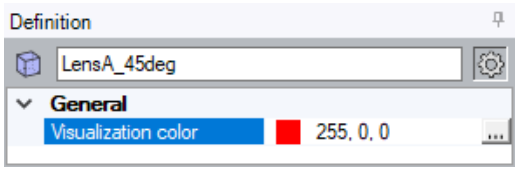
DRAFT

5.1.7.3. Applying a Visualization Color to a Material

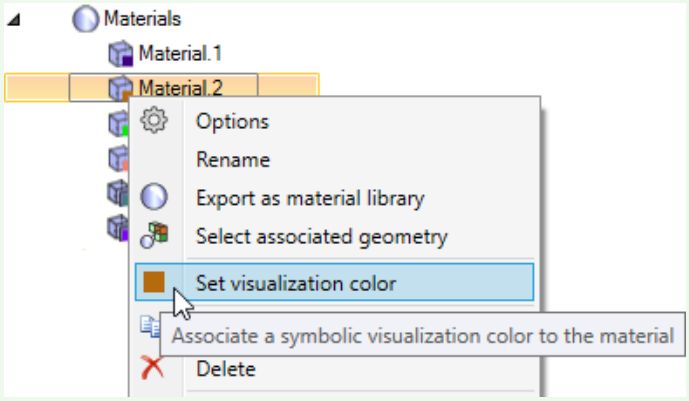
The following procedure helps you apply a visualization color to a material in order to quickly find the material in the 3D view. Thus, when you use the locate material tool, the geometries on which the material is applied are hightlighted in 3D view with the chosen color.

To apply a visualization color to a material:

1. Open a material.
2. In the **Material** definition, click **Optional or advanced settings**  .

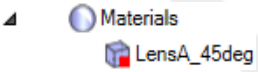


3. Click  to open the **Color** tool.

> **Tip:** You can directly right-click a material and click **Set visualization color** to open the **Color** tool, or you can right-click a geometry when the Locate Material tool is activated and click **Material visualization color**.
>
> 
>

4. Define the color to apply.

The visualization color is applied to the material and the material will be highlighted with this color in the 3D view.

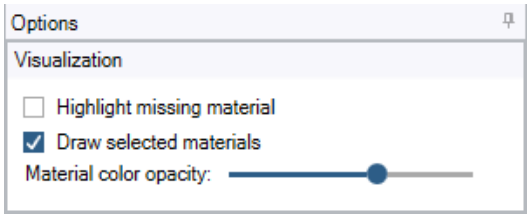


5.1.7.4. Defining the Visualization Options

The following procedure helps you adjust the visualization options of Locate Material tool.

To define the Visualization options:

1. Open the **Locate Material** tool.
2. Open the **Options** panel.



3. With the **Visualization** options, you can:
   - **Highlight missing material**

     When activated, geometries with no material will be highlighted in red when using the Locate Material tool.
   - **Draw selected material**

     When activated, the colored materials are highlighted in the 3D view when using the Locate Material tool. Otherwise the materials are only highlighted in the tree.
   - Define the **Material color opacity**.

     This option allows you to adjust the level of transparency of the material highlight so that you can see through transparent geometries or better see the geometry selection.

4. To apply the modifications, in the 3D view, click **Search for applied material**  .

The modifications defined in the Visualization options are visible in the 3D view.

5.2. Texture Mapping

This section introduces the tools that allows you to create and apply texture onto geometries in Speos.

5.2.1. Texture Mapping Overview

Texture mapping is a process that allows you to simulate material texture to improve realism. A texture mapping can be applied on a surface, a face, an outer surface of bodies or on geometry groups.

Texture mapping allows you to simulate physics-based textures in your projects. In Speos, you can create a texture mapping from any image, normal map or from optical properties.

To create a texture mapping, two elements are necessary: the texture itself (coming from an image, a normal map or a BSDF) and the UV mapping technique (that is, the application method).

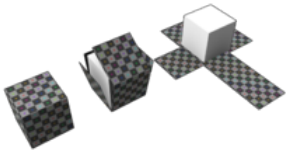
> **Note:** A typical workflow of a texture mapping creation is described in the following mapping process overview.

UV Mapping Process

The Mapping is projected on the geometry using the UV mapping process. UV mapping is the process of wrapping a 2D image on a 3D mesh. U and V are used to denote the axes of the plane because X,Y and Z are already used for the coordinates of the 3D object. "U" denotes the horizontal axis of the mapping projection and "V" denotes the vertical axis of the mapping projection.

The mapping can be planar, spherical, cylindrical or cubic.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



Types of Texture Mapping

- A texture mapping using an **image texture** modifies the distribution of light. They are often used to simulate materials with important color variations (wood), or for printings, logos, stickers, etc. You can load an image and define its properties (position, size, repeatability, projection direction etc.) from the interface.
- A **normal map** modifies the aspect of the surface by simulating bumps and wrinkles. It is used to simulate depth for materials like leather or grained plastics. To set a normal map, you can load a standard image or a normal map and define its properties (position, size, repeatability, projection direction, roughness etc.) from the interface.
- A texture mapping taking into account **anisotropic direction** is used for materials that are directionally dependant like carbons or fiber-reinforced composites.
- A combination of different mapping techniques for materials that require it, like fabrics. Fabrics often need a combination of image texture and normal map.



Multi Layer System

The texture mapping works in a principle of layer superposition.

On each layer you can stack different types of mapping. You can apply an image texture and a normal map, or an image texture along with optical properties (a BSDF anisotropic file for example).

You can also work with images containing transparency.

With this multi-layer system you can mix several optical behaviors to create a global texture realistically rendered.

Related concepts

Texture Mapping Process Overview on page 87
This page illustrates a standard texture mapping process.

Understanding UV Mapping on page 84
This page describes the UV mapping principle along with the different types of mapping available in Speos.

Understanding Texture Mapping on page 85
This page describes how the texture mapping multi layer system works and how the rays interact with it.

5.2.2. Understanding UV Mapping

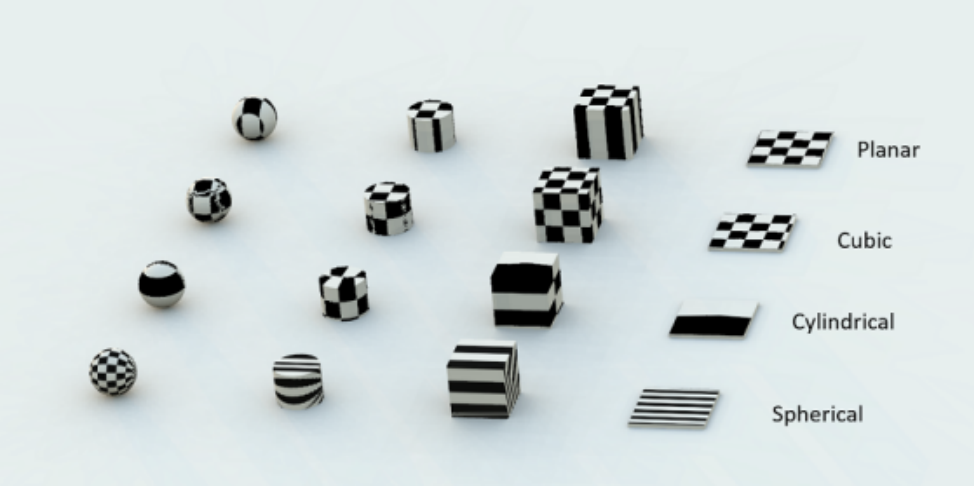
This page describes the UV mapping principle along with the different types of mapping available in Speos.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

UV mapping is the process of wrapping a 2D image on a 3D mesh. It determines how the mapping is projected on the geometry.

Different mapping types exist to suit and fit the shape of the geometry on which to apply a texture. The mapping can be planar, cubic, cylindrical or spherical as illustrated below.

Figure 5. Effect of the mapping type depending on the geometry's shape



In Speos, a UV mapping feature corresponds to one mapping type and is linked to a unique set of geometries that you define. A geometry cannot be included in different UV mapping features. It is necessarily stored in one feature.

If you want to apply two textures on a same surface and want them to be projected or rendered differently, you can create new UV maps under the same UV mapping feature.

For example, if you want to create a texture mapping on a surface with a texture image with a planar mapping type and a normal map with a specific projection, you need to create a UV mapping feature containing your surface and two UV maps under it containing your two mappings.

Related concepts

Understanding Texture Mapping on page 85
This page describes how the texture mapping multi layer system works and how the rays interact with it.

Texture Mapping Overview on page 83
Texture mapping is a process that allows you to simulate material texture to improve realism. A texture mapping can be applied on a surface, a face, an outer surface of bodies or on geometry groups.

Texture Mapping Process Overview on page 87
This page illustrates a standard texture mapping process.

5.2.3. Understanding Texture Mapping

This page describes how the texture mapping multi layer system works and how the rays interact with it.

Texture Mapping Process

When a Texture Mapping is created on a surface or a face, it is considered as merged with that surface.

The Texture Mapping can contain one or several layers. On each layer you can define an image type of texture, a normal map and/or apply specific optical properties. You can even superimpose different types of mapping and combine a normal map and an image texture for example.

When rays hit the Texture Mapping, they interact with the layer(s) you created. Depending on the **level of transparency** (alpha) of the layers, a certain amount of information can be processed on the current layer or on the next. If you created a Texture Mapping with an image texture containing alpha channels, a certain amount (expressed in %) of the rays will get through these areas of transparency to reach the next layer and continue interacting with it.

Then, the rays that interact with a layer are submitted to direction and color change according to the texture mapping created.

This process, described below, is repeated on each layer.

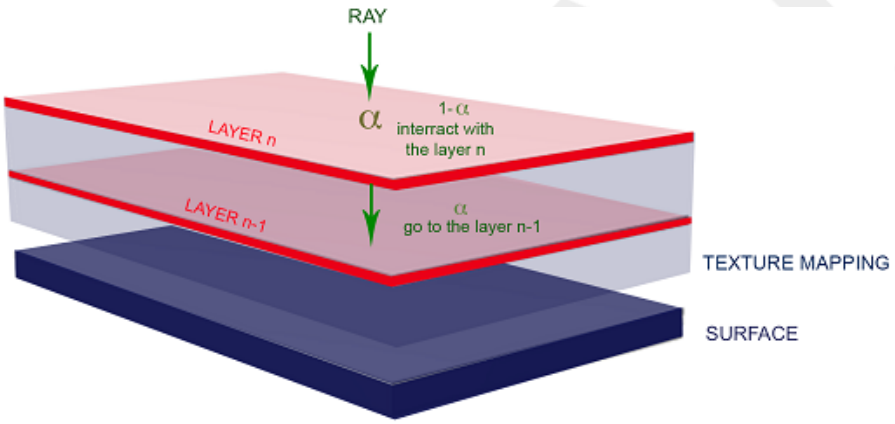
Phase One: The Layer Selection

The layer selection determines if the ray can carry on interacting with the layer (L_n) or goes to the next layer (L_{n-1}).

To determine the layer selection, the algorithm checks the **transparency level** (α of the layer L_n Image Texture pixel on which the ray is interacting.

α is a component of the Image Texture which corresponds to the transparency layer in addition to other R,G,B layers and is coded on 8-bit with values being within the 0-255 range.

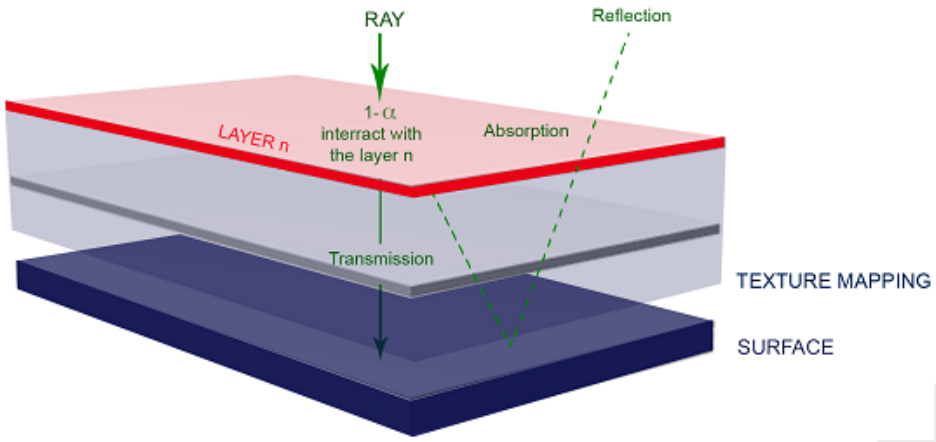
The probability to interact with the layer L_n or to go to the layer L_{n-1} is calculated according to the level of α, for $0 < \alpha < 255$. There is α-in-255 chance that a ray uses the Layer Ln optical properties.



- If α = 255, the pixel of the texture file is opaque and the ray interacts only with the Surface State of the layer L_n.
- If α = 0, the pixel is transparent and the ray does not consider optical properties from the Layer Ln, it goes to the layer L_{n-1}.

Phase Two: Surface State Interaction

When the ray carries on interacting with the layer L_n, the second step is the **surface state interaction**. The surface state corresponds to the optical properties and the texture normalization applied to the layer.



The ray is then submitted to:

- Direction change according to the Optical Properties of the layer.

  Once a ray interacts with the Surface State of a layer, the ray no longer interacts with other layers and is either transmitted toward the volume, reflected, or absorbed.

- Color change according to the Texture normalization parameter selected:
  - **Color from BRDF**: ray color changes according the Optical Properties of the layer.
  - **Color from Texture**: ray color changes according the Image Texture RGB components.
  - **Color from None**: ray color changes according the Optical Properties and the Image Texture RGB components.

Related concepts

Understanding UV Mapping on page 84
This page describes the UV mapping principle along with the different types of mapping available in Speos.

Texture Mapping Process Overview on page 87
This page illustrates a standard texture mapping process.

Related information

Creating a Texture Mapping on page 92
This section shows how to create a texture mapping over one or several geometries.

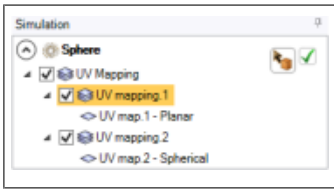
5.2.4. Texture Mapping Process Overview

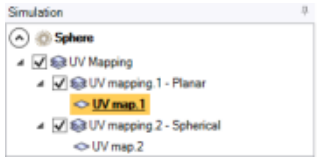
This page illustrates a standard texture mapping process.

The texture mapping can be separated into two major steps: the **UV mapping creation** and the **texture creation**.

UV Mapping Creation

UV mapping consists in assigning a mapping type to the geometries you want to apply a texture to.

	The "feature" level allows you to select the geometries on which to apply the mapping. One UV mapping feature can only contain one unique set of geometries. A geometry cannot be selected into two different UV Mapping features.

	The **UV map** level allows you to define the mapping type/technique used to apply the texture (planar, spherical, etc.). You can create as many UV maps as you need under a UV Mapping feature. Each UV map has an index that depends on its position under the feature. For example if you need to superimpose an image texture and a normal map on a same surface but you want to apply them differently, you can create two UV maps under the same UV Mapping feature.

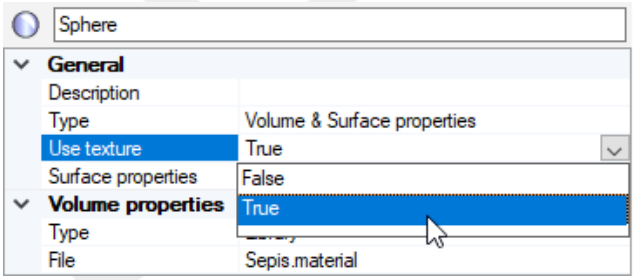
Texture Creation - Materials

Material Creation

Materials allow to define texture.

The material level allows you to select the geometries on which to apply the texture.

Then in each material created, you can activate the **Use texture** option.

	
Material level - geometry selection	Texture activation

Texture creation

Activating the texture creates another **Surface layer** on which you can define an image texture, normal map and/or apply specific optical properties.

In the surface layer, you need to define the texture properties and a **UV map index**. This index allows you to indicate which **UV map** layer should be used to apply the texture.

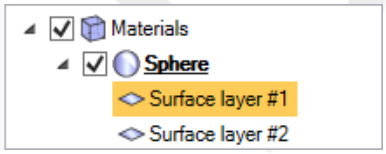
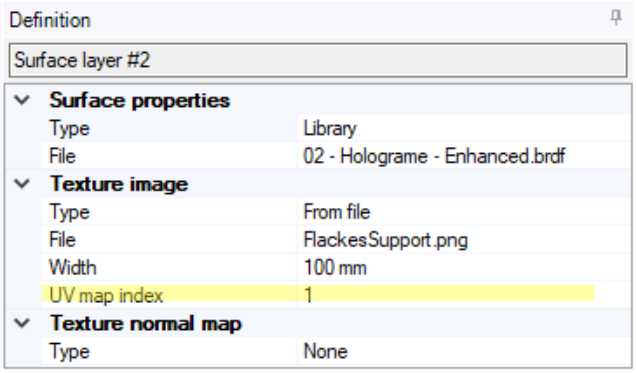


Figure 6. Texture Image definition



DRAFT

Figure 7. Normal map definition

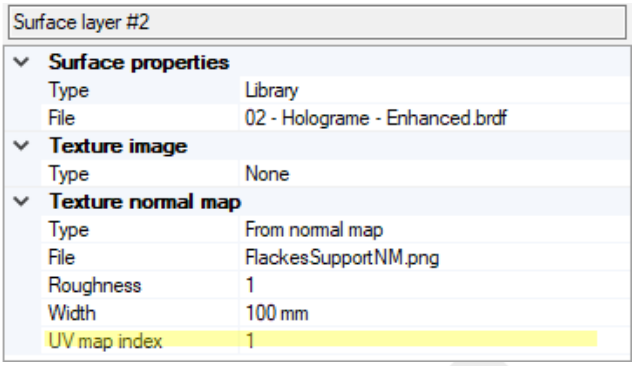
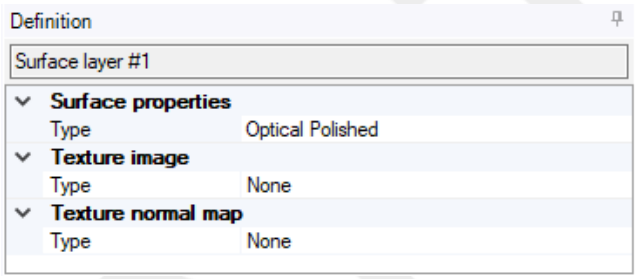


Figure 8. Optical Property Definition



Algorithm Check

Once the UV mapping and materials are created, the algorithm confronts the geometries contained in the material and the ones selected in the UV mapping features.

When a geometry is detected in both elements, the **UV map index** defined in the surface layer is used to select the correct **UV map** to apply the texture.

Related concepts

Understanding UV Mapping on page 84
This page describes the UV mapping principle along with the different types of mapping available in Speos.

Understanding Texture Mapping on page 85
This page describes how the texture mapping multi layer system works and how the rays interact with it.

Related information

Creating a Texture Mapping on page 92
This section shows how to create a texture mapping over one or several geometries.

5.2.5. Texture Mapping Preview

Speos provides the possibility to dynamically preview the texture mappings in the 3D view while working on them so that you can directly see if textured materials are correctly defined.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Description

Texture mapping preview allows you to:

- immediately access to texture preview on geometries while editing the UV mapping properties. This way you can have a dynamic overview of the mapping and understand if a textured material is well defined.
- See the texture size on geometries while setting the texture parameters.
- See the alignments between textures on multiple objects when setting UV mapping.
- See the UV orientation when setting anisotropic properties.

Texture Mapping Preview:

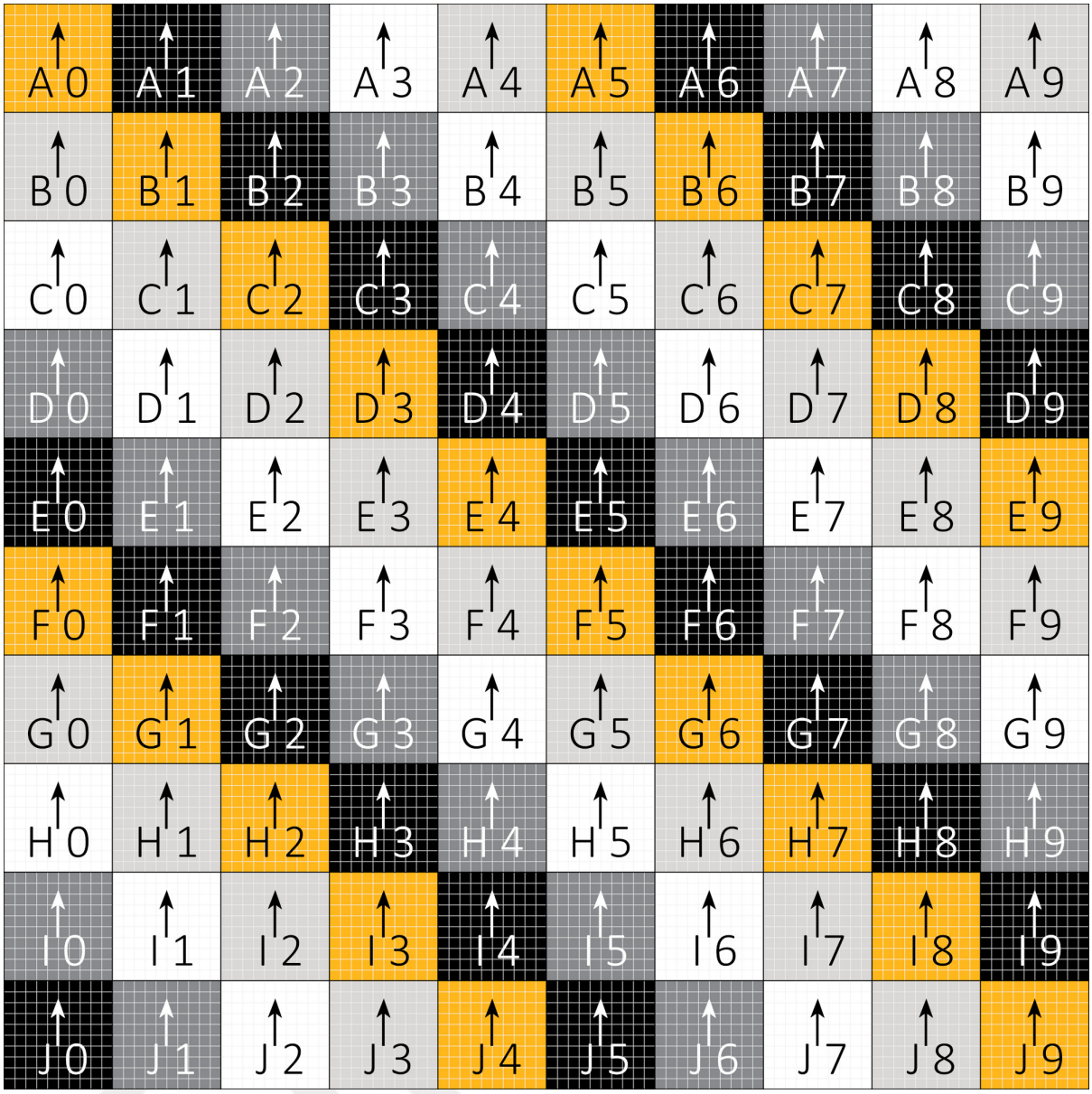
- can be activated permanently.
- is automatically displayed when you are in the UV map definition or in the Surface Layer definition, until you exit the definition.

The preview of the textures corresponds to the simulation results and the Live Preview.

If no texture or material is defined, a default rendering (checker texture) is displayed for the texture or the normal map to help you set the UV map:

- The arrow indicates the vertical axis to help you orientate anisotropic materials.
- The checker texture measures 100x100 mm, and each colored square measures 10x10 mm. This can help you assess the distortions due to the projection on the geometries.

DRAFT



Example

The following example is a combination of two textured materials:

- Textured Material 1 is applied on the solid and composed of one Surface Layer
- Textured Material 2 is applied on a face and composed of two Surface Layers

> **Note:** The Texture image of both Surface Layers 1 is a real image that looks like the default rendering when no texture is applied.

	Texture Material 1	**Texture Material 2**	
	Surface Layer 1	**Surface Layer 1**	**Surface Layer 2**
Surface properties	PerlGold.brdf	PerlGold.brdf	PerlGold.brdf

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

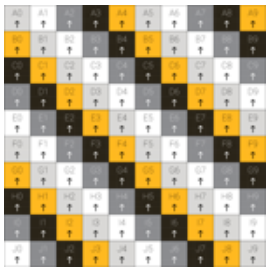
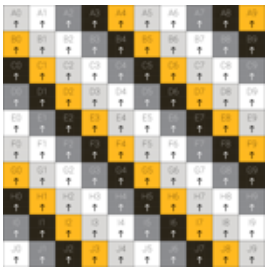
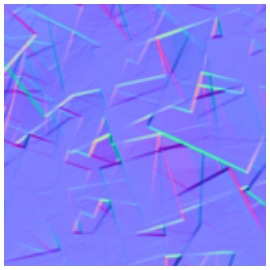
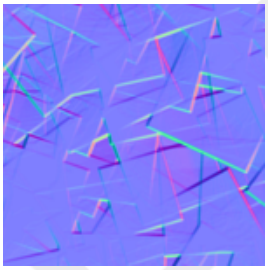
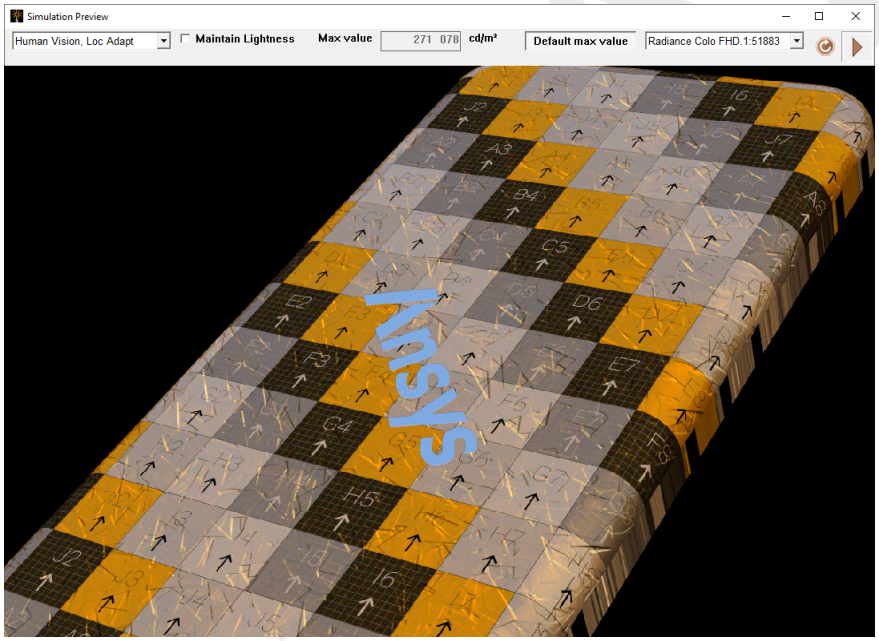
	Texture Material 1	Texture Material 2	
	Surface Layer 1	Surface Layer 1	Surface Layer 2
Texture image			
Texture normal map			

Figure 9. Result Generated using the Live Preview



5.2.6. Creating a Texture Mapping

This section shows how to create a texture mapping over one or several geometries.

5.2.6.1. Creating the UV Mapping

This page describes the first steps of a texture mapping creation, that is the definition of the UV mapping of geometries on which texture is applied.

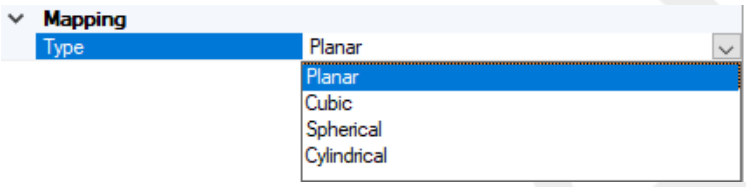
To create a UV Mapping:

1. From the **Light Simulation** tab, click **UV Mapping**  .
2. In the 3D view, click  , select the geometries on which to apply the current mapping and click  to validate.

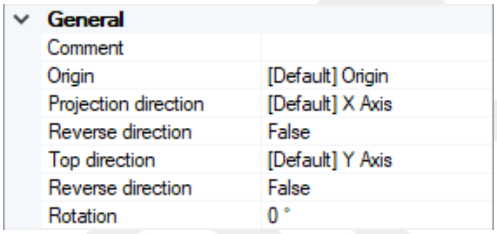
   During UV Mapping edition, the texture set or the checker texture is displayed in the 3D view on the associated geometries and is updated upon modifications.

   > **Tip:** If you need a certain mapping type (a planar mapping for example) for several geometries, select them all and create only one UV mapping that will be used for each geometry.

3. In the first UV map, select the mapping type you need.



- **Planar**



- Define the Axis System of the Planar mapping type:
  - For **Origin**, click 
    and select the middle point of the image texture you want to apply.
  - For **Projection direction**, click 
    and select a line defining the direction in which the image texture should be projected on the plane.
  - For **Top direction**, click  and select a line defining the orientation of the image texture on the plane.

    > **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- or click
  
  and select a coordinate system to autofill the Axis System.

- If you want to apply a rotation to the current image, define a rotation value in degrees.

Cubic

General	
Comment	
Origin	[Default] Origin
Projection direction	[Default] X Axis
Reverse direction	False
Top direction	[Default] Y Axis
Reverse direction	False
Rotation	0 °

- Define the Axis System of the Planar mapping type:
  - For **Origin**, click
    
    and select the middle point of the image texture you want to apply.
  - For **Projection direction**, click
    
    and select a line defining the direction in which the image should be projected on the plane.
  - For **Top direction**, click  and select a line defining the orientation of the image texture on the plane.

    > **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

  - or click
    
    and select a coordinate system to autofill the Axis System.
- If you want to apply a rotation to the current image, define a rotation value in degrees.

Spherical

General	
Comment	
Origin	[Default] Origin
Projection direction	[Default] X Axis
Reverse direction	False
Top direction	[Default] Y Axis
Reverse direction	False
Rotation	0 °
Perimeter	100 mm

- Define the Axis System of the Planar mapping type:

DRAFT

- For **Origin**, click
  
  and select a point placed in the center of the sphere.
- For **Projection direction**, click
  
  and select a line defining the direction in which the texture should be projected.
- For **Top direction**, click  and select a line defining the orientation of the image texture on the plane.

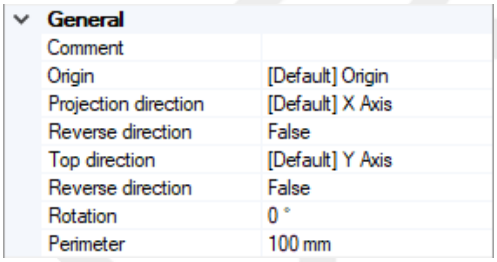
> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

- or click
  
  and select a coordinate system to autofill the Axis System.

- If you want to apply a rotation to the texture, define a rotation value in degrees.
- In **Perimeter**, define the perimeter of the sphere in mm.

> **Tip:** If the geometry was build with SpaceClaim modeler, you can obtain the perimeter by clicking **Measure**  and clicking on the geometry.

Cylindrical



- Define the Axis System of the Planar mapping type:
  - For **Origin**, click
    
    and select the point on the revolution axis of the cylinder.
  - For **Projection direction**, click
    
    and select a line defining the direction in which the texture should be projected.
  - For **Top direction**, click  and select a line defining the orientation of the image texture on the plane.

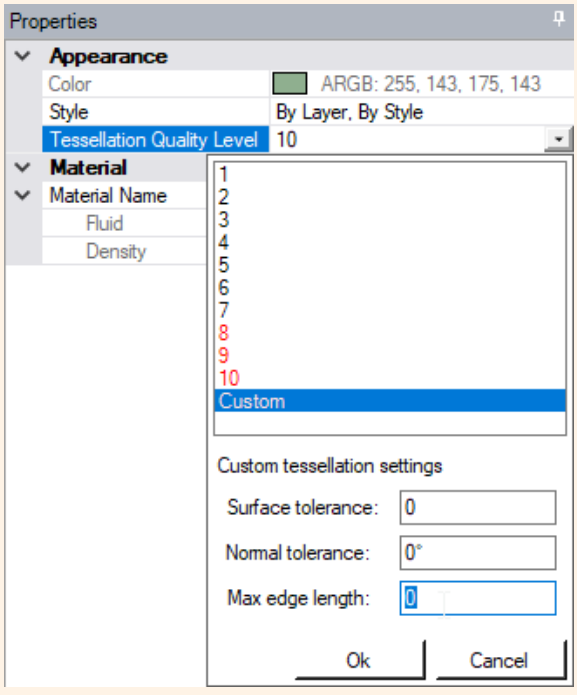
> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

- or click
  
  and select a coordinate system to autofill the Axis System.

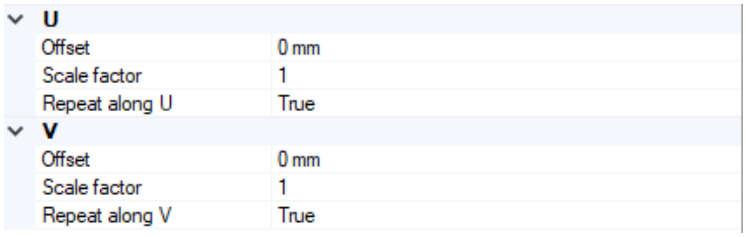
- If you want to apply a rotation to the texture, define a rotation value in degrees.
- In **Perimeter**, define the perimeter of the cylinder basis in mm.

> **Tip:** If the geometry was build with SpaceClaim modeler, you can obtain the perimeter by clicking **Measure**  and clicking on the geometry.

> **CAUTION:** When a cylindrical mapping is applied on a cylindrical geometry, the preview of the texture mapping may be inconsistent. To avoid such an issue:
>
> **a.** Open the **Properties** of the body.
> **b.** Set **Tessalation Quality Level** to **Custom**.
> **c.** Set **Max edge length** to a value consistent with the body size.



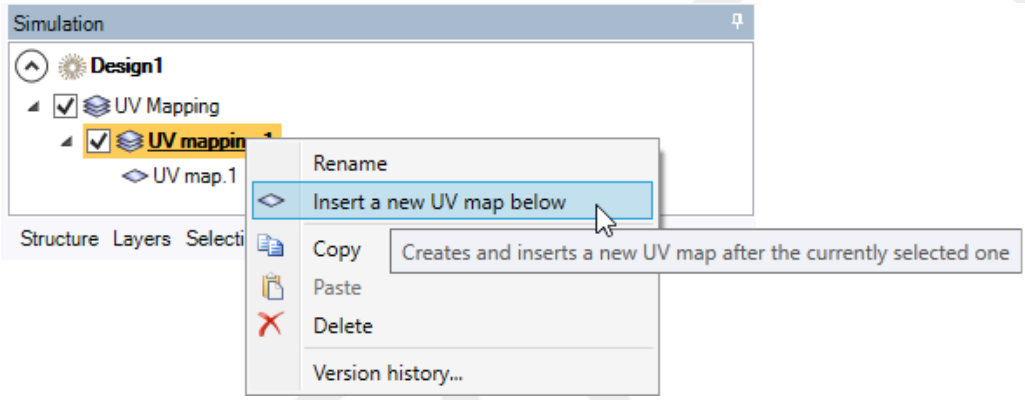
4. In **U** and **V** sections, respectively denoting the horizontal and vertical axes of the mapping projection:



a) Define the horizontal and vertical positioning of the texture using the U/V **Offset.**
b) If you want to define a specific scale on U and/or V, adjust the **Scale factor**.
c) Activate/deactivate the repeatability of the texture on U and/or V axes.

> **Note:** This option is activated by default. Repeating the texture on the geometry ensures that the integrality of the surface is covered by the texture.

5. If you selected a set of geometries that require different mapping applications:



a) Right-click the **UV mapping** feature and click **Insert a new UV map below**.
b) Repeat the steps 3 and 4 for the newly created UV map.
Once all your UV mappings are created for your geometries, move on to the texture application .

Related tasks

Applying Textures on page 97
This section gathers the three types of texture application that can be performed in Speos.

5.2.6.2. Applying Textures

This section gathers the three types of texture application that can be performed in Speos.

5.2.6.2.1. Applying an Image Texture

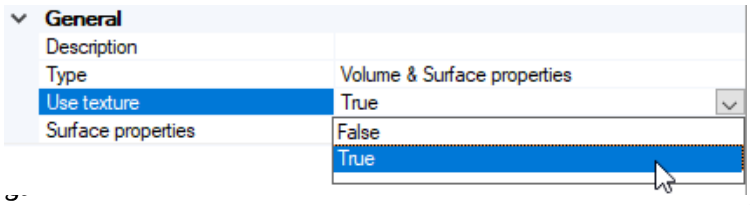
This page shows how to apply an image texture. Image textures are often used to simulate materials with color variations like wood.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

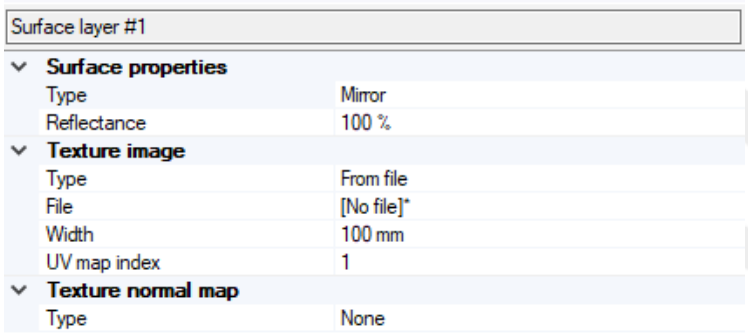
To apply an Image Texture:

One or several UV mappings should already be created.

1. From the **Light Simulation** tab, click **Material** .
2. In the 3D view, click , select the geometries on which to apply the texture and click to validate.
3. In **General**, set **Use Texture** to **True**.
   A surface layer is created under the material in the simulation panel and allows you to define the surface properties of the



4. Define the volume properties of the geometry.
5. In the surface layer, from the **Texture imageType** drop-down list, select **From file** to activate the image texture.



- In **File**, double-click in the field to browse and load a .jpeg or .png file.
- Define the **Image width** of the image in mm.

  The Image size is calculated proportionally on U and V.

- Define the **UV map index**you want to use to apply this texture.

  The UV map index determines which UV mapping should be used to apply the texture. This index refers to the UV map or "layer" that should be selected within a **UV mapping** feature.

The Image texture is created and applied on the geometry.

Related concepts

Understanding UV Mapping on page 84
This page describes the UV mapping principle along with the different types of mapping available in Speos.

Texture Mapping Process Overview on page 87
This page illustrates a standard texture mapping process.

5.2.6.2.2. Applying a Normal Map

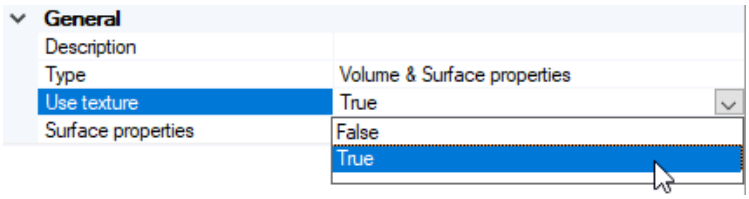
This page shows how to apply a normal map on a geometry to create an illusion of depth. The normal map is often used to simulate textured materials such as leather or plastics with a grain.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

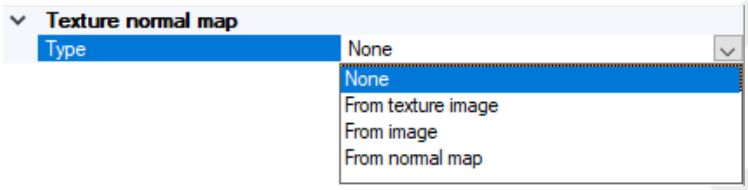
To apply a Normal Map:

One or several UV mappings should already be created.

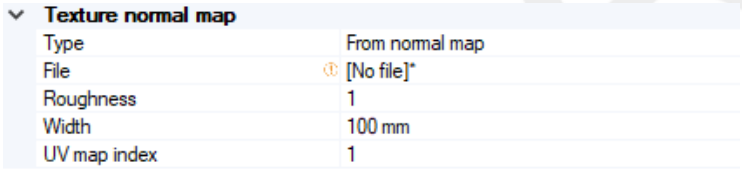
1. From the **Light Simulation** tab, click **Material** .
2. In the 3D view, click , select the geometries on which to apply the texture and click to validate.
3. In **General**, set **Use texture** to **True**.
   A surface layer is created under the material in the simulation panel.



4. In the surface layer, from the **Type** drop-down list, select:



- **From texture image** to create a normal map from a previously defined texture image and define the **Roughness** ratio of the normal.
- **From image** if you do not have a normal map and want to generate a normal map from a .jpeg or .png file.
- **From normal map** to generate a normal map from a .bmp file.



  - In **File**, double-click in the field to browse and load an image file.
  - **Roughness** corresponds to a ratio applied on the normal to the surface of a normal map:
    - A roughness ratio 1 applies the normal map parameters saved.
    - A roughness ratio different from 1 applies a multiplying factor on the normal, therefore accentuating or flattening normals. This gives the impression of changing roughness, therefore of different level of depth of the relief.
  - Define the **Image width** of the image in mm.
  - Define the **UV map index**you want to use to apply this texture.

    The UV map index determines which UV mapping should be used to apply the texture. This index refers to the UV map or "layer" that should be selected within a **UV mapping** feature.

The texture normal map is created and applied on the geometry.

Related concepts

Understanding UV Mapping on page 84
This page describes the UV mapping principle along with the different types of mapping available in Speos.

DRAFT

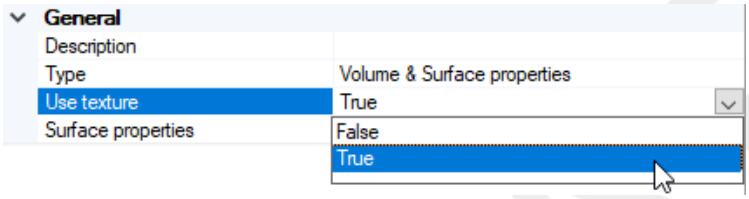
Texture Mapping Process Overview on page 87
This page illustrates a standard texture mapping process.

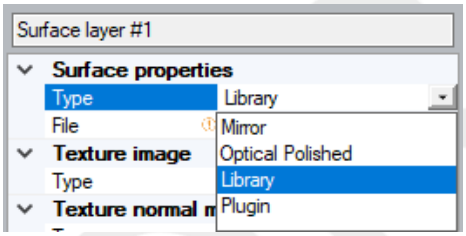
5.2.6.2.3. Applying an Anisotropic Mapping

This page shows how to create a mapping taking into account anisotropic direction. The anisotropic mapping is used for materials that are directionally dependant, like carbons.

To apply an Anisotropic mapping:

One or several UV mappings should already be created.

1. From the **Light Simulation** tab, click **Material**  .
2. In the 3D view, click  , select the geometries on which to apply the texture and click  to validate.
3. In **General**, set **Use texture** to **True**.
   A surface layer is created under the material in the simulation panel.
   

4. In the surface layer, define the surface properties to be applied on the geometry:
   a) From the **Type** drop-down list, select **Library**.
   b) In **File**, double-click in the field to browse and load an .anisotropicbsdf material.
   

The anisotropic mapping is created and applied on the geometry.

Related concepts

Understanding UV Mapping on page 84
This page describes the UV mapping principle along with the different types of mapping available in Speos.

Texture Mapping Process Overview on page 87
This page illustrates a standard texture mapping process.

5.2.7. Activating the Texture Mapping Preview

The following page helps you activate the texture mapping preview. This way the previewed texture mapping will permanently be displayed in the 3D view unless you deactivate it.

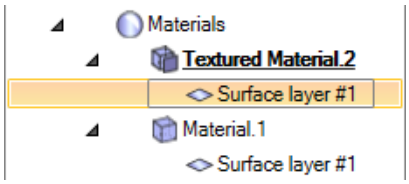
DRAFT

Note: Do not confuse the permanent preview with the preview displayed when you are in the UV map definition or in the Surface Layer definition which is displayed only until you exit the definition.

To activate a texture mapping preview:

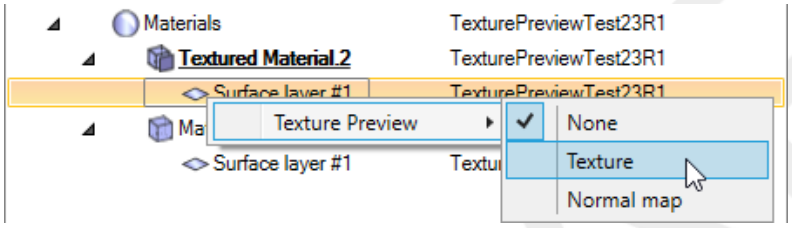
You must have set Use Texture to **True** on a material.

1. In the **Speos** tree, select a **Surface Layer**.



The Surface Layer must be selected, otherwise you cannot activate the texture mapping preview.

2. Right-click the **Surface Layer** and select the texture to preview:



- **None**: deactivates the Texture Mapping Preview.

  Note: The preview never deactivates itself, you need to see it to none if you no longer want to the preview to be displayed.

- **Texture**

  When Texture is activated, the Surface Layer is flagged with a black dot in the tree.

- **Normal map**

  When Normal map is activated, the Surface Layer is flagged with a purple dot in the tree.

Note: If a textured material has several surface layers, you can only preview one surface layer at a time.

Note: If no texture or material is defined, a default rendering is displayed for the texture or the normal map to help you set the UV map.

Note: If you have not deactivated a preview and you saved the project, the preview is displayed at the next opening of the project.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

The preview of the surface layer textures are now permanently activated until you deactivate them.

5.2.8. Texture Normalization

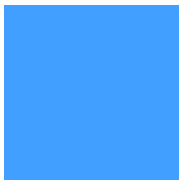
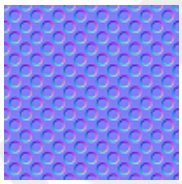
Texture normalization allows you to control the rendering of the results.

5.2.8.1. Understanding Texture Normalization

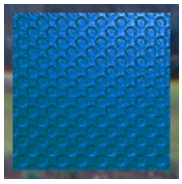
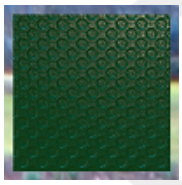
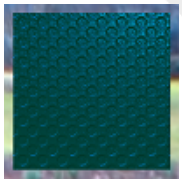
This page helps to understand the texture normalization and how texture mapping modifies the interaction between light and faces for all kind of interactions.

To understand the behavior of the texture normalization, let's consider the following context:

- A geometry with the following optical properties: **SOP**: red plastic scattering file, **VOP**: opaque.
- And a texture mapping with one surface layer containing:

An image texture	A normal map	An optical property
A uniform blue image	A lego like bump image	A BSDF
		A semi mat green BSDF

The simulation's results depend on the texture normalization selected:

	**Color from Texture** Color from Texture means that the simulation result uses the color and the color lightness of the Image Texture. In our context, as we applied a normal map, the simulation result also takes into account the grey scale color lightness of the normal map.
	**Color from BSDF** Color from BSDF means that the simulation result uses the BSDF information of the texture mapping optical properties. In our context, as we applied a normal map, the simulation result also takes into account the grey scale color lightness of the normal map.
	**None** With None, the simulation results uses both the image texture and the texture mapping optical properties. The simulation result also takes into account the grey scale color lightness of the normal map.

Related tasks

Setting the Texture Normalization on page 103

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Texture application can have an impact on the simulations results. To control what is taken into account for simulation, a texture normalization mode must be selected.

5.2.8.2. Setting the Texture Normalization

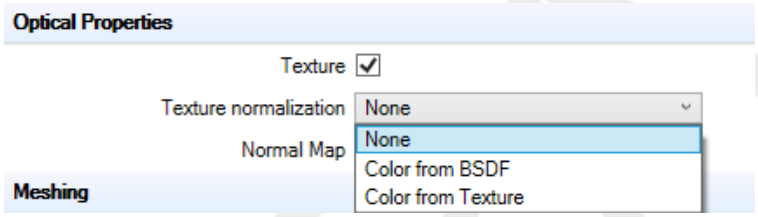
Texture application can have an impact on the simulations results. To control what is taken into account for simulation, a texture normalization mode must be selected.

> **Note:** You can only select one texture normalization mode for all texture mappings created in the entire assembly.

To define a Texture Normalization:

A simulation must already be created.

1. Open a simulation.
2. Right-click the simulation and click **Options**.
3. In the **Optical Properties** section, make sure **Texture** is checked.
   Texture is checked by default when creating a simulation.
4. From the **Texture normalization** drop-down list, select **None**, **Color from BSDF** or **Color from Texture**.



5. Click **Close**.

The texture normalization is selected and will be taken into account for simulation to determine the rendering of the texture.

Related concepts

Understanding Texture Normalization on page 102
This page helps to understand the texture normalization and how texture mapping modifies the interaction between light and faces for all kind of interactions.

Related information

Simulations Overview on page 362
Simulations allow you to give life to the optical system in order to generate results.

5.3. Polarization

5.3.1. Understanding Polarization

This page describes the polarization principle and the main application of it.

Overview

Polarization is an electromagnetic wave like radio, radar, X rays or gamma rays. The difference is a question of wavelength. A wave is something vibrating, in the case of a piano or a guitar, it is a cord. When talking about polarization, it is the orientation of the electromagnetic field of a propagating light wave (an electric and a magnetic field which are vibrating together).

The software only considers the electric field, as the magnetic field can be deduced from it in materials used for light propagation.

A polarization state is the geometrical trajectory followed by the electric field vector while propagating.

As the polarization state is elliptical, the polarization is defined by the azimuth angle of the ellipse, the ellipticity and its rotation sense.

> **Note:** Birefringent materials, polarizer surface or optical polished surfaces (Fresnel) use the polarization. Lambertian reflection is depolarizing. It converts polarized light into unpolarized light by changing randomly its polarization while processing the reflection. However, for Gaussian scattering, the model has been designed not to depolarize light.

Application

The polarization is the main physical property LCDs are working with. LCDs are used together with polarizer.

According to the applied voltage, they rotate or not the polarization axis of the light by 90°. So when this light tries to cross the polarizer, it is stopped (black state) in case of a 90° angle of the polarization axis with the easy axis of the polarizer or transmitted (white state) if this angle is 0°.

We saw that even the simplest surface quality (optical polished) has an effect on polarization. This is the surface quality used each time one deal with a light guide in automotive (dashboards) or in telephony (to lighten the keypad for example).

Since such devices are using multiple reflections inside their light guides, it is important to have an accurate model to describe the light behavior on this surface.

It is possible to build a light guide with a birefringent material to build a special function for the polarization. For example, a backlight for a LCD without any polarizer between the backlight and the LCD reducing the losses due to the polarizer.

5.3.2. Creating a Polarizer

This page shows how to create a polarizer thanks to the Speos features Material and UV Mapping.

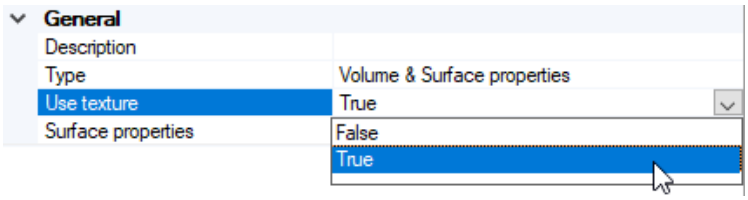
To create a polarizer:

1. From the **Design** tab, create the surface and the origin of the polarizer.
2. From the **Light Simulation** tab, create a material:
   a) click **Material**  .
   b) In the 3D view, click  , select the geometries on which to apply optical properties and click  to validate.
   The selection appears in **Geometries** as linked objects.
   c) In **General**, set the **Type** to **Volume & Surface properties**.
   d) Set **Use Texture** to **True**.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

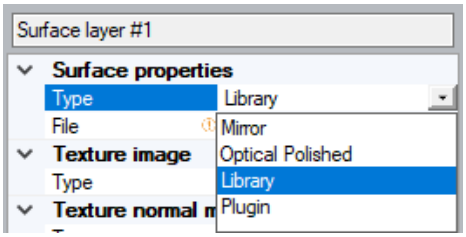
Published: 2023-10-23T14:08:14.04-04:00

A surface layer is created under the material in the simulation panel and allows you to define the surface properties of the geometry.



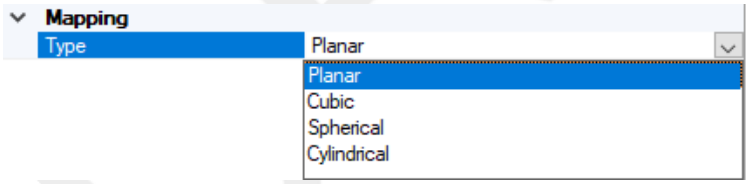
e) In **Volume properties**, set the **Type** to **Opaque** or **None**.

3. In the **Simulation** tree, open the **Surface Layer**.
4. In the surface layer, define the surface properties to be applied on the geometry:
   a) From the **Type** drop-down list, select **Library**.
   b) In **File**, double-click in the field to browse and load a .polarizer material.



> **Note:** You can create a *.polarizer file with the **Polarizer Surface Editor**.

5. Create a UV mapping:
   a) click **UV Mapping**  .
   b) In the 3D view, click  , select the geometry on which to apply the current mapping and click  to validate.
   c) In the first UV map, select the mapping **Type** you need.



   d) Define the **Origin**, **Projection Direction** and **Top Direction** according to the mapping type selected.

> **Note:** For more information, refer to Creating the UV Mapping.

The polarizer is created.

6: Local Meshing

Local meshing properties allow you to lighten memory resources by creating specific areas of focus on body parts.

6.1. Understanding Meshing Properties

This page describes the different parameters to set when creating a Meshing and helps you understand how meshing properties impact performance and result quality.

> **Note:** For same values of meshing, meshing results can be different between the CAD platforms in which Speos is integrated.

> **Note:** In case of a thin body, make sure to apply a fixed meshing sag mode and a meshing sag value smaller than the thickness of the body. Otherwise you may generate incorrect results.

Creating a meshing on an object, a face or a surface allows you to mobilize and concentrate computing power on one or certain areas of a geometry to obtain a better level of detail in your results. In a CAD software, meshing helps you to subdivide your model into simpler blocks. By breaking an object down into smaller and simpler pieces such as triangular shapes, you can concentrate more computing power on them, and therefore improve the quality of your results. During a simulation, it will no longer be one single object that interprets the incoming rays but a multitude of small objects.

Meshing Migration Warning

For files created in version 2021 R1 or before: if **Sag** / **Step** type was **Proportional**, the file is migrated to used **Proportional to Face size** in version 2022 R2.

For files created in versions 2021 R2 or 2022 R1: if **Sag** / **Step** type was **Proportional to Body size**, the file is migrated with the same settings in version 2022 R2.

For file created before version 2022 R2: if **Sag** / **Step** type was **Fixed**, the file is migrated with no change.

> **Warning:** if you created a file in version 2021 R1, then migrated to 2021 R2 and changed the values for **Sag** / **Step** type (when it became **Proportional to Body size**), these values may not be good in 2022 R2 when the document is migrated back to **Proportional to Face size**. You cannot know that the values were changed over the versions.

Meshing Mode: Fixed, Proportional to Face size Proportional to Body size

The meshing values can be set as Proportional to Body size, Proportional to Face size or Fixed:

- **Proportional to Face size** means that the tolerance adapts and adjusts to the size of each face of the object. The sag and maximum step size will, therefore, depend on the size of each face.
- **Proportional to Body size** means that the tolerance adapts and adjusts to the size of the object. The sag and maximum step size will, therefore, depend on the size of the body.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- **Fixed** means that the tolerance will remain unchanged no matter the size or shape of the object. The mesh of triangles will be forced on the object. The sag and maximum step size is, therefore, equal to the tolerance you entered in the settings.

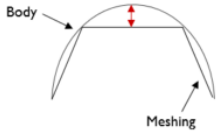
> **Note:** From 2022R2, the new default value is **Proportional to Face size**. Selecting between Proportional to Face size and Proportional to Body size may slightly affect the result according to the elements meshed.

> **Note:** When setting the meshing to **Proportional to Face size**, the results may return more faces than **Proportional to Body size**. These additional faces should be really small and they should not influence the ray propagation.

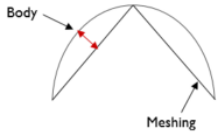
> **Note:** When running a simulation for the first time, Speos caches meshing information if the Meshing mode is **Fixed** or **Proportional to Body size**. This way, when you run a subsequent simulation and you have not modified the Meshing mode, the initialization time may be a bit faster than the first simulation run.

Sag Tolerance

The sag tolerance defines the maximum distance between the geometry and the meshing.



Small Sag Value



Large Sag Value

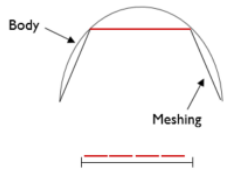
By setting the sag tolerance, the distance between the meshing and the surface changes. A small sag tolerance creates triangles that are smaller in size and generated closer to the surface. This will increase the number of triangles and potentially computation time. A large sag tolerance will generate looser triangles that are placed farther from the surface. A looser meshing can be used on objects that do not require a great level of detail.

> **Note:** If the **Meshing sag value** is too large compared to the body size, Speos recalculate with a **Meshing sag value 128** to better correspond to the body size.

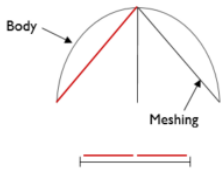
Maximum step size

> **Note:** For a Heavyweight body, the **Meshing step value** precision decreases when applying a value below 0.01mm.

The maximum step size defines the maximum length of a segment.



Small maximum step size



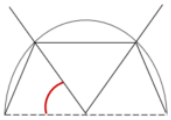
Large maximum step size

A small maximum step size generates triangles with smaller edge lengths. This usually increases the accuracy of the results.

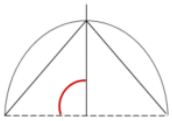
A greater maximum step size generates triangles with bigger edge lengths.

Angle Tolerance

The angle tolerance defines the maximum angle tolerated between the normal of the tangent formed at each end of the segments.



Small angle tolerance



Large angle tolerance

Related tasks

Creating a Local Meshing on page 108
Creating a local meshing allows you to identify areas requiring a high level of detail and optimize simulation time by creating a fine meshing on specific areas only.

6.2. Creating a Local Meshing

Creating a local meshing allows you to identify areas requiring a high level of detail and optimize simulation time by creating a fine meshing on specific areas only.

> **Note:** For same values of meshing, meshing results can be different between the CAD platforms in which Speos is integrated.

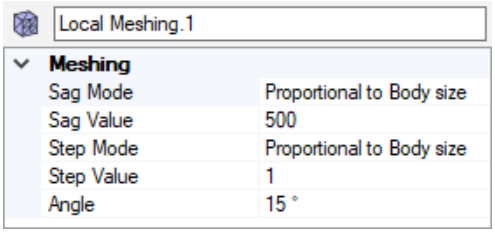
To create a Local Meshing:

> **Note:** Only one local meshing can be applied to a geometry.

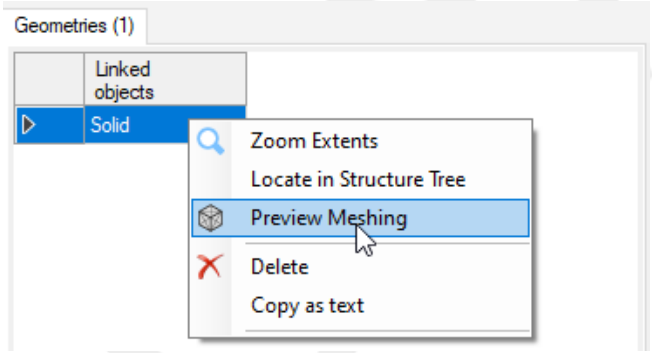
1. From the **Light Simulation** tab, click **Local Meshing**  .

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00



2. In the 3D view, click , select the geometries on which to apply optical properties and click to validate. The selection appears in the **Geometries** list as linked objects.
3. From the **Sag Mode** drop-down list:
   - Select **Proportional to Face size** to create a mesh of triangles that are proportional to the size of each face of the object. The sag and step value therefore depend on the size of each face.
   - Select **Proportional to Body size** to create a mesh of triangles that are proportional to the size of the object. The sag and step value therefore depend on the size of the body.
   - Select **Fixed** to create a mesh of triangles fixed in size regardless of the size of the body or faces.
4. In **Sag Value**, define the maximum distance between a segment and the object to mesh.
5. From the **Step Mode** drop-down list:
   - Select **Proportional to Body size** to create a mesh of triangles that are proportional to the size of the object.
   - Select **Fixed** to create a mesh of triangles fixed in size regardless of the size of the body.
6. In **Step Value**, define the maximum size of a triangle of the meshing.
7. In **Angle**, define the maximum angle tolerated between the normal formed at each end of the segments and the related tangents.
8. If you want to display the current meshing properties in the 3D view, from the **Geometries** list, right-click the geometry and click **Preview Meshing**.



The local meshing is created and applied to the selected geometries. The local meshing prevails over the simulation meshing properties for the selected geometries.

Related concepts

Understanding Meshing Properties on page 106
This page describes the different parameters to set when creating a Meshing and helps you understand how meshing properties impact performance and result quality.

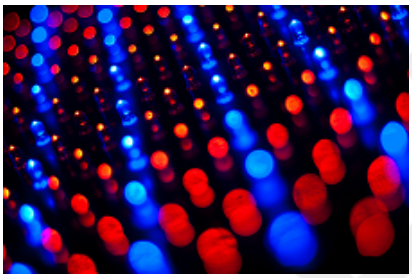
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

7: Sources

Sources allow you to virtually create and generate all kinds of light sources. The sources can be used to directly simulate and validate any lighting system or to simulate unwanted induced effects of light (glare for example).

7.1. Sources Overview

The Sources correspond to the light sources propagating rays in an optical system.


The light sources are a key component of optical simulation. Speos feature allows you to model these light sources and their interaction with an optical system.

A wide variety of sources are available to cover different needs and configurations. Speos allows you to model natural light or artificial light, LED's, streetlights, indoor lighting, back lighting, displays etc.

The Sources can be used to test, adjust or validate lighting systems themselves or can be used as a tool to analyze the induced effects of light.

Types of Light Sources

- The Surface Source allows you to model the light emission of a source taking into account its flux, spectrum, exitance and intensity. The Surface Source can suit numerous configurations.
- The Ambient Sources allows you to model and generate ambient natural light in a scene (sun, sky, stars). With Ambient Sources, you can also follow CIE standards or create background setups thanks to HDR images.
- The Display Source allows you to simulate the rendering of a lit display. You can visualize the image and the luminosity of the display after simulation.
- The Luminaire Source allows you to model artificial light for indoor or outdoor lighting. This source is mainly used to simulate street lighting.
- The Ray File Source allows you to model a light source thanks to a virtual source (rayfile), created from a measured light source. The ray file, therefore, contains information like direction, power or wavelength.
- The Interactive Source allows you to link two geometries to analyze the propagation of the rays that is made through an optical system. This source is useful to understand the behavior of a light beam in a given optical system.
- The Light Field Source allows you to read the optical light field file *.olf generated by the Light Field sensor to emit light.

In Speos

You can manage all the characteristics of the light source: its power, spectrum, emission pattern etc. To create a light source you have to define several parameters: (The parameters vary based on the type of source.)

DRAFT

- The **power** of the source or flux can be defined from the interface or inherited from a file (e.g from a ray file).
- The **spectrum** of the light source can either be defined from the interface or downloaded from the Ansys Optical Library or created with the Spectrum Editor.
- The direction of the emission or **intensity distribution** can be defined from the interface thanks to standard emission patterns or inherited from an intensity distribution file that contains information about the intensity profile of the source, that is to say how the source redistributes light in space. These files can be downloaded from the Ansys Optical Library.

Related concepts

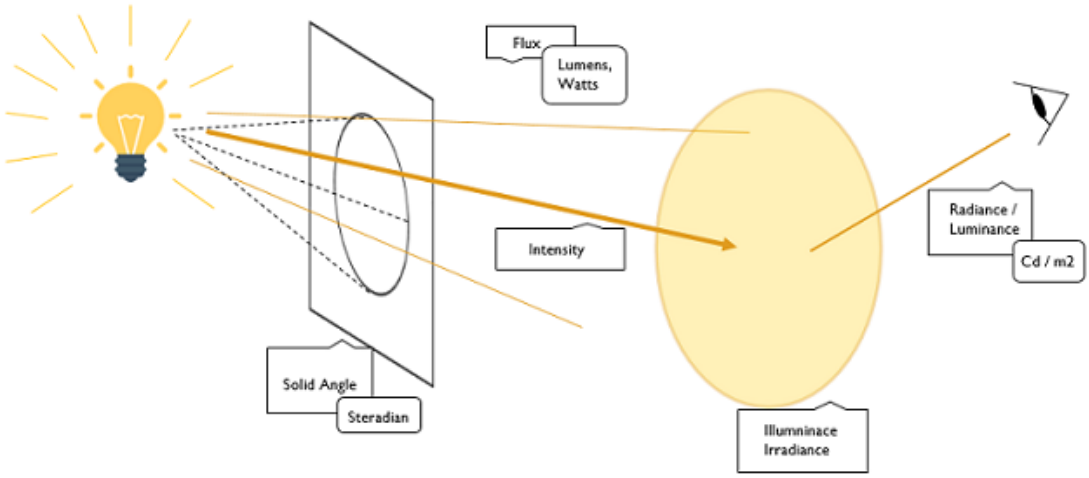
Introduction to Optics on page 111
This page describes core optical principles.

Related information

Sources Creation on page 114
You can create all kinds of light sources thanks to different sources types.

7.2. Introduction to Optics

This page describes core optical principles.



A light source has several characteristics and is defined by:

- Its emissivity / its power: the flux
- Its intensity
- Its spectrum

The Flux

The flux corresponds to the total energy emitted by a light source.

In photometry, as the light emits within the visible spectrum, then the flux is referred to as a **luminous flux** (expressed in lumens).

> **Note:** In case of a photometric result generation, the International Commission on Illumination (CIE) defines the visible spectrum as follows: "There are no precise limits for the spectral range of visible radiation since they depend upon the amount of radiant flux reaching the retina and the responsivity of the observer. The lower limit is generally taken between 360 nm and 400 nm and the upper limit between 760 nm and 830 nm".

In radiometry, the energy emitted by the light source is referred to as the **radiant flux** (expressed in watts).

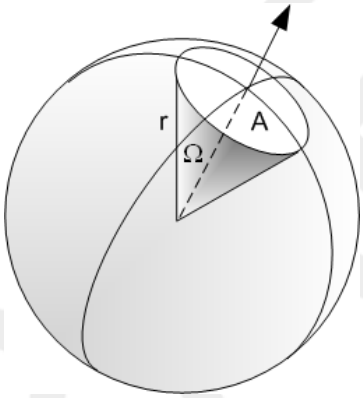
The Intensity

The intensity is the power emitted by a light source in a particular direction per unit solid angle.

The unit of measure is the candela. 1cd = 11m/sr.

The radiant intensity is expressed in watts per steradian ($W.sr^{-1}$.)

The luminous intensity is expressed in lumens per steradian ($lm.sr^{-1}$.)

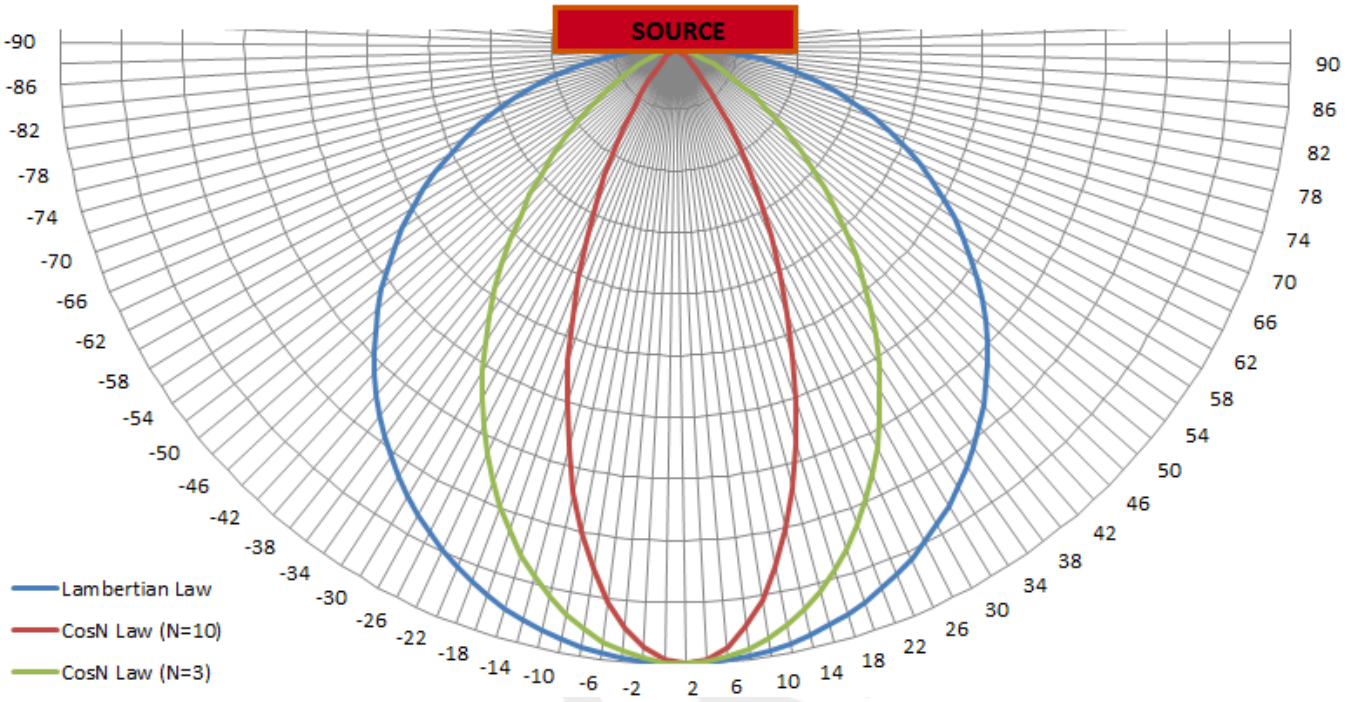


Solid Angle = Field of view from a specific point covered by an observer or an object.

DRAFT

The Intensity Distribution

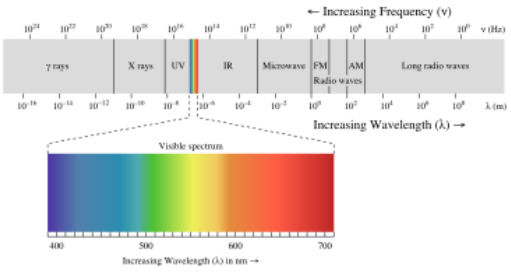
Figure 14. A source emitting on an hemisphere with different intensity distributions.



The intensity distribution describes the emission direction of a light source.

- A lambertian emission ensures that the source has a uniform distribution. The source theoretically distributes the same amount of light in every direction. The source has therefore the same luminance whatever the observation angle is.
- With a Cos distribution, the intensity follows the cosine law. The higher the intensity, the narrower the intensity diagram will appear. You can modify the order of the law to make the rays converge or diverge.
- A gaussian distribution follows a gaussian function and can be symmetric or asymmetric.
- Intensity files are data measured files that provide an accurate intensity profile.

Spectrum



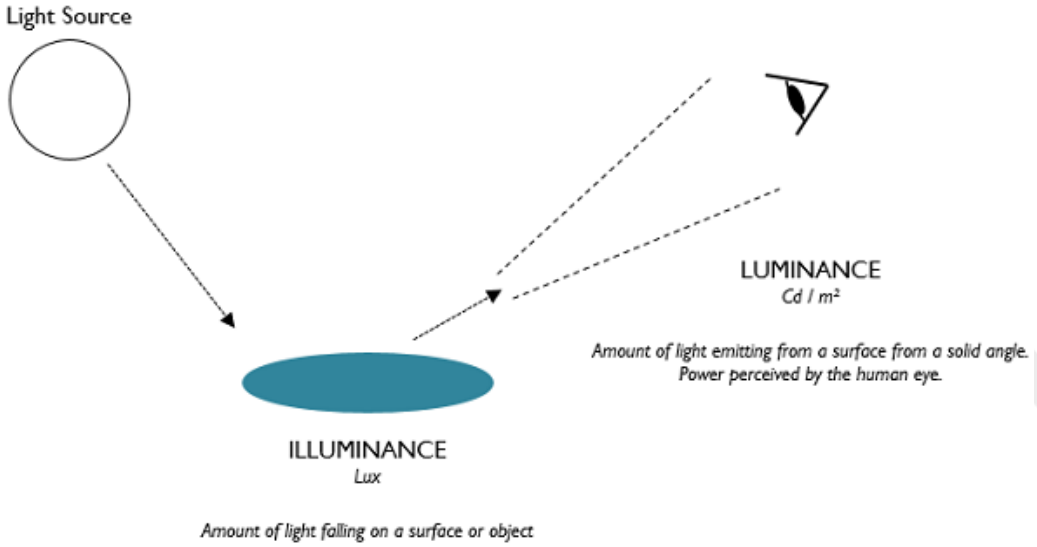
The entire light spectrum encompasses:

- The visible spectrum: what the human eye can perceive (usually 390nm to 700nm).
- The infrared
- The ultraviolets

Luminance / Illuminance

Luminance or radiance is the amount of light coming from a surface that the human eye perceives.

Illuminance or irradiance is the amount of visible energy falling upon an object's surface.



Related information

Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

Sources Creation on page 114
You can create all kinds of light sources thanks to different sources types.

7.3. Sources Creation

You can create all kinds of light sources thanks to different sources types.

7.3.1. Creating an Interactive Source

An Interactive Source generates specific and monochromatic light rays which are useful to understand the behavior of a light beam through an optical system.

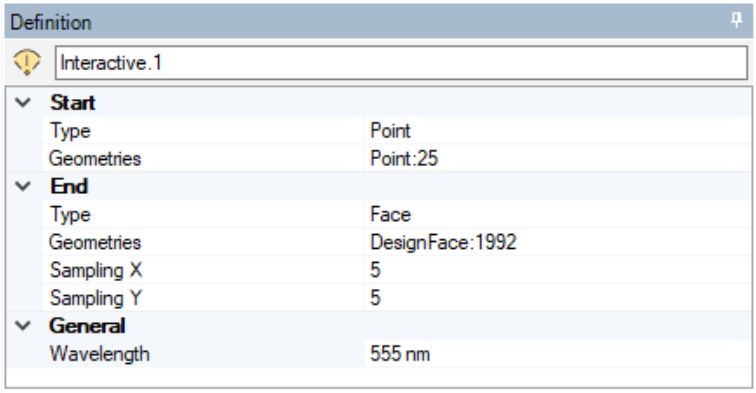
> **Note:** The purpose of the interactive source is not to model the emission of a real light source (like a LED or a filament), but to generate specific light rays that can help dimension an optical system but never validate it. The Interactive Sources are generally created to be used in an Interactive Simulation.

To create an Interactive Source:

You must have two geometries, one for the propagation's start and one for the propagation's end.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

1. From the **Light Simulation** tab, click **Interactive**  .



2. From the **Type** drop-down list, select the start geometry's type.
3. Click  to select the start geometry in the 3D view:
   - For a **Point**, select a point.
   - For a **Curve**, select a line or an edge and specify the sampling (number of rays) on X axis.
   - For a **Face**, select a face and specify the sampling on X and Y axes.
   - For a **Direction**, select a line or a body edge.
4. From the **Type** drop-down list, select the end geometry's type.
5. Click  to select the end geometry in the 3D view.

   A preview of the rays appear in the 3D view.
6. Edit the Wavelength value according to the light you want to simulate.

*The interactive source is created and appears in the **Simulation** tree and in the 3D view.*

Related concepts

Introduction to Optics on page 111
This page describes core optical principles.

Related information

Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

7.3.2. Creating a Ray File Source

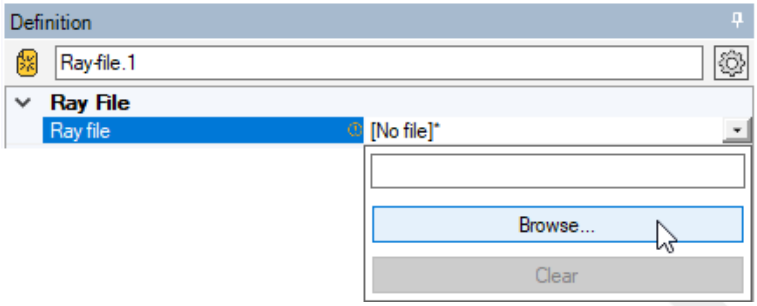
A Ray File Source emits specific rays according to a pre-calculated source (a .ray file) that contains information regarding the position, direction and wavelength of each ray.

DRAFT

Note: Standard ray file format (.ray), standard IES TM25 ray file format (.tm25ray), as well as LightTools® and TracePro® ray file formats (.ray) are compatible with the Ray File Source and can be used to describe the emission of a light source.

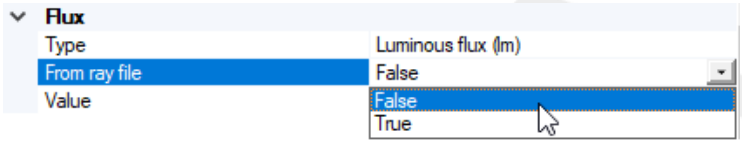
To create a Ray File Source:

1. From the **Light Simulation** tab, click **Ray-file**  .
2. To import the ray file, click in the file field and click **Browse** to select a .ray or a .tm25ray file.



The file is imported and the flux is inherited from the file.

3. If you do not want to inherit the flux values from the ray file:



a) From the drop-down list, select **False**.
b) From the **Type** drop-down list, define the flux as luminous or radiant.

> **Note:** Flux expressed in Watt **(w)** is the radiant energy or radiant power of a light-emitting source.
>
> Flux expressed in Lumen **(lm)** is the luminous flux or luminous power of a light-emitting source.

c) In **Value**, specify the luminous flux (lumens) or electric power (watts) of the source.

> **Note:** If only one flux type (radiometric or photometric) is available in the ray file, you cannot select another flux type.
>
> If you define an old ray file that does not contain values in lumen, you cannot change the flux unit. To convert the file to a more recent file format, use the **Ray File Editor** to get values in lumen.
>
> When loading a ray file, it may not be optimized. In this case, click **Optimize ray file for Speos simulation**. If the ray file does not contain any spectrum information, the option will not display until you define the spectrum of the source.
>
> Ray files generated during a Direct Simulation are automatically optimized.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

4. In the 3D view, set the **Axis System** of the source by clicking  to sequentially select one point for the origin and two lines for X and Y axes or click  and select a coordinate system to autofill the Axis System.

Axis System	
Origin	[Default] Origin
X direction	[Default] X Axis
Reverse direction	False
Y direction	[Default] Y Axis
Reverse direction	False

The X and Y directions define the propagation direction of the rays.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

If you need to adjust the ray's propagation direction, set **Reverse direction** to **True**.

5. If the ray file does not contain any spectrum information, define the **Spectrum** of the source:
   - Select **Blackbody** to set the temperature of the source in kelvins.
   - Select **Library** and from the drop-down list, click **Browse** to load a .spectrum file.

     If you want to see the file properties or edit the file, from the drop-down list, click **Open file** to open the **Spectrum Editor**.

6. If you want to associate geometries to the ray file source, in the 3D view click the face(s) to be considered as the exit geometry of the source.

Geometry	
Exit geometry	DesignFace:2016

7. In **Optional or advanced settings**, adjust the **Number of Rays and Length of rays** to display in the 3D view.

The Ray File Source is created and visible both in the **Simulation** panel and in the 3D view.

> **Tip:** Sometimes, to save simulation time, it is useful to split a simulation in two parts. The first simulation can be dedicated to simulate the light propagation in parts with a definitive design (for instance the filament, the bulb and the socket of a lamp). The second simulation can be dedicated to simulate the light propagation in parts currently in the design process (for instance a reflector). You can create a Ray File source with a ray file generated by the first simulation. Then, you can use the ray file source to replace the first part of the optical system in the second simulation. At each simulation done to optimize the second part of the optical system, the simulation time dedicated to the ray propagation in the first part is saved. Generally, with this tip, you can save between 20% and 80% of the simulation time.

Related concepts

Introduction to Optics on page 111
This page describes core optical principles.

Related information

Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

7.3.3. Light Field Source

A Light Field Source reads the optical light field file *.olf generated by the Light Field sensor to emit light during an inverse simulation run.

> **Note:** The Light Field feature is in BETA mode for the current release.

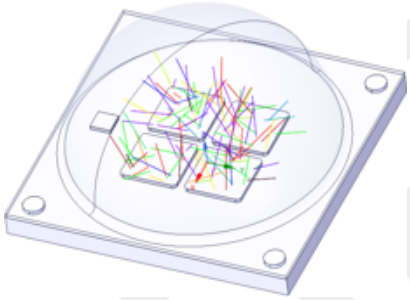
7.3.3.1. Light Field Overview

The Light Field allows you to create a propagation result of a sub-optical system to be reused in a more complex optical system to gain time when computing simulation.

> **Note:** The Light Field feature is in BETA mode for the current release.

Optical systems can be composed of sub-optical systems. When you focus on the main optical system, recalculating every time the propagation inside those sub optical systems can be time-consuming. This time can be optimized: the goal is to speed up the simulation by pre-computing the propagation of the sub-optical systems.

To proceed to the pre-calculation of those sub-optical systems, the Light Field feature generates a *.olf (Optical Light Field) file format thanks to a Light Field sensor, that is then used as a Light Field source in the main optical system. Thus, the simulation does not have to compute the propagation of the sub-optical system, reducing the simulation time.

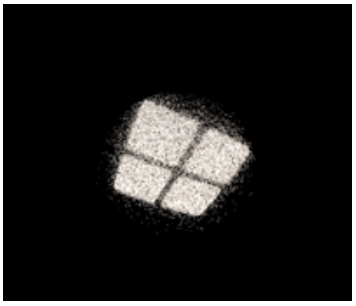


LED including chips with a lens on top



Original Radiance Simulation of the LED

Reference time = T



Radiance Simulation of the Light Field representing the LED

Simulation time = 0.43 * T

General Workflow

1. Create a Local Meshing of the surface to be integrated in the Optical Light Field file.

   -Or-

   At Step 3, use the Meshing Options of the Direct Simulation to generate the meshing.

   > **Note:** As no optical properties are defined on a Light Field meshing, the Light Field is fully absorbing.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

2. Create a Light Field Sensor to define how the *.olf file will be generated.
3. Create a Direct Simulation, and if no Local Meshing is applied on the Light Field surface, define the Meshing Options.
4. Run the Direct Simulation to generate the *.olf file.
5. Create a Light Field Source that uses the generated *.olf file as input.
6. Create and run an Interactive, Direct or Inverse Simulation of the main optical system, using the Light Field Source as input.

7.3.3.2. Understanding the Parameters of a Light Field Source

This page describes the parameters to set when creating a Light Field Source.

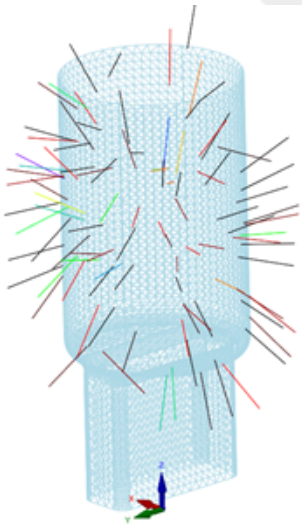
> **Note:** The Light Field feature is in BETA mode for the current release.

Display Meshing Mode

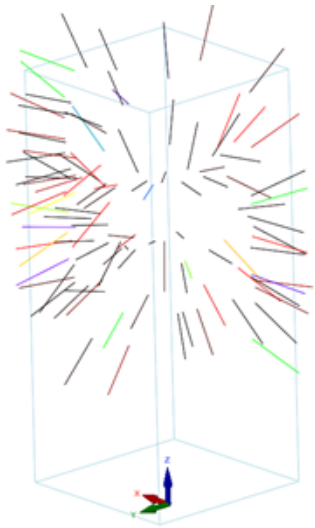
The Light Field Source can be represented in several ways in the 3D view.



None: does not display information. Only rays are visible.



Meshing: displays the meshing according to the Meshing settings defined.



Bounding Box: displays the bounding box of the source without the meshing.

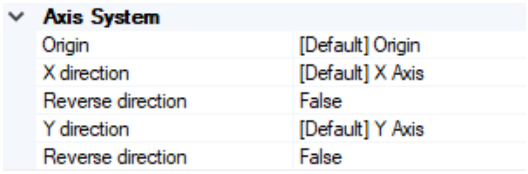
7.3.3.3. Creating a Light Field Source

A Light Field Source reads the optical light field file *.olf generated by the Light Field sensor to emit light.

> **Note:** The Light Field feature is in BETA mode for the current release.

To create a Light Field Source:

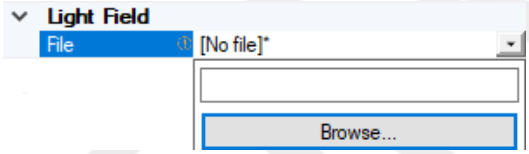
1. From the **Light Simulation** tab, click **Light Field**  .
2. In the 3D view, define the **Axis system** of the Light Field source.



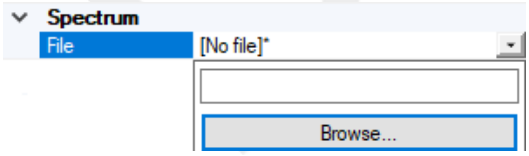
- 
  to select an origin point.
- 
  to select a line defining the horizontal direction.
- 
  to select a line defining the vertical direction.
- or click
  
  and select a coordinate system to autofill the Axis System.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

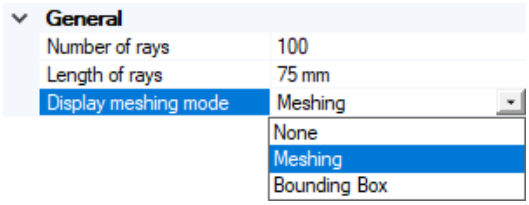
3. From the **Light Field file** drop-down list, click **Browse** to load an optical light field file *.olf.



4. If the selected optical light field file contains radiometric or photometric data, in **Wavelength**, select the a spectrum file.



5. In **Optional or advanced settings**  :



Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- adjust the **Number of rays** and **Length of rays** to display in the 3D view.
- define the **Display meshing mode** to display in the 3D view.

The Light Field Source is created and appears in the **Simulation** panel and in the 3D view.

Create and run an Interactive, Direct or Inverse Simulation containing the Light Field Source to benefit from the Light Field.

7.3.4. Surface Source

A Surface Source can be created on any face of a geometry and uniformly emits light on each point of a surface.

7.3.4.1. Understanding the Parameters of a Surface Source

This page describes the parameters to set when creating a Surface Source.

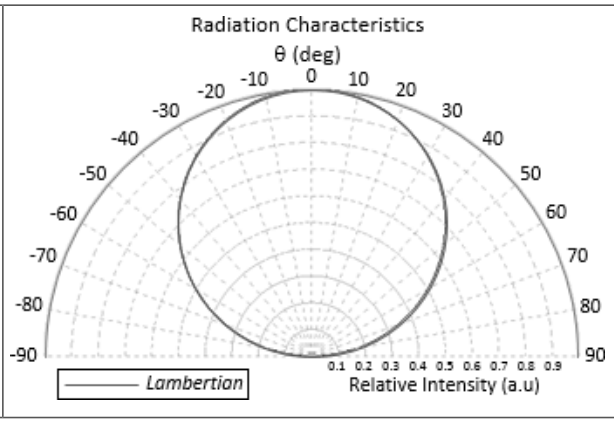
Intensity Distribution

The Intensity Distribution describes the emission pattern of a light source. You can choose among different distribution profiles:

- A **Lambertian** emission ensures that the source has a uniform distribution. The source theoretically distributes the same amount of light in every direction and has, therefore, the same luminance whatever the observation angle is.
- With a **Cos** distribution, the intensity follows the cosine law. The higher the intensity, the narrower the intensity diagram will appear. You can modify the order of the law to make the rays converge or diverge.
- A gaussian distribution follows a gaussian function and can be symmetric or asymmetric.
- Intensity files are data measured files that provide an accurate intensity profile. The supported formats are:
  - iesna (.ies);
  - eulumdat (.ldt);
  - XMP maps with conoscopic intensity (.xmp).

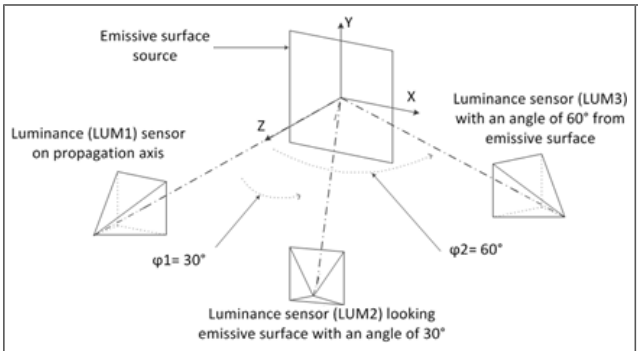
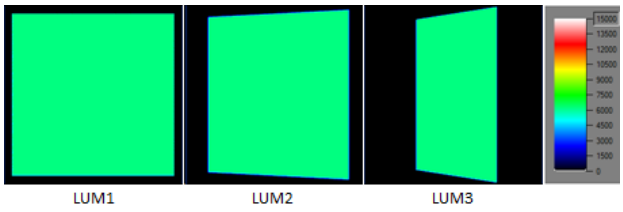
Lambertian Distribution

A lambertian source evenly distributes light in every direction of the half space. The deflection angle(q) corresponds to the total angle of emission of the light source.

I = A* cos(q) A: Intensity in propagation axis q: Deflection angle	
Radiation laws and relative intensity diagram, characteristic of a lambertian source emitting on a half sphere.	

DRAFT

A source with a lambertian distribution has the same luminance whatever the observation angle is, as illustrated below:

	
Set-up of emissive source with three radiance sensors.	Radiance map of the lambertian source set-up above. The Luminance is constant no matter the angle of observation.

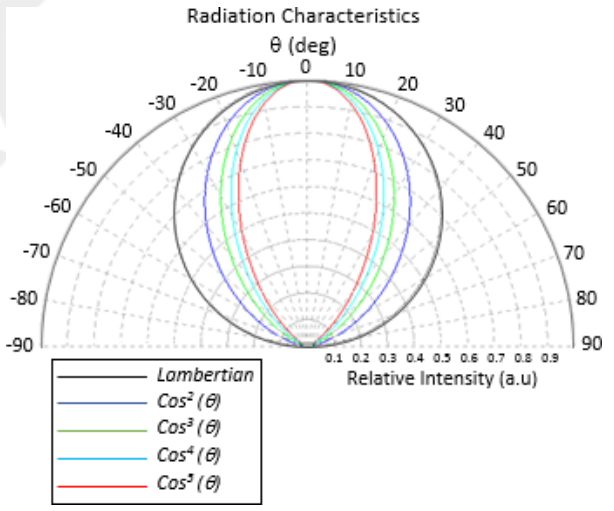
Cos - Lambert's Cosine Law

The Lambert's cosine law basically states that the illumination of a surface is proportional to the cosine of the angle between the direction of the incident light and the surface normal.

The cos distribution follows a cosines law at nth order.

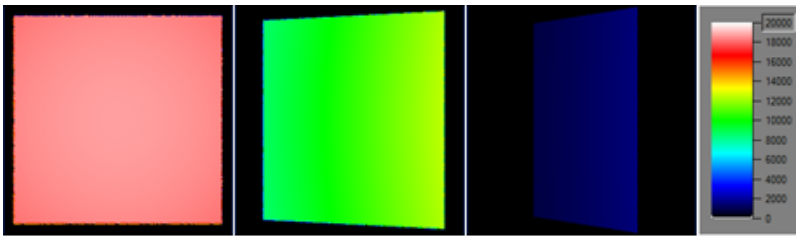
The **Total Angle** is the maximum angle of emission of the light source.

The **N** parameter sets the order of the cosines law.

$I = A^* \cos^n(q)$ A: Intensity in propagation axis q: Deflection angle n: Order of cos law	
Radiation laws of cos function.	Radiation diagram at 2nd, 3rd, 4th and 5th order compared to a lambertian distribution. The higher the intensity, the narrower the intensity diagram will appear.

A source with cos distribution has a luminance varying according to the observation angle, as illustrated below:

DRAFT

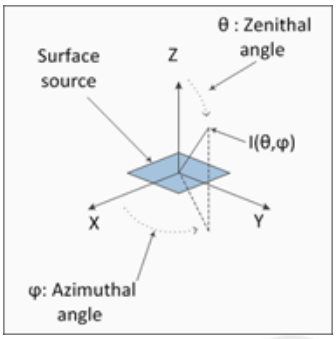


Radiance map of the Cos source

Gaussian

The intensity distribution of a source can follow a gaussian distribution.

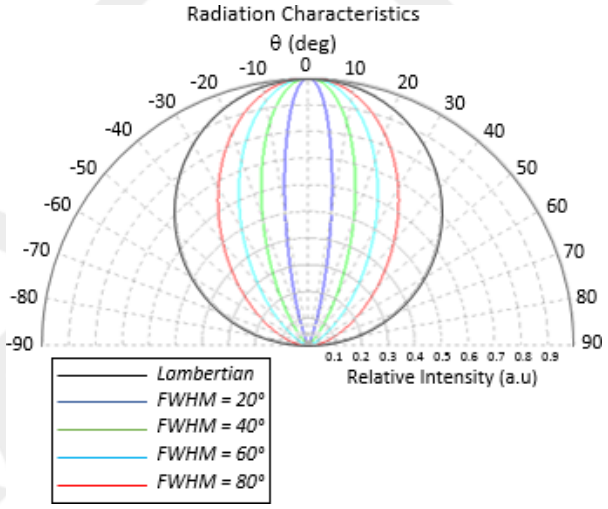
The **Total Angle** defines the angle of emission of the light source.



$$I = exp[-(\frac{x^2}{2 \times \sigma_x^2} + \frac{y^2}{2 \times \sigma_y^2})$$

$$where:$$
$$x = \theta \times \cos\phi$$
$$y = \theta \times \sin\phi$$

$$\sigma_x = \frac{FWHM_x}{2 \times \sqrt{2\ln 2}}$$
$$\sigma_y = \frac{FWHM_y}{2 \times \sqrt{2\ln 2}}$$



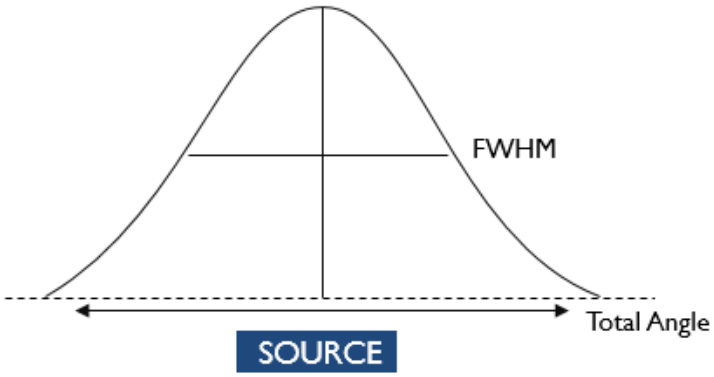
Gaussian distribution laws and relative radiations distribution of gaussian compared to a lambertian distribution.

FWHM

The Full Width At Half Maximum (**FWHM Angle**) is used to describe the width of a curve at half its maximum amplitude. It means that the source reaches half its power potential between (0°) the normal of the emitting surface and the FWHM.

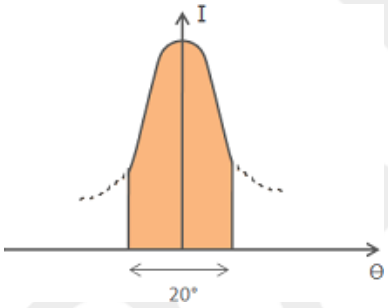
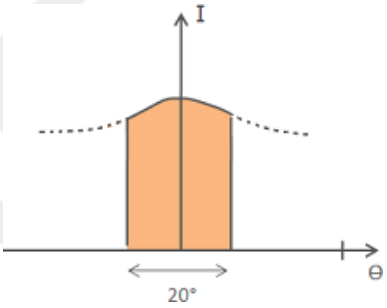
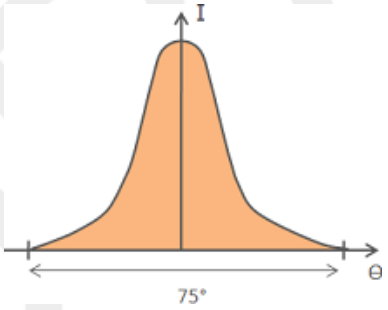
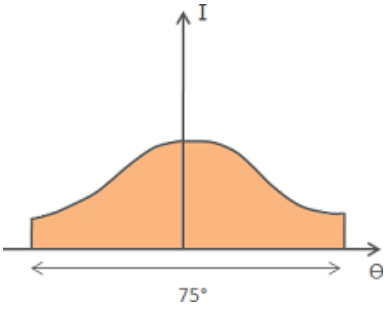
It allows you to alter the emission profile of the light source.

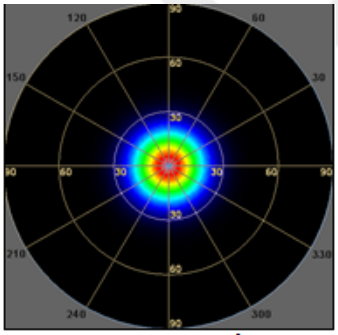
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



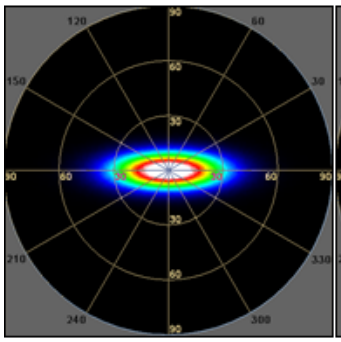
As illustrated below, a small FWHM value tends to restrain and concentrate the light beam. A large FWHM value results in a broader, more widespread light emission. If the source is symmetric, then the FWHM Angle is the same on both axes.

If the source is asymmetric, the FWHM Angle can be edited on X and Y.

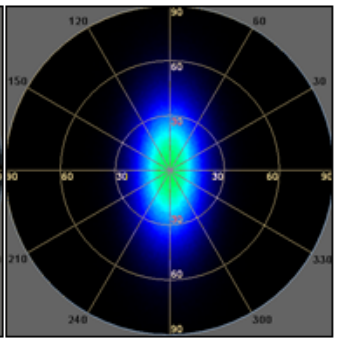
	FWHM = 15°	FWHM = 45°
Total Angle = 20°		
Total Angle = 75°		



FWHM_x = 30°
FWHM_y = 30°



FWHM_x = 45°
FWHM_y = 15°



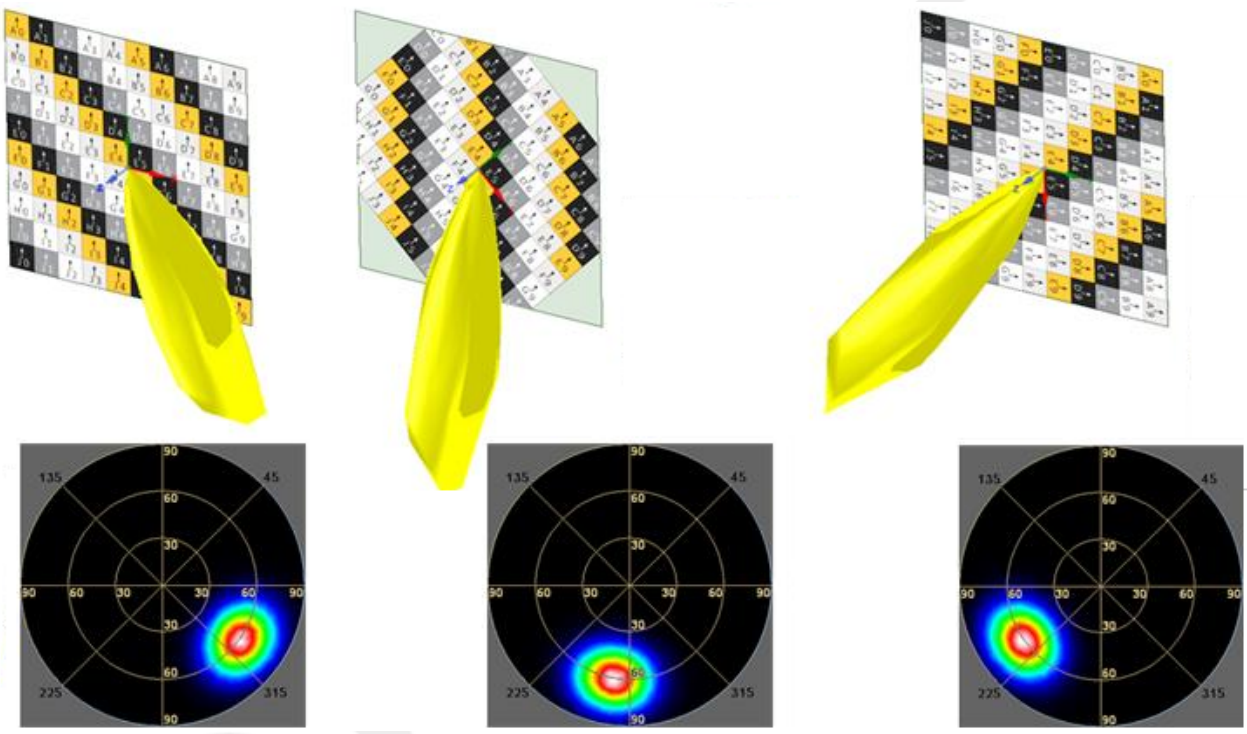
FWHM_x = 30°
FWHM_y = 60°

Library - Normal to UV Map

Normal to UV map allows you to define the intensity distribution as normal to the selected emissive surface and its orientation on the emissive surface.

Normal to UV map is particularly useful in case of an asymmetrical intensity distribution as it allows you to define accurately its orientation on the surface.

> **Important:** For the Normal to UV map to work you need to create a Texture Mapping on the emitting face. Refer to the Surface Source procedure for more information on how to create a Texture Mapping.



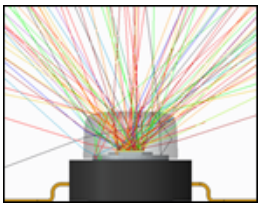
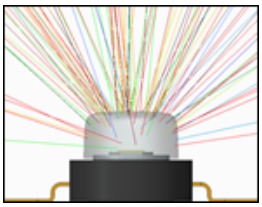
Exit Geometries

The exit geometries represent the geometries present during source measurement (the bulb of a light bulb or the case of a LED) that could potentially influence the optical behavior or intensity distribution of the source.

Selecting exit geometries allows you to define a new emissive geometry to avoid recalculation of the geometry's effect on the source.

For example, if you selected an iesna file (.ies) corresponding to a light bulb, the bulb geometry is taken into account in the data of the iesna file. To avoid the recalculation of the light bulb's effect, the bulb must be selected as the exit geometry.

> **CAUTION:** When using a Surface Source in a CPU simulation, rays from the Exit Geometries are only propagated forward according to the emitted rays from the Surface Source.

	
Intensity distribution without specific exit geometry.	Intensity distribution with lens defined as exit geometry.

Related tasks

Creating a Surface Source on page 126
The Surface Source models the light emission of a source taking into account its physical properties as the flux, the spectrum, the emittance and the intensity.

Related information

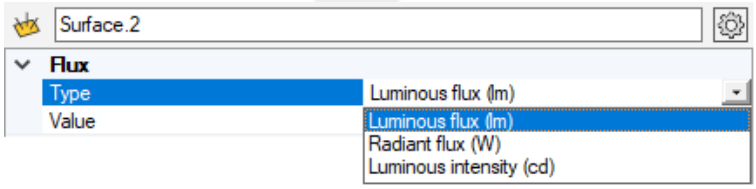
Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

7.3.4.2. Creating a Surface Source

The Surface Source models the light emission of a source taking into account its physical properties as the flux, the spectrum, the emittance and the intensity.

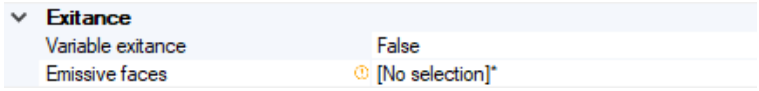
To create a Surface Source:

1. From the **Light Simulation** tab, click **Surface**  .
2. In the 3D view, click  , select the exitance/emissive face and click to  validate.
3. According to the flux you want to define, from the **Type** drop-down list:



   - Select **Luminous Flux (lm)** and define its value in lumens.
   - Select **Radiant Flux (W)** and define its value in watts.
   - Select **Luminous Intensity (cd)** and define its value in candelas.

4. In the Exitance section:



   - Leave the **Variable exitance** set as **False** to have a constant ray energy over the surface source and select the emissive face(s) in the 3D view.

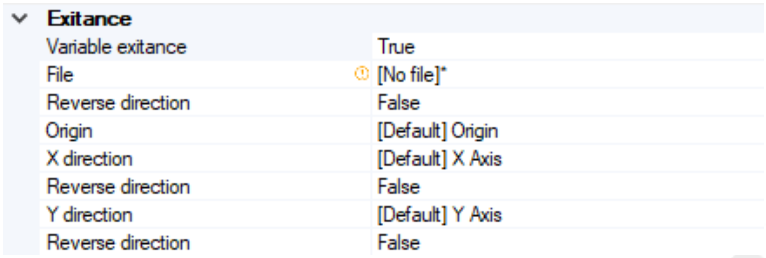
     The face(s) appear in the **List of Selected Objects**. A preview of the ray's distribution and color appears in the 3D view.

DRAFT

If you need to adjust the ray's propagation direction, check **Reverse normal**.

> **Note:** The selection of faces from an imported *.obj file is not compatible with the Surface Source.

- Set the **Variable exitance** as **True** to have a variable ray energy over the surface source depending on the xmp energy distribution.



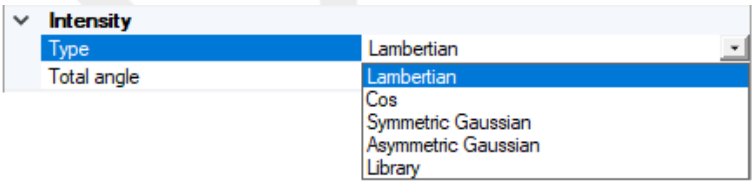
  - Click in the **File** field and click **Browse** to load an xmp file.

    If you want to see the file properties or edit the xmp map, click the file's drop-down list and click **Open file**.
  - If you need to adjust the ray's propagation direction, set **Reverse Direction** to **True**.
  - Set the coordinate system of the XMP map by clicking one point for the origin point and two lines for X and Y axes or click  and select a coordinate system to autofill the Axis System.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

> **Note:** If you selected a spectral map, you cannot set another spectrum.
>
> If you selected a non-spectral map, you need to specify a spectrum type.

5. In **Intensity**, set the intensity distribution of the light source. From the **Type** drop-down list:



- Select **Lambertian** for a uniform distribution and set the total angle of the surface source's emission.

> **Note:** By default **Total angle** is set to 180° so that the source emits on a hemisphere. Output light is set to 0 cd, for deflection angles (q) bigger than half the total angle.

- Select **Cos** for a distribution that follows a cosine law at n^{th} order and set the total angle of the surface source's emission.
  - In **N**, set the order of the cosine law.
  - In **Total Angle** set the angle of emission of the source.
- Select **Symmetric Gaussian**:
  - Set the total angle of emission of the source.
  - Set the **FWHM angle**.

    FWHMAngle has the same value for x and y and is computed on both axes.
- If you want to define different FWHM values on X and Y, select **Asymmetric Gaussian**:
  - Set the total angle of emission of the source.
  - Set the **FWHM Angle** for X and Y.
  - In the 3D view, click two lines to define X direction and Y direction.

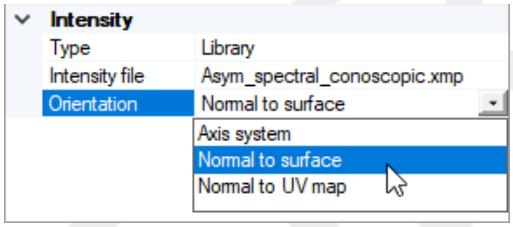
> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

- Select **Library** and click **Browse** to load an intensity file.

  If you want to see the file properties, click **Open file** to open the **IESNA Viewer**.

  If you want to inherit the flux value from the intensity file, in the **Flux** section, set **From intensity file** field as **True**.

6. If you selected **Library**, set the orientation of the source intensity distribution:



- select **Axis system** and click two lines to define X and Y direction.
- select **Normal to surface** to define the intensity distribution as normal to the selected surface.
- select **Normal to UV map** to define the intensity distribution as normal to the selected emissive surface and its orientation on the emissive surface.

  Normal to UV map is particularly useful in case of an asymmetrical intensity distribution as it allows you to define accurately its orientation on the surface.

> **Note:** When **Normal to UV map** is used, the intensity distribution preview is defined as **Normal to Surface** due to performance issue.

DRAFT

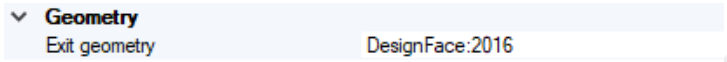
Important: For the Normal to UV map to work you need to create a Texture Mapping on the emitting face:

a. Create a UV map to apply on the emitting face of the surface source.
b. Create and apply a FOP or a SOP material with Use Texture set to True on the emitting face.
c. As Use Texture is activated, define at least one Surface Layer with a Texture Image on the emitting face.

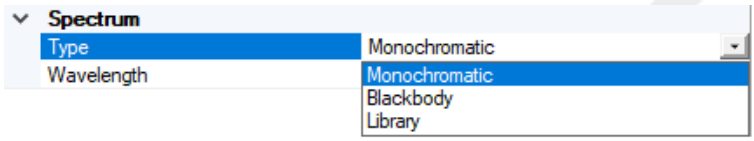
Any image is appropriate. The image is required only to consider the UV map on the surface.

d. In the Simulation options (Interactive, Direct, Inverse), in the Geometry tab, check Texture.

7. If you selected **Library**, you can select exit geometries by clicking them in the 3D view.



8. In **Spectrum**, from the **Type** drop-down list:

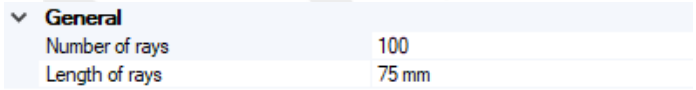


- Select **Monochromatic** and specify the wavelength in nanometers.
- Select **Blackbody** to set the temperature of the source in kelvins.
- Select **Library** and from the drop-down list click **Browse** to load a .spectrum file.

If you want to see the file properties or edit the file, click **Open file** to open the **Spectrum Editor**.

Note: If you select a XMP map with spectral conoscopic intensity, the spectral information of the map is displayed in the **Spectrum** group box.

9. In **Optional or advanced settings**:



- Adjust the **Number of rays** to display in the 3D view.
- If needed, adjust the **Length of rays** to display in the 3D view.

10. If you are using a variable exitance, click **Compute** to apply the XMP file to the surface source.

The Surface Source is created and should appear in the 3D view and in the **Simulation** panel.

Related concepts

Introduction to Optics on page 111
This page describes core optical principles.

Understanding the Parameters of a Surface Source on page 121
This page describes the parameters to set when creating a Surface Source.

Related information

Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

7.3.5. Display Source

The Display Source allows you to simulate the rendering of lit displays.

7.3.5.1. Understanding the Parameters of a Display Source

This page describes the parameters to set when creating a Display Source.

Contrast Ratio

The **Contrast Ratio** is a characteristic of any display. It corresponds to the ratio of the luminance of the brightest pixel (white color) to that of the darkest pixel (black color).

The higher the contrast ratio, the better the colors will appear.

An **Infinite Contrast Ratio** considers the brightest pixel at 255 255 255 and the darkest pixel at 0 0 0.

Standard contrast ratios range from 500:1 to 1000:1 for a LCD.

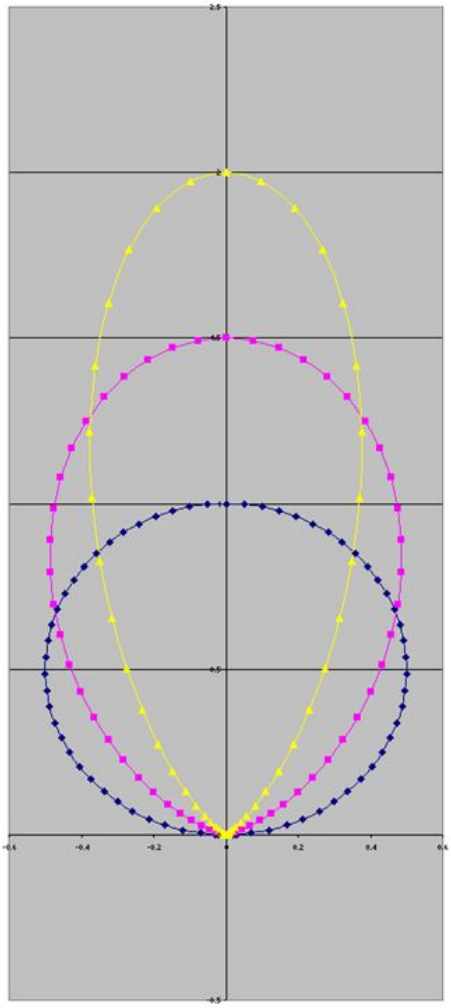
Intensity Distributions

> **Note:** When using an intensity file, the maximum value of the intensity diagram is always considered at the normal incidence. However if the real maximum value of the intensity diagram is not located at the normal incidence, you may face with unexpected measured values.

The Intensity Distribution describes the emission pattern of a light source. You can choose among different distribution profiles.

The following image shows the intensity diagram for a Lambertian law (blue curve), a Cosnθ law (purple curve) and a Gaussian law (yellow curve).

DRAFT



Lambertian

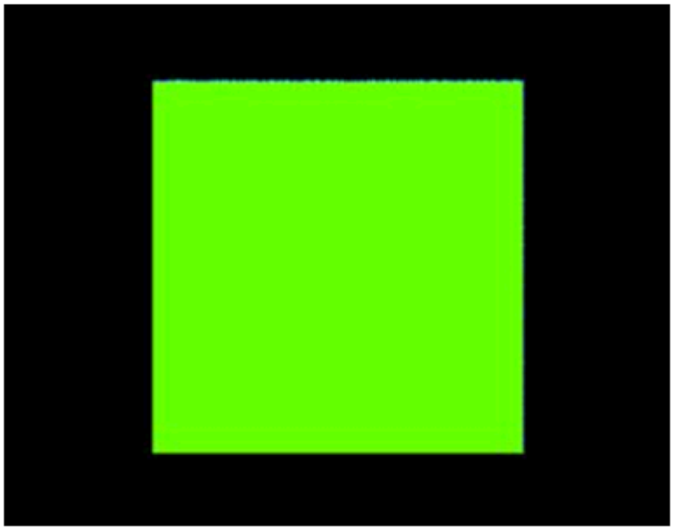
The simplest model of light distribution is the lambertian model. This model ensures that the source has a uniform distribution and equal probabilities of distribution in every direction.

The **Total angle** of emission of the source is by default set to 180° so that the source emits on a hemisphere.

The intensity formula for Lambertian is I = cos(theta). Cos: I = cosn(theta).

DRAFT

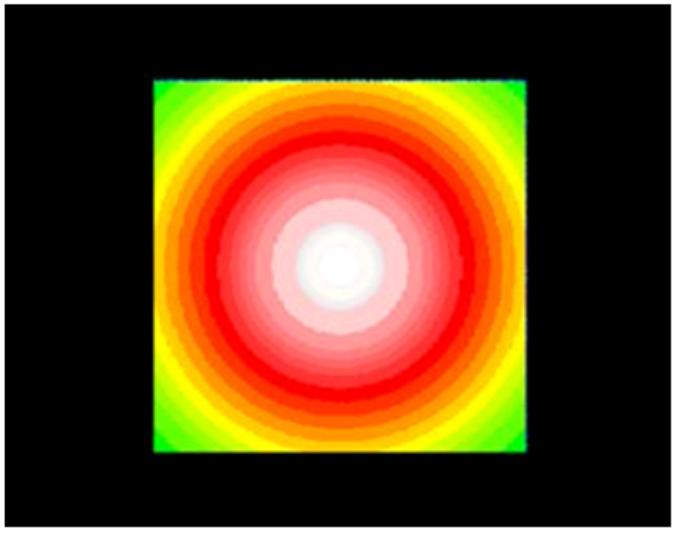
Figure 15. Luminance map of a source with a lambertian distribution



Cos

A **Cos** distribution follows a cosines law (Lambert's Cosine law). With a Cos distribution, you can modify **N** (order of the cosine law) to alter the intensity so that the rays converge or diverge.

Figure 16. Luminance map of a source with Cos distribution



Gaussian

A gaussian distribution follows a gaussian function and can be symmetric or asymmetric.

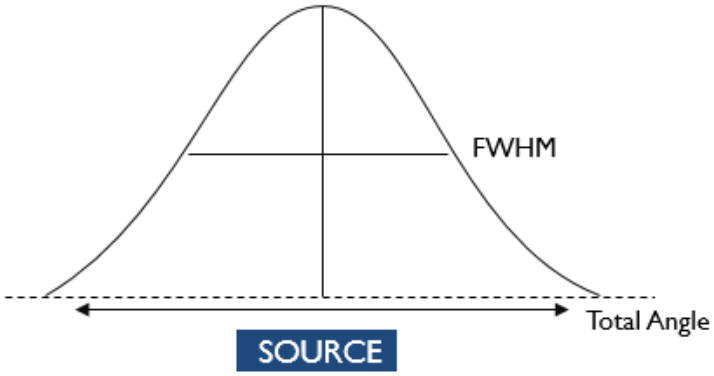
The intensity formula for **Gaussian** is I = exp(-(theta/a)^2). **a** is calculated in a way that the FWHM (Full Width at Half Maximum) angle of the Gaussian is the one given by the user.

The Full Width At Half Maximum (**FWHM Angle**) is used to describe the width of a curve at half its maximum amplitude. It means that the source reaches half its power potential between 0° and the FWHM you define.

It allows you to alter the emission profile of the light source.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

A small FWHM value tends to restrain and concentrate the light beam. A large FWHM value results in a broader, more widespread light emission.



If the source is **symmetric**, then the FWHM Angle is the same on both axes.

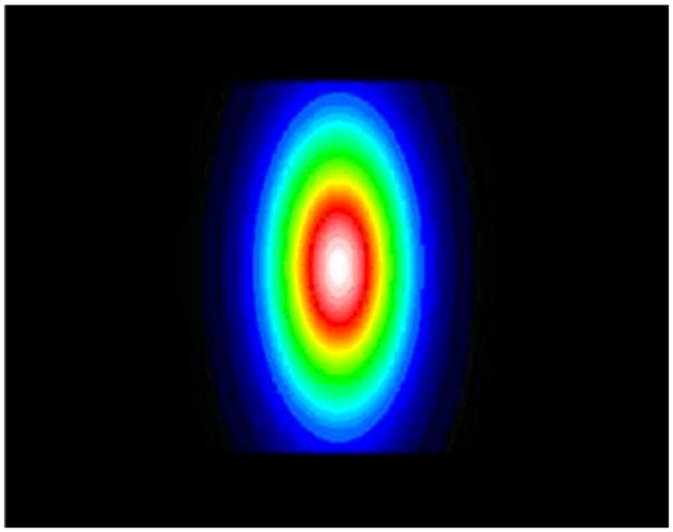
If the source is **asymmetric**, the FWHM Angle can be edited on X and Y and an axis can be defined. Defining the axis is optional for an asymmetric gaussian when FWHM values on X and Y are identical.

The axis can be global or local:

- Global axis: The orientation of the intensity diagram is related to the axis system.
- Local axis: The orientation of the intensity diagram is related to the normal at the surface.

If no axis is selected, the axis is local.

Figure 17. Luminance map of a source with Gaussian distribution



Related tasks

Creating a Display Source on page 134
The Display Source allows you to model the light emission of a display (LCD, control panel etc.) taking into account its physical properties such as the flux, the spectrum, the emittance and the intensity.

Related information

Sources Overview on page 110

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

The Sources correspond to the light sources propagating rays in an optical system.

7.3.5.2. Creating a Display Source

The Display Source allows you to model the light emission of a display (LCD, control panel etc.) taking into account its physical properties such as the flux, the spectrum, the emittance and the intensity.

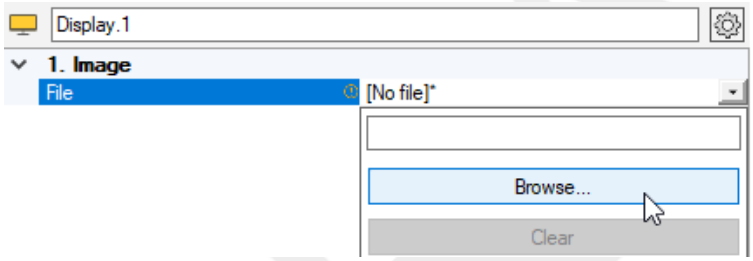
> **Important:** This feature is only available under Speos Premium or Enterprise license.

To create a Display Source:

1. From the **Light Simulation** tab, click **Display**  .
2. In **Image**, from the **File** drop-down list, click **Browse** to load a .jpeg. or png file.

   A preview of the image and the rays appears in the 3D view.

   > **Note:** If you do not see the image correctly, adjust the axis system. The image must be projected on Z (normal to the display). This ensures that what you see in the 3D view corresponds to the XMP result.



3. Define the dimensions of the display:

X range	
Mirror extent	False
Start	-50 mm
End	50 mm
Y range	
Mirror extent	False
Start	-50 mm
End	50 mm

   a) If you want light from all space, set **Mirror extent** to **True** to link the start and end values.
   b) Edit the X and Y coordinates of the start and end points of the display either by entering the values or by using the manipulators in the 3D view.

4. In **Flux**, specify the luminance of the brightest pixel (white pixel). The luminance is calculated according to this reference pixel.

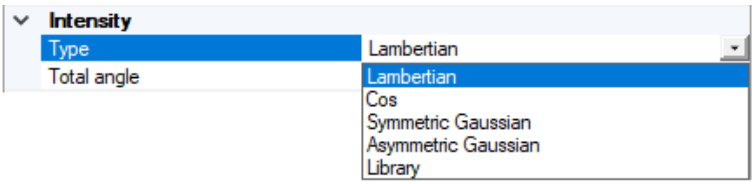
Flux	
Luminance	50 cd/m2
Infinite contrast ratio	True

5. If you want to edit the Contrast Ratio of the display, set **Infinite contrast ratio** to **False**.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

> **Tip:** The **contrast ratio** is a property of display systems. It is the measurement of the ratio between the darkest blacks and the brightest whites.
>
> Standard values range from 500:1 to 1000:1 for a LCD.

6. In **Intensity**, set the intensity distribution of the display source. From the **Type** drop-down list:



- Select **Lambertian** for a uniform distribution and in **Max Angle** set the angle of emission of the display source.

  > **Note:** By default **MaxAngle** is set to 180° so that the source emits on a hemisphere.

- Select **Cos** for a distribution that follows a cosine law at n^{th} order and set the total angle of the surface source's emission.

  In **N**, set the order of the cosine law.

- Select **Symmetric Gaussian** and set the **FWHM Angle**.

  FWHM Angle *has the same value for x and y and is computed on both axes.*

- If you want to define different FWHM values on X and Y, select **Asymmetric Gaussian**:
  - Set the FWHM angle for X and Y.
  - In the 3D view, click two lines to define X direction and Y direction.

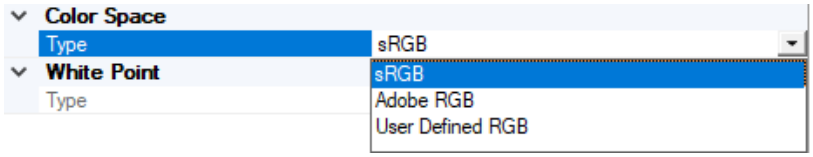
  > **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

- Select **Library** and click **Browse** to load an .ies .ldt file.
  - If you want to see the file's properties or edit the file, click **Open file** to open the **IESNA Viewer**.
  - To set the orientation of the source intensity distribution, click two lines to define X and Y direction.

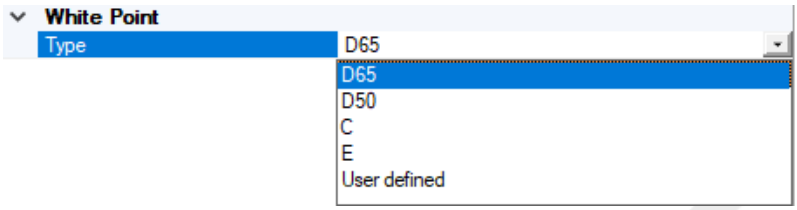
  > **Note:** When using an intensity file, the maximum value of the intensity diagram is always considered at the normal incidence. However if the real maximum value of the intensity diagram is not located at the normal incidence, you may face with unexpected measured values.

7. In **Color Space**, from the **Type** drop-down list, select which color space based model to use according to your needs and screen's capacities.

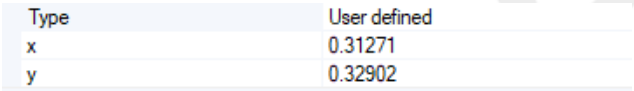
DRAFT



- Select **sRGB** to use the standard and most commonly used RGB based model.
- Select **Adobe RGB** to use a larger gamut.
- Select **User Defined RGB** to manually define the white point of the standard illuminant. From the **White Point Type** drop-down list:



  - Select **D65** to use a standard daylight illuminant that provides accurate color perception and evaluation.
  - Select **D50** to use a natural, horizon light.
  - Select **C** to use an average daylight illuminant.
  - Select **E** to use an illuminant that gives equal weight to all wavelengths.
  - Select **User defined** if you want to edit the x and y coordinates of the white point (the reference point of the model).

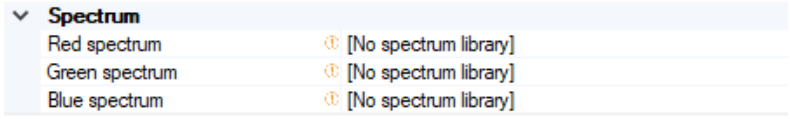


> **Note:** For more information about color models or white points of standard illuminants, see Colorimetric illuminants .

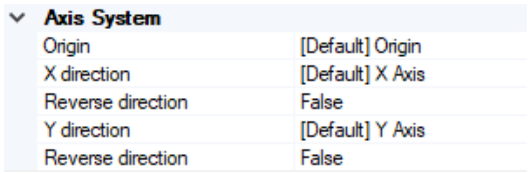
8. If you selected **User Defined RGB** from the **Color Space** drop-down list, load a spectrum file for each primary color.

   If you want to modify or create a .spectrum file, click **Open file** to open the **Spectrum Editor**.

> **Tip:** You can also download spectrum files from the Optical Library.



9. To orientate the image, set its **Axis system** by clicking one point for the origin point and two lines for X and Y directions or click  and select a coordinate system to autofill the Axis System.



> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

If you need to adjust the propagation direction of the rays, use **Reverse direction**.

10. In **Optional or advanced settings** :
    - Adjust the **Number of rays** and the **Length of rays** to display in the 3D view.
    - If you selected an intensity distribution file, set **Show Intensity Distribution** to **True** to display the intensity diagram in the 3D view.

*The Display Source is created and appears in the **Simulation** panel and in the 3D view.*

Related concepts

Understanding the Parameters of a Display Source on page 130
This page describes the parameters to set when creating a Display Source.

Related information

Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

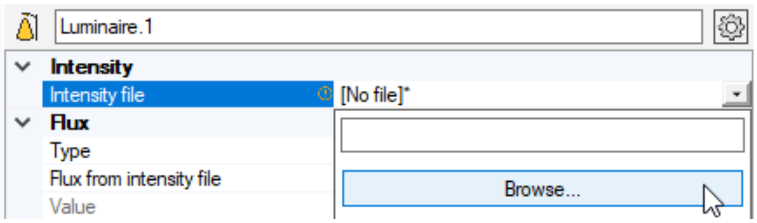
7.3.6. Creating a Luminaire Source

The Luminaire Source allows you to model artificial light for outdoor and indoor lighting.

To create a Luminaire Source:

1. From the **Light Simulation** tab, click **Luminaire** .
2. From the **Intensity file** drop-down list, click **Browse** to load an intensity distribution file (.ies, .ldt file).

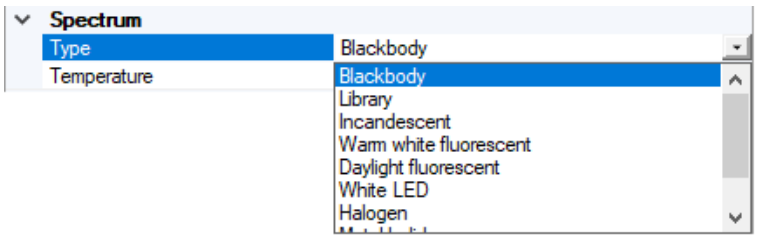
   A preview of the rays' distribution appears in the 3D view.



3. If you do not want to inherit the flux from the file, set **Flux from intensity file** to **False** and edit the flux in lumens, watts or candelas.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

4. From the **SpectrumType** drop-down list:

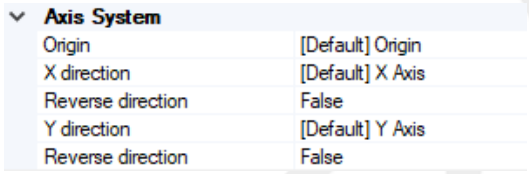


- Select **Blackbody** to set the temperature of the source in kelvins.
- Select **Library** and click **Browse** to load a .spectrum file.

  If you want to see the file's properties or edit the file, click **Open file** to open the **Spectrum Editor**.
- Select one of the predefined types available : **Incandescent**, **Warmwhite fluorescent**, **Daylight fluorescent**, **White LED**, **Halogen**, **Metal Halide**, **High Pressure Sodium**.

> **Note:** Each type of source uses a standard spectrum.

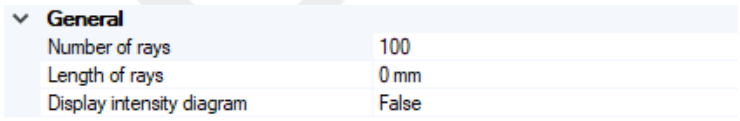
5. Set the **Axis System** of the source by clicking  in the 3D view and select one point for the origin and two lines for X and Y axes or click  and select a coordinate system to autofill the Axis System.



> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

If you need to adjust the ray's propagation direction, use **Reverse Direction** on X and / or Y.

6. In **Optional or advanced settings**  :



- Adjust the **Number of Rays** and **Ray Length** to display in the 3D view.
- If you want to display the intensity diagram in the 3D view, set the option to **True**.

  This is a visualization parameter. It displays the intensity diagram of the intensity distribution file used to define the source. If the default size is not big enough, you can increase it to observe the 3D diagram.

*The Luminaire Source is created and appears in the **Simulation** panel and in the 3D view.*

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Related concepts

Introduction to Optics on page 111
This page describes core optical principles.

Related information

Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

7.3.7. Thermic Source

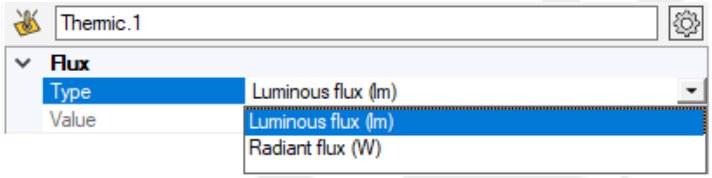
You can create a Thermic Source by directly selecting the emissive faces of the source in the interface or use a temperature field file bounding the field of action of the thermic source.

7.3.7.1. Creating a Thermic Source

A thermic surface can define a source for which the total flux and the spectrum are defined by the source's temperature and the optical properties of the support geometry. This page shows how to define a Thermic Surface Source using the faces of a geometry.

To create a Thermic Source:

1. From the **Light Simulation** tab, click **Thermic Source**  .
2. From the Flux drop-down list:

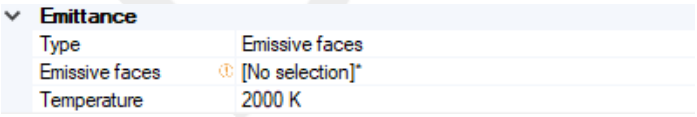


   - Select **Luminous flux (lm)** to set the flux in lumens.

     Luminous flux is the photometrically weighted radiant energy, that is to say measuring the total radiant energy across the spectrum, weighted by the sensitivity of the human eye to different wavelengths.

   - Select **Radiant flux (W)** to set the flux in Watts.

3. In **Emittance**, select **Emissive Faces** from the type drop-down list.



4. In the 3D view, click the emissive face(s) of the thermic source.

   If you need to adjust the propagation direction of the rays, check **Reverse normal**.

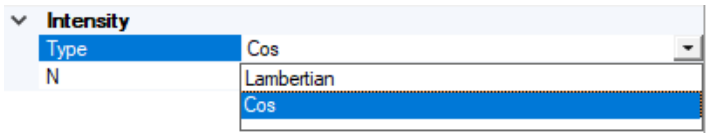
   > **Note:** The selection of faces from an imported *.obj file is not compatible with the Thermic Source.

5. Set the temperature of the source in kelvins.

DRAFT

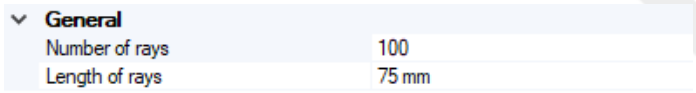
> **Note:** You cannot edit the flux values, they are automatically computed.
>
> The flux depends on the blackbody temperature and the absorption of the surface optical properties and is determined by calculating the emittance's integral on the geometry of the source.

6. From the Intensity **Type** drop-down list, select the intensity distribution of the source:



   - Select **Lambertian** for a uniform distribution.
   - Select **Cos** for a distribution that follows a cosine law at n^{th} order and in **N** set the order of the cosine law.

7. In **Optional or advanced settings**, adjust the **Number of rays** and **Length of rays** (in mm) to display in the 3D view.



8. Click **Compute** to validate the thermic source definition.

   Rays are displayed in the 3D view. The flux of the source is computed in lumens or watts according the flux type selected.

9. Press **F4** to leave the edition mode and validate the feature.

*The Thermic Source is created and appears in the **Simulation** panel and in the 3D view.*



Example of Thermic Source with emissive faces.

DRAFT

Related concepts

Introduction to Optics on page 111
This page describes core optical principles.

Related tasks

Creating a Thermic Source using a Temperature Field File on page 141
A thermic surface can define a source for which the total flux and the spectrum are defined by the source's temperature and the optical properties of the support geometry. This page shows how to create a Thermic Source using a temperature field file.

Related information

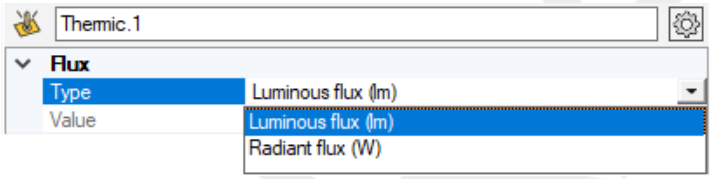
Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

7.3.7.2. Creating a Thermic Source using a Temperature Field File

A thermic surface can define a source for which the total flux and the spectrum are defined by the source's temperature and the optical properties of the support geometry. This page shows how to create a Thermic Source using a temperature field file.

To create a Thermic Source using a Temperature Field File:

1. From the **Light Simulation** tab, click **Thermic Source** .
2. From the Flux drop-down list:



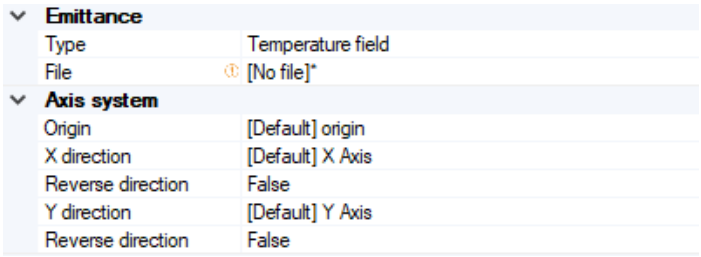
   - Select **Luminous flux (lm)** to set the flux in lumens.

     Luminous flux is the photometrically weighted radiant energy, that is to say measuring the total radiant energy across the spectrum, weighted by the sensitivity of the human eye to different wavelengths.
   - Select **Radiant flux (W)** to set the flux in Watt.

     > **Note:** You cannot edit the flux values, they are automatically computed.
     >
     > The flux value depends on the blackbody temperature and the surface optical properties and is determined by calculating the emittance's integral on the geometry of the source.

3. In **Emittance**, from the **Type** drop-down list, select **Temperature field**.

DRAFT



- Double click in the file field to browse and load an **.OPTTemperatureField** file.

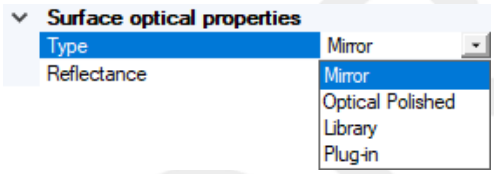
> **Note:** The **.OPTTemperatureField** file format includes description line, number of summits (Ns), number of triangles (Nt), coordinates x,y,z of summits (x Ns), coordinates l,m,n of normals (x Ns), index of summits of each triangle (x Nt), temperature of each triangle (x Nt).

- To orientate the file, set its axis system by clicking in the 3D view one point for the origin point and two lines for X and Y directions or click  and select a coordinate system to autofill the Axis System.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

If you need to adjust the direction of the vectors, use the **Reverse direction** option.

4. In **Surface optical properties**, from the **Type** drop-down list:

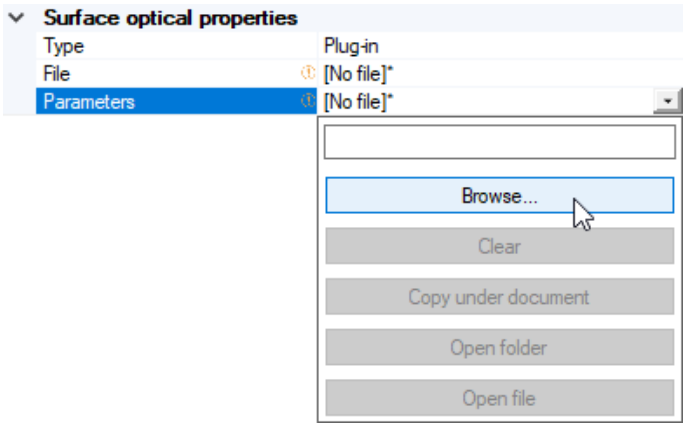


- Select **Mirror** for a perfect specular surface and edit the **Reflectance** value if needed.
- Select **Optical Polished** for a transparent or perfect polished material (glass, plastic).
- Select **Library** and double-click the file field to browse and load a SOP file.

  If you want to edit the file, click the file and click **Open file** to open it with **a surface optical property editor**.
- Select **Plug-in**, and click **Browse** to select a custom made *.sop plug-in as **File** and the **Parameters** file for the plug-in.

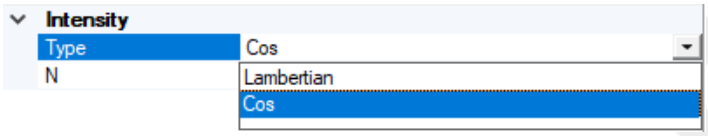
> **Note:** Make sure each plug-in created has a different GUID.

DRAFT



For more information, refer to Surface State Plugin.

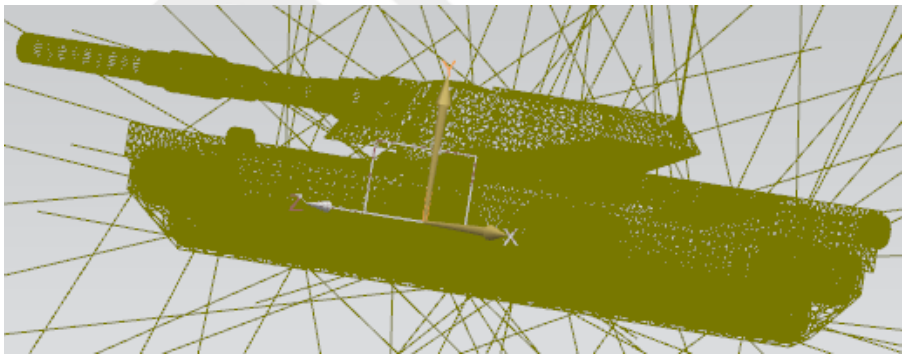
5. From the Intensity **Type** drop-down list, select the intensity distribution of the source:



- Select **Lambertian** for a uniform distribution.
- Select **Cos** for a distribution that follows a cosine law at n^{th} order and in **N** set the order of the cosine law.

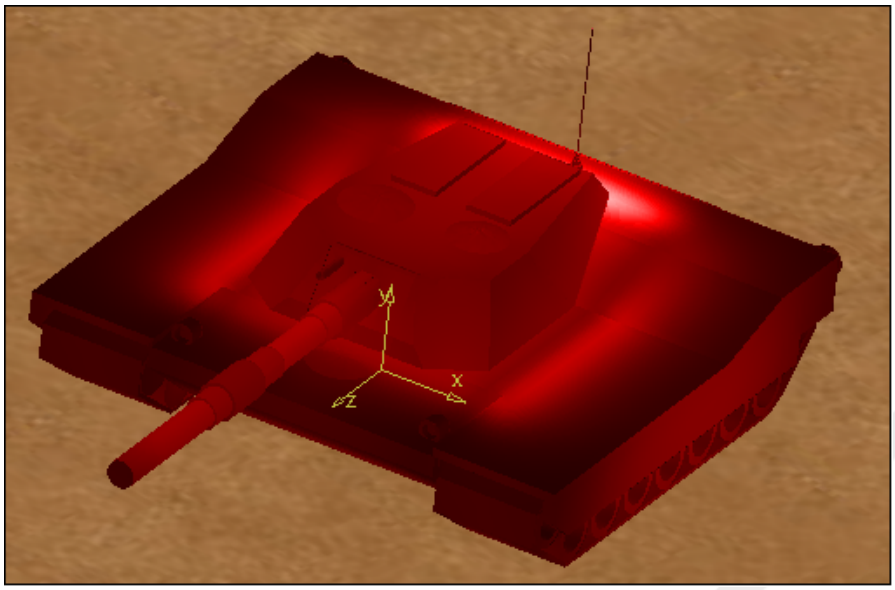
6. In **Optional or advanced settings** :
   - Adjust the **Number of rays** and **Length of rays** (in mm) to display in the 3D view.
   - If you want to display the meshing of the temperature field file in the 3D view, select **Meshing** or **Bounding box**.
7. Click **Compute** to validate the thermic source definition.
8. Press **F4** to leave the edition mode and validate the feature.

The Thermic Source is created and appears in the **Simulation** panel and in the 3D view.



Temperature field thermic source (GUI view)

DRAFT



Temperature field thermic source (simulation result)

Related concepts

Introduction to Optics on page 111
This page describes core optical principles.

Related tasks

Creating a Thermic Source on page 139
A thermic surface can define a source for which the total flux and the spectrum are defined by the source's temperature and the optical properties of the support geometry. This page shows how to define a Thermic Surface Source using the faces of a geometry.

Related information

Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

7.3.8. Ambient Sources

With an Ambient Source, you can model natural environment light such as the sun and sky.

> **Note:** Ambient Sources can be used in inverse or direct simulations. However, in direct simulations, only 2D and 3D illuminance/irradiance sensors are taken into account.

7.3.8.1. Environment Source

An Environment Source allows you to create a background using an image file.

DRAFT

7.3.8.1.1. Environment Type Overview

Environment sources are often generated based on HDR environment maps. This page shows the different mapping techniques used to generate environment maps (latitude map, light probe map etc).

Conventions

- For all the different north selected types, Zenith defines the main direction for the ambient source.
- If the North is not perpendicular to the Zenith, it is projected in the perpendicular plan.

Figure 18. Longitude/Latitude Map Type



Figure 19. Light Probe Type

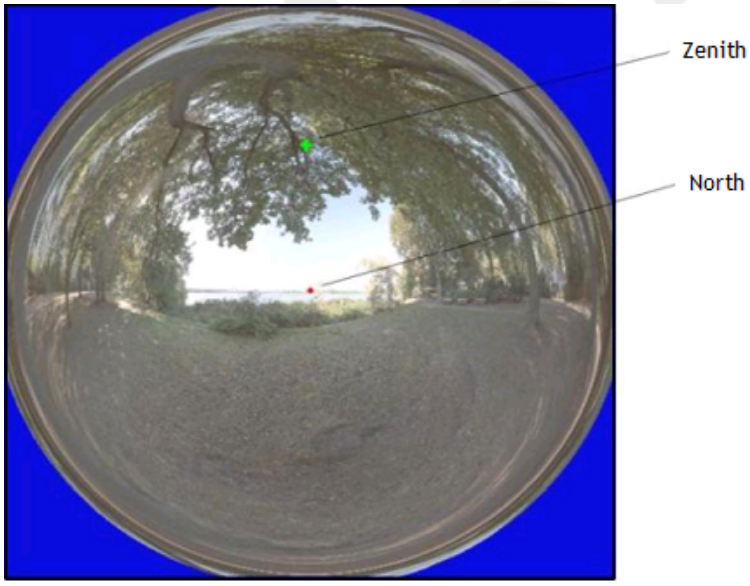


Figure 20. Horizontal Cross Type

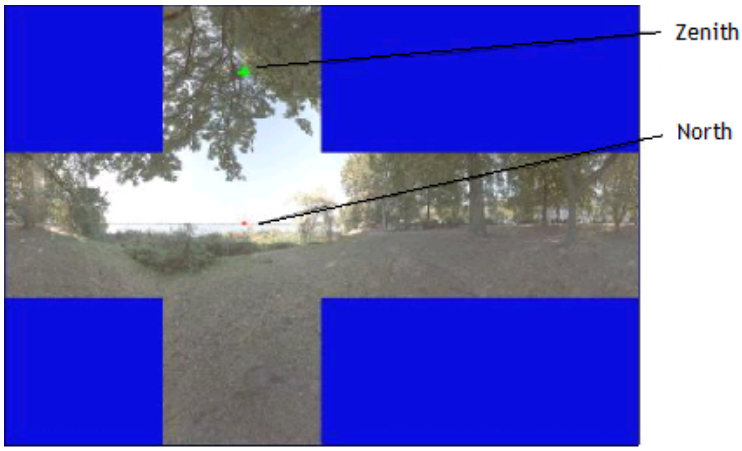
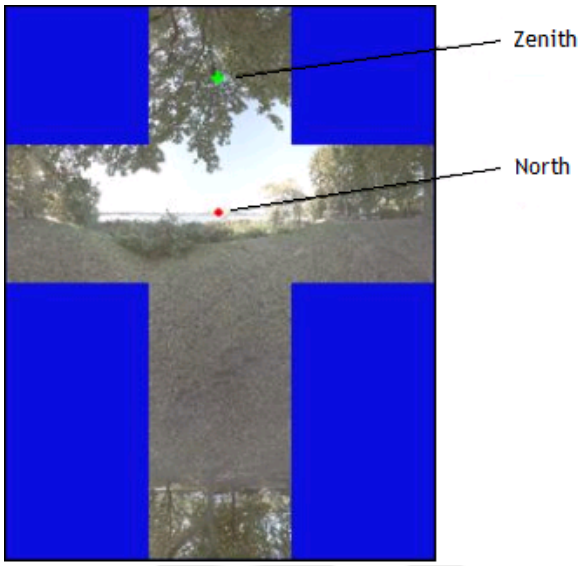
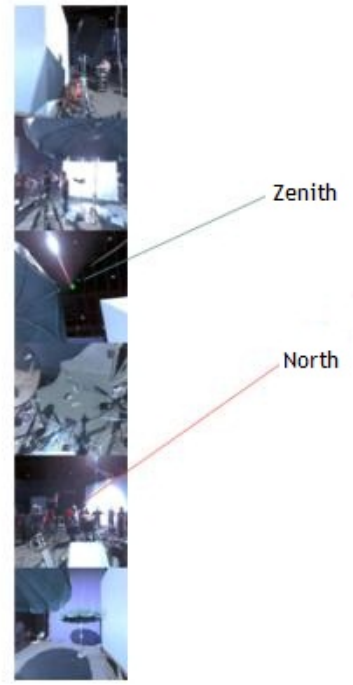


Figure 21. Vertical Cross Type



DRAFT

Figure 22. OpenEXR Type



Related tasks

Creating an Environment Source on page 147
An Environment Source generates light from an hdr image (HDRI) according to the RGB components of each pixel.

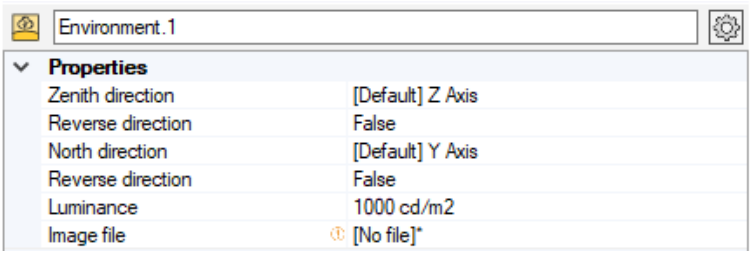
7.3.8.1.2. Creating an Environment Source

An Environment Source generates light from an hdr image (HDRI) according to the RGB components of each pixel.

> **Tip:** Where to download a .hdr file:
>
> HDRI Haven
>
> Openfootage
>
> Light Probe Image Gallery (free)
>
> Dosch Design

To create an Environment Source:

1. From the **Light Simulation** tab, click **Ambient** and click **Environment**.



2. In the 3D view:
   - click a line (normal to the ground) to set the **Zenith direction**.
   - click a line corresponding to the X axis to set the North direction.
   - or click

     and select a coordinate system to autofill the Axis System.

   If you need to adjust the axes' orientation, use **Reverse direction**.

   > **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

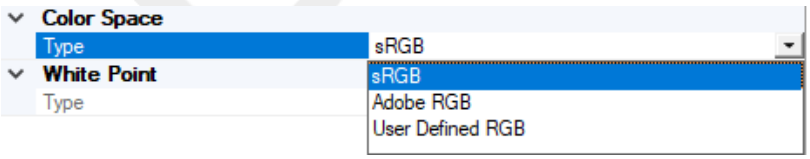
3. In **Luminance**, define the pixels' luminance.

   > **Note:** The luminance set here is the floating point representation of the reference white color (1,1,1). The luminance of each pixel is calculated according to its RGB components and the reference pixel. The luminance usually varies between 1000 and 20000 $^{cd/m2}$.

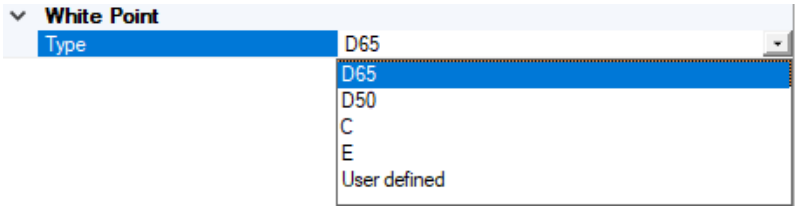
4. In **Image file**, click **Browse** to load an image or HDRI file.

   > **Note:** HDRIs have relative luminance values. If you set the luminance to 1000 cd/m^2, all the (1,1,1) pixels will have 1000 cd/m^2. The other colors are defined relatively to this one.

5. In **Color Space**, from the **Type** drop-down list, select which color space based model to use according to your needs and to the image file's own color space:

   - Select **sRGB** to use the standard and most commonly used RGB based model.
   - Select **Adobe RGB** to use a larger gamut.
   - Select **User Defined RGB** to manually define the white point of the standard illuminant. From the **White Point Type** drop-down list:



- Select **D65** to use a standard daylight illuminant that provides accurate color perception and evaluation.
- Select **D50** to use a natural, horizon light.
- Select **C** to use an average daylight illuminant.
- Select **E** to use an illuminant that gives equal weight to all wavelengths.
- Select **User defined** if you want to edit the x and y coordinates of the white point (the reference point of the model).

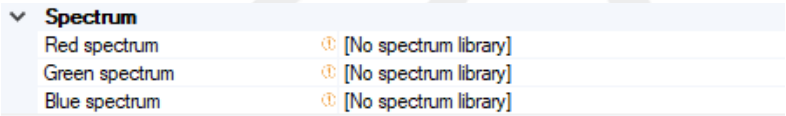
Type	User defined
x	0.31271
y	0.32902

> **Note:** For more information about color models or white points of standard illuminants, see Colorimetric illuminants.

6. If you selected **User Defined RGB** from the **Color Space** drop-down list, load a spectrum file for each primary color.

   If you want to modify or create a .spectrum file, click **Open file** to open the **Spectrum Editor**.

> **Tip:** You can also download spectrum files from the Ansys Optical Library.



7. If you selected an HDR image and want to define a ground plane, click a point in the 3D view to determine the ground origin and type the **Height** of the environment shooting.

   The HDR image is displayed on the ground plane. Thus, the ground plane acts like a textured geometry that can reflect/absorb light from other sources.

> **Tip:** To maintain the scale, use the real height of the environment shooting. If you do not have that information, the standard height is 1m.

The Environment Source is created and appears in the **Simulation** panel. The luminance is calculated according to the reference pixel. If a ground plane is defined, it is taken into account for simulation.



Without a ground plane



With a ground plane

Related concepts

Introduction to Optics on page 111
This page describes core optical principles.

Related information

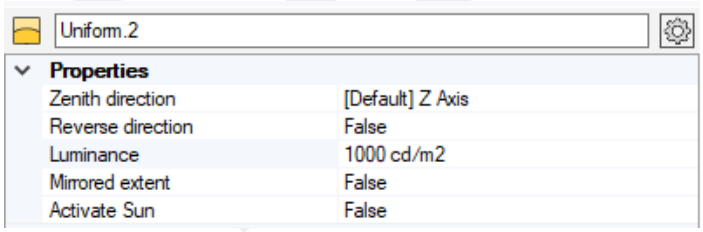
Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

7.3.8.2. Creating a Uniform Ambient Source

The Uniform Ambient Source allows you to set a specific and common luminance for the entire sky without any contribution of the sun. The sun has a specific value, automatically calculated according to its position.

To create a Uniform Ambient Source:

1. From the **Light Simulation** tab, click **Ambient** and click **Uniform** .



2. In the 3D view, click a line (normal to the ground) to set the **Zenith direction**.
3. Set the Luminance of the entire sky.

> **Tip:** The value usually ranges from a 1000 to 20000 cd/m².

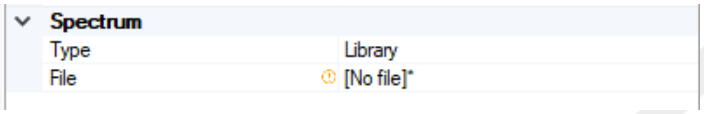
> **Note:** If you want to use the sun only in simulations, set the luminance to 0 cd/m²

4. If you want to generate the ambient light from all space and not only from the upper half space, set **Mirrored Extent** to **true**.
5. To add the sun to the ambient source, set **Activate Sun** to **True**.

   The sun is represented in the 3D view.

   - In the 3D view, select a line to set the **Sun Direction**.
   - If you need to adjust the sun direction, use **Reverse direction**.
   - If you need to adjust the sun position, use the manipulators in the 3D view

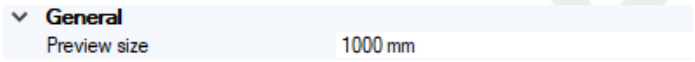
6. From the **Spectrum** drop-down list:



   - Select **Blackbody** to set the temperature of the source spectrum in Kelvins.
   - Select **Library** and click **Browse** to load a .spectrum file.

     If you want to see the file's properties or edit the file, click **Open file** to open the **Spectrum Editor**.

7. In **Optional or advanced settings**  , adjust the size of the rays displayed in the 3D view.



*The Ambient Source is created and appears in the **Simulation** panel and in the 3D view. The four cardinal points are displayed in the 3D view to visualize the orientation of the source. If you activated the sun in the scene, a sun is represented in the 3D view. The sun of the Uniform Ambient Source changes of power and color according to its orientation.*

Related concepts

Introduction to Optics on page 111
This page describes core optical principles.

Related information

Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

7.3.8.3. Creating a CIE Standard General Sky Ambient Source

The CIE Standard General Sky Ambient Source allows you to generate a specific distribution for the sky according to a zenith luminance value.

> **Important:** This feature is only available under Speos Premium or Enterprise license.

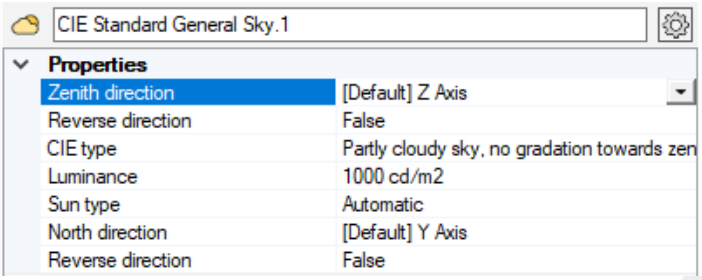
DRAFT

This sky model is based on the publication of the spatial distribution of daylight - CIE standard general sky ISO 15469:2004(E)/CIE S 011/E:2003 .

To create a CIE Standard General Sky Ambient Source:

1. From the **Light Simulation** tab, click **Ambient** and click **CIE Standard General Sky** .

   A preview of the sun and the 4 cardinal points appear in the 3D view. The sun's position is computed according to the timezone and location set.



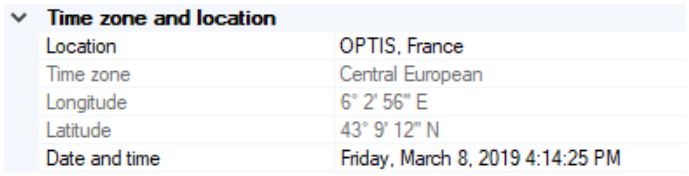
2. In the 3D view:
   - If you want to modify the **Zenith direction** (corresponding by default to the Z axis), in the 3D view click  and select a line.
   - If you want to modify the **North direction** (corresponding by default to the Y axis), in the 3D view, click  and select a line.
   - or click  and select a coordinate system to autofill the Axis System.

   The North direction corresponds to the rotation axis of the earth projected on the ground.

   > **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

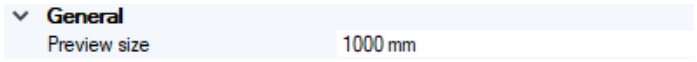
3. From the **CIE type** drop-down list, select a sky model.
4. Set the Luminance of the sky at the zenith.
5. From the **Sun type** drop-down list:
   - Select **Direction** to manually set the sun's position and direction thanks to the manipulators in the 3D view.
   - Select **Automatic** to automatically calculate the sun position according to the north direction and time zone and location you set.
     a. If you want to set the time and location manually, from the **Location** drop-down list, select **User**.
     b. Set the time zone and location of your choice.
6. Set the **Time zone and location**:

DRAFT



- If you want to set the coordinates manually (longitude and latitude), from the **Location** drop-down list, select **User**.
- If you want to use the night sky model, adjust the **Date and time**.

7. In **Optional or advanced settings** , adjust the size of the rays displayed in the 3D view.



*The Ambient Source appears in the **Simulation** panel. In the 3D view, the four cardinal points and a representation of the sun are displayed.*

Related concepts

Introduction to Optics on page 111
This page describes core optical principles.

Related information

Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

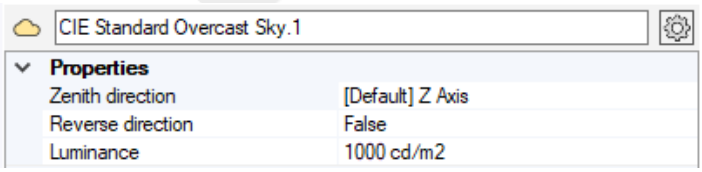
7.3.8.4. Creating a CIE Standard Overcast Sky Ambient Source

This sky model is based on the publication of the spatial distribution of daylight - CIE standard overcast and clear sky ISO 15469:2004(E)/CIE S 011/E:2003 .

> **Important:** This feature is only available under Speos Premium or Enterprise license.

To create a CIE Standard Overcast Sky Source:

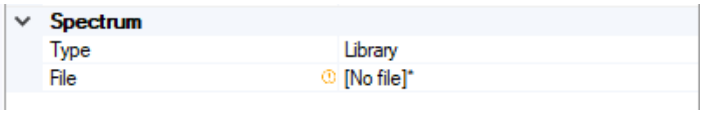
1. From the **Light Simulation** tab, click **Ambient** and click **Overcast** .



2. If you want to modify the **Zenith direction**, in the 3D view, select a line (normal to the ground).
3. Set the Luminance of the entire sky in cd/m^2.

> **Tip:** The value usually ranges from a 1000 to 20000 cd/m².

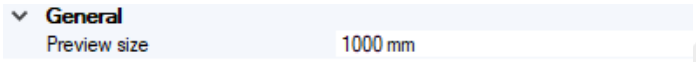
4. From the **Spectrum** drop-down list:



   - Select **Blackbody** to set the temperature of the source spectrum in Kelvins.
   - Select **Library** and click **Browse** to load a .spectrum file.

5. If you want to see the file's properties or edit the file, click **Open file** to open the **Spectrum Editor** .

6. In **Optional or advanced settings** , adjust the size of the rays displayed in the 3D view.



*The Ambient Source appears in the **Simulation** panel. In the 3D view, the four cardinal points are displayed.*

Related concepts

Introduction to Optics on page 111
This page describes core optical principles.

Related information

Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

7.3.8.5. Natural Light Ambient Source

The Natural Light Ambient Source allows you to generate natural lighting based on a day time or a night time model. With the day time model, you can generate the sun and the sky with a radiation ranging from 380nm to 780nm, and adjust the luminance of the source by modifying the sky's cloudiness. With the night time model, you can generate a sky radiation ranging from 380nm to 1100nm (near infrared), while taking into account moonlight, starlight, zodiacal light and airglow.

7.3.8.5.1. Creating a Natural Light Ambient Source

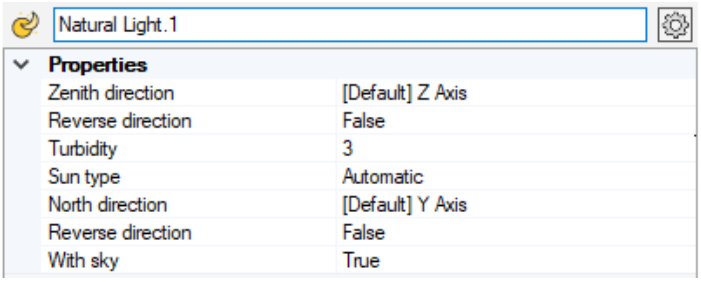
A Natural Light Ambient Source generates light from the sky according to the time, location and the turbidity. The luminance varies according to where you look.

> **Important:** This feature is only available under Speos Premium or Enterprise license.

To create a Natural Light Ambient Source:

1. From the **Light Simulation** tab, click **Ambient** and click **Natural Light** .

A preview of the sun and the 4 cardinal points appear in the 3D view. The sun's position is computed according to the timezone and location set.



2. In the 3D view:
   - If you want to modify the **Zenith direction**, click
     
     and select a line (normal to the ground).
   - If you want to modify the North direction, click
     
     and select a line corresponding to the Y axis.
   - or click
     
     and select a coordinate system to autofill the Axis System.

   Note: The North direction corresponds to the rotation axis of the earth projected on the ground.

   Note: If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

3. From the **Sun type** drop-down list:
   - Select **Automatic** to automatically calculate the sun position according to the north direction and the selected **Timezone and location**.
   - Select **Direction** to manually set the **Sun Direction** by clicking a line in the 3D view.
4. Set the **Turbidity** of the sky, that is the cloudiness of the environment. The lower the turbidity, the clearer the environment.

   The luminance is automatically calculated according to the level of turbidity of the sky.

   Tip: The turbidity usually varies between 1.9 and 9.9.

   Note: When turbidity is higher than 6, a part of the luminance distribution of the sky is computed using the overcast sky luminance distribution formula. We recommend you not using a turbidity higher than 6 with natural light ambient source. Otherwise, limit the altitude of sun to reduce the error.

DRAFT

5. If you want to use the sun only in simulations, uncheck **With Sky**.
6. Set the **Time zone and location**:

Time zone and location	
Location	OPTIS, France
Time zone	Central European
Longitude	6° 2' 56" E
Latitude	43° 9' 12" N
Date and time	Friday, March 8, 2019 4:14:25 PM

   - If you want to set the coordinates manually (longitude and latitude), from the **Location** drop-down list, select **User**.
   - If you want to use the night sky model, adjust the **Date and time**.

7. In **Optional or advanced settings**  , adjust the size of the rays displayed in the 3D view.

General	
Preview size	1000 mm

The Ambient Source appears in the **Simulation** panel. In the 3D view, the four cardinal points and a representation of the sun are displayed.

Related concepts

Introduction to Optics on page 111
This page describes core optical principles.

Using Turbidity for a Natural Light Ambient Source on page 156
This page describes the impact of the turbidity on the simulations' results.

Related information

Sources Overview on page 110
The Sources correspond to the light sources propagating rays in an optical system.

7.3.8.5.2. Using Turbidity for a Natural Light Ambient Source

This page describes the impact of the turbidity on the simulations' results.

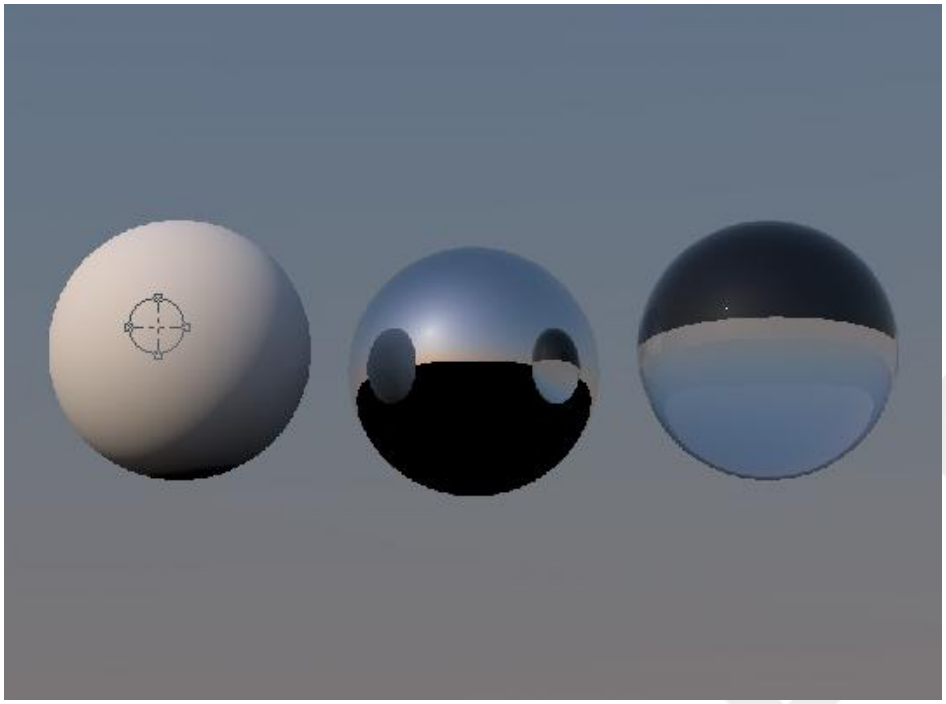
Definition of turbidity (*A Practical Analytic Model for Daylight*, A. J. Preetham, Peter Shirley, Brian Smits): Turbidity is a measure of the fraction of scattering due to haze as opposed to molecules. This is a convenient quantity because it can be estimated based on visibility of distant objects.

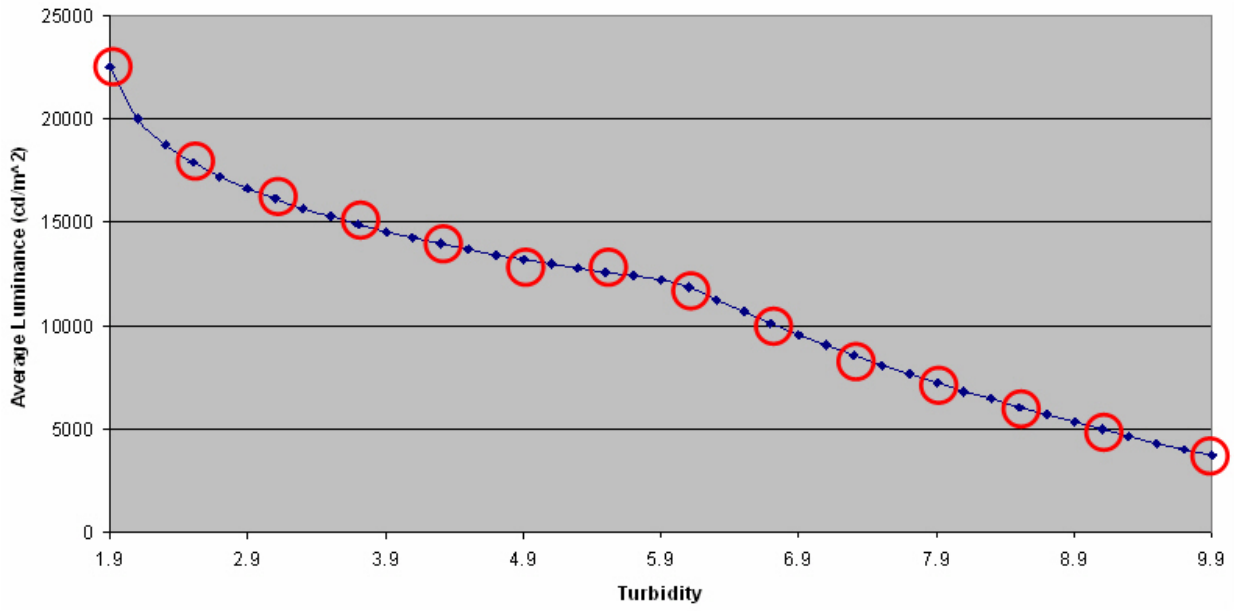
In Speos, the turbidity allows you to adjust the luminance of the sky but it does not alter the density of the air. The turbidity does not impact the water particles present in the air. Therefore, the source does not generate fog or haze.

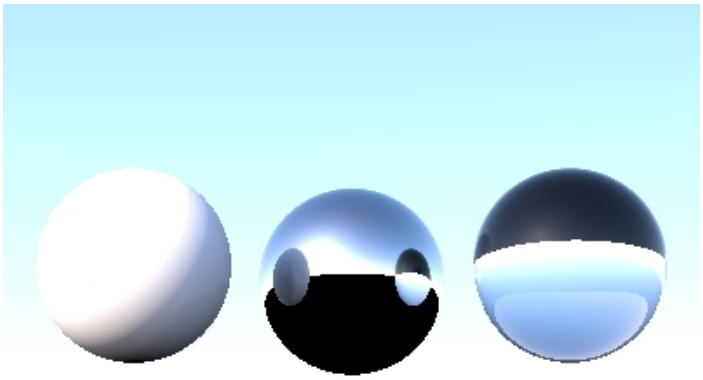
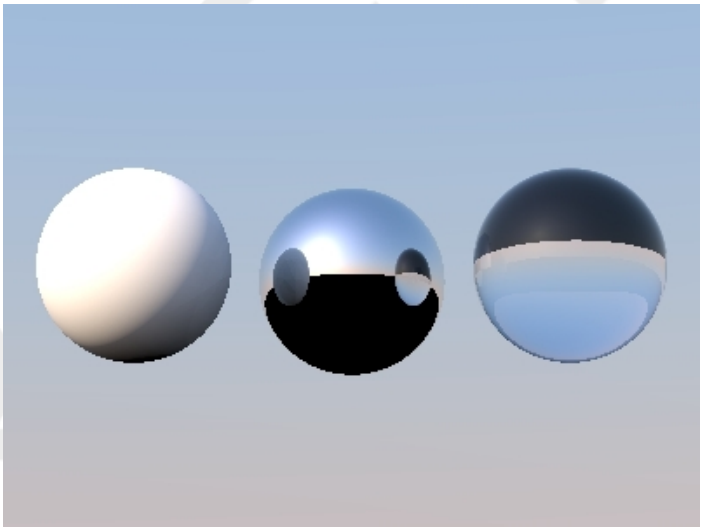
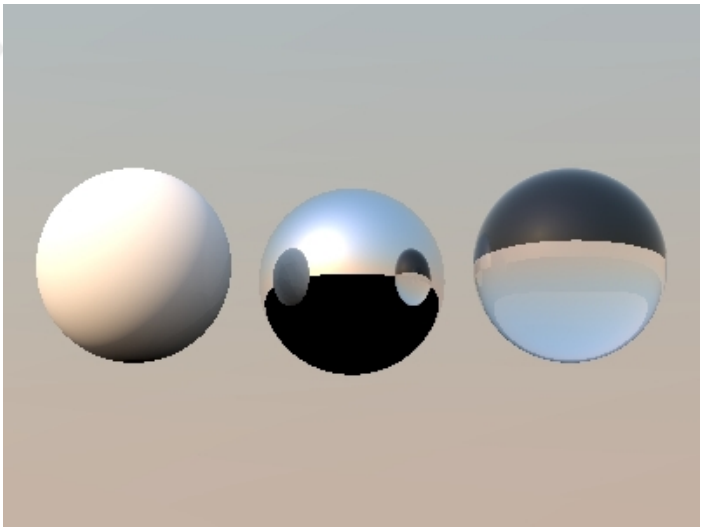
To generate fog, the Ambient Material must be modified by loading a new material or adjusting the refractive index of the air.

In the following example, we consider the average luminance value of the measurement point indicated on the left plastic ball and we let the turbidity vary between 1.9 and 9.1.

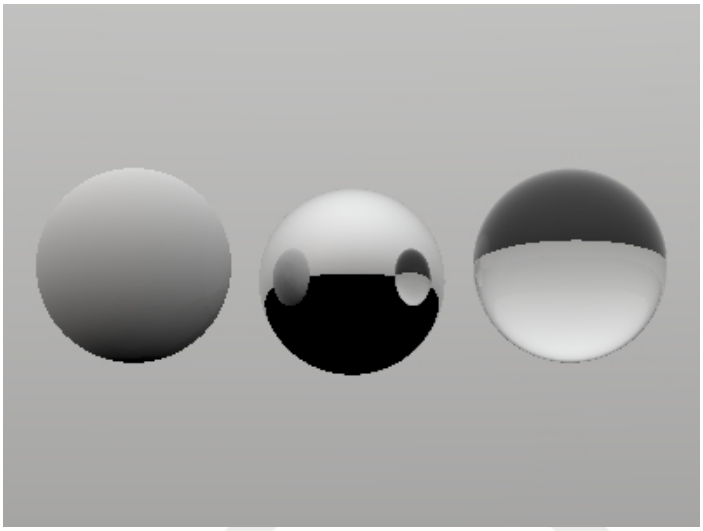
DRAFT




25000
20000
15000
10000
5000
0
Average Luminance (cd/m^2)
1.9
2.9
3.9
4.9
5.9
6.9
7.9
8.9
9.9
Turbidity

Turbidity	Normalized Simulation Result
1.9	
3.1	
5.5	

DRAFT

Turbidity	Normalized Simulation Result
9.1	

Note: When turbidity is higher than 6, a part of the luminance distribution of the sky is computed using the overcast sky luminance distribution formula. We recommend you not using a turbidity higher than 6 with natural light ambient source. Otherwise, limit the altitude of sun to reduce the error.

Related tasks

Creating a Natural Light Ambient Source on page 154
A Natural Light Ambient Source generates light from the sky according to the time, location and the turbidity. The luminance varies according to where you look.

7.3.8.6. U.S. Standard Atmosphere 1976 Source

The U.S. Standard Atmosphere Ambient Source follows a specific atmospheric model to simulate a sky radiation ranging from 280nm into ultraviolet to 4µm into infrared. This source is mainly used for irradiance computation of optical systems involving near infrared contribution such as solar concentrators, LiDAR or camera sensors.

7.3.8.6.1. Understanding the Characteristics of the U.S. Standard Atmosphere 1976 Source

This page describes the characteristics, specificities and typical use case scenario of the U.S. Standard Atmosphere 1976 Ambient Source.

The Ambient Source U.S. Standard Atmosphere 1976 is an ambient source that simulates a sky with a spectrum ranging from 280nm into ultraviolet to 4µm into infrared.

This Ambient Source is mainly used to compute irradiance. The visual rendering may then not appear realistic, and may not be considered as such. For a realistic rendering, rather use the CIE Standard Overcast or General Sky Ambient Source.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Characteristic of the Source

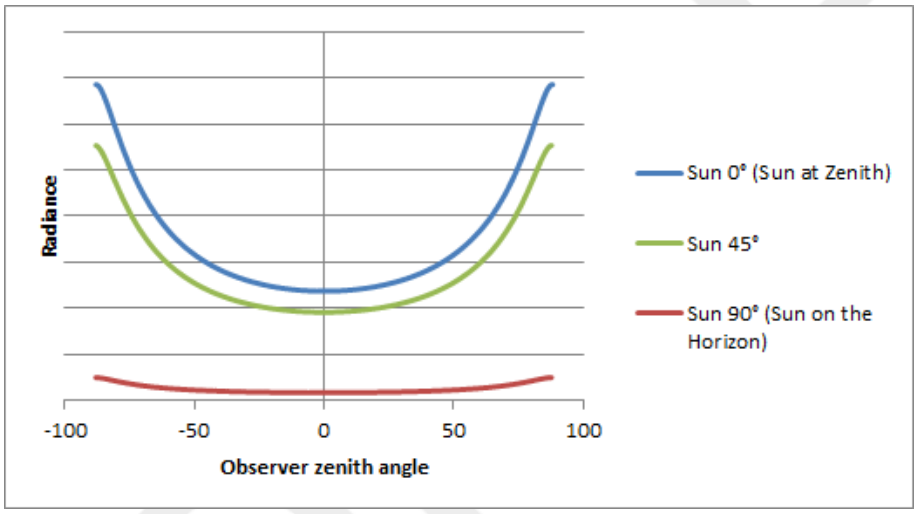
- Sky, sun spectra and radiance vary according to the sun position:
  - Sky spectrum depends on the sun elevation and is constant over the whole sky for a given position of the sun.
  - Sun spectrum depends on its elevation.
  - Radiance levels depend on the solar zenith angle. Azimuthal variations are not considered.
- The atmospheric and meteorological conditions are defined in the U.S. Standard Atmosphere, 1976 and cannot be modified.

  The conditions are a mid-latitude northern hemisphere model of daytime atmospheric temperature and pressure, for a clear sky with large visibility distance.

The chart below shows the radiance of the sky for a given sun position according to the solar zenith angle.

> **Note:** The radiance of the sun is not considered in the chart.

Figure 23. Figure 1. Radiance behavior according to solar zenith angle.



The more the solar zenith angle, the more radiance:

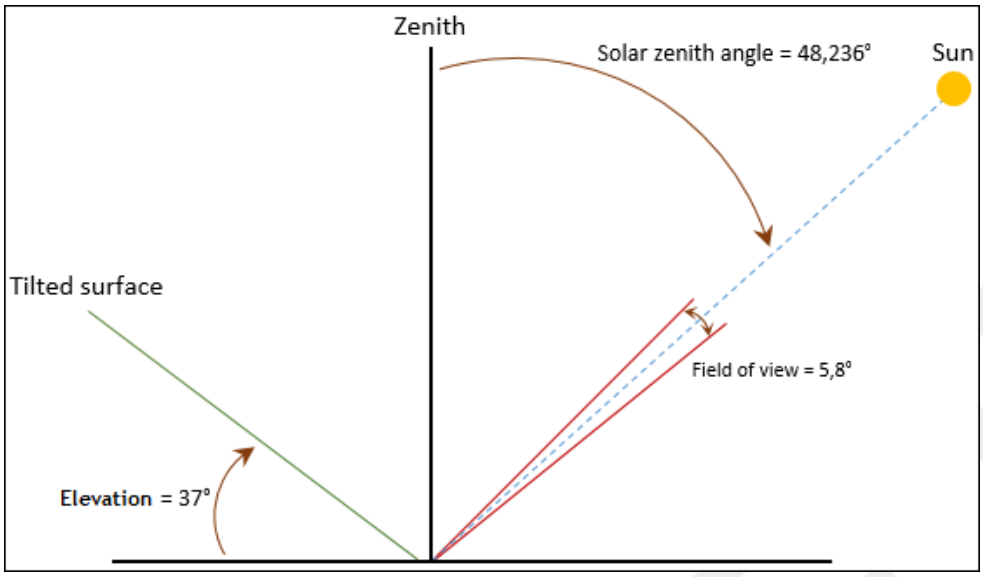
- When the sun is at the zenith, the radiance level is high and increases when the sensor orientation approaches the horizon.
- When the sun is on the horizon, the radiance is low (dawn or dusk condition) and presents a flat profile (with a slight increase at the horizon).

Typical Use Case

The U.S. Standard Atmosphere Ambient Source can be used to compute the irradiance on solar panels.

To be realistic, the Ambient Source follows the ASTM-G173 Standard when used in the conditions of this standard, that is when the solar zenith angle is 48.236° (air mass 1.5).

Figure 24. Figure 2. ASTM-G173 Standard conditions representation.

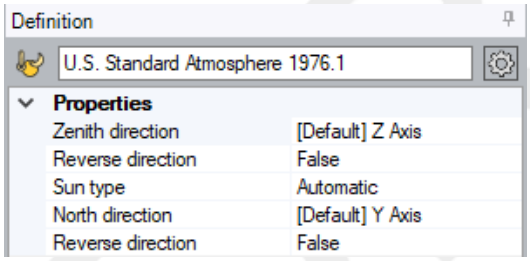


7.3.8.6.2. Creating a U.S. Standard Atmosphere 1976 Source

The U.S Standard Atmosphere 1976 source allows you to generate light from the sky according to a time and location. With this source, signals can be collected outside the visible spectrum including in the infrared.

To create a U.S. Standard Atmosphere Ambient Source:

1. From the **Light Simulation** tab, click **Ambient** and click **U.S. Standard Atmosphere 1976** .



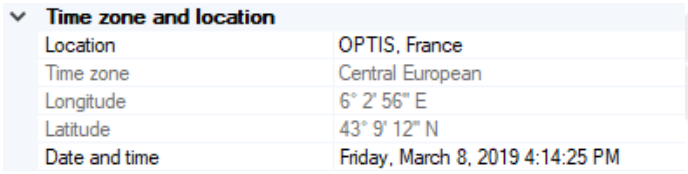
   A preview of the sun and the 4 cardinal points appear in the 3D view. The sun's position is computed according to the timezone and location set.
2. In the 3D view:
   - If you want to modify the **Zenith direction**, click
     
     and select a line (normal to the ground).
   - If you want to modify the **North direction**, click
     
     and select a line collinear to north direction.
   - or click
     
     and select a coordinate system to autofill the Axis System.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

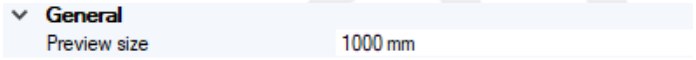
> **Note:** The North direction corresponds to the rotation axis of the earth projected on the ground.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

3. From the **Sun type** drop-down list:
   - Select **Automatic** to automatically calculate the sun position according to the north direction and the selected **Timezone and location**.
   - Select **Manual** to manually set the **Sun direction** by clicking a line in the 3D view or by using the 3D view manipulators.
4. Set the **Time zone and location**:



   - If you want to set the coordinates manually (longitude and latitude), from the **Location** drop-down list, select **User**.
   - If you want to use the night sky model, adjust the **Date and time**.
5. In **Optional or advanced settings**  , adjust the size of the rays displayed in the 3D view.



The Ambient Source appears in the **Simulation** panel. Four cardinal points and a representation of the sun are displayed are displayed in the 3D view.

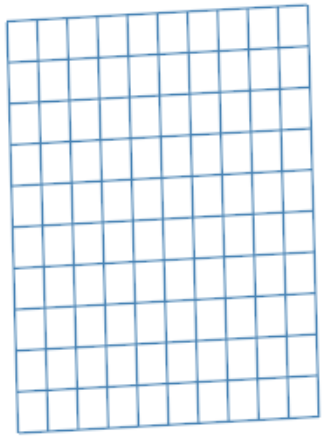
DRAFT

8: Sensors

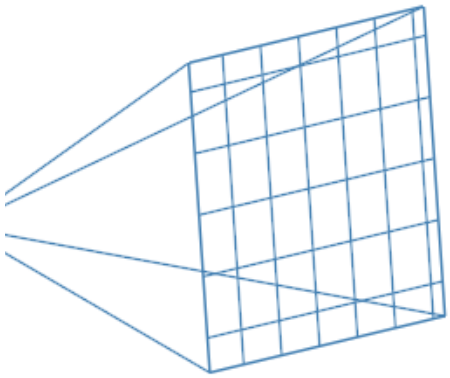
Sensors allow to render the optical result of an optical system by integrating rays coming from the sources.

8.1. Sensors Overview

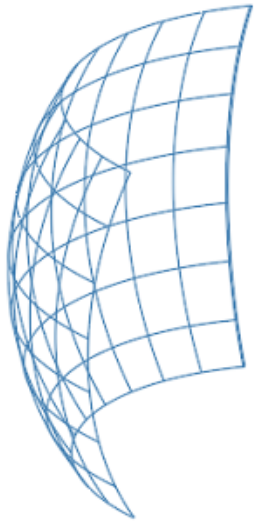
Sensors integrate rays coming from the source to analyze the optical result.



Irradiance Sensor



Radiance Sensor



Intensity Sensor

Sensors allow to integrate and analyze rays coming from the sources contained in an optical system.

A wide variety of sensors are available to cover different needs and configurations.

The sensors can be used to compute power and analyze how a source is emitting and what is its intensity/emission pattern or to create perspective and viewpoints to see how a system is perceived by an eye, an observer.

Types of Sensors

- The Radiance Sensor allows you to compute radiance (in watt/sr/m^2) or luminance (in candela/m^2).
- The Irradiance Sensor allows you to compute irradiance (in watt/m^2) or illuminance (in Lux).
- The Intensity Sensor allows you to compute radiant intensity (in watt/sr) or luminance intensity (in candela).
- The 3D Irradiance Sensor allows you to compute irradiance of volume bodies or faces.
- The Human Eye Sensor allows you to accurately simulate human vision by considering the physiological characteristics of the eye.
- The Immersive Sensor allows you to observe all the objects surrounding a defined point of view. A point is placed in the scene and the sensor restitutes what is viewed from the scene from this specific point.
- The 3D Energy Density Sensor allows you to compute the energy density carried out by the light in Lumen/m^3 or Watt/m^3 which can be useful when working with highly diffusive materials, wanting to track some hot spots or wanting to visualize the rays' distribution inside the volume itself.
- The Observer Sensor allows you to create an observer point in the scene.
- The Light Field Sensor allows you to measure the distribution of light hitting a surface and to generate a Light Field file storing light distribution on this selected surface.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- The Camera Sensor allows you to simulate rays integration as in a real camera.
- The Physical Camera allows you to create a lens system with its own optimized propagation engine, to be included in a bigger optical system, which improves the convergence when propagating rays and considers ghost images.
- The LiDAR Sensor allows you to create a LiDAR source and sensor that can be used for LiDAR simulation.
- The Geometric Rotating LiDAR sensor allows you to perform field of view studies to quickly identify how to optimize your LiDAR system.
- The Light Expert Group is dedicated to perform a Light Expert analysis for several sensors at the same time.

In Speos

You can manage all the characteristics of the sensors: its size, position, orientation, wavelength sensitivity etc.

To create a sensor you have to define several parameters: (The parameters vary based on the type of sensor.)

Related information

Sensor Creation on page 171
This section gathers all procedures allowing to create a sensor. At least one sensor must be created in an optical system to gather and interpret rays.

8.2. Generic Sensor Parameters and Tools

This section gathers parameters and tools that are relevant to the definition and usage of almost every sensor available in Speos.

8.2.1. Integration Angle

This section describes the Integration Angle parameter to better understand its possible impacts on the simulation's results and provides pieces of advices to correctly set it. The integration angle must be defined for various sensors.

8.2.1.1. Integration Angle Overview

The following page presents you briefly the Integration Angle and its capabilities.

Definition

Integration Angle has been introduced in case of a Direct Simulation to provide results in a system with specular contributions.

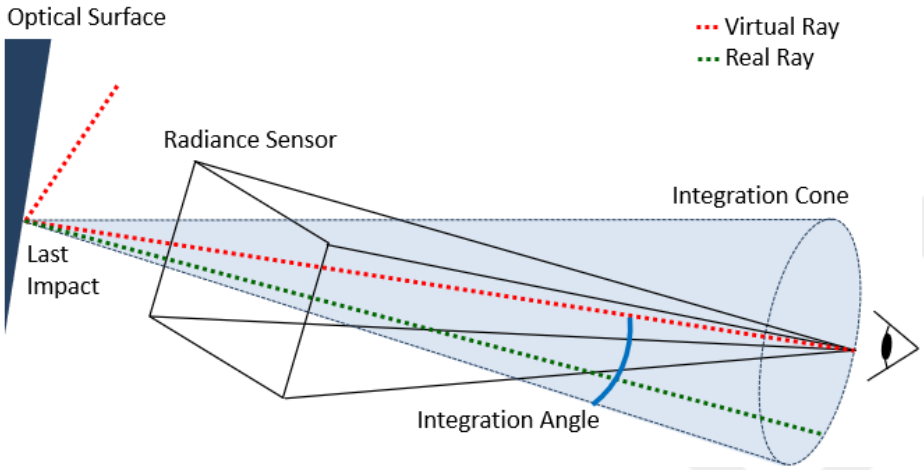
In a Direct Simulation, the rays come from the source and are propagated according to physic law (Monte-Carlo engine). The probability that a ray coming from a specular interaction has exactly the same direction as the integration direction (pixel on sensor plan to focal point), is almost null and the specular contribution is almost unsolvable.

To solve the issue, the sensors Radiance, Observer, Immersive use an internal algorithm (non-visible for users) called Gathering which basically forces/helps specular rays to find the sensors. The Integration Angle approximation will help the Gathering algorithm into avoiding the forced rays to be too deflected compared to its original direction (after last impact), allowing rays to be integrated into the solid angle (defined thanks to the Integration Angle). Thanks to this approximation, rays are integrated if the angular deflection between their propagation direction and the sensor integration direction is smaller than the integration angle.

DRAFT

The approximation makes possible that an infinitesimal small ray hits an infinitesimal small point, which is impossible without it because ray cannot hit a point, but a cone can. That's why we need an integration angle to generate a cone.

Figure 25. Integration Angle Representation

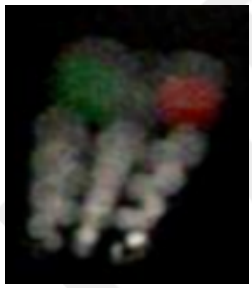
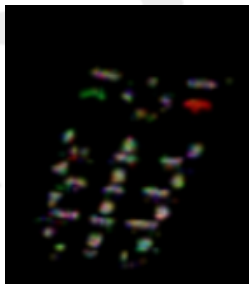
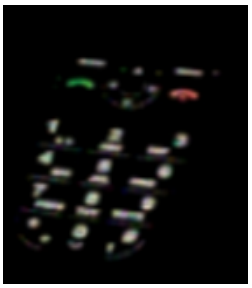


- **Virtual Ray**: The integration direction (nominal direction) for one pixel to reach the focal point of the sensor (represented by the eye in the picture).
- **Real Ray**: The ray computed by the simulation.

Integration Angle Effect on Result

Three examples are provided to illustrate the integration angle effect on a Radiance map (image of a phone seen through a window (specular surface)).

Each result is obtained with an equal number of rays launched into the simulation.

		
Blurry A blurry result means the integration angle is too big. Too many rays are integrated by the sensor.	**Noisy** A noisy result means the integration angle is too small. Not enough rays are integrated by the sensor.	**Adequate** A balanced result is obtained when the integration angle is well adjusted to the scene.

8.2.1.2. Direct Simulation with Integration Angle VS Inverse Simulation

Before finding an Integration Angle for your system, this page provides you with comparisons of direct simulations with an Integration Angle VS inverse simulations.

DRAFT

This will show you the main differences, but keep in mind that there is no right or wrong answer. The examples consider that an adequate integration angle is set.

Example

Conclusion

Direct Simulation Advantages

- The bigger your system is (adding diffusing material, hidden sources, etc.), the more the direct simulation outperforms the inverse simulation.
- The more the sources are hidden behind optics, the longer the light path is from the source to the sensor, the better the direct simulation compared to the inverse simulation.
- The direct simulation is usually faster than the inverse simulation when there are diffuse materials.

Direct Simulation Drawbacks

The Integration Angle is and will always be an approximation. If you set a correct Integration Angle you will narrow down the approximation, but you will not completely remove it.

Direct Simulation Advantages

The Inverse Simulation does not need an Integration Angle. So, the simulation result will be correct without any approximation.

Direct Simulation Drawbacks

The Inverse Simulation is slow and requires more resources (HPC, Cores, etc.) than a Direct Simulation.

8.2.1.3. Setting an Integration Angle

The following page gives you advices to help you find an integration angle that could fit your system.

We assume that if you are on this page, you have decided to use a Direct Simulation and so define an Integration Angle. However, if you have enough resources and you do not want to define an integration angle that will bring some approximation, use the inverse simulation. It will take longer than a direct simulation, but you will have a result without approximation.

There is no definite rule that will allow you to find the perfect integration angle each and every time. To find a good integration angle, only the empirical way is possible, because it highly depends on your system: the intensity distribution of the system, the scene size, the different materials and geometries.

Case Types

Narrowing the Integration Angle

The Integration Angle needs to be smaller than a specific threshold. To find this threshold, you'll need to carry out empirical tests. To do so, you can:

1. Define a Surface Source a diffuser.
2. Run a quick Direct Simulation of the system with the Surface Source (without the diffuser).
3. Analyse the result.

DRAFT

4. From now, try defining an Integration Angle. According to the size of the smaller element (or source?) and the observer point of the sensor, you have good start to define the Integration Angle: use the trigonometry that will help you find the maximum Integration Angle that you could use.
5. Run the real Direct Simulation with the diffuser.
6. Iterate to narrow down the Integration Angle until you generate a suitable result.

8.2.1.4. Influence Factors

The following section provides information regarding factors that could influence the Integration Angle definition.

8.2.1.4.1. Scene Size Influence

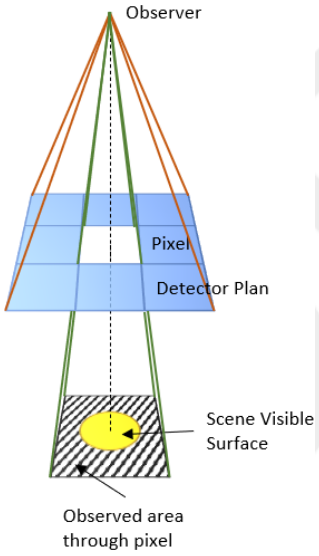
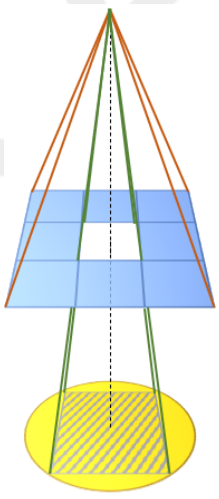
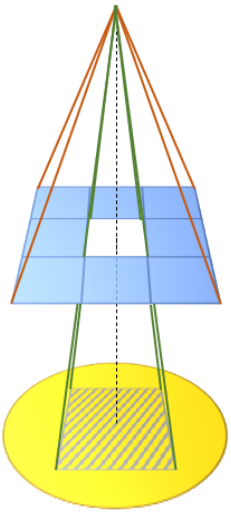
This page describes how the size of a scene can impact the sensor's perception of the scene's illumination.

The distance between the sensor plan and the emitting scene impacts the sensor's perception of the scene illumination.

If the sensor plan is placed at a certain distance from the emitting scene, the sensor defines a cone under which the observer sees an area through the pixel.

The luminance is then integrated over this observed area.

Therefore, there are at least two cases to consider if you want to study the influence of the emitting scene dimensions and the integration angle of the sensor on the luminance value.

Visible surface of the scene is smaller than the observed area	Visible surface of the scene is bigger than the observed area	
		

When the visible surface of the scene is smaller than the area observed through the pixel aperture, the luminance is integrated both over the visible scene and the unilluminated area. The average luminance value is then lower than the average luminance value of the scene because the average is made on the illuminated and unilluminated areas.	When the visible surface of the scene is bigger than the area observed through the pixel aperture, the luminance is integrated over the visible surface of the scene. We observe the luminance variation as a function of the integration angle and the visible surface of the scene.

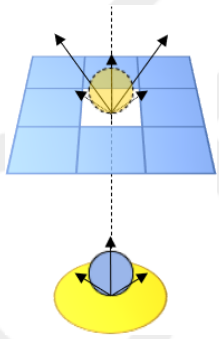
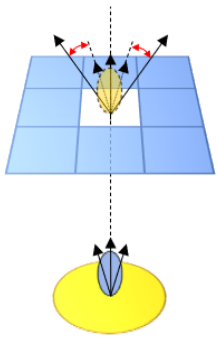
8.2.1.4.2. Gaussian Intensity Distribution Influence

This page describes the influence a scene with a gaussian intensity distribution has on the illuminance interpretation.

Integration and FWHM Angle

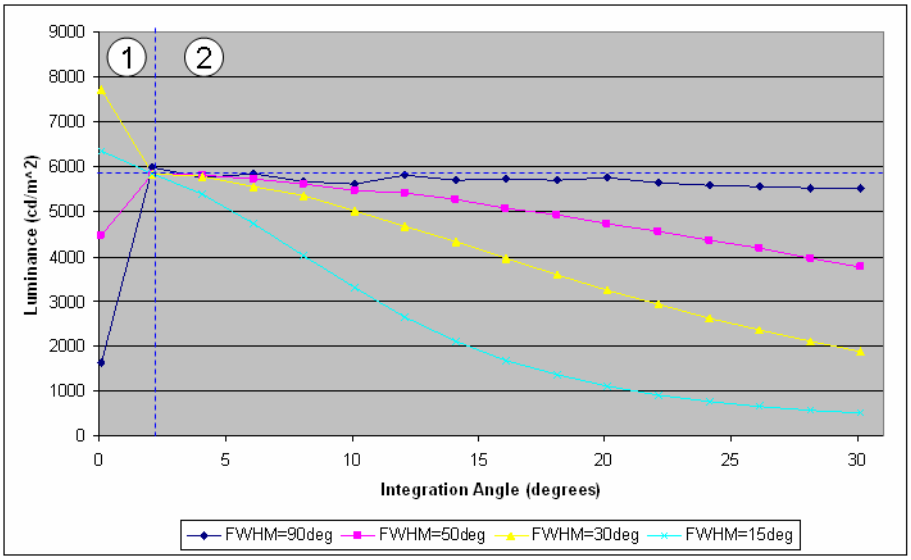
So far, only scenes with a lambertian intensity distribution have been studied. To study the effect of the angular distribution in the luminance simulation, let us consider a scene with a symmetrical gaussian intensity distribution and a FWHM variable angle. The visible surface of the scene is bigger than the pixel dimensions and the sensor is positioned in order to be close to the scene.

We need to consider at least two cases if we want to study the influence of the integration angle of the sensor and the FWHM angle of the scene on the luminance value:

Integration angle smaller than the FWHM angle of the scene	**Integration angle bigger than the FWHM angle of the scene**
	
When the integration angle is smaller than the FWHM angle of the scene, the luminance value is supposed to stay constant, as the photons are integrated under the same cone as the emission cone of the scene. There is a limitation when the integration angle is too small. Too few photons are integrated by the pixel, meaning that the luminance value is noisy and therefore not reliable.	When the integration angle is greater than the FWHM angle of the scene, the luminance value decreases as the integration angle increases, as the number of photons does not change, but is integrated under a bigger cone than the emission cone of the scene.

DRAFT

Luminance Variation



For a too small angle (1), the luminance is too noisy, therefore not reliable. Then the luminance starts to decrease as the integration angle increases (2).

As the value of the FWHM angle increases, the integration angle ranges for which the luminance stays constant raises.

On this example, the luminance starts to decrease even if the incident angle is lower than the FWHM angle as the luminance calculation is also influenced by other parameters such as the scene size, the distance between the scene and the sensor.

8.2.2. Automatic Framing

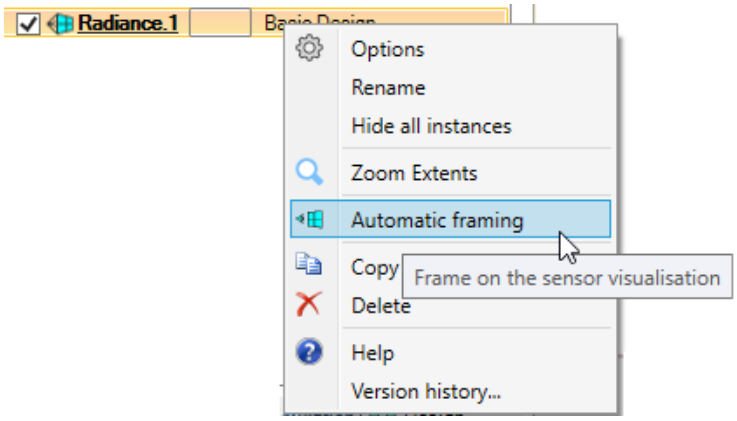
This page describes the Automatic Framing tool that allows you to reframe the 3D view camera to obtain a sensor's point of view.

> **Note:** Automatic framing is only available for Radiance, Human Eye, Camera, LIDAR, Observer and Immersive sensors.

Automatic framing is a tool used to visualize the point of view of a sensor to see what this sensor will capture during simulation. It is useful to identify an incorrect framing or test different fields of view before computing a simulation.

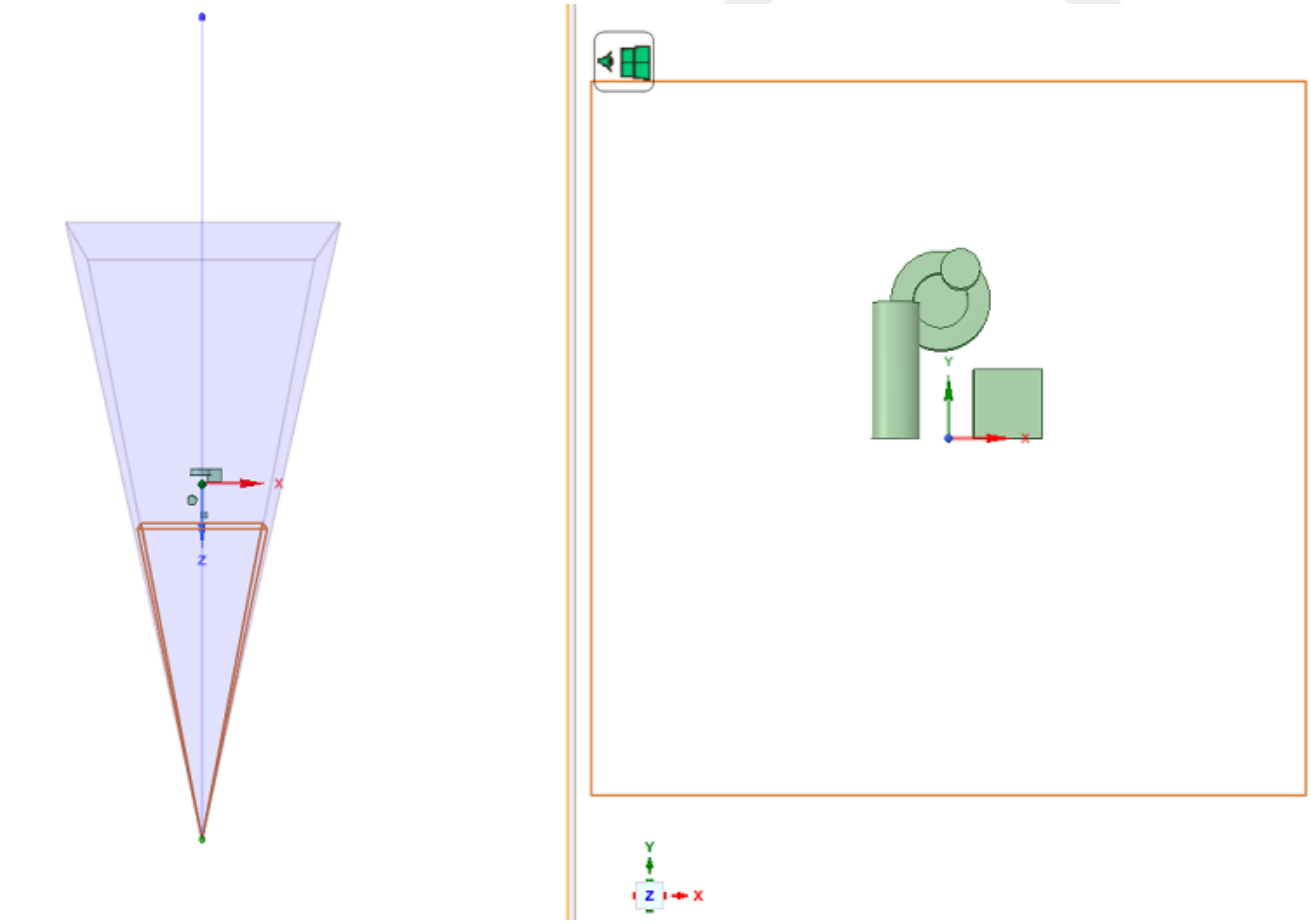
The automatic framing viewing mode can be activated at any time during a sensor edition from the 3D view  or from the sensor's contextual menu. To deactivate this view, just click back either on the icon or on the option in the contextual menu.

> **Note:** Deactivating the Automatic Framing does not return to the view displayed before activating the Automatic Framing.



This tool also offers a dynamic visualization as the view automatically updates the sensor's field of view when editing sensor parameters like the sampling, definition type, active vision field etc

Figure 26. Example of Automatic Framing and Camera View for a Radiance Sensor



Visualization of the sensor's active vision field using the **Camera** options

Automatic Framing option activated

Related tasks

Creating an Observer Sensor on page 216
Creating an observer sensor allows you to create a specific view point from where you can observe the system. This sensor is useful to visualize the optical system from user defined points of view.

Creating an Immersive Sensor on page 212
This page shows how to create an Immersive Sensor to visualize what is viewed from a specific point of view of a scene. An immersive sensor allows you to generate an .speos360 file that can be visualized in Virtual Reality Lab.

Related information

Sensor Creation on page 171
This section gathers all procedures allowing to create a sensor. At least one sensor must be created in an optical system to gather and interpret rays.

8.3. Sensor Creation

This section gathers all procedures allowing to create a sensor. At least one sensor must be created in an optical system to gather and interpret rays.

8.3.1. Irradiance Sensor

An Irradiance Sensor allows you to compute irradiance (in watt/m^2) or illuminance (in Lux).

8.3.1.1. Creating an Irradiance Sensor

This page shows how to create an Irradiance Sensor that computes and analyzes irradiance and illuminance distribution. The irradiance sensor can be created with different integration types allowing you to integrate specific light directions in the sensor.

To create an Irradiance Sensor:

1. From the **Light Simulation** tab, click **Irradiance**  .
   The sensor appears in the 3D view and is placed on the origin of the assembly.
2. In the 3D view, set the **Axis System** of the sensor by clicking  to select one point for the origin,  to select a line for the X axis,  to select a line for the Y axis or click  and select a coordinate system to autofill the Axis System.

   > **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

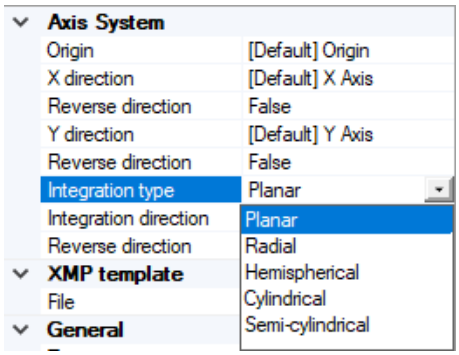
   Three bold arrows indicate the axis system of the sensor. The blue arrow corresponds to the Z axis and indicates the integration direction of the sensor.

DRAFT

> **Note:** Make sure the sensor is not tangent to a geometry.

> **Tip:** To adjust the sensor position and orientation dynamically from the 3D view, you can also use the **Move** option (**Design** tab).

3. From the **Integration type** drop-down list, select how the light should be integrated to the sensor.



- Select Planar for an integration that is made orthogonally with the sensor plan.
- Select Radial, Hemispherical, Cylindrical, Semi-cylindrical if you need to follow specific street lighting illumination regulations.

> **Note:** The **Planar** integration type suits most of the configurations.
>
> Other integration types available are dedicated to specific lighting regulations as they allow to study light in a specific plan (it can be useful to analyze the light contribution on road signs, pedestrian walkways or the road itself).

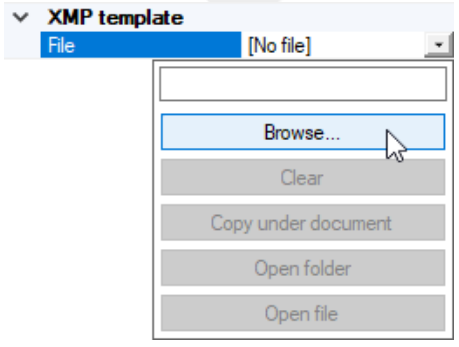
4. If you selected **Planar** or **Semi-cylindrical** integration types, define an integration direction:

   a) In the 3D view, click  .

   - For **Planar**, select a direction in the 3D view.
   - For **Semi-cylindrical**, select a line that is parallel to the sensor plan.

   a) If you need to adjust the integration direction axis, use **Reverse direction**.

5. If you want to use a XMP Template to define the sensor, in **XMP Template**, click **Browse** to load a *.xml file.



A XMP Template is a *.xml file generated from a XMP result. It contains data and information related to the options of the XMP result (dimensions, type, wavelength and display properties).

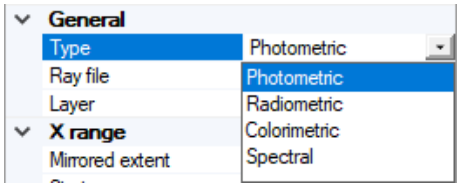
When using a XMP Template, measures are then automatically created in the new *.xmp generated during the simulation based on the data contained in the template file.

- If you want to inherit the axis system of the sensor from the XMP Template file, set **Dimensions from file** to **True**.

  The dimensions are inherited from the file and cannot be edited from the definition panel.

- If you want to define the radiance sensor according to display settings (grid, scale etc.) of the XMP Template, set **Display properties from file** to **True**.

6. In **General**, from the **Type** drop-down list:



- Select **Photometric** if you want the sensor to consider the visible spectrum and get the results in lm/m^2 or lx.

> **Note:** In case of a photometric result generation, the International Commission on Illumination (CIE) defines the visible spectrum as follows: "There are no precise limits for the spectral range of visible radiation since they depend upon the amount of radiant flux reaching the retina and the responsivity of the observer. The lower limit is generally taken between 360 nm and 400 nm and the upper limit between 760 nm and 830 nm".

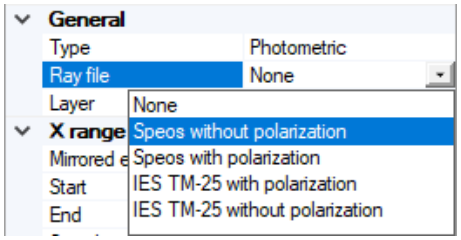
- Select **Radiometric** if you want the sensor to consider the entire spectrum and get the results in W/m^2.

> **Note:** With both Photometric and Radiometric types, the illuminance levels are displayed with a false color and you cannot make any spectral or color analysis on the results.

- Select **Colorimetric** to get the color results without any spectral data or layer separation (in lx or W/m^2).
- Select **Spectral** to get the color results and spectral data separated by wavelength (in lx or W/m^2).

> **Note:** Spectral results take more time to compute as they contain more information.

7. If you want to generate a ray file containing the rays that will be integrated to the sensor, from the **Ray file** drop-down list, select the ray file type:



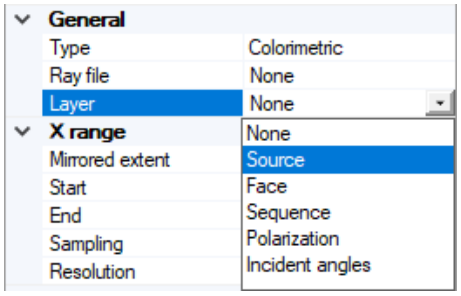
- Select **SPEOS without polarization** to generate a ray file without polarization data.

- Select **SPEOS with polarization** to generate a ray file with the polarization data for each ray.
- Select **IES TM-25 with polarization** to generate a .tm25ray file with polarization data for each ray.
- Select **IES TM-25 without polarization** to generate a .tm25ray file without polarization data.

> **Note:** The size of a ray file is approximately 30MB per million of rays. Consider freeing space on your computer prior to launching the simulation.

> **CAUTION:** If the surface that will be used for the creation of the ray file is fully absorbent (SOP set to Mirror 0%), all rays are absorbed and no ray will be integrated in the ray file. This ray file will be empty.

8. From the **Layer** drop-down list:



- Select **None** to get the simulation's results in one layer.
- Select **Source** if you have created more than one source and want to include one layer per active source in the result.

> **Note:** You can change the source's power or spectrum with the Virtual Lighting Controller in the **Virtual Photometric Lab** or in the **Virtual Human Vision Lab**.

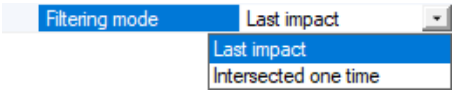
- Select **Face** to include one layer per surface selected in the result.

> **Tip:** Separating the result by face is useful when working on a reflector analysis.

- In the 3D view click  and select the contributing faces you want to include for layer separation.

> **Tip:** Select a group (**Named Selection**) to separate the result with one layer for all the faces contained in the group.

- Select the **filtering mode** to use to store the results (*.xmp):



- **Last Impact**: with this mode, the ray is integrated in the layer of the last hit surface before hitting the sensor.

- **Intersected one time**: with this mode, the ray is integrated in the layer of the last hit selected surface if the surface has been selected as a contributing face or the ray intersects it at least one time.

- Select **Sequence** to include one layer per sequence in the result.
  - Define the **Maximum number of sequences** to calculate.
  - Define the sequences per **Geometries** or **Faces**.

> **Note:** Separating the result by sequence is useful if you want to make a Stray Light Analysis. For more information, refer to Stray Light Analysis.

- Select **Polarization** to include one layer per Stokes parameter using the polarization parameter.

  Stokes parameters are displayed using the layers of the **Virtual Photometric Lab**.

- Select **Incident angles** to include one layer per range of incident angles, and define the **Sampling**.

  For more information on the data separated by Incident angles, refer to Understanding the Incident Angles Layer Type.

9. Define the dimensions of the sensor:

X range	
Mirrored extent	False
Start	-50 mm
End	50 mm
Sampling	100
Resolution	1 mm
Y range	
Mirrored extent	False
Start	-50 mm
End	50 mm
Sampling	100
Resolution	1 mm

- If you want to symmetrize the sensor by linking **Start** and **End** values, set **Mirrored extent** to **True**.
- Edit the **Start** and **End** positions of the sensor on X and Y axes.

> **Tip:** You can either use the manipulators of the 3D view to adjust the sensor or directly edit the values from the definition panel.

- Adjust the sampling of the sensor. The sampling corresponds to the number of pixels of the XMP map.

> **Note:** Speos supports a maximum resolution of 23170 * 23170 pixels.

10. If you selected **Spectral** or **Colorimetric** as sensor type, set the spectral range to use for simulation.

Wavelength	
Start	400 nm
End	700 nm
Sampling	13
Resolution	25 nm

DRAFT

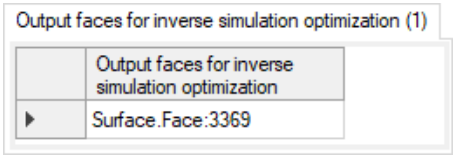
- Edit the **Start** (minimum wavelength) and **End** (maximum wavelength) values to determine the wavelength range to be considered by the sensor.
- If needed, in **Sampling**, adjust the number of wavelengths to be computed during simulation.

  The Resolution is automatically computed according to the sampling and wavelength start and end values.

11. If you intend to use the sensor for an inverse simulation, define the Output faces that the rays generated from the sensor will aim at during the simulation to improve performance:

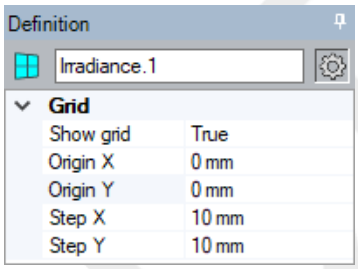
a) In the 3D view, click  .

b) Select the **Output faces** you want the sensor to focus on for the inverse simulation.



> **Important:**
>
> In simulation:
>
> - In CPU, for each pixel per pass, if the ray emitted by the CPU does not intersect an output face, the CPU will emit again a ray until the ray intersects an output face.
> - In GPU, for each pixel per pass, the GPU emits one ray whatever it intersects or not an output face. GPU does not emit again if the ray does not intersect an output face.
>
> That means, for a same number of pass, CPU does converge better than GPU. To get the same result on GPU, you need to increase the number of pass.

12. If you want to display sensor grid in the 3D view, in **Optional or advanced settings**  set **Show grid** to **True**.



The Irradiance Sensor is created and appears both in Speos tree and in the 3D view.

Related concepts

Understanding Integration Types on page 177
This section gathers and describes the different integration types available in Speos. An integration type allows you to define how light is going to be integrated to a sensor.

Related information

Sensors Overview on page 163
Sensors integrate rays coming from the source to analyze the optical result.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

8.3.1.2. Understanding Integration Types

This section gathers and describes the different integration types available in Speos. An integration type allows you to define how light is going to be integrated to a sensor.

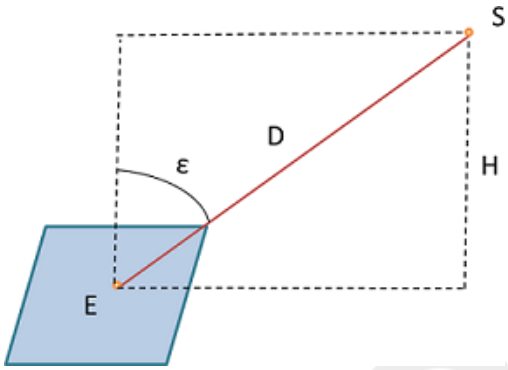
8.3.1.2.1. Planar Illuminance

This page describes how Planar Illuminance is calculated and integrated to the sensor (horizontally or vertically).

Planar Illuminance Overview

The illuminance on a point is calculated by the cosine on the angle of incidence ε. The formula is $E = \frac{I_t \cos \varepsilon}{D^2}$.

Figure 27. Planar mode with an integration direction normal to the sensor plan



Types of Integration Direction of Planar Illuminance

On the following examples, the source is simulated by a local point source.

> **Note:** The source can also be an extended light source like luminaire, ambient, surface source, etc.
>
> The pixel is only sensitive on one side. Its sensitivity is lambertian.

Horizontal plan

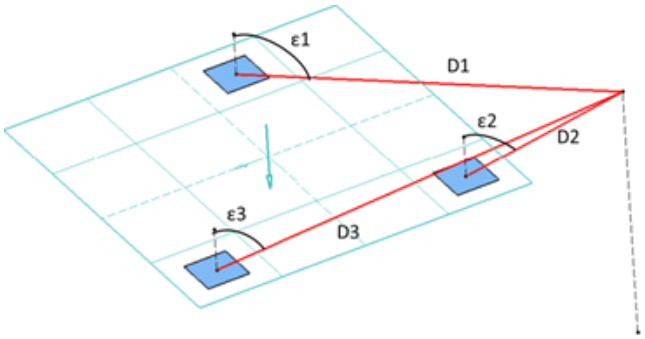
The horizontal illuminance is the most common way to calculate illuminance.

- The integration direction is perpendicular to the horizontal plan and the surface sensor.
- The normal illuminance follows the Bouguer law.
- The formula is

$$E_{Horizontal} = \frac{I_t \cos \varepsilon}{D^2}$$

DRAFT

Figure 28. Illuminance on horizontal sensor



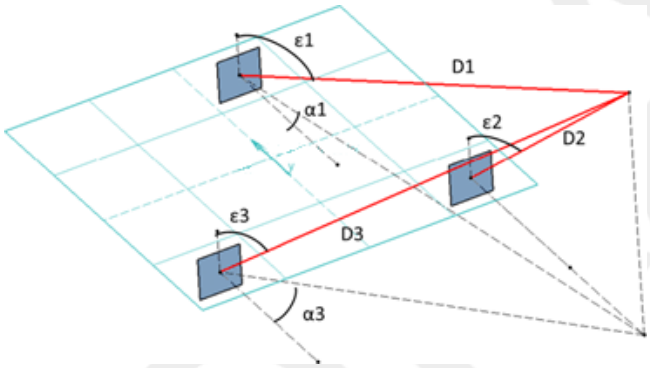
Vertical plan

When the surface sensor is applied vertically, the lateral orientation becomes an important parameter to determinate the illuminance.

The integration direction is perpendicular to the vertical plan and parallel to the surface sensor (that is, a wall on the road).

The formula is $E_{Vertical} = \frac{I_t \cos\alpha \sin\varepsilon}{D^2}$

Figure 29. Illuminance on vertical sensor with lateral deviation (angle α)



> **Tip:** In the specific case where α is equal to 0, the illuminance calculation is the same as for the horizontal type. It does not depend on the α factor. Only the mechanical plan is different, so the two coordinates systems have different orientations. $E_{Vertical\,Specific} = \frac{I_t \sin\varepsilon_2}{D_2{}^2}$

General case

In the general case, you must define the integration direction.

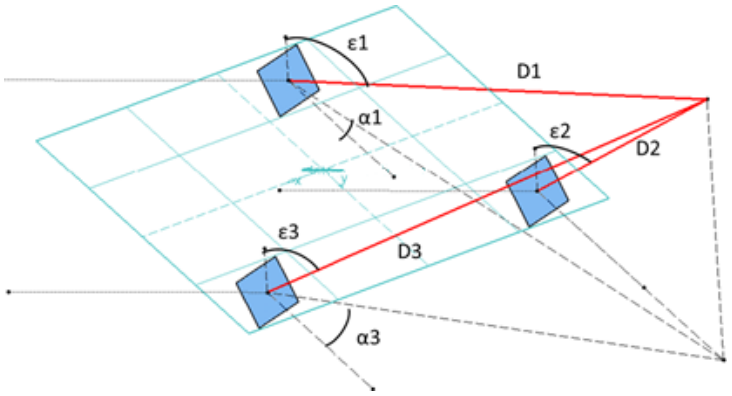
The same integration direction is applied on each pixel of the sensor.

On the figure below, the integration direction is perpendicular to the blue mechanical plans.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Figure 30. Planar illuminance in general case



Related tasks

Creating an Irradiance Sensor on page 171
This page shows how to create an Irradiance Sensor that computes and analyzes irradiance and illuminance distribution. The irradiance sensor can be created with different integration types allowing you to integrate specific light directions in the sensor.

Creating an Irradiance Sensor
Creating an Irradiance Sensor

8.3.1.2.2. Radial Illuminance

This page describes how Radial Illuminance is calculated.

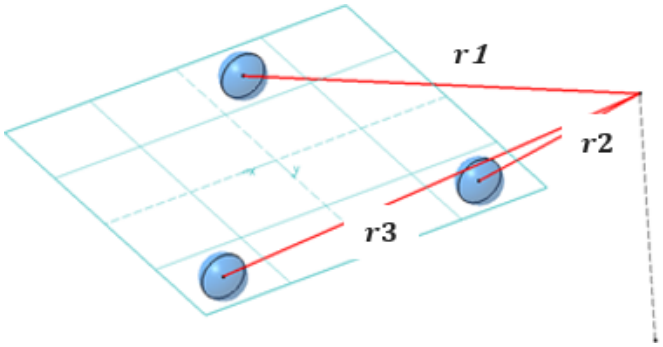
On the following example, the source is simulated by a local point source.

> **Note:** The source can also be an extended light source like luminaire, ambient, surface source, etc.
>
> The sensitivity of the pixel does not depend on where the rays are coming from.

The calculation is based on the standard EN-13201, which gives mathematical formulas equivalent to different types of illuminance. Compared to the EN-13201 standard, several parameters are simplified.

The integration direction is the incident flux. This direction is on the vertical plan at right-angle to the surface. Then, the angle of incident ε is equal to 0° and cos ε = 1.

The illuminance formula is $E_{Radial} = \frac{I_t}{r^2}$



The formula $E_{Radial} = \frac{I_t}{r^2}$ is derived from

$E = \frac{d\phi_t}{dA}$ Insert $\frac{d\Omega}{d\Omega}$

$E = \frac{d\phi_t}{d\Omega} \cdot \frac{d\Omega}{dA}$ Simplify with $I_t = \frac{d\phi_t}{d\Omega}$

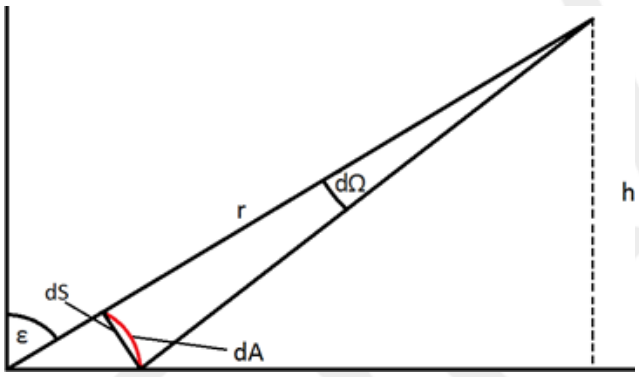
$E = I_t \cdot \frac{d\Omega}{dA}$ Insert $\frac{dS}{dS}$

$E = I_t \cdot \frac{d\Omega}{dS} \cdot \frac{dS}{dA}$ Simplify with $d\Omega = \frac{dS}{r^2}$

$E = \frac{I_t}{r^2} \cdot \frac{dS}{dA}$

At this step, the integration directions are distinct in radial, plane.

$dA \approx dS$ insert in Formula (dA is a spherical surface)



dA and dS are surfaces.

Related tasks

Creating an Irradiance Sensor on page 171
This page shows how to create an Irradiance Sensor that computes and analyzes irradiance and illuminance distribution. The irradiance sensor can be created with different integration types allowing you to integrate specific light directions in the sensor.

Creating an Irradiance Sensor

Creating an Irradiance Sensor

8.3.1.2.3. Hemispherical Illuminance

This page describes how Hemispherical Illuminance is calculated.

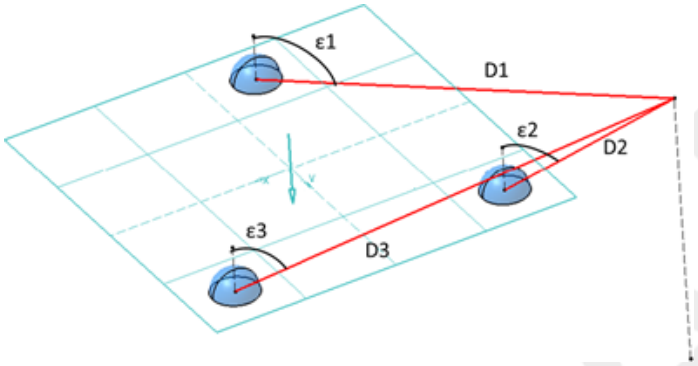
On the following example, the source is simulated by a local point source.

> **Note:** The source can also be an extended light source like luminaire, ambient, surface source, etc.
>
> The sensor is sensible to light incoming from all directions except the direction exactly opposed to the integration direction.

The hemispherical illuminance is an addition of horizontal and radial illuminance.

- The integration direction is perpendicular to the sensor plane.
- The integration direction is the same as horizontal illuminance (perpendicular to the horizontal plan).
- The formula is

$$E_{Hemispherical} = \frac{I_t(1+\cos\varepsilon)}{4\,D^2}$$



Related tasks

Creating an Irradiance Sensor on page 171
This page shows how to create an Irradiance Sensor that computes and analyzes irradiance and illuminance distribution. The irradiance sensor can be created with different integration types allowing you to integrate specific light directions in the sensor.

Creating an Irradiance Sensor

Creating an Irradiance Sensor

8.3.1.2.4. Cylindrical Illuminance

This page describes how Cylindrical Illuminance is calculated.

On the following example, the source is simulated by a local point source.

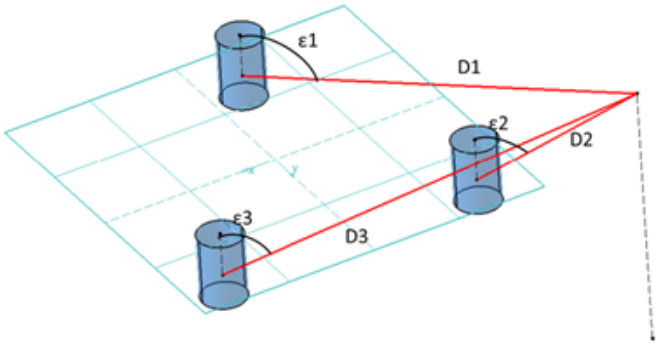
> **Note:** The source can also be an extended light source like luminaire, ambient, surface source, etc.
>
> The sensor is sensible to light coming from all directions except the direction exactly normal to the sensor plane.

The cylindrical illuminance can be defined by the specific case of vertical illuminance (when α = 0°).

DRAFT

Because of the rotational symmetry (around z axis) only the angle ε is important. In that case, you do not need a specific integration direction.

The formula is $E_{Cylindrical} = \frac{I_t \sin \varepsilon}{\pi D^2}$



Related tasks

Creating an Irradiance Sensor on page 171
This page shows how to create an Irradiance Sensor that computes and analyzes irradiance and illuminance distribution. The irradiance sensor can be created with different integration types allowing you to integrate specific light directions in the sensor.

Creating an Irradiance Sensor
Creating an Irradiance Sensor

8.3.1.2.5. Semi-cylindrical Illuminance

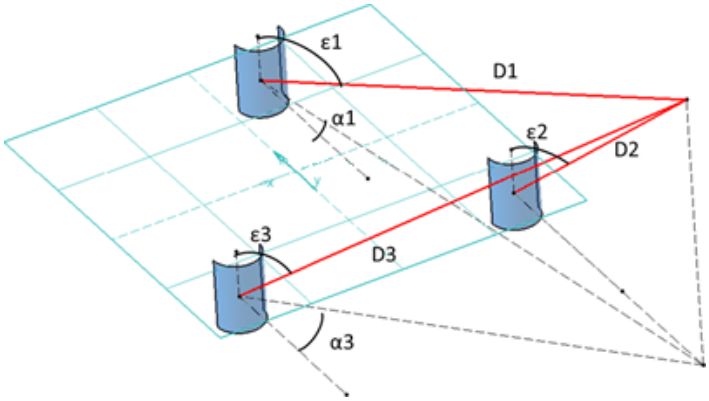
This page describes how Semi-cylindrical Illuminance is calculated.

On the following example, the source is simulated by a local point source.

> **Note:** The source can also be an extended light source like luminaire, ambient, surface source, etc.
>
> The sensor is sensible to light coming from all directions, except the directions included in a half plan delimited by the cylinder axis and situated behind the half cylinder.

Contrary to the cylindrical illuminance, you need an integration direction to calculate the semi-cylindrical illuminance. In addition, the illuminance depends on the lateral deviation (like the vertical illuminance).

The formula is $E_{Hemicylindrical} = \frac{I_t \sin \varepsilon \ (1 + \cos \alpha)}{\pi D^2}$



Related tasks

Creating an Irradiance Sensor on page 171
This page shows how to create an Irradiance Sensor that computes and analyzes irradiance and illuminance distribution. The irradiance sensor can be created with different integration types allowing you to integrate specific light directions in the sensor.

Creating an Irradiance Sensor
Creating an Irradiance Sensor

8.3.1.3. Understanding the Incident Angles Layer Type

The layer type Incident Angles separates the data by ranges of incident angles. That means, one layer corresponds to one incident angles range.

Ranges are defined according to different factors:

- The sampling of the layer: the sampling determines how many layer will be generated.
- The type of integration: according to the integration type, the total integration angle is different.
  - In Planar integration, the integration is made in [0;90]
  - In all other integration, the integration is made in [0;180]
- The Integration Direction in case of a Planar or Semi-cylindrical integration type.

  Rays are integrated in the different layers according to the Integration Direction. A ray with an angle theta in relation to the integration direction will be integrated in the layer min angle <= theta < max angle.

Planar Integration Case

The Planar Integration is calculated on an angle of integration included in [0;90] with 0 corresponding to the direction of integration.

The direction of integration is used to determine how pixels are oriented on the sensor plane. The direction of integration corresponds to the inverse of the normal to the pixels.

Example

The following example presents you two sensors with a planar integration type separated by incident angles.

The sampling defined is 6 meaning you will generate the 6 following layers:

- Layer 0-15

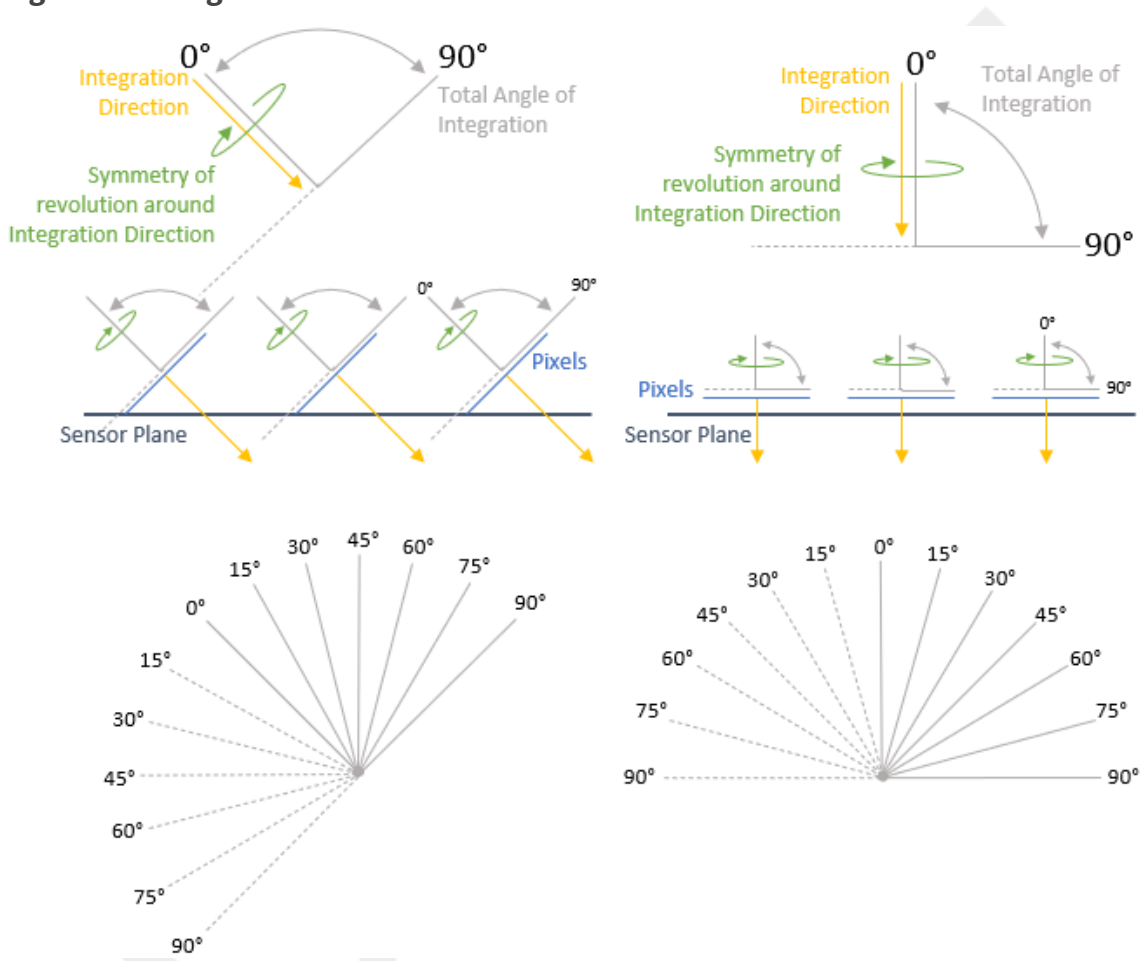
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

- Layer 15-30
- Layer 30-45
- Layer 45-60
- Layer 60-75
- Layer 75-90

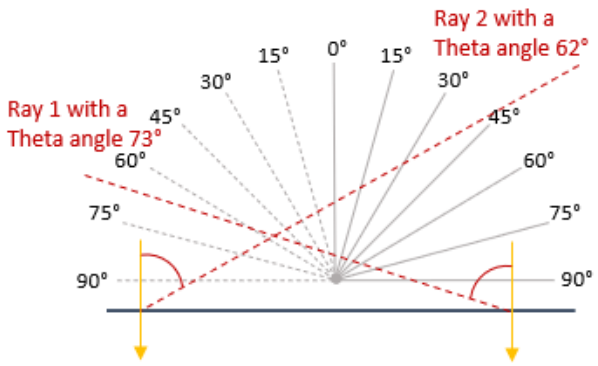
In the figure 1, you can see that the Integration Direction determines how pixels are oriented, which therefore will generate different results according to the rays that will be integrated in each layer.

Figure 31. Integration Direction effect



In the figure 2, both Ray 1 and Ray 2 will be integrated in the Layer 60-75.

Figure 32. Rays Integration



Semi-Cylindrical Integration Case

The Semi-Cylindrical Integration is calculated on an angle of integration included in [0;180]; 0 being the arrow tail of the integration direction and 180 the arrowhead.

The Integration Direction of a Semi-Cylindrical integration is always parallel to the sensor plane.

Example

The following example presents you a sensor with a semi-cylindrical integration type separated by incident angles.

The sampling defined is 12 meaning you will generate the 12 following layers:

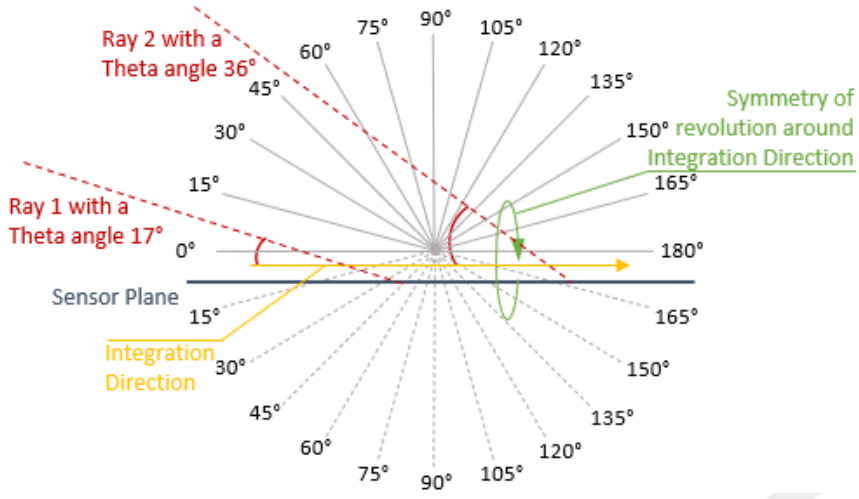
- Layer 0-15
- Layer 15-30
- Layer 30-45
- Layer 45-60
- Layer 60-75
- Layer 75-90
- Layer 90-105
- Layer 105-120
- Layer 120-135
- Layer 135-150
- Layer 150-165
- Layer 165-180

The ray 1 hits the sensor plane with a theta angle of 17° in relation to the Integration Direction. Therefore, it will be integrated in the Layer 15-30.

The ray 2 hits the sensor plane with a theta angle of 36° in relation to the Integration Direction. Therefore, it will be integrated in the Layer 30-45.

DRAFT

Figure 33. Rays Integration in Semi-Cylindrical Integration Type

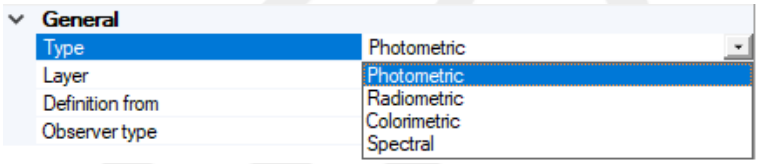


8.3.2. Creating a Radiance Sensor

This page shows how to create a Radiance Sensor that computes and analyzes radiance (in watt/m^2) and luminance (in candela/m^2) distribution. The radiance sensor can be created from two points of view (from the point of view of an observer or from the frame).

To create a Radiance Sensor:

1. From the **Light Simulation** tab, click **Radiance**  .
   The sensor appears in the 3D view and is placed on the origin of the assembly.
2. In **General**, from the **Type** drop-down list:



- Select **Photometric** if you want the sensor to consider the visible spectrum and get the results in cd/m^2 or lm/sr/m^2.

> **Note:** In case of a photometric result generation, the International Commission on Illumination (CIE) defines the visible spectrum as follows: "There are no precise limits for the spectral range of visible radiation since they depend upon the amount of radiant flux reaching the retina and the responsivity of the observer. The lower limit is generally taken between 360 nm and 400 nm and the upper limit between 760 nm and 830 nm".

- Select **Radiometric** if you want the sensor to consider the entire spectrum and get the results in W/sr/m^2.

> **Note:** With both Photometric and Radiometric types, the illuminance levels are displayed with a false color and you cannot make any spectral or color analysis on the results.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- Select **Colorimetric** to get the color results without any spectral data or layer separation (in cd/m^2 or $W/sr/m^2$).
- Select **Spectral** to get the color results and spectral data separated by wavelength (in cd/m^2 or $W/sr/m^2$).

> **Note:** Spectral results take more time to compute as they contain more information.

3. From the **Layer** drop-down list:
   - Select **None** to get the simulation's results in one layer.
   - Select **Source** if you have created more than one source and want to include one layer per active source in the result.

> **Tip:** You can change the source's power or spectrum with the Virtual Lighting Controller in the **Virtual Photometric Lab** or in the **Virtual Human Vision Lab**.

4. In **Definition from**, select the point of view you want to use for the sensor:
   - **Observer**
     a. In **Focal**, define the distance between the sensor plane and the observer point.
     b. Set the **Axis System** of the sensor by placing one point and two directions in the scene:

Axis System	
Observer point	[Default] Origin
Front direction	[Default] Z Axis
DirectionFront_Reverse	False
Top direction	[Default] Y Axis
DirectionTop_Reverse	False

     - In the 3D view, click  and select one point to place the observer point in the scene.
     - In the 3D view, click  and select a line to define the **Front direction** (corresponding by default to Z axis).
     - In the 3D view, click  and select a line to define the **Top direction** (corresponding by default to Y axis).
     - or click  and select a coordinate system to autofill the Axis System.

> **Note:** Make sure the sensor is not tangent to a geometry.

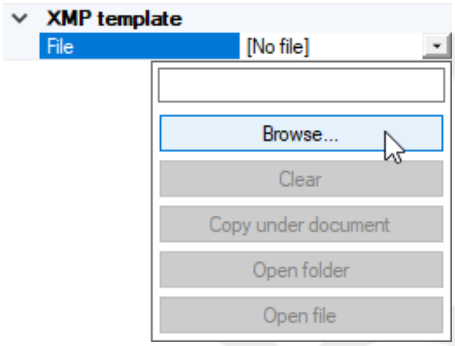
DRAFT

> **Tip:** To adjust the sensor position and orientation dynamically from the 3D view, you can also use the Move option (**Design** tab).

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

c. If you want to use an XMP Template to define the sensor, in **XMP Template**, click **Browse** to load an .xmp file.

> **Note:** An XMP Template is an .xml file generated from an XMP result. It contains data and information related to the options of the XMP result (dimensions, type, wavelength and display properties).
>
> When using an XMP Template, measures are then automatically created in the new .xmp generated during the simulation based on the data contained in the template file.

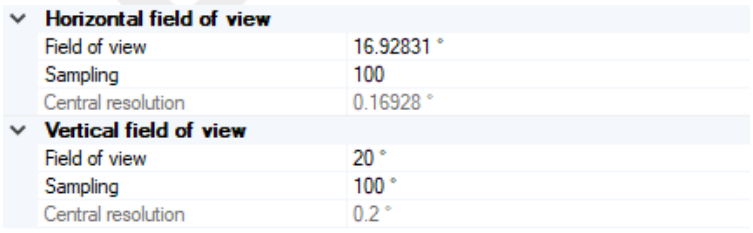


- If you want to inherit the axis system of the sensor from the XMP Template file, set **Dimensions from file** to **True**.

  The dimensions are inherited from the file and cannot be edited from the definition panel.

- If you want to define the radiance sensor according to display settings (grid, scale etc.) of the XMP Template, set **Display properties from file** to **True**.

d. Determine the field of view of the sensor:



- Adjust the field of view both horizontally and vertically.

> **Tip:** You can either use the manipulators of the 3D view to adjust the sensor or directly edit the values from the definition panel.

- Adjust the sampling of the sensor. The sampling corresponds to the number of pixels of the XMP map.

  *The **Central resolution** is automatically calculated and depends on the **Sampling** value.*

> **Note:** The **Central resolution** is the central angular resolution which corresponds to the angular resolution of the pixel located in front of the observer.

- **Frame**

  a. From the **Observer Type** drop-down list, choose how you want to define the distance between the sensor plane and the observer point:

  - Select **Focal** to manually define the distance between the sensor plane and the observer point in mm.
  - Select **Observer** to select a **Focal point** from the 3D view.

    In this case, the focal is automatically computed between the sensor plane and the selected point.

  b. In the 3D view, set the **Axis System** of the sensor by clicking  to select one point for the origin and  ,  to select two lines for X and Y axes or click  and select a coordinate system to autofill the Axis System.

Axis System	
Origin	[Default] Origin
X direction	[Default] X Axis
Reverse direction	False
Y direction	[Default] Y Axis
Reverse direction	False

  If you need to adjust the ray's propagation direction, set **Reverse direction** to **True**.

> **Note:** Make sure the sensor is not tangent to a geometry.

> **Tip:** To adjust the sensor position and orientation dynamically from the 3D view, you can also use the Move option (**Design** tab).

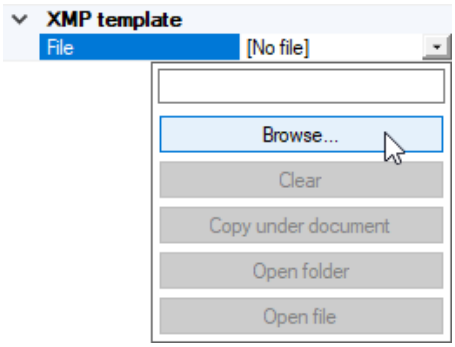
> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

  c. If you want to use an XMP Template to define the sensor, in **XMP Template**, click **Browse** to load an .xmp file.

DRAFT

Note: An XMP Template is an .xml file generated from an XMP result. It contains data and information related to the options of the XMP result (dimensions, type, wavelength and display properties).

When using an XMP Template, measures are then automatically created in the new .xmp generated during the simulation based on the data contained in the template file.

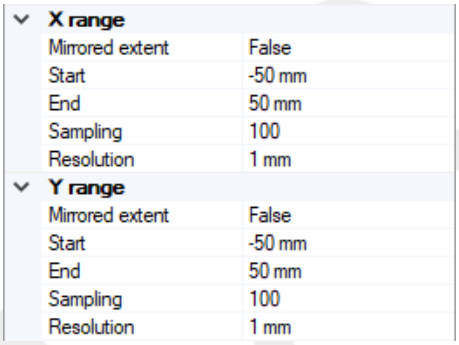


- If you want to inherit the axis system of the sensor from the XMP Template file, set **Dimensions from file** to **True**.

  The dimensions are inherited from the file and cannot be edited from the definition panel.

- If you want to define the radiance sensor according to display settings (grid, scale etc.) of the XMP Template, set **Display properties from file** to **True**.

d. Define the dimensions of the sensor:



- If you want to symmetrize the sensor by linking **Start** and **End** values, set **Mirrored extent** to **True**.
- Edit the **Start** and **End** positions of the sensor on X and Y axes.

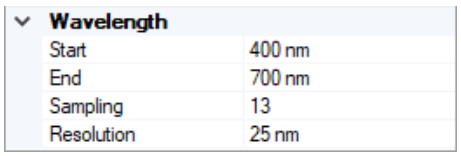
Tip: You can either use the manipulators of the 3D view to adjust the sensor or directly edit the values from the definition panel.

- Adjust the sampling of the sensor. The sampling corresponds to the number of pixels of the XMP map.

Note: Speos supports a maximum resolution of 23170 * 23170 pixels.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

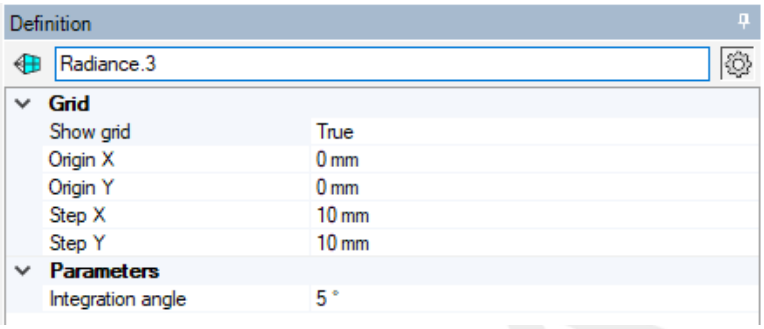
5. If you selected **Spectral** or **Colorimetric** as sensor type, set the spectral excursion to use for simulation.



   - Edit the **Start** (minimum wavelength) and **End** (maximum wavelength) values to determine the wavelength range to be considered by the sensor.
   - If needed, in **Sampling**, adjust the number of wavelengths to be computed during simulation.

     The Resolution is automatically computed according to the sampling and wavelength start and end values.

6. In **Optional or advanced settings** :



   - To display the sensor grid in the 3D view, set **Show grid** to **True**.
   - If needed, adjust the Integration angle.

The Radiance Sensor is created and appears in Speos tree and in the 3D view.

Related concepts

Integration Angle on page 164
This section describes the Integration Angle parameter to better understand its possible impacts on the simulation's results and provides pieces of advices to correctly set it. The integration angle must be defined for various sensors.

Related information

Sensors Overview on page 163
Sensors integrate rays coming from the source to analyze the optical result.

8.3.3. Intensity Sensor

An Intensity Sensor allows you to compute radiant intensity (in watt/sr) or luminance intensity (in candela).

8.3.3.1. Understanding the Parameters of an Intensity Sensor

This page describes advanced settings to set when creating an Intensity Sensor.

DRAFT

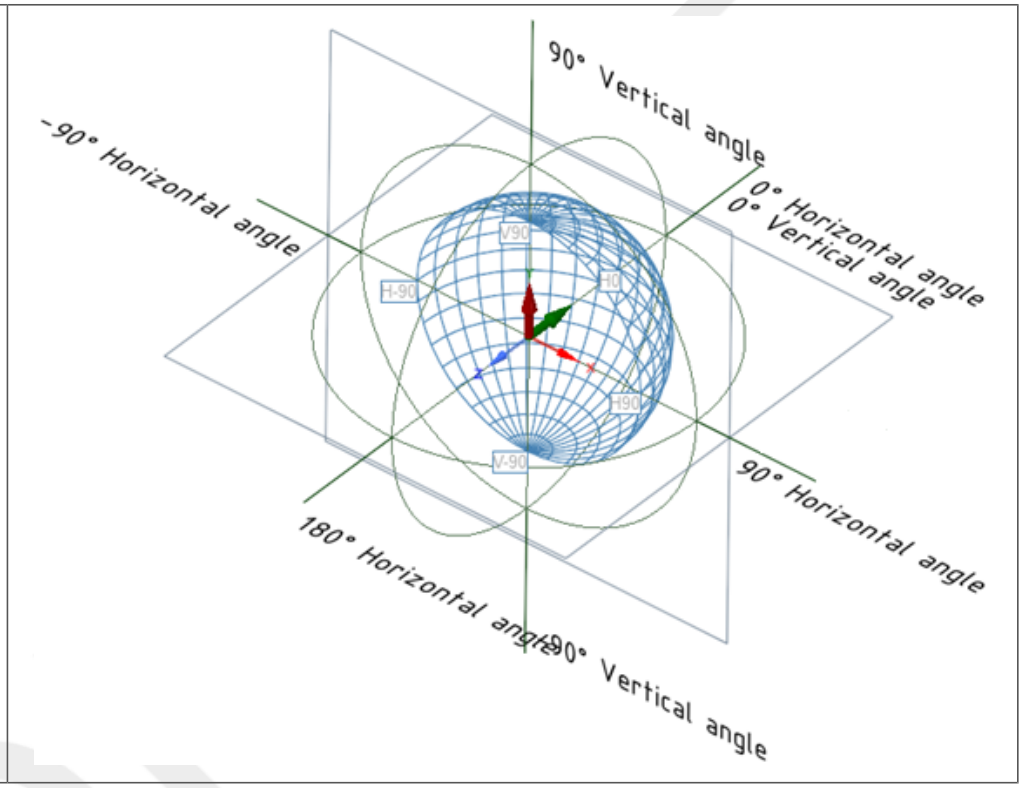
Intensity Sensor Format

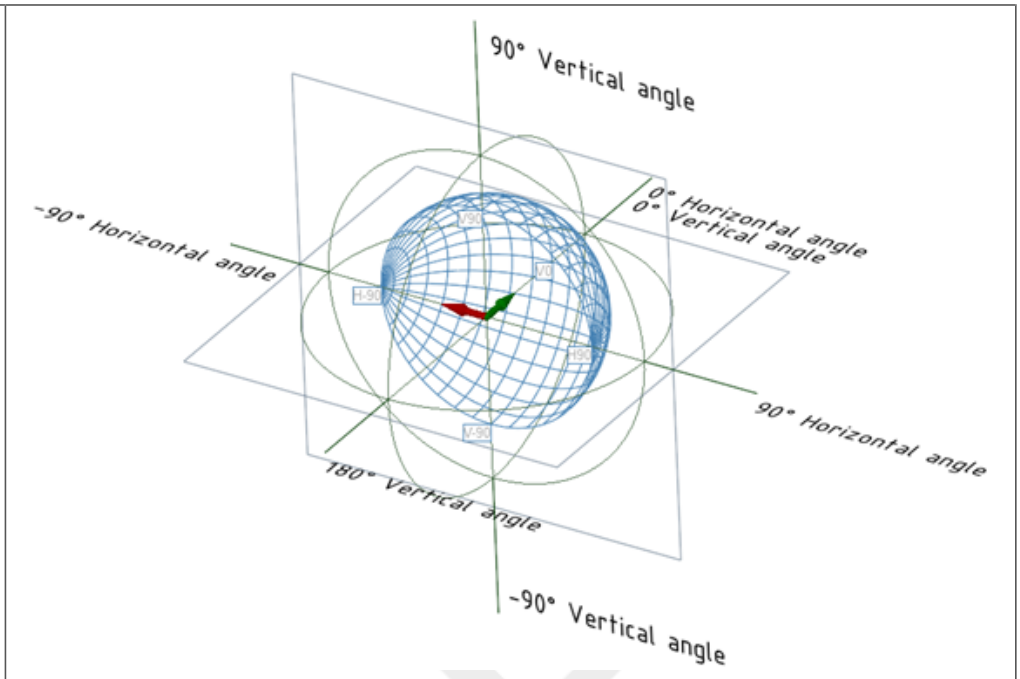
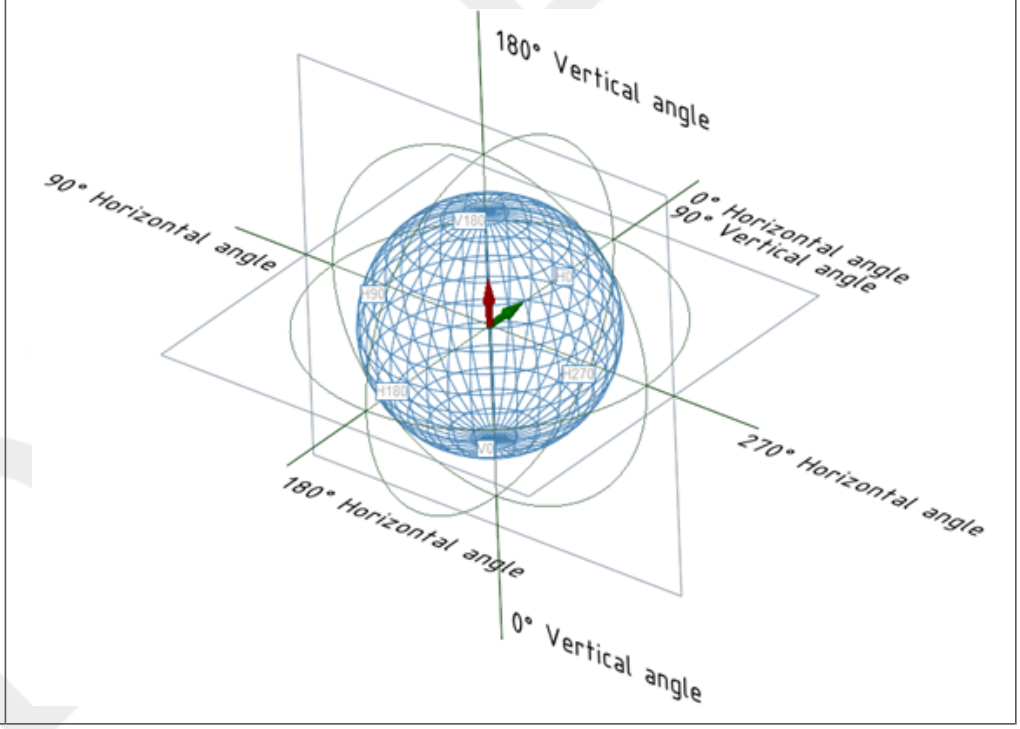
Various sensor formats are available when defining an intensity sensor.

These formats correspond to light distribution standards and will generate different file formats as an output.

Each standard have its own specificities (axes, orientation, integration plan of light etc.).

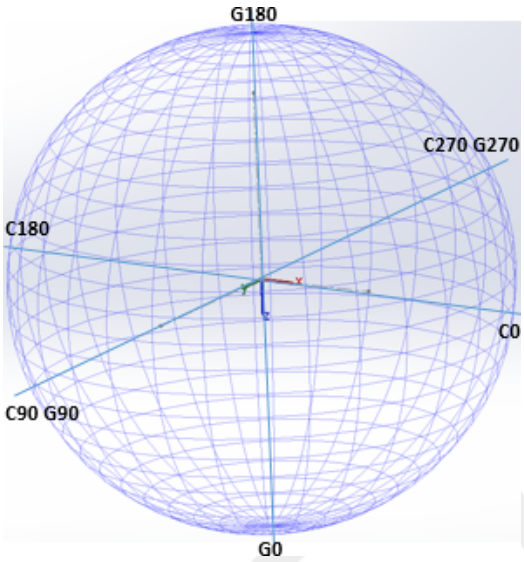
- IESNA files (*.ies, structured text file format) correspond to the U.S. standard data format for storing the spatial light output distribution for luminaires. **IESNA A**, **IESNA B** and **IESNA C** correspond to three different specifications of IES.
- **Eulumdat** files (*.ldt, structured text file format) correspond to the European standard data format for storing the spatial light output distribution for luminaires.

IES A	

IES B	
IES C	

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Eulumdat	

Near Field

Description

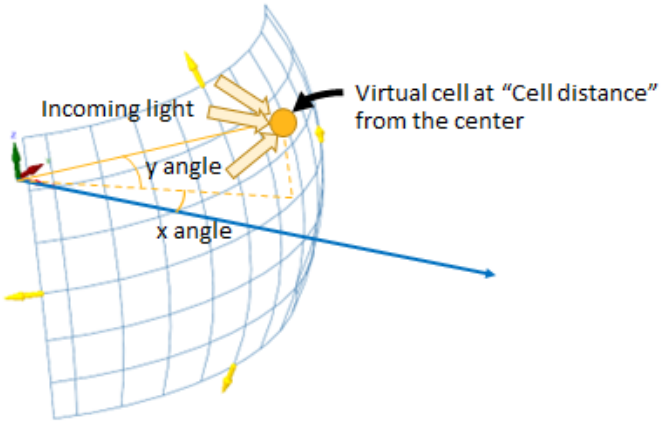
In practice, actual measurement devices measure intensity at a finite distance. In Speos, the Near Field option allows you to model the integration at the finite distance by defining the actual cell of an intensity bench.

The sensor cell is brought closer to the source, therefore bringing the measuring field of the sensor closer to the source. A near-field sensor is useful when wanting to compare simulated intensity to measured intensity on small devices. This option generates results that are physically closer to reality.

Light Contribution

For each pixel of the Intensity Sensor corresponds a virtual cell located at a Cell distance drom the center of the sensor, with a size of Cell diameter.

All incoming light on the cell contributes to the corresponding intensity level.



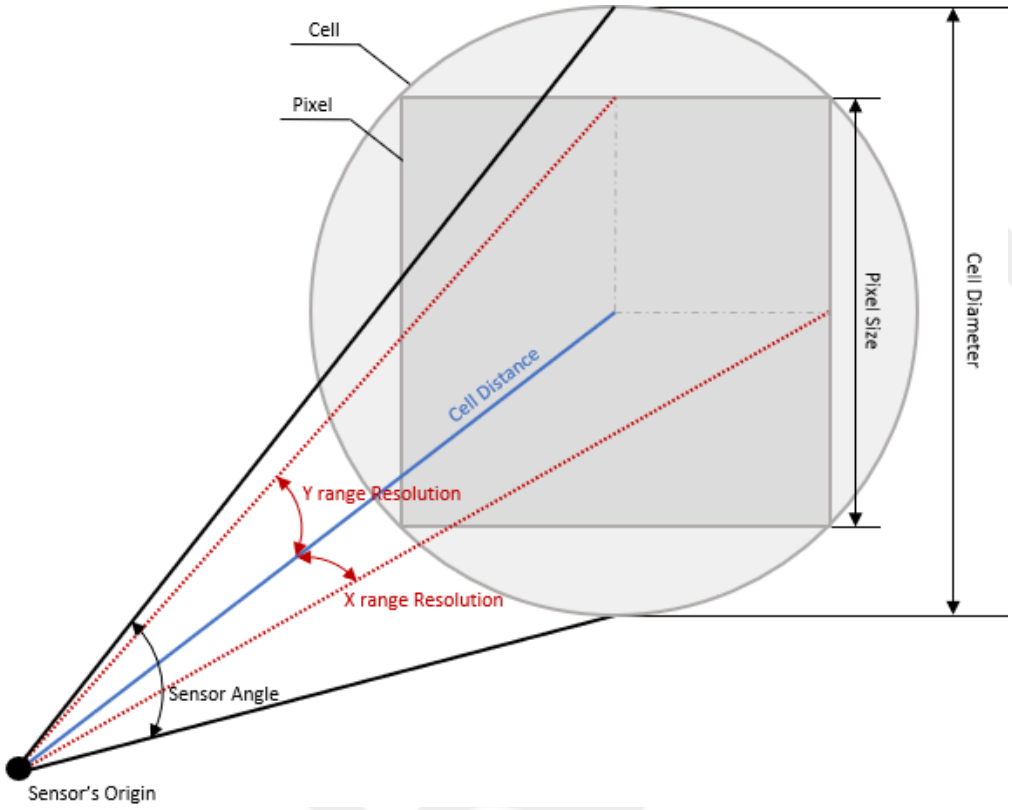
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Cell Description

In practice, the sensor cell is a disk which size is determined by the **Cell diameter** value. This diameter corresponds to the size of the photosensitive sensor that would be used to perform the measure in reality.

The **Cell distance** value defines the sensor visualization in the 3D view.

Figure 34. Cell representation



Make sure to define a **Cell Diameter** bigger than the pixel size. Otherwise, the sensor angle is smaller than the pixel, and some rays will pass between the pixels and will not be considered. These rays are uselessly calculated, however, the result is still correct.

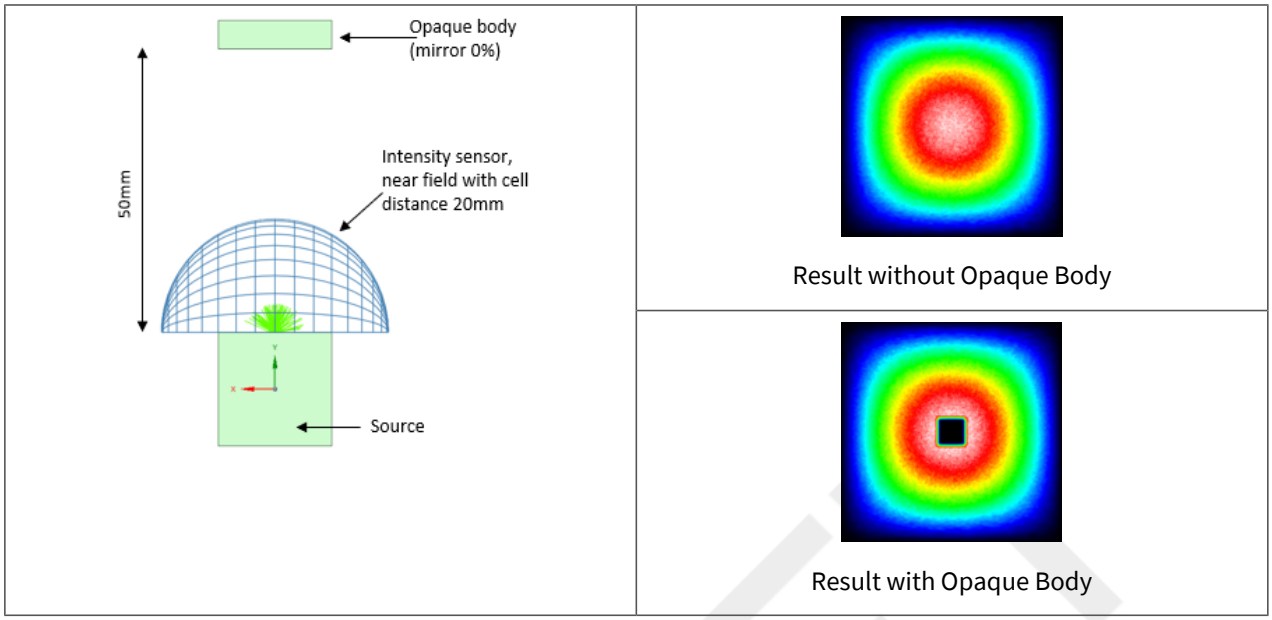
> **Note:** The Sensor angle corresponds to the Integration angle parameter (read-only) of the Intensity Sensor definition. Do not confuse the Sensor angle with the Integration Angle.

If the **Resolution** of X range is different from the **Resolution** of Y range, take the biggest resolution as reference to calculate your sensor system.

To calculate the adequate system use the following equation: **Cell Diameter = tan(res)*Cell Distance*√2**

Near Field and Geometries

When defining a near field intensity sensor, geometries located further than the sensor contribute to the result (see example below).

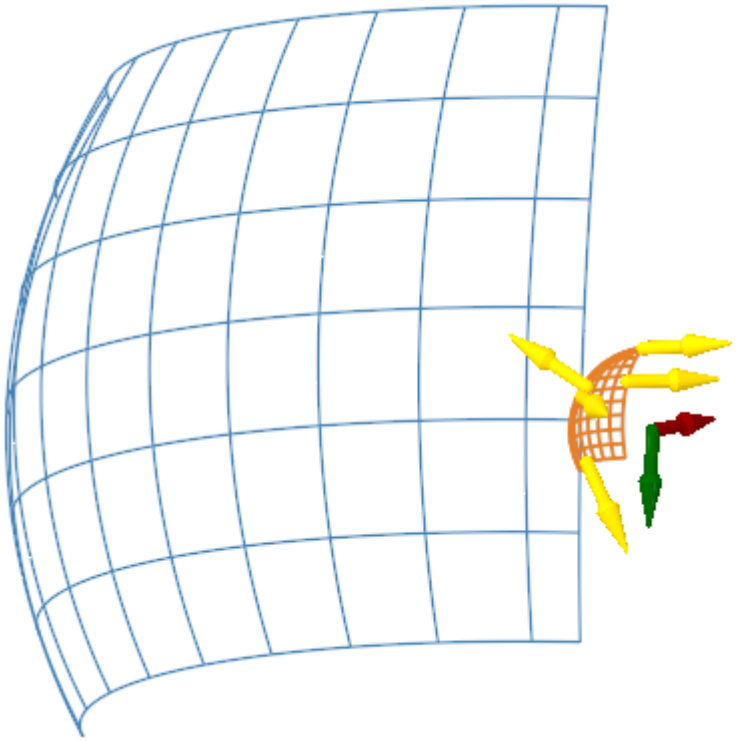


Near Field Limitations

You cannot use the result of near-field sensor to model the near field of a light source.

The result obtained with a near field sensor can be inaccurate on the edge of the map, over a width equal to the radius of a cell.

Figure 35. Near field intensity sensor displayed in the 3D view.



Adaptive Sampling

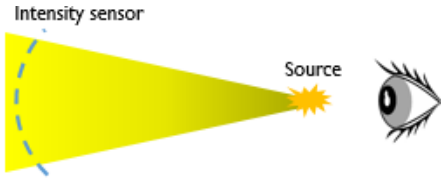
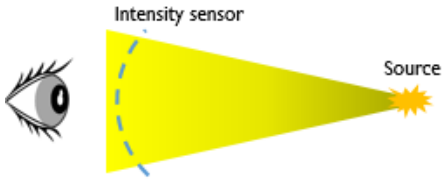
Adaptive sampling allows you to browse a .txt file describing the angles of result to be used for the sensor.

The format of the adaptive sampling file is the following:

For IESNA A and B	For IESNA C and Eulumdat
• Line 1: Header OPTIS Intensity Distribution sampling file v1.0 • Line 2: Comments Example of IES B sampling file • Line 3: Number of samples on H plane and angle values HSamplingNumber -90 Angle 1 Angle 2... AngleN 90. • Line 4: Number of samples on V plane and angle values VSamplingNumber -90 Angle 1 Angle 2... AngleN 90. When the sampling goes from -90 to 90 degrees for an IES A, the sample list has to begin at -90 and finish at 90.	• Line 1: Header OPTIS Intensity Distribution sampling file v1.0 • Line 2: Comments Example of IES C sampling file • Line 3: Number of samples on H plane and angle values HSamplingNumber 0 Angle 1 Angle 2... AngleN 360. • Line 4: Number of samples on V plane and angle values VSamplingNumber -90 Angle 1 Angle 2... AngleN 180.

Intensity Result Viewing Direction

The intensity result viewing direction defines the viewing direction of the observer.

From source looking at sensor: The viewing direction of the observer is the same as the light direction emitted.	
From sensor looking at source: The viewing direction of the observer is in the opposite of the light direction.	

8.3.3.2. Creating an Intensity Sensor

This page shows how to create an Intensity Sensor that computes and analyzes radiant or luminous intensity.

To create an Intensity Sensor:

1. From the **Light Simulation** tab, click **Intensity**  .

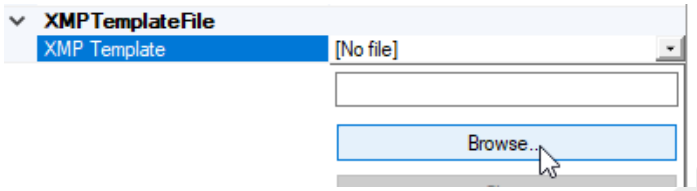
DRAFT

2. Set the **Axis System** of the sensor by clicking to select the origin, to define X axis and to define the Y axis or click and select a coordinate system to autofill the Axis System.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

The rays are integrated on Z plane.

3. If you want to import a template file to define the sensor, in **XMPTemplateFile**, click **Browse** to load an .xml file.



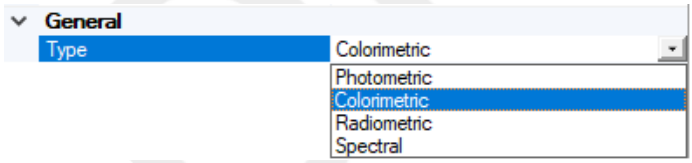
> **Note:** An XMP Template is an .xml file generated from an XMP result. It contains data and information related to the options of the XMP result (dimensions, type, wavelength and display properties).
>
> When using an XMP Template, measures are then automatically created in the new .xmp generated during the simulation based on the data contained in the template file.

- If you want to inherit the axis system of the sensor from the XMP Template file, set **Dimensions from file** to **True**.

  The dimensions are inherited from the file and cannot be edited from the definition panel.

- If you want to define the radiance sensor according to display settings (grid, scale etc.) of the XMP Template, set **Display properties from file** to **True**.

4. In **General**, from the **Type** drop-down list:



- Select **Photometric** if you want the sensor to consider the visible spectrum and get the results in cd.

> **Note:** In case of a photometric result generation, the International Commission on Illumination (CIE) defines the visible spectrum as follows: "There are no precise limits for the spectral range of visible radiation since they depend upon the amount of radiant flux reaching the retina and the responsivity of the observer. The lower limit is generally taken between 360 nm and 400 nm and the upper limit between 760 nm and 830 nm".

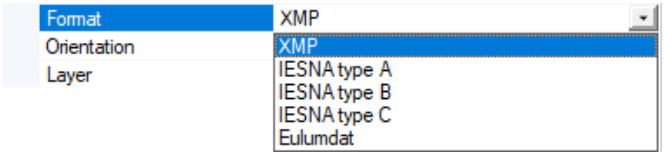
- Select **Radiometric** if you want the sensor to consider the entire spectrum and get the results in W/sr.

> **Note:** With both **Photometric** and **Radiometric** types, the illuminance levels are displayed with a false color and you cannot make any spectral or color analysis on the results.

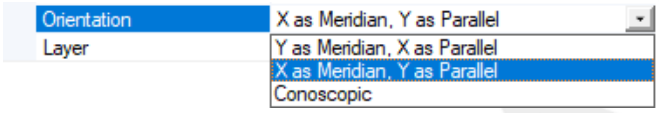
- Select **Colorimetric** to get the color results without any spectral data or layer separation (in cd or W/sr).
- Select **Spectral** to get the results and spectral data separated by wavelength (in cd or W/sr).

> **Note:** Spectral results take more time to compute as they contain more information.

5. From the **Format** drop-down list, select **XMP** to integrate light according to a standard coordinate system.



6. From the **Orientation** drop-down list, define which axis represents the polar axis:



- Select **X as meridian and Y as parallel** to define X as the polar axis.
- Select **Y as meridian and X as parallel** to define Y as the polar axis.

> **Note:** These orientations are often used for automotive regulations.
>
> X as meridian corresponds to the orientation of an IESNA B format
>
> Y as meridian corresponds to the orientation of an IESNA A format.

- Select **Conoscopic** to take Z as the polar axis.

7. From the **Layer** drop-down list:
   - Select **None** to get the simulation's results in one layer.
   - Select **Source** if you have created more than one source and want to include one layer per active source in the result.

> **Tip:** You can change the source's power or spectrum with the Virtual Lighting Controller in the **Virtual Photometric Lab** or in the **Virtual Human Vision Lab**.

   - Select **Face** to include one layer per surface selected in the result.

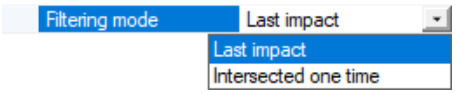
> **Tip:** Separating the result by face is useful when working on a reflector analysis.

DRAFT

- In the 3D view click  and select the contributing faces you want to include for layer separation.

> **Tip:** Select a group (**Named Selection**) to separate the result with one layer for all the faces contained in the group.

- Select the **filtering mode** to use to store the results (in the *.xmp map):



- **Last Impact**: with this mode, the ray is integrated in the layer of the last hit surface before hitting the sensor.
- **Intersected one time**: with this mode, the ray is integrated in the layer of the last hit selected surface if the surface has been selected as a contributing face or the ray intersects it at least one time.

- Select **Sequence** to include one layer per sequence in the result.

8. Define the dimensions of the sensor on X and Y axes:

X range	
Mirrored extent	False
Start	-45 °
End	45 °
Sampling	180
Resolution	0.5 °
Y range	
Mirrored extent	False
Start	-30 °
End	30 °
Sampling	120
Resolution	0.5 °

- If you want to link the start and end points values, set **Mirror extent** to **True**.
- Define the **Start** and **End** points of the sensor on X and Y axes by editing the values in degrees.
- In **Sampling**, define the number of pixels of the XMP map.

  *The **Resolution** is automatically calculated.*

> **Note:** Speos supports a maximum resolution of 23170 * 23170 pixels.

9. If you selected **Colorimetric** or **Spectral** as sensor type, in Wavelength, define the spectral range the sensor needs to consider:

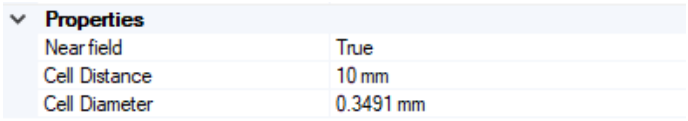
Wavelength	
Start	400 nm
End	700 nm
Sampling	13
Resolution	25 nm

- Edit the **Start** (minimum wavelength) and **End** (maximum wavelength) values to determine the wavelength range to be considered by the sensor.
- If needed, in **Sampling**, adjust the number of wavelengths to be computed during simulation.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

The Resolution is automatically computed according to the sampling and wavelength start and end values.

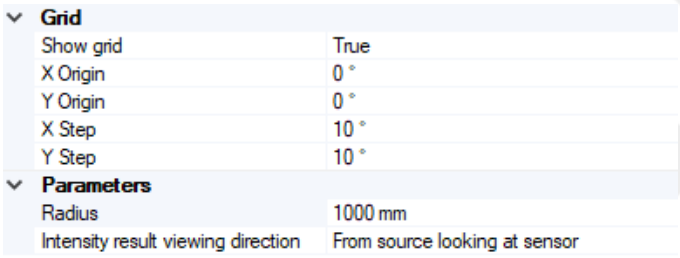
10. If you want to bring the measuring field of the sensor closer to the source, in **Properties** set Near field to **True**.



> **Note:** If **Near field** is deactivated, the intensity is located at the infinite.
>
> The results obtained with a near-field sensor can be inaccurate on the edge of the map, over a width equal to the radius of a cell.

a) In **Cell Distance**, define the position of the sensor in regard to the origin in the 3D view.
b) In **Cell Diameter**, define the actual size of the photosensitive sensor.
Make sure to define a Cell Diameter bigger than the pixel size. For more information refer to Near field.

11. In **Optional or advanced settings**  :



- If needed, adjust the preview of the grid by editing the values.
- If you want to adjust the position preview of the sensor, change the radius (in mm).
- From Intensity result viewing direction, define where the sensor is placed regarding to the source:
  - Select **From source looking at sensor** to position the observer point from where light is emitted.
  - Select **From sensor looking at source** to position the observer in the opposite of light direction.

The intensity sensor is created and is visible both in Speos tree and in the 3D view.

Related tasks

Creating a Polar Intensity Sensor on page 202
This page shows how to create an Intensity Sensor that computes and analyzes radiant or luminous intensity following standard and intensity formats (IESNA format, Eulumdat format).

Related information

Sensors Overview on page 163
Sensors integrate rays coming from the source to analyze the optical result.

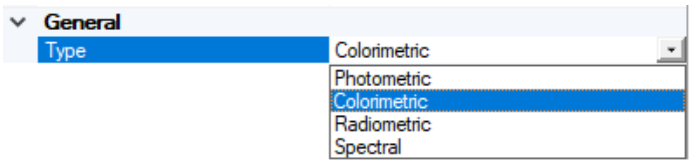
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

8.3.3.3. Creating a Polar Intensity Sensor

This page shows how to create an Intensity Sensor that computes and analyzes radiant or luminous intensity following standard and intensity formats (IESNA format, Eulumdat format).

To create a Polar Intensity Sensor:

1. From the **Light Simulation** tab, click **Intensity**  .
2. In **General**, from the **Type** drop-down list:



- Select **Photometric** if you want the sensor to consider the visible spectrum and get the results in cd.

> **Note:** In case of a photometric result generation, the International Commission on Illumination (CIE) defines the visible spectrum as follows: "There are no precise limits for the spectral range of visible radiation since they depend upon the amount of radiant flux reaching the retina and the responsivity of the observer. The lower limit is generally taken between 360 nm and 400 nm and the upper limit between 760 nm and 830 nm".

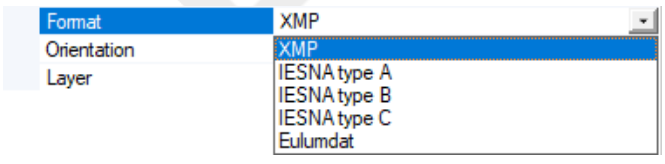
- Select **Radiometric** if you want the sensor to consider the entire spectrum and get the results in W/sr.

> **Note:** With both **Photometric** and **Radiometric** types, the illuminance levels are displayed with a false color and you cannot make any spectral or color analysis on the results.

- Select **Colorimetric** to get the color results without any spectral data or layer separation (in cd or W/sr).
- Select **Spectral** to get the results and spectral data separated by wavelength (in cd or W/sr).

> **Note:** Spectral results take more time to compute as they contain more information.

3. From the **Format** drop-down list, select the type of standard you want to follow:



- Select IESNA type A, B or C if you want to generate an .ies file and follow US standard data format for storing the spatial light output distribution.
- Select **Eulumdat** if you want to generate an .ldt file and follow the European standard data format for storing the spatial light output distribution.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

4. From the **Layer** drop-down list:
   - Select **None** to get the simulation's results in one layer.
   - Select **Source** if you have created more than one source and want to include one layer per active source in the result.

     > **Tip:** You can change the source's power or spectrum with the Virtual Lighting Controller in the **Virtual Photometric Lab** or in the **Virtual Human Vision Lab**.

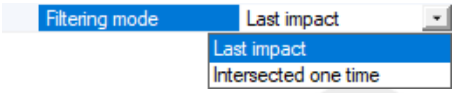
   - Select **Face** to include one layer per surface selected in the result.

     > **Tip:** Separating the result by face is useful when working on a reflector analysis.

     - In the 3D view click  and select the contributing faces you want to include for layer separation.

       > **Tip:** Select a group (**Named Selection**) to separate the result with one layer for all the faces contained in the group.

     - Select the **filtering mode** to use to store the results (in the *.xmp map):

     - **Last Impact**: with this mode, the ray is integrated in the layer of the last hit surface before hitting the sensor.
     - **Intersected one time**: with this mode, the ray is integrated in the layer of the last hit selected surface if the surface has been selected as a contributing face or the ray intersects it at least one time.
     - Select **Sequence** to include one layer per sequence in the result.

5. Set the **Axis System** of the sensor:
   - If you selected **IESNA type B**, click  to select an origin,  to select a line to define the **Polar Axis** and  to select a line to define **VO Axis**.
   - If you selected **IESNA type A**, **IESNA type C** or **Eulumdat**, click  to select an origin,  to select a line to define the **Polar Axis** and  to select a line to define **HO Axis**.

     > **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

6. Define the sensor's sampling for **H Plane** (horizontal plane) and **V Plane** (vertical plane).

   The Resolution is automatically computed.

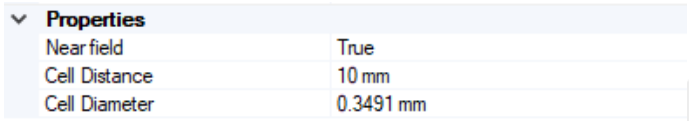
DRAFT

> **Note:** X and Y range define the sensor's dimensions and depend on the sensor format previously selected.

7. If you want to apply a specific sampling to the sensor, in **Adaptive sampling**, click **Browse** to load a .txt file.
8. If you want to bring the measuring field of the sensor closer to the source, in **Properties** set Near field to **True**.

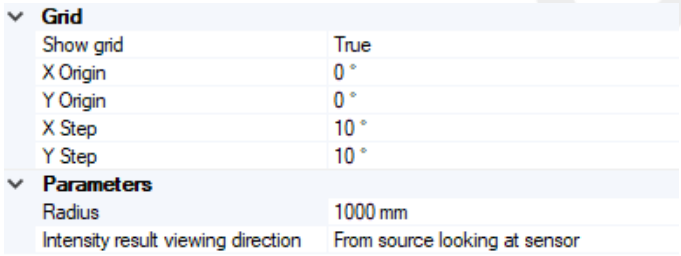
   The near-field sensor appears in the 3D view.

> **Note:** If **Near field** is deactivated, the intensity is located at the infinite.
>
> The results obtained with a near-field sensor can be inaccurate on the edge of the map, over a width equal to the radius of a cell.



- In **Cell Distance**, define the position of the sensor in regard to the origin in the 3D view.
- In **Cell Diameter**, define the actual size of the photosensitive sensor. The size must be superior to one pixel.

9. In **Optional or advanced settings**  :



- If needed, adjust the preview of the grid by editing the values.
- If you want to adjust the position preview of the sensor, change the radius (in mm).
- From Intensity result viewing direction, define where the sensor is placed regarding to the source:
  - Select **From source looking at sensor** to position the observer point from where light is emitted.
  - Select **From sensor looking at source** to position the observer in the opposite of light direction.

The intensity sensor is created and is visible both in Speos tree and in the 3D view.

Related reference

Understanding the Parameters of an Intensity Sensor on page 191
This page describes advanced settings to set when creating an Intensity Sensor.

8.3.4. Human Eye Sensor

The Human Eye Sensor is a physics-based sensor that allows you to accurately simulate human vision by considering the field of view, visual acuity and depth of field (linked to the pupil diameter) of an observer.

> **Important:** This feature is only available under Speos Enterprise license.

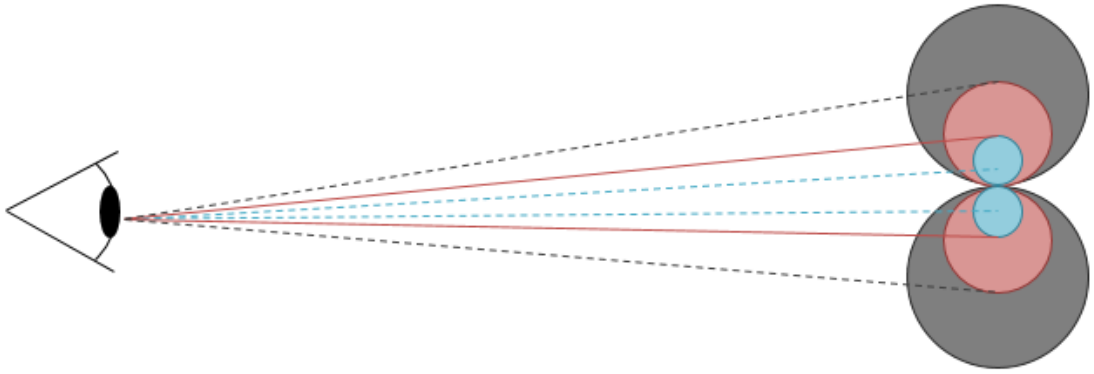
8.3.4.1. Understanding the Parameters of a Human Eye Sensor

This page describes some advanced settings to set when creating a Human Eye Sensor.

Resolution

The Resolution corresponds to the visual acuity of the observer. The visual acuity is commonly expressed in arc minutes (60 arc minute = 1 degree).

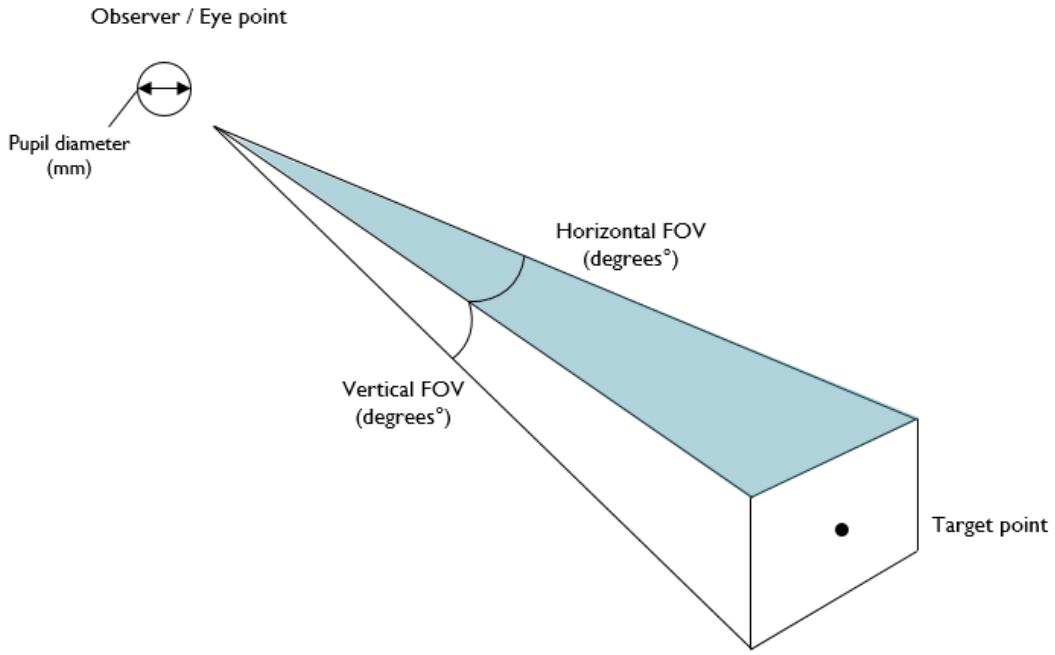
The value taken by as a reference (1 arc minute=0.0167 degrees) corresponds to the normal visual acuity of an observer. The better the visual acuity, the smaller the angle of resolution gets and the finer the details are perceived by the eye.



Normal visual acuity = 1 arc minute
Lower visual acuity = 2 arc minutes
Better visual acuity = 0.5 arc minutes

Field Of View

The standard field of view of an observer is 45° (-45°, 45°).



8.3.4.2. Creating a Human Eye Sensor

The Human Eye Sensor takes into account the physiological characteristics of the eye to accurately replicate human vision and perception.

> **Note:** The peculiarity of the human eye sensor lies in its capacity to replicate the physiological characteristics of the eye. It is precisely what makes it accurate and at the same time inherently less suited for certain configurations.
>
> This sensor should not be used with transparent diffusive materials in direct simulation because:
>
> - Its ray-acquisition capability appears fairly low with diffusive parts (due to the smallness of the integration angle that drastically reduces the probability for enough rays to fall into it.)
> - Simulations might take too much time to run.
> - Simulation results might present noise, grain or blur whilst appearing more sharp.

To create a Human Eye Sensor:

1. From the **Light Simulation** tab, click **Human Eye**

2. Set the **Axis System** of the sensor by placing two points in the scene:
   - In the 3D view, click

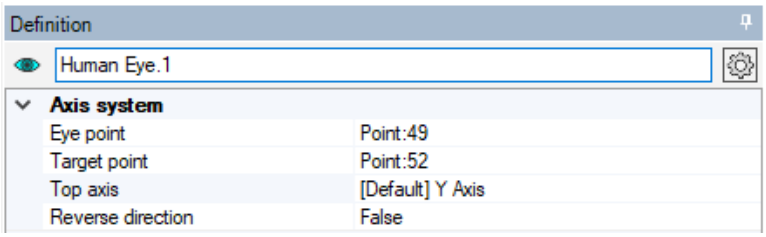
     and select one point to place the eye in the scene.

- In the 3D view, click

  and select one point to define a target point.
- If you want to modify the vertical direction (Y by default), select a line in the 3D view.

  A preview of the sensor appears in the 3D view.



- or click

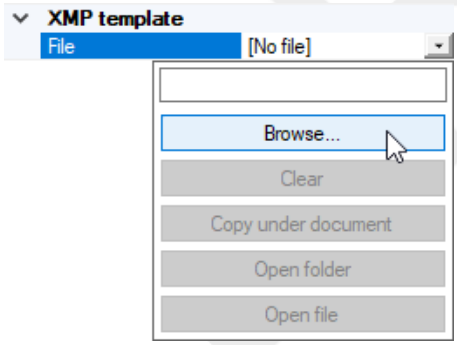
  

  and select a coordinate system to autofill the Axis System.

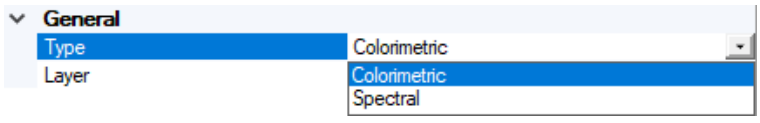
> **Note:** Make sure the sensor is not tangent to a geometry.

3. If you want to use an XMP Template to define the sensor, in **XMP Template**, click **Browse** to load an .xmp file.

> **Note:** An XMP Template is an .xml file generated from an XMP result. It contains data and information related to the options of the XMP result (dimensions, type, wavelength and display properties).
>
> When using an XMP Template, measures are then automatically created in the new .xmp generated during the simulation based on the data contained in the template file.



4. In **General**, from the **Type** drop-down list:



- Select **Colorimetric** to get the color results without any spectral layer separation (in cd/m^2 or $W/sr.m^2$).
- Select **Spectral** to get the color results and spectral data separated by wavelength (in cd/m^2 or $W/sr.m^2$).

5. From the **Layer** drop-down list:

- Select **None** to get the simulation's results in one layer.
- Select **Source** if you have created more than one source and want to include one layer per active source in the result.

> **Tip:** You can change the source's power or spectrum with the Virtual Lighting Controller in **Virtual Photometric Lab** or **Virtual Human Vision Lab**.

6. Determine the field of view of the observer:

Horizontal field of view	
Mirrored extent	False
Start	-20 °
End	20 °
Sampling	2352
Resolution	0.01701 °
Vertical field of view	
Mirrored extent	False
Start	-10 °
End	10 °
Sampling	1175
Resolution	0.01702 °

- Adjust the **Start** and **End** values of the horizontal and vertical fields view.

> **Tip:** You can either use the manipulators of the 3D view to adjust the sensor or directly edit the values from the definition panel.

- Define the Resolution of the observer's eye in degrees.

  The sampling is automatically calculated and depends on the eye's resolution.

> **Tip:** The **Resolution** corresponds to the visual acuity of the observer. The value taken here as a reference (1 arc minute = 0.0167 degrees) corresponds to the normal visual acuity of an observer.

- Set **Mirrored Extent** to **True** if you want to symmetrize the sensor by linking Start and End values.

7. In **Wavelength**, edit the sampling if needed.

> **Note:** The Wavelength Start and End values are inherited from the human eye sensitivity.

Wavelength	
Start	360 nm
End	830 nm
Sampling	13
Resolution	39.16667 nm

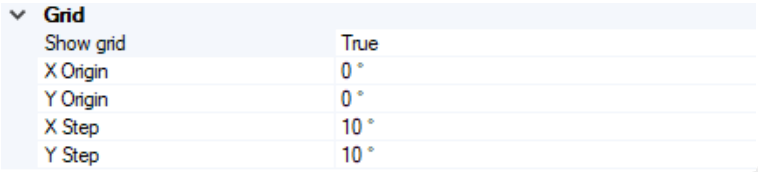
8. If you need to modify the pupil diameter of the observer, type a value in mm.

Properties	
Pupil diameter	4 mm

The pupil diameter corresponds to the physical aperture of the eye. By default, the diameter is set to 4mm.

> **Note:** The pupil dilation strongly depends on the environment. The default value used here (4mm) corresponds to the pupil dilation in broad daylight. In a bright environment the pupil contracts and can reach 2mm. In a dark environment the pupil dilates and can reach 9mm. The pupil's dilation capacity also depends on a lot of factors (age of the observer, disease etc).

9. In **Optional or advanced settings** , adjust the preview to display in the 3D view if needed.



The Human Eye Sensor is created and appears in Speos tree and in the 3D view.

Related information

Sensors Overview on page 163
Sensors integrate rays coming from the source to analyze the optical result.

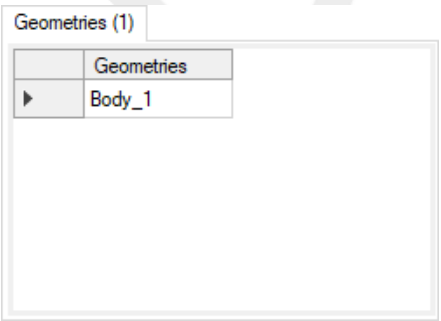
8.3.5. Creating a 3D Irradiance Sensor

The 3D Irradiance Sensor allows you to analyze the light contributions for each face of a geometry. Each selected face becomes a sensor and computes irradiance (in Watt/m^2) or illuminance (in Lux).

> **Important:** This feature is only available under Speos Premium or Enterprise license.

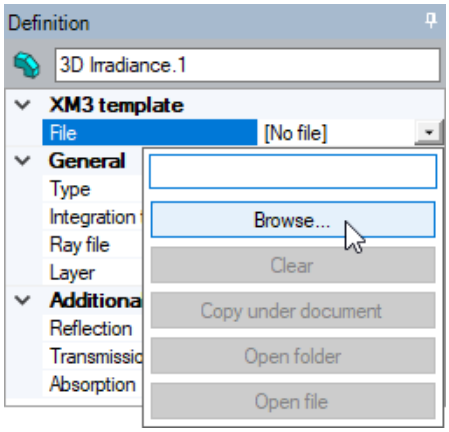
To create a 3D Irradiance Sensor:

1. From the **Light Simulation** tab, click **3D Sensors** > **3D Irradiance** .
2. In the 3D view, click and select one or more faces or bodies to include to the sensor.



3. If you want to use a XM3 template to define the sensor, in **XM3 template**, click **Browse** and load a *.xml file.

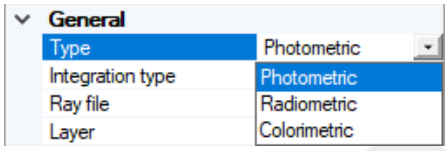
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



A XM3 template allows you to apply a *.xml file containing measure data exported from an existing *.xm3 file. Measures are automatically created in the new *.xm3 file generated during the simulation based on the data contained in the template file.

> **Note:** The *.xm3 file generated is not related to the measure template file. If you modify the template file, the *.xm3 file generated remains the same.

4. In the **General** section, select the **Type** of the sensor:



- **Photometric** to compute the luminous intensity (in cd) and generate an extended map for Virtual Photometric Lab.

  The illuminance levels are displayed with a false color and you cannot make a spectral or a colorimetric analysis with an extended map.

  > **Note:** In case of a photometric result generation, the International Commission on Illumination (CIE) defines the visible spectrum as follows: "There are no precise limits for the spectral range of visible radiation since they depend upon the amount of radiant flux reaching the retina and the responsivity of the observer. The lower limit is generally taken between 360 nm and 400 nm and the upper limit between 760 nm and 830 nm".

- **Radiometric** to compute the radiant intensity (in $W.sr^{-1}$) and generate an extended map for Virtual Photometric Lab.

  The illuminance levels are displayed with a false color and you cannot make a spectral or a colorimetric analysis with an extended map.

- **Colorimetric** to compute the color results without any spectral layer separation (in cd or $W.sr^{-1}$)

5. From the **Integration type** drop-down list, define how the illuminance is integrated in the sensor:



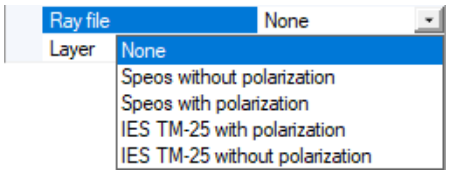
- Select Planar for an integration that is made orthogonally with the sensor plan.

- Select Radial if you need to follow specific street lighting illumination regulations.

  With the **Radial** type, the calculation is based on the standard EN-13201 which gives mathematical formulas equivalent to different types of illuminance.

  The integration direction must be orthogonal to avoid wrong flux computation.

6. If you want to generate a ray file containing the rays that will be integrated to the sensor, select the **Ray file** type:



- **SPEOS without polarization** generates a ray file without polarization data.
- **SPEOS with polarization** generates a ray file with the polarization data for each ray.
- **IES TM-25 with polarization** generates a *.tm25ray file with polarization data for each ray.
- **IES TM-25 without polarization** generates a *.tm25ray file without polarization data.

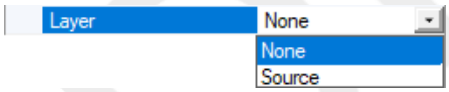
The ray file is only generated at the end of the simulation.

> **Note:** According to the geometry on which you use the 3D Irradiance Sensor, a same ray can be stored several times at different locations leading to an over-estimation of the flux.

> **Note:** The size of a ray file is roughly 30 MB per Mray. Check the free space on the hard drive before generating a ray file.

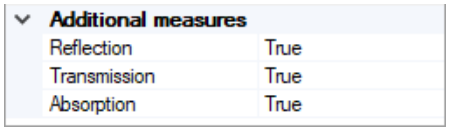
> **CAUTION:** If the surface that will be used for the creation of the ray file is fully absorbent (SOP set to Mirror 0%), all rays are absorbed and no ray will be integrated in the ray file. This ray file will be empty.

7. From the **Layer** drop-down list, select:

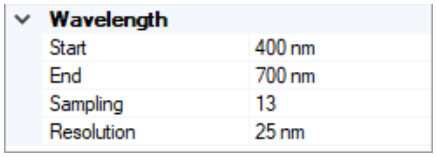


- **None** to get the results of the simulation in one layer.
- **Source** if you have created more than one source and want to create one layer per active source in the result.

8. If you selected **Photometric** or **Radiometric**, in the **Additional measures** section, define which type of contributions (transmission, absorption, reflection) need to be taken into account for the integrating faces of the sensor.



9. If you selected **Colorimetric**, in the **Wavelength** section, define the wavelength characteristics:



a) With **Start** and **End**, define the wavelength interval of the spectral data.
b) With **Sampling** or **Resolution**, define the number of wavelengths or the step between each wavelength to take into account in the wavelength interval.

The 3D Irradiance Sensor is created in the 3D view and the Speos tree.

8.3.6. Creating an Immersive Sensor

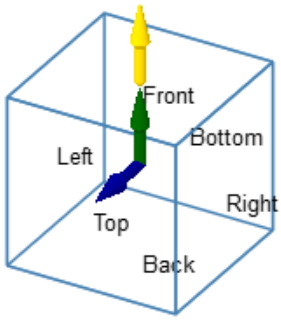
This page shows how to create an Immersive Sensor to visualize what is viewed from a specific point of view of a scene. An immersive sensor allows you to generate an .speos360 file that can be visualized in Virtual Reality Lab.

> **Important:** This feature is only available under Speos Premium or Enterprise license.

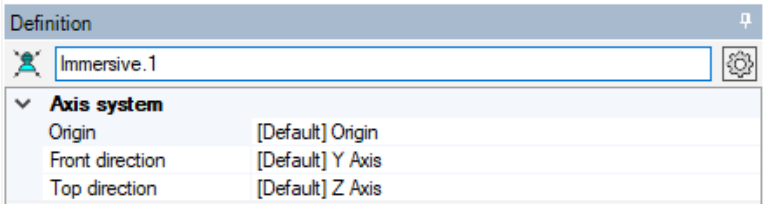
To create an Immersive Sensor:

1. From the **Light Simulation** tab, click **VR** > **Immersive** .

   Spatial indicators appear in the 3D view to demonstrate how the sensor is oriented in space (front, back, bottom, top, left and right positions are indicated).



2. If you want to adjust the **Axis system** of the sensor:



- to select another origin than the absolute origin of the assembly.
- and select a line to define the **Front direction** (corresponding by default to Y axis).

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

- 
  and select a line to define the **Top direction** (corresponding by default to Z axis).
- or click
  
  and select a coordinate system to autofill the Axis System.

> **Note:** Make sure the sensor is not tangent to a geometry.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

3. In **General**, edit the **Sampling** value to recalculate the **Central resolution**.
   The **Central resolution** is automatically calculated and depends on the **Sampling** value.

General	
Sampling	600
Central resolution	0.19099 °
Stereo	True
Interocular distance	65 mm
Layer	None

4. If you want to define an interocular distance, set **Stereo** to **True** and type the interpupillary distance in mm.

> **Note:** When you define a stereo sensor, make sure that the Front direction is horizontal, the Top direction is vertical, and the Central resolution matches the intended 3D display.

5. In **Wavelength**, define the spectral excursion of the sensor:

Wavelength	
Start	400 nm
End	700 nm
Sampling	13
Resolution	25 nm

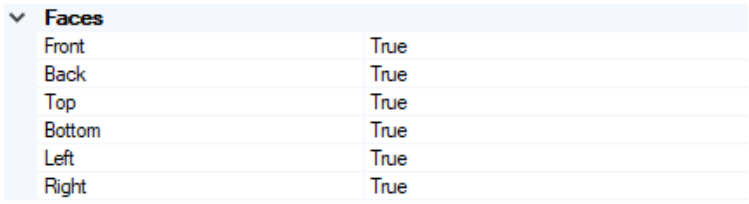
   - Edit the **Start** (minimum wavelength) and **End** (maximum wavelength) values to determine the wavelength range to be considered by the sensor.
   - If needed, in **Sampling**, adjust the number of wavelengths to be computed during simulation.

     The Resolution is automatically computed according to the sampling and wavelength start and end values.

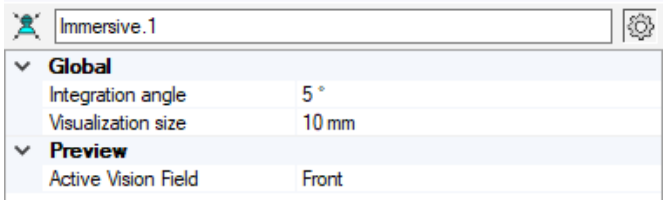
> **Note:** The sensor sampling is used for each face.
>
> Every wavelength beyond the defined borders are not taken into account by the sensor.

6. In **Faces**, define which faces to include/exclude.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

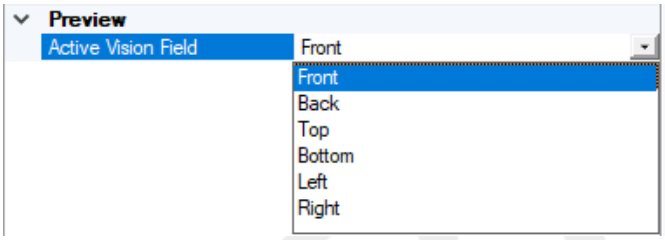


7. In **Optional or advanced settings** :



- If needed, adjust the Integration angle.
- If needed, in **Visualization size**, adjust the preview of the sensor's size.
- In **Preview**, from the **Active Vision Field** drop-down list, select which face of the sensor should be used as default viewpoint for the **Automatic framing**

  option.



The Immersive Sensor is created and appears both in Speos tree and in the 3D view.

Related information

Sensors Overview on page 163
Sensors integrate rays coming from the source to analyze the optical result.

8.3.7. Creating a 3D Energy Density Sensor

This page shows how to create a 3D Energy Density Sensor. This sensor allows you to compute the absorbed energy density in Lumen/m^3 or Watt/m^3 which can be useful when working with highly diffusive materials, wanting to track some hot spots or wanting to visualize the rays' distribution inside the volume itself.

> **Important:** This feature is only available under Speos Premium or Enterprise license.

To create a 3D Energy Density Sensor:

1. From the **Light Simulation** tab, click **3D Energy Density** .

The sensor appears in the 3D view and is placed on the origin of the assembly.

2. If you want to modify the axis system of the sensor, click to select the origin, to define the X axis and to define the Y axis or click and select a coordinate system to autofill the Axis System.

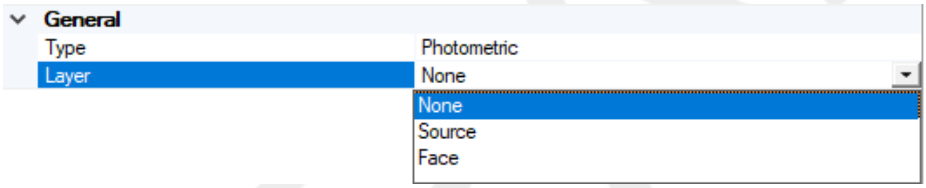
> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

3. In **General**, from the **Type** drop-down list:
   - Select **Photometric** if you want the sensor to consider the visible spectrum and get results in lumens/m^3.

> **Note:** In case of a photometric result generation, the International Commission on Illumination (CIE) defines the visible spectrum as follows: "There are no precise limits for the spectral range of visible radiation since they depend upon the amount of radiant flux reaching the retina and the responsivity of the observer. The lower limit is generally taken between 360 nm and 400 nm and the upper limit between 760 nm and 830 nm".

   - Select **Radiometric** if you want the sensor to consider the entire spectrum and get results in watts/m^3.

4. From the **Layer** drop-down list:



   - Select **None** to get the simulation's results in one layer.
   - Select **Source** if you have created more than one source and want to include one layer per active source in the result.

> **Tip:** You can change the source's power or spectrum with the Virtual Lighting Controller in **Virtual 3D Photometric Lab**.

   - Select **Face** to include one layer per surface selected in the result.

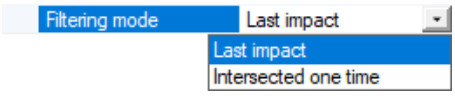
> **Tip:** Separating the result by face is useful when working on a reflector analysis.

   - In the 3D view click and select the contributing faces you want to include for layer separation.

> **Tip:** Select a group (**Named Selection**) to separate the result with one layer for all the faces contained in the group.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- Select the **filtering mode** to use to store the results (*.xm3):



- **Last Impact**: with this mode, the ray is integrated in the layer of the last hit surface before hitting the sensor.
- **Intersected one time**: with this mode, the ray is integrated in the layer of the last hit selected surface if the surface has been selected as a contributing face or the ray intersects it at least one time.

5. If needed, adjust the dimensions of the sensor by either entering the values or using the 3D view manipulators.
6. Adjust the sampling of the sensor on X, Y and Z axes.

Dimensions	
X size	50 mm
X sampling	100
Y size	54 mm
Y sampling	100
Z size	50 mm
Z sampling	100

7. Press **F4** to leave the edition mode.

The 3D Energy Density sensor is created and appears both in Speos tree and in the 3D view.

Related information

Sensors Overview on page 163
Sensors integrate rays coming from the source to analyze the optical result.

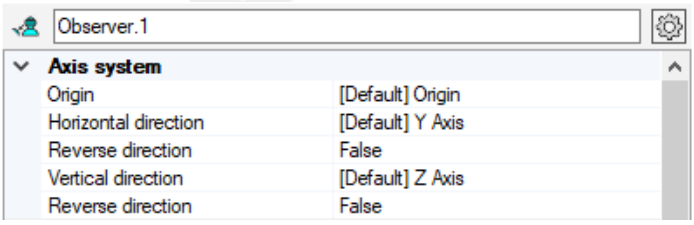
8.3.8. Creating an Observer Sensor

Creating an observer sensor allows you to create a specific view point from where you can observe the system. This sensor is useful to visualize the optical system from user defined points of view.

> **Note:** If your computer has enough memory, deactivate the **VR Sensor Memory Management** (accessible from the general options) for better performance.

To create an Observer Sensor:

1. From the **Light Simulation** tab, click **VR** > **Observer**.
2. In the 3D view, define the **Axis system** of the observed object.



DRAFT

Note: With the Observer Sensor, the axis system does not define the sensor's position but the observed object/scene position. A sphere is then generated around the defined origin allowing you to adjust the sensor's position by adjusting the sphere itself.

- 
to select an origin point.
- 
to select a line defining the horizontal direction.
- 
to select a line defining the vertical direction.
- or click

and select a coordinate system to autofill the Axis System.

Note: Make sure the sensor is not tangent to a geometry.

Note: If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

3. In **Distance**, adjust the radius of the sphere to narrow or widen the global field of vision.

General	
Distance	1000 mm
Focal	50 mm
Stereo	False
Layer	None

4. In **Focal**, adjust the distance between the sensor radiance plan and the origin point of the observed object. The larger the focal, the closer to the object.
5. If you want to define an **Interocular distance**, set **Stereo** to **True** and type a value in mm.

Note: When you define a stereo sensor, make sure the stereo sensor size (height, width) matches your display and the stereo sensor focal distance matches the focal distance from your display.

6. From the **Layer**drop-down list, define how you want the data to be separated in the results:
   - Select **None** if you want the simulation to generate a Speos360 file with one layer containing all sources.
   - Select **Source** if you have created more than one source and want to include one layer per active source in the result.
7. In **Wavelength**, define the spectral range the sensor needs to consider:

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Wavelength	
Start	400 nm
End	700 nm
Sampling	13
Resolution	25 nm

- Edit the **Start** (minimum wavelength) and **End** (maximum wavelength) values to determine the wavelength range to be considered by the sensor.
- If needed, in **Sampling**, adjust the number of wavelengths to be computed during simulation.

  The Resolution is automatically computed according to the sampling and wavelength start and end values.

8. In **Observer locations**, define the sensor's position on the sphere around the target point:

Observer locations	
H start	-180 °
H end	180 °
H sampling	9
H resolution	40 °
Mirrored extent	False
V start	-90 °
V end	90 °
V sampling	9
V resolution	22.5 °
Mirrored extent	False

> **Tip:** You can use the 3D view manipulators to adjust the Vertical and Horizontal Start and End positions of the sphere.

- Set the observer location horizontally by adjusting H start and end values.
- Set the observer location vertically by adjusting V start and end values.
- Adjust the V and H sampling.

  The resolution is automatically computed.

9. In **Size**, adjust the sensor's size:

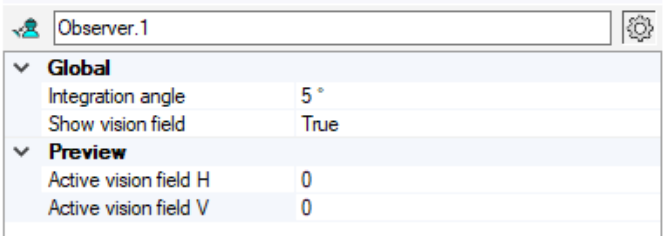
Size	
H start	-50 mm
H end	50 mm
H sampling	100
H resolution	1 mm
Mirrored extent	False
V start	-50 mm
V end	50 mm
V sampling	100
V resolution	1 mm
Mirrored extent	False

- Set the horizontal size of the sensor by adjusting H start and end values (in mm).
- Set the vertical size of the sensor by adjusting V start and end values (in mm).
- Adjust the V and H sampling.

  The resolution is automatically computed.

> **Note:** Speos supports a maximum resolution of 23170 * 23170 pixels.

10. In **Optional or advanced settings** :

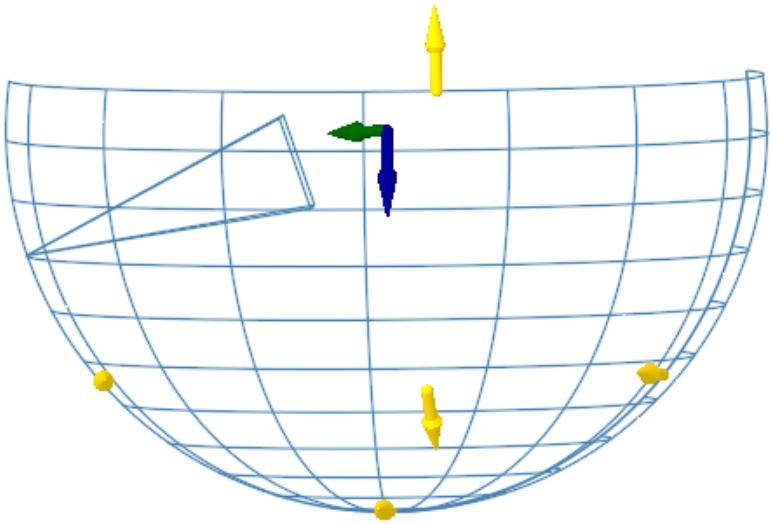


- If needed, adjust the Integration angle.
- If you do not want to display the vision field (the sphere), set **Show vision field** to **False**.
- If you activated the vision field, in **Preview**, you can adjust the active vision field horizontally and/or vertically to place the observer on a specific location.

> **Note:** When defined, this position is used as the default viewpoint for the **Automatic framing** option.

The Observer Sensor is created and appears both in Speos tree and in the 3D view.

Figure 36. Example of Result



Related information

Sensors Overview on page 163
Sensors integrate rays coming from the source to analyze the optical result.

Integration Angle

8.3.9. Light Field Sensor

The Light Field Sensor measures the distribution of light hitting a surface and generates a Light Field file storing this light distribution.

> **Note:** The Light Field feature is in BETA mode for the current release.

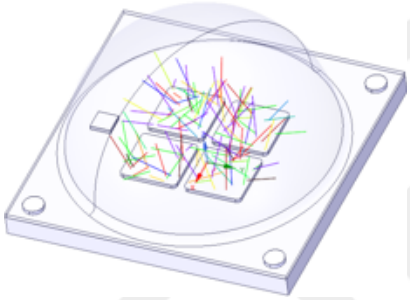
8.3.9.1. Light Field Overview

The Light Field allows you to create a propagation result of a sub-optical system to be reused in a more complex optical system to gain time when computing simulation.

> **Note:** The Light Field feature is in BETA mode for the current release.

Optical systems can be composed of sub-optical systems. When you focus on the main optical system, recalculating every time the propagation inside those sub optical systems can be time-consuming. This time can be optimized: the goal is to speed up the simulation by pre-computing the propagation of the sub-optical systems.

To proceed to the pre-calculation of those sub-optical systems, the Light Field feature generates a *.olf (Optical Light Field) file format thanks to a Light Field sensor, that is then used as a Light Field source in the main optical system. Thus, the simulation does not have to compute the propagation of the sub-optical system, reducing the simulation time.

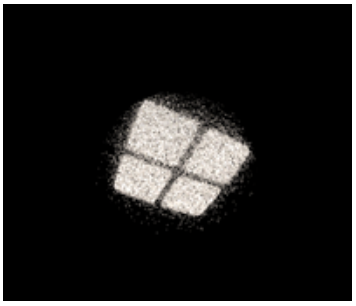


LED including chips with a lens on top



Original Radiance Simulation of the LED

Reference time = T



Radiance Simulation of the Light Field representing the LED

Simulation time = 0.43 * T

General Workflow

1. Create a Local Meshing of the surface to be integrated in the Optical Light Field file.

   -Or-

   At Step 3, use the Meshing Options of the Direct Simulation to generate the meshing.

   > **Note:** As no optical properties are defined on a Light Field meshing, the Light Field is fully absorbing.

2. Create a Light Field Sensor to define how the *.olf file will be generated.
3. Create a Direct Simulation, and if no Local Meshing is applied on the Light Field surface, define the Meshing Options.
4. Run the Direct Simulation to generate the *.olf file.
5. Create a Light Field Source that uses the generated *.olf file as input.
6. Create and run an Interactive, Direct or Inverse Simulation of the main optical system, using the Light Field Source as input.

8.3.9.2. Understanding the Parameters of a Light Field Sensor

This page describes the parameters to set when creating a Light Field Sensor.

> **Note:** The Light Field feature is in BETA mode for the current release.

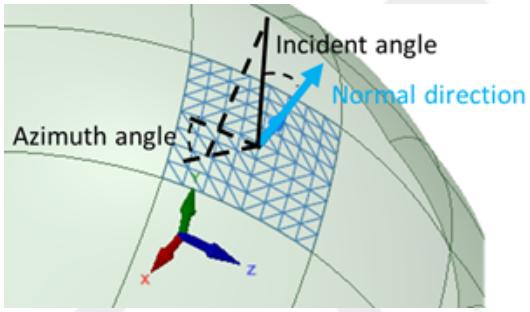
Oriented Faces

Light Field sensor measures the distribution of light hitting selected Oriented Faces after a reflection or a transmission.

The faces are meshed using either Direct simulation's options or Local meshing, and the light distribution is stored for each triangle.

Incident and Azimuth Angles

To measure the light distribution, the sensor requires a sampling of the angular coordinates defined by the incident and azimuth angles.



The smaller the angular resolution and the thinner the meshing, the larger memory used to generate the Light Field file and the bigger the file size.

8.3.9.3. Creating a Light Field Sensor

This page shows how to create a Light Field Sensor that measures the distribution of light hitting a surface and generates a Light Field file (".olf" file for "optical light field" file) storing light distribution on this selected surface.

> **Note:** The Light Field feature is in BETA mode for the current release.

To create a Light Field Sensor:

1. From the **Light Simulation** tab, click **Light Field**  .
   The sensor appears in the 3D view and is placed on the origin of the assembly.
2. In the 3D view, define the **Axis system** of the Light Field sensor.

Axis System	
Origin	[Default] Origin
X direction	[Default] X Axis
Reverse direction	False
Y direction	[Default] Y Axis
Reverse direction	False

   - 
     to select an origin point.
   - 
     to select a line defining the horizontal direction.
   - 
     to select a line defining the vertical direction.
   - or click
     
     and select a coordinate system to autofill the Axis System.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

3. In **General**, from the **Type** drop-down list:

General	
Type	Photometric

   - Select **Photometric** if you want the sensor to consider the visible spectrum and get the results in lm/m^2 or lx.

> **Note:** In case of a photometric result generation, the International Commission on Illumination (CIE) defines the visible spectrum as follows: "There are no precise limits for the spectral range of visible radiation since they depend upon the amount of radiant flux reaching the retina and the responsivity of the observer. The lower limit is generally taken between 360 nm and 400 nm and the upper limit between 760 nm and 830 nm".

   - Select **Radiometric** if you want the sensor to consider the entire spectrum and get the results in W/m^2.

> **Note:** With both Photometric and Radiometric types, the illuminance levels are displayed with a false color and you cannot make any spectral or color analysis on the results.

   - Select **Spectral** to store the spectral data according to the wavelength sampling defined (in lx or W/m^2).

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

> **Note:** Spectral results take more time to compute as they contain more information.

4. In the 3D view, click  to select the oriented faces on which to measure the light distribution.
   The selected faces appear in the list as **Oriented Faces**.
5. In **Incident angles**, define the angular sampling or the angular resolution.

Incident Angles	
Start	0 °
End	90 °
Sampling	10
Resolution	9 °

> **Note:** In the 2022 R1 version, the Start and End values are fixed to 0° and 90°.

6. In **Azimuth angles**, define the angular sampling or the angular resolution.

Azimuth Angles	
Start	0 °
End	360 °
Sampling	10
Resolution	36 °

> **Note:** In the 2022 R1 version, the Start and End values are fixed to 0° and 360°.

7. If you selected **Spectral** as sensor type, set the spectral excursion to use for simulation.

Wavelength	
Start	400 nm
End	700 nm
Sampling	13
Resolution	25 nm

   - Edit the **Start** (minimum wavelength) and **End** (maximum wavelength) values to determine the wavelength range to be considered by the sensor.
   - If needed, in **Sampling**, adjust the number of wavelengths to be computed during simulation.

> **Note:** The Resolution is automatically computed according to the sampling and wavelength start and end values.

The Light Field Sensor is created and appears both in Speos tree and in the 3D view.

Create and run a Direct Simulation containing the Light Field Sensor to generate the Light Field file.

8.3.10. LiDAR Sensor

LIDAR is a remote sensing technology using light to measure the distance to a target. The LiDAR sensor allows you to model static, scanning or rotating LiDAR systems to conduct detection tests or field of views studies.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Important: This feature is only available with the **Speos Optical Sensor Test** add on.

8.3.10.1. LiDAR Sensor Models

This page provides an overview of LiDAR sensor principles and presents the models that are available in Speos.

A LiDAR (Light Detection and Ranging) is a system composed of a light source (an emitter channel - Tx) and a sensor (a receiver channel - Rx) meant to measure the distance to a target by sending pulsed laser light. The emitter illuminates a target by sending laser pulses and evaluates the distance to that target based on the time the reflected pulse took to hit the receiver. LiDAR technology is widely used in the automotive industry as it holds a key role in the development of autonomous vehicles by being able to detect and render a model of a vehicle's surroundings.

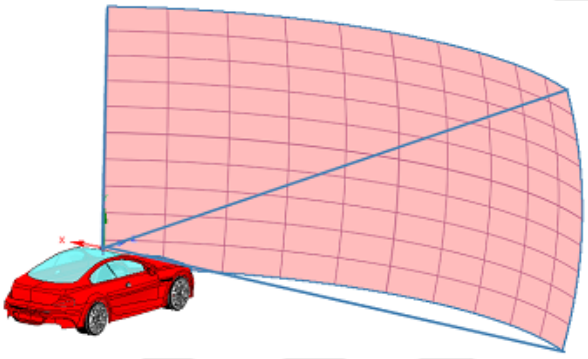
Three different types of LiDAR sensors are available to cover different needs and configurations.

Types of LiDAR Sensors

Static

Solid-State or flashing LiDARs are static systems with no moving parts. With a camera-like behavior, the flashing LiDAR sends one large laser pulse in one direction to detect and model its environment.

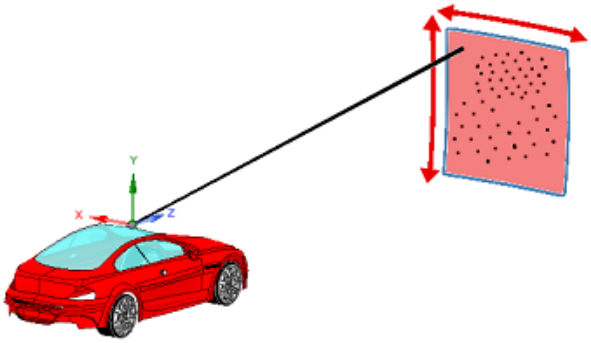
This type of LiDAR, however, rarely exceeds a 120° field of view.



Scanning

Scanning LiDAR systems use dual-axis mirrors to deflect laser beams into specific scanning angles. With this type of LiDAR, you can define a custom beam detection pattern by describing the azimuth and elevation shooting angles of the LiDAR. These systems have a detection range that strongly depends on the optical components and the field of view of the imager. They can cover, at most,180 degrees.

DRAFT

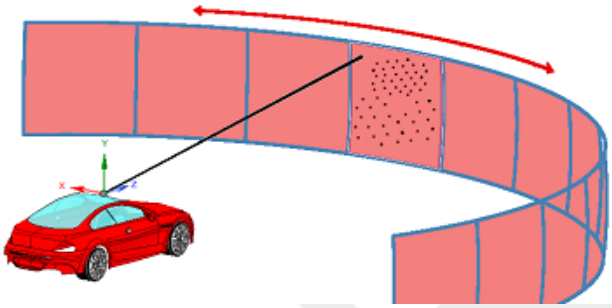


Rotating

Rotating LiDARs are systems with a mechanical rotation offering controlled aiming directions over a 360 field of view. This type of LiDAR enables a multi-direction detection and a high speed scanning.

With this configuration, the scanning pattern is repeated sequentially for each rotation step of the LiDAR.

Figure 37. Example of Rotating LiDARs simulated with Speos



In Speos

- From solid-state to rotating LiDARs, define LiDAR systems based on currently commercialized LiDAR models.
- Manage all the characteristics of your LiDAR system: its energy, range, emission pattern etc.

Related concepts

LiDAR Sensor Parameters on page 226
The following section provides more information on static, scanning or rotating LiDAR sensor parameters.

Related tasks

Creating a LiDAR Sensor on page 246
This section gathers the LiDAR sensor creation procedures for the three LiDAR sensor types available in Speos

Related information

Understanding LiDAR Simulation on page 441
This page gives a global presentation on LiDAR principles and simulation.

DRAFT

8.3.10.2. LiDAR Sensor Parameters

The following section provides more information on static, scanning or rotating LiDAR sensor parameters.

8.3.10.2.1. Understanding LiDAR Sensor Parameters

This page describes the parameters to set when creating a static, scanning or rotating LiDAR sensor.

A LiDAR sensor definition can be broken down into several parts corresponding to each component of the LiDAR system: the system definition, the source definition and the sensor definition.

System Definition

The system definition corresponds to the definition of the LiDAR physical module and its emission pattern.

- **Axis System**: the system's axis system defines the position and orientation of the "physical" LiDAR module. This axis is only required for rotating LiDARs as it is used as the revolution axis of the system.

  During the simulation, the Y vector of this axis is used as reference to allow the other axes (the source's and sensor's axes) to revolve around it.

- **Trajectory file**: a trajectory describes the different positions and orientations of features in time. The Trajectory file is used to simulate dynamic objects.

  > **Note:** For more information, see Trajectory File on page 236.

- **Firing Sequence**: firing sequence files are files used to describe a LiDAR's emission pattern.

  Two types of firing sequence files are available in Speos allowing you to define the emission pattern of a scanning or a rotating LiDAR.

  > **Note:** For more information, see **Firing Sequence Files**.

Source Definition

The source definition corresponds to the definition of the emitter channel of the LiDAR.

- **Axis system**: sets the position and orientation of the source channel.

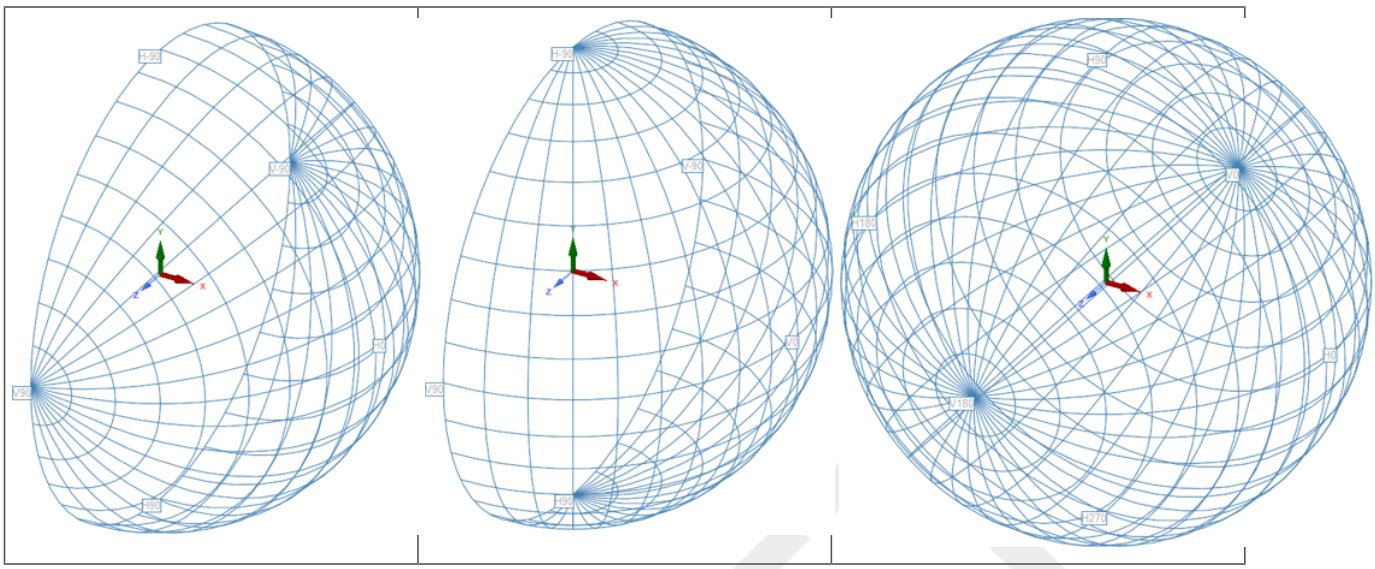
  > **Note:** When using an intensity distribution file, verify the orientation of the IES file to correctly orient the beam which changes according to the IES type (IESNA type A B or C).

  Example:

  Source X direction = X axis

  Source Y direction = Y axis

IES A	IES B	IES C



- **Spectrum**: for static LiDAR source, the spectrum is monochromatic. Only one wavelength must be defined to set the spectral excursion of the LiDAR.

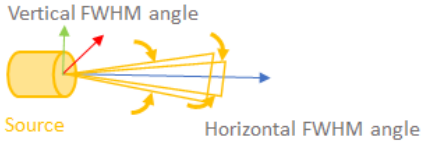
  For scanning or rotating LiDARs, the spectrum can either be monochromatic or defined by a spectrum file (*.spectrum) as input.

- **Intensity**: for static LiDARs, the intensity of the source is defined using an IES (*.ies) or eulumdat (*.ldt) file.

  For scanning or rotating LiDARs, the intensity of the source can be defined either by using an intensity distribution file (*.ies or *.ldt file) as input or by manually defining an asymmetrical gaussian profile.



Source defined by IES file



Source defined with a gaussian profile

- **Energy**: the source energy corresponds to the laser pulse energy expressed in Joules (J).
- The **Minimum intensity threshold** is the threshold under which the signal of the LiDAR source is not taken into account. This helps LiDAR differentiate what should be used or ignored to calculate LiDAR's field of view.

> **Note:** For a static LiDAR, the energy and minimum intensity threshold are defined in the interface.
>
> For scanning or rotating LiDARs, only the pulse energy must be defined in the scanning sequence file.

Sensor Definition

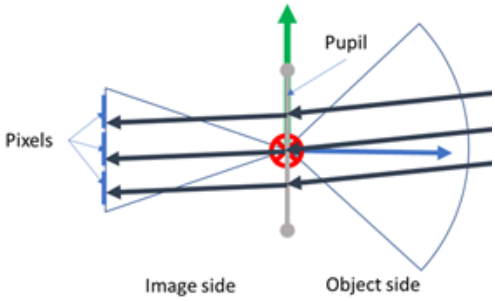
The sensor definition corresponds to the receiver channel of the LiDAR.

- **Axis System**: sets the position and orientation of the sensor channel.
- The **Optics** section allows you to model the optical system in front of the imager.
  - **Distortion file**: the .OPTDistortion file is used to introduce/replicate the optical distortion of the lens.
  - The **Transmittance** corresponds to the capacity of the lens to allow the light to pass through.

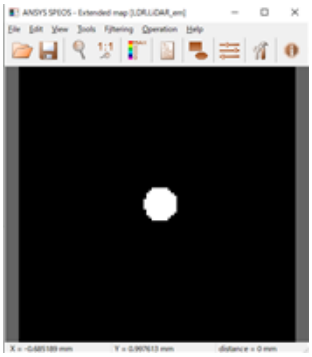
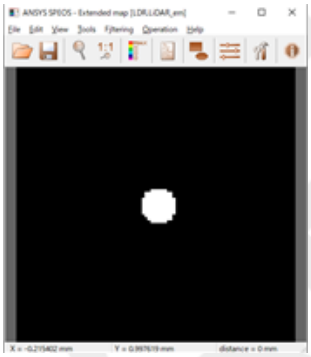
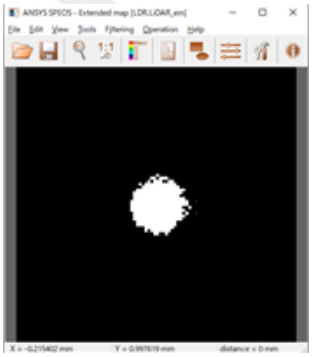
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- The **Focal length** defines the distance between the center of the optical system to the focus.
- The **Pupil** is the diameter of the LiDAR sensor'saperture.

  According to the Pupil size, rays coming with the same direction are not integrated in the same pixel.



  If you keep increasing the pupil size, you will observe blur as shown below:

		
Pupil = 0.5	Pupil = 1.0	Pupil = 10.0

- The **Horizontal and Vertical FOV** correspond to the fields of view calculated by the sensor using the Focal Length, Distortion File, Imager Width and Height.

- The **Imager** represents the LiDAR sensor (receiver channel).
  - The Imager **Width** and **Height** correspond to the horizontal and vertical size of the sensor.

    Note: For scanning and rotating LiDARs, only the Width and Height of the imager must be defined as the sensor is a one-pixel sensor.

  - The Horizontal and Vertical Pixels represent the sensor resolution.
- **Range and accuracy**
  - The **Start** and **End** values represent the distance from, and up to which the LiDAR is able to operate and integrate rays.
  - The **Spatial accuracy** defines the discrimination step used to save the Raw time of flight signal.

    This magnitude is directly linked to the hardware capability of the LiDAR system regarding its distance resolution and sampling frequency.

    Timestamp resolution can be converted to a spatial resolution, using the time of flight.

- The **Aiming area** allows you to define the position, size and shape of the LiDAR's protective sheet.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Note: For more information, see LiDAR Sensor Aiming Area Parameters.

Related concepts

LiDAR Sensor Parameters on page 226
The following section provides more information on static, scanning or rotating LiDAR sensor parameters.

Related tasks

Creating a LiDAR Sensor on page 246
This section gathers the LiDAR sensor creation procedures for the three LiDAR sensor types available in Speos

8.3.10.2.2. Understanding Advanced LiDAR Sensor Parameters

The following section describes some advanced parameters to set when creating a LiDAR sensor.

8.3.10.2.2.1. Firing Sequence Files

This page describes the format of firing sequence files, used to define the scanning and/or rotation pattern of a LiDAR system.

Description

Firing sequence files are used to describe a LiDAR's emission pattern. These files embed several information like the firing time, firing angles and source power, that will be used during the simulation to draw the scanning sequence.

Two types of firing sequence files are available in Speos allowing you to define the emission pattern of a scanning or a rotating LiDAR.

Scanning Sequence File

Scanning Sequence File Overview

Scanning sequence files describe the horizontal and vertical emission pattern for one beam or multiple beams of the LiDAR. They are required for both scanning and rotating LiDAR sensors.

Two scanning sequence file formats are available:

- *.txt Scanning Sequence File which describes:
  - the power of each beam
- *.OPTScanSequence Scanning Sequence File which describes:
  - the power of each beam
  - the intensity profile and intensity distribution of each beam
  - the emissive surface shape of the beam
  - the spectrum of each beam

For more information on how to generate a scanning sequence file, refer to Generating a Scanning Sequence File.

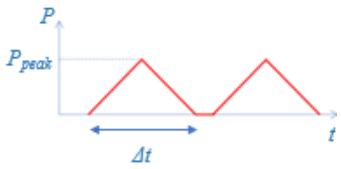
DRAFT

*.OPTScanSequence Scanning Sequence File Content

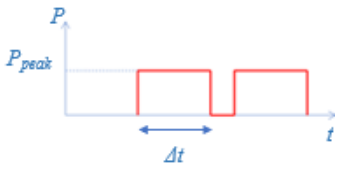
The *.OPTScanSequence scanning sequence file comprises the following parameters:

- **Type of pulses**: DIRAC, Gaussian, Rectangular, Triangular
- **Width of pulse (s)**: duration of one pulse in seconds (s), Δt (only for Gaussian, Triangular, Rectangular pulse types)

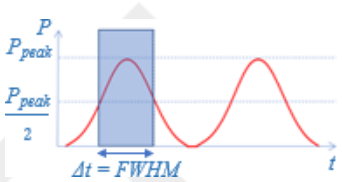
> **Note:** When using the Dirac pulse type, the Width of pulse is skipped.



Triangular pulse type



Rectangular pulse type



Gaussian pulse type

- **Shooting Time (s)**: time when the pulse is emitted in seconds (s)
- **Pulse energy (J)**: Energy of the pulse in Joules
- **Pulse type**: From pulses, DIRAC, Gaussian, Rectangular, Triangular

> **Note:** **From pulse** means that the **Pulse type** inherits the **Type of pulses** value.

- **Pulse width (s)**: duration of one pulse in seconds (s), Δt (only for Gaussian, Triangular, Rectangular pulse types)

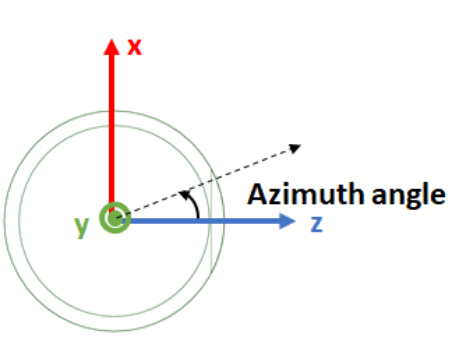
> **Note:** If you enter a **Pulse width** value different from the **Width of pulse** value, the **Pulse width** has priority.

> **Note:** When using the Dirac pulse type, the Width of pulse is skipped.

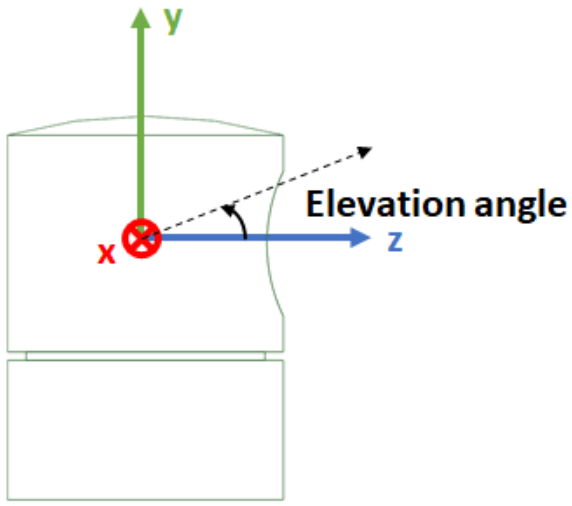
- **Azimuth angle (°)**: Azimuth angles within the range [0;360[expressed in degrees
- **Elevation angle (°)**: Elevation angles within the range [-90;90] expressed in degrees

> **Note:** Azimuth and elevation angles are expressed with respect to the source's axis system of the LiDAR sensor.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



Azimuth angle (horizontal pattern)



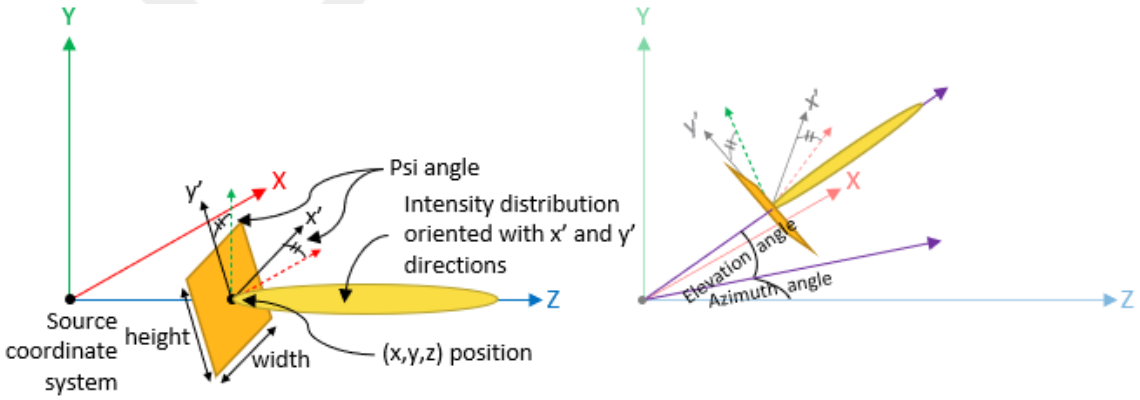
Elevation angle (vertical pattern)

- **Position X Y Z (mm)**: Starting position of the beam with respect to the source's axis system of the LiDAR sensor. That means the beam can be offset.
- **Emissive Surface Description**: corresponds to the emissive surface of the beam.
  - **Surface shape**: Point, Rectangular, Elliptic
  - **Width**: width of the beam (for Rectangular and Elliptic shapes)
  - **Height**: height of the beam (for Rectangular and Elliptic shapes)
  - **Psi angle (°)**: corresponds to the rotation of the surface shape around the Z axis
- **Intensity Description**
  - **Intensity type**: From source definition, Sampled, Gaussian

    If you set **From source definition**, the **Source - Intensity** section's parameters from the LiDAR sensor definition are used.

    If you set **Sampled**, define an **Intensity file** (IES file)

    If you set **Gaussian** define the three following **Gaussian** parameters.
  - **Intensity file**: IES file that will be converted into a readable intensity distribution when generating the *.OPTScanSequence (or *.txt) file. The intensity value is defined on 2π steradian.
  - **Gaussian FWHM along X and Y (°)**: Gaussian intensity distribution of the beam
  - **Gaussian total angle (°)**: Total angle of the beam for a Gaussian intensity distribution



- **Spectrum description**

- **Spectrum type**: From source definition, Sampled, Monochromatic

  If you set **From source definition**, the **Source - Spectrum** section's parameters from the LiDAR sensor definition are used.

  If you set **Sampled**, define an **Spectrum file**

  If you set **Monochromatic** defined **Wavelength (nm)**

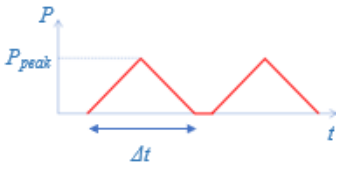
- **Spectrum file**: Specific spectrum file for the beam
- **Wavelength (nm)**: wavelength for monochromatic spectrum type

*.txt Scanning Sequence File Content

The *.txt scanning sequence file comprises the following parameters:

- **Pulse Type**: DIRAC, Gaussian, Rectangular, Triangular
- **Width of the pulse Δt** in nano seconds (ns) (only for Gaussian, Triangular, Rectangular pulse types)

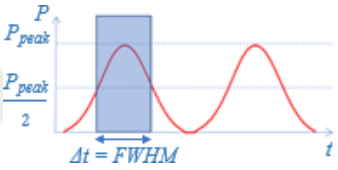
> **Note:** When using the DIRAC pulse type (line 3), the Width of the pulse (line 4) is skipped, meaning no blank line.



Triangular pulse type



Rectangular pulse type

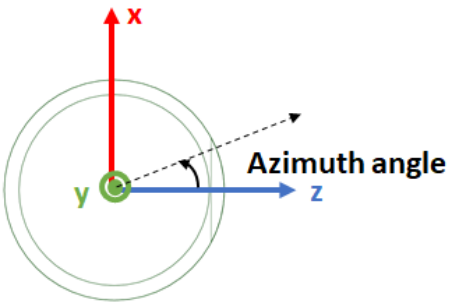
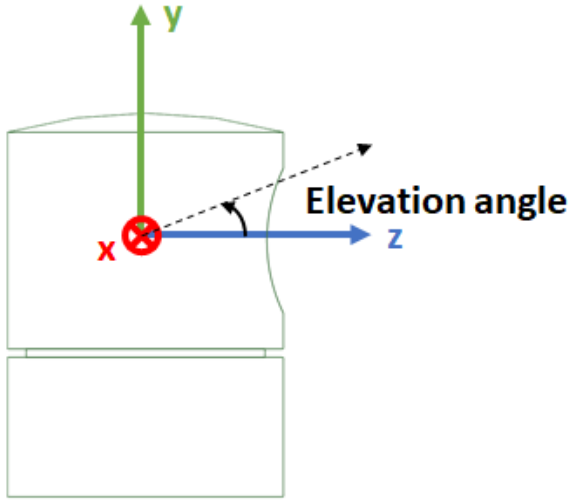


Gaussian pulse type

- **Fire_time_N**: Shooting time in micro seconds (µs)
- **Fire_pulse_energy_N**: Pulse energy in Joules (J)
- **Azimuth_angle_N**: Azimuth angles within the range [0;360[expressed in degrees
- **Elevation_angle_N**: Elevation angles withing the range [-90;90] expressed in degrees

> **Note:** Azimuth and elevation angles are expressed with respect to the source's axis system of the LiDAR sensor.

DRAFT

	
Azimuth angle (horizontal pattern)	Elevation angle (vertical pattern)

Format Description	Example
(line 1) ANSYS - LiDAR scanning sequence file v1.0 (line 2) Comments (line 3) Pulse Type (DIRAC or Gaussian or Rectangular or Triangular) (line 4) Width of the pulse Δt (line 5) Total number of firing samples (line 6) Fire_time_1 Fire_pulse_energy_1 Azimuth_angle_1 Elevation_angle_1 (line 7) Fire_time_2 Fire_pulse_energy_2 Azimuth_angle_2 Elevation_ angle_2 ... (line 5+N) Fire_time_N Fire_pulse_energy_N Azimuth_angle_N Elevation_angle_N	ANSYS - LiDAR scanning sequence file v1.0 Scanning Sequence File Example Rectangular 2 4 0.0 3.00E-7 60 0 1.0 3.00E-7 25 0 2.0 3.00E-7 -15 0 3.0 3.00E-7 -45 0

Rotating Sequence File

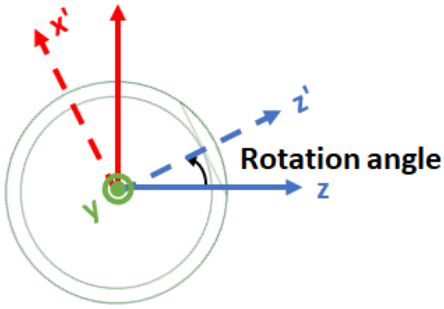
Rotating sequence files describe the rotation pattern of the LiDAR system. The rotating sequence file and scanning pattern file need to be combined to model the definition of a rotating LiDAR. Indeed, with this configuration, the scanning pattern (defined in the scanning sequence file) is repeated sequentially for each rotation step of the LiDAR (defined in the rotating sequence file).

The rotating sequence (*.txt) file comprises only two parameters:

- Rotation time

> **Note:** The rotation time is only indicative, it does not impact the simulation.

- Azimuth angles within the range [0;360]



Azimuth angles - Rotating LiDAR rotation step

Format Description	Example
(line 1) ANSYS - LiDAR rotation sequence file v1.0 (line 2) Comments (line 3) Rotation Definition Mode (Sampled only) (line 4) Total number of rotation samples (line 5) Rotation_time_1 Azimuth_angle_1 (line 6) Rotation_time_2 Azimuth_angle_2 ... (line N+4) Rotation_time_N Azimuth_angle_N	ANSYS - LiDAR rotation sequence file v1.0 Rotation Sequence File Example Sampled 4 0.0 25 1.0 60 2.0 315 3.0 345

Related tasks

Creating a LiDAR Sensor on page 246
This section gathers the LiDAR sensor creation procedures for the three LiDAR sensor types available in Speos

Related reference

Understanding LiDAR Sensor Parameters on page 226
This page describes the parameters to set when creating a static, scanning or rotating LiDAR sensor.

8.3.10.2.2.2. LiDAR Sensor Aiming Area Parameters

This page provides more information on the **Aiming Area** parameter, used during the definition of a LiDAR Sensor.

LiDAR sensors are often placed behind protective glasses or sheets.

The **Aiming Area** section allows you to define the position, size and shape of this protective glass also called cover lens. When defined, the cover lens is taken into account for simulation allowing the LiDAR to recover target signatures through it.

The LiDAR propagation algorithm makes sure that rays launched from the LiDAR Source aim to this area:

1. A primary ray is emitted by the LiDAR Source and propagated in the system.
2. Then, for all impacts of this primary ray propagation, a secondary ray is propagated in the direction of the Aiming Area if the BSDF at the impact point has a diffuse contribution

As the cover lens is likely to be placed right by the sensor, the axis system of the aiming area should (in most cases) be the same as the sensor's.

DRAFT

Axis System

The Axis System of the aiming area can either be inherited from the sensor's axis system or defined manually. In the 3D view, a preview of the aiming area is also displayed to help position it to the desired location.

Dimensions

The dimensions of the aiming area are either inherited from the sensor's pupil diameter or have to be edited manually.

Definition

The aiming area is defined by its shape and size.

Two shape types are available, **Rectangular** or **Elliptical**.

Figure 38. Rectangular Aiming Area

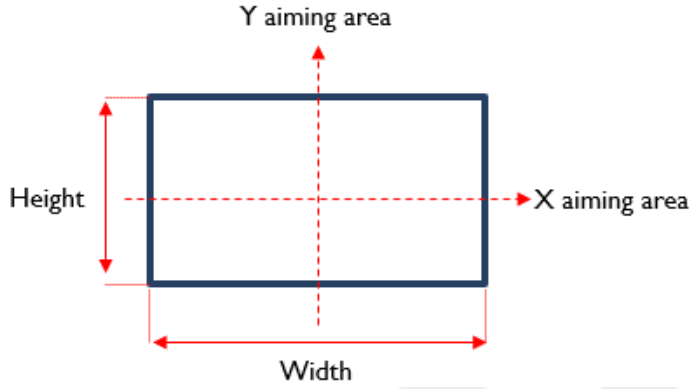
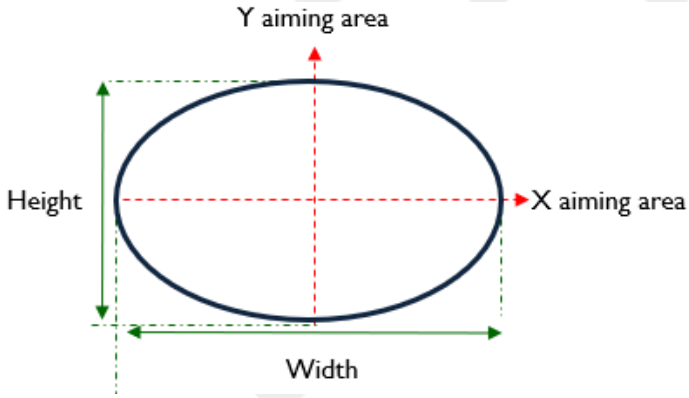


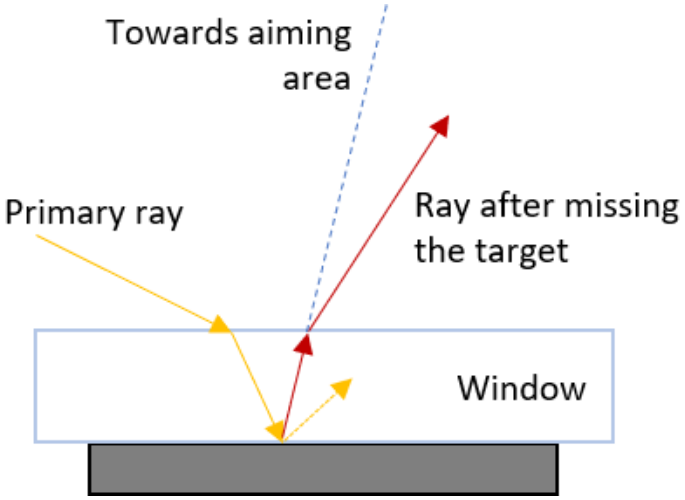
Figure 39. Elliptical Aiming Area



Transparent Material Case

When there is a tangency with no gap between a transparent material and an opaque material, the secondary ray aimed at the aiming area starts in the transparent material. However, when exiting the transparent material, the refraction changes the rays direction, and misses the target. Only primary rays can make it to the sensor but the probability is very low. This can generate unexpected result

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



To avoid such issues, we recommend you to add a small gap between the transparent material and the opaque material. When there is an air gap, the rays are deviated twice:

1. first when entering the transparent material
2. then when existing again

When the two faces are parallel, the exit direction is the same as the direction before entering. Therefore, the exiting rays are still aimed at the area.

8.3.10.2.2.3. Trajectory File

This page describes the format of trajectory files, used to define the position and orientations of a Speos Light Box, a LiDAR sensor, or a Camera sensor in time.

Description

Positions and orientations are expressed with the respect to a reference coordinate system, so the trajectory is relative to this coordinate system.

For instance, the same trajectory file can be used to describe a translation movement of a car as well as the LiDAR sensor placed on it.

Trajectory is described in a *.json file that contains each chronological sample:

DRAFT

Format Description	Example

• "Time" in second.
• "Origin" with "X", "Y", "Z" coordinates in mm, that corresponds to the relative position with the respect to reference coordinate system.
• "Direction_X" with "X", "Y", "Z" coordinates that corresponds to the relative X direction with the respect to reference coordinate system.
• "Direction_Y" with "X", "Y", "Z" coordinates that corresponds to the relative Y direction with the respect to reference coordinate system.
	<pre>"Trajectory": {
	{
	"Time": 0.0,
	"Origin": {
	"X": 0.0,
	"Y": 0.0,
	"Z": 0.0
	},
	"Direction_X": {
	"X": 1.0,
	"Y": 0.0,
	"Z": 0.0
	},
	"Direction_Y": {
	"X": 0.0,
	"Y": 1.0,
	"Z": 0.0
	}
	},
	"Time": 0.033333333333,
	"Origin": {
	"X": 0.0,
	"Y": -0.0,
	"Z": 900.0
	},
	"Direction_X": {
	"X": 1.0,
	"Y": 0.0,
	"Z": 0.0
	},
	...</pre>

Script Example

Trajectory file can be easily accessed (read or write) using dedicated scripting interfaces available in IronPython and Python.

> **Note:** Make sure to use the **3.9 version** of IronPython or Python language to write your scripts.

IronPython Example

```
import sys
import clr

sys.path += [R"C:\Program Files\ANSYS Inc\v2XX\Optical Products\SPEOS\Bin"]

clr.AddReferenceToFile("Optis.Core_net.dll")
clr.AddReferenceToFile("Optis.Data_net.dll")

# References to Optis.Core and Optis.Data
import Optis.Core as OptisCore
import Optis.Data as OptisData
```

```
try:
  firstData = OptisCore.DAxisSystemData()
  firstData.Time = 0.0
  firstData.Origin.Init(0, 0, 0)
  firstData.Direction_X.Init(1, 0, 0)
  firstData.Direction_Y.Init(0, 1, 0)

  lastData = OptisCore.DAxisSystemData()
  lastData.Time = 7.0
  lastData.Origin.Init(0, 0, 3000)
  lastData.Direction_X.Init(1, 0, 0)
  lastData.Direction_Y.Init(0, 1, 0)

  dmTrajectory = OptisCore.DAxisSystemTrajectory()
  dmTrajectory.Trajectory.Resize(2)
  dmTrajectory.Trajectory.Set(0, firstData)
  dmTrajectory.Trajectory.Set(1, lastData)

  strPathTrajectoryFile = OptisCore.String.From(R"C:\trajectory.json")
  pathTrajectoryFile = OptisCore.Path(strPathTrajectoryFile)

  cAxisSystemTrajectoryWriter = OptisData.CAxisSystemTrajectoryWriter()
  cAxisSystemTrajectoryWriter.Open(pathTrajectoryFile)
  cAxisSystemTrajectoryWriter.Write(dmTrajectory)

  cAxisSystemTrajectoryReader = OptisData.CAxisSystemTrajectoryReader()
  cAxisSystemTrajectoryReader.Open(pathTrajectoryFile)
  dmTrajectory = cAxisSystemTrajectoryReader.Read()

  cAxisSystemTrajectoryReader.Close()

  print "Done"

except:
  print "Exception raised"
```

Python Example

```
import sys

sys.path += [R" C:\Program Files\ANSYS Inc\v2XX\Optical Products\SPEOS\Bin"]

# References to Optis.Core and Optis.Data
import IllumineCore_pywrap as OptisCore
import IllumineData_pywrap as OptisData

try:
  firstData = OptisCore.DAxisSystemData()
  firstData.Time = 0.0
  firstData.Origin.Init(0, 0, 0)
  firstData.Direction_X.Init(1, 0, 0)
  firstData.Direction_Y.Init(0, 1, 0)

  lastData = OptisCore.DAxisSystemData()
  lastData.Time = 7.0
  lastData.Origin.Init(0, 0, 3000)
  lastData.Direction_X.Init(1, 0, 0)
```

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

```
  lastData.Direction_Y.Init(0, 1, 0)

  dmTrajectory = OptisCore.DAxisSystemTrajectory()
  dmTrajectory.Trajectory.Resize(2)
  dmTrajectory.Trajectory.Set(0, firstData)
  dmTrajectory.Trajectory.Set(1, lastData)

  strPathTrajectoryFile = OptisCore.String(R"C:\trajectory.json")
  pathTrajectoryFile = OptisCore.Path(strPathTrajectoryFile)

  cAxisSystemTrajectoryWriter = OptisData.CAxisSystemTrajectoryWriter()
  cAxisSystemTrajectoryWriter.Open(pathTrajectoryFile)
  cAxisSystemTrajectoryWriter.Write(dmTrajectory)

  cAxisSystemTrajectoryWriter.Close()

  print("Done")

except:
  print("Exception raised")
```

8.3.10.2.2.4. Trajectory Script Example

The following page gives you a full script example to create a trajectory in Speos.

> **Note:** Make sure to use the **3.9 version** of IronPython or Python language to write your scripts.

```
# © 2012-2021 ANSYS, Inc. All rights reserved. Unauthorized use, distribution,
 or duplication is
prohibited.

# THIS ANSYS SOFTWARE PRODUCT AND PROGRAM DOCUMENTATION INCLUDE TRADE SECRETS
AND ARE CONFIDENTIAL AND PROPRIETARY PRODUCTS OF ANSYS, INC., ITS SUBSIDIARIES,
 OR
LICENSORS.  The software products and documentation are furnished by ANSYS,
Inc., its subsidiaries,
or affiliates under a software license agreement that contains provisions
concerning non-disclosure,
copying, length and nature of use, compliance with exporting laws, warranties,
 disclaimers,
limitations of liability, and remedies, and other provisions.  The software
products and
documentation may be used, disclosed, transferred, or copied only in accordance
 with the terms and
conditions of that software license agreement.

import sys
import clr
import os
from os import path

sys.path += [R"C:\Program Files\ANSYS Inc\v211\Optical Products\SPEOS\Bin"]

clr.AddReferenceToFile("Optis.Core_net.dll")
clr.AddReferenceToFile("Optis.Data_net.dll")
```

```
# References to Optis.Core and Optis.Data
import Optis.Core as OptisCore
import Optis.Data as OptisData

def GetAxisSystemData(iTime, iFrame):
    dAxisSystemData = OptisCore.DAxisSystemData()
    dAxisSystemData.Time = iTime
    dAxisSystemData.Origin.Init(1000 * iFrame.Origin.X,
                                1000 * iFrame.Origin.Y,
                                1000 * iFrame.Origin.Z)

    tmpVector = OptisCore.Vector3_double()
    tmpVector.Init(iFrame.DirX.X,
                   iFrame.DirX.Y,
                   iFrame.DirX.Z)
    tmpVector.Normalize()

    dAxisSystemData.Direction_X.Init(tmpVector.Get(0),
                                     tmpVector.Get(1),
                                     tmpVector.Get(2))

    tmpVector.Init(iFrame.DirY.X,
                   iFrame.DirY.Y,
                   iFrame.DirY.Z)
    tmpVector.Normalize()

    dAxisSystemData.Direction_Y.Init(tmpVector.Get(0),
                                     tmpVector.Get(1),
                                     tmpVector.Get(2))

    return dAxisSystemData

# Working directory
workingDirectory = path.dirname(GetRootPart().Document.Path.ToString())

# SpaceClaim InputHelper
ihTrajectoryName = InputHelper.CreateTextBox("Trajectory.1", 'Trajectory name:
 ', 'Enter the name of the trajectory')
ihFrameFrequency = InputHelper.CreateTextBox(30, 'Timeline frame rate:', 'Set
 timeline frame rate (s-1)', ValueType.PositiveInteger)
ihReverseDirection = InputHelper.CreateCheckBox(False, "Reverse direction",
"Reverse direction on trajectory")
ihObjectSpeed = InputHelper.CreateTextBox(50, 'Object speed:', 'Set the moving
 object speed (km/h)', ValueType.PositiveDouble)

ihCoordinateSystem = InputHelper.CreateSelection("Reference coordinate system",
 SelectionFilter.CoordinateSystem, 1, True)
ihTrajectory = InputHelper.CreateSelection("Trajectory curve",
SelectionFilter.Curve, 1, True)

InputHelper.PauseAndGetInput('Trajectory parameters', ihTrajectoryName,
ihFrameFrequency, ihObjectSpeed, ihReverseDirection, ihCoordinateSystem,
ihTrajectory)

# Animation frame rate (s-1)
frameFrequency = float(ihFrameFrequency.Value)

# Object speed (km/h)
objectSpeed = float(ihObjectSpeed.Value)
```

```
# Trajectory file
trajectoryName = str(ihTrajectoryName.Value)
trajectoryAcquisitionFile = workingDirectory + "\\" + trajectoryName

# Reference coordinate system
coordSysSelection = ihCoordinateSystem.Value
trajectoryCoordSys = coordSysSelection.Items[0]

# Trajectory curve
trajCurveSelection = ihTrajectory.Value
trajectoryCurve = trajCurveSelection.Items[0]

# Reversed trajectory (Boolean)
isReversedTrajectory = ihReverseDirection.Value

# Acquisition of positions
def GetPositionOrientation(i_CoordSys, i_ReferenceCoordSys):
    # change base of current position
    newMatric = Matrix.CreateMapping(i_ReferenceCoordSys.Frame)

    currentPosition = newMatric.Inverse  * i_CoordSys.Frame.Origin
    currentPosition = Point.Create(round(currentPosition.X, 6),
round(currentPosition.Y, 6), round(currentPosition.Z, 6))

    # change base of current iDirection
    iDirectionX = newMatric.Inverse  * i_CoordSys.Frame.DirX

    # change base of current iDirection
    jDirectionX = newMatric.Inverse  *i_CoordSys.Frame.DirY

    # Return new frame
    return Frame.Create(currentPosition, iDirectionX, jDirectionX)

def MoveAlongTrajectory(i_trajectoryCoordSys, i_trajectoryCurve,
i_isReversedTrajectory, i_frameFrequency, i_objectSpeed):
    selectedCoordSys = Selection.Create(i_trajectoryCoordSys)
    newselectedCoordSys = i_trajectoryCoordSys

    pathLength = i_trajectoryCurve.Shape.Length
    selectedCurve = Selection.Create(i_trajectoryCurve)

    # Convert speed in m/s
    convObjectSpeed = float(i_objectSpeed * 1000 / 3600)

    currentPosition = 0.0
    timeStamp = 0.0
    positionTable = []
    timeTable = []

    while currentPosition < 1:
        options = MoveOptions()

        if currentPosition == 0:
            options.Copy = True
        else:
            options.Copy = False

        if i_isReversedTrajectory:
            result = Move.AlongTrajectory(selectedCoordSys, selectedCurve,
1-currentPosition, options)
```

```
            if currentPosition == 0:
              newselectedCoordSys = result.GetCreated[ICoordinateSystem]()[0]

                selectedCoordSys = Selection.Create(newselectedCoordSys)
                if newselectedCoordSys.Frame.Origin !=
i_trajectoryCoordSys.Frame.Origin:
                    options.Copy = False
                    result = Move.AlongTrajectory(selectedCoordSys,
selectedCurve, currentPosition, options)
        else:
            result = Move.AlongTrajectory(selectedCoordSys, selectedCurve,
currentPosition, options)

            if currentPosition == 0:
              newselectedCoordSys = result.GetCreated[ICoordinateSystem]()[0]

                selectedCoordSys = Selection.Create(newselectedCoordSys)
                if newselectedCoordSys.Frame.Origin !=
i_trajectoryCoordSys.Frame.Origin:
                    options.Copy = False
                    result = Move.AlongTrajectory(selectedCoordSys,
selectedCurve, currentPosition, options)

        if(result):
            movedFrame = GetPositionOrientation(newselectedCoordSys,
i_trajectoryCoordSys)
            positionTable.append(movedFrame)
            timeTable.append(timeStamp)

        currentPosition += (convObjectSpeed / i_frameFrequency)/pathLength
        timeStamp +=  1/float(i_frameFrequency)

    result = Delete.Execute(selectedCoordSys)

    return timeTable, positionTable

# Length of path (m)
pathLength = trajectoryCurve.Shape.Length

# Get time stamps and relative coordinate systems
timeTable, positionTable = MoveAlongTrajectory(trajectoryCoordSys,
trajectoryCurve, isReversedTrajectory,  frameFrequency, objectSpeed)

dAxisSystemData_Table = []
for time in timeTable:
    timeIndex = timeTable.index(time)
    fFrame = positionTable[timeIndex]

    dAxisSystemData = GetAxisSystemData(time, fFrame)
    dAxisSystemData_Table.append(dAxisSystemData)

if len(dAxisSystemData_Table) > 0:
    dmTrajectory = OptisCore.DAxisSystemTrajectory()
    dmTrajectory.Trajectory.Resize(len(dAxisSystemData_Table))

    for dAxisSystemData in dAxisSystemData_Table:

dmTrajectory.Trajectory.Set(dAxisSystemData_Table.index(dAxisSystemData),
dAxisSystemData)

    pathTrajectoryFile = str(workingDirectory + "\\" +
```

```
str(ihTrajectoryName.Value) + ".json")
    strPathTrajectoryFile = OptisCore.String.From(pathTrajectoryFile)
    pathTrajectoryFile = OptisCore.Path(strPathTrajectoryFile)

    cAxisSystemTrajectoryWriter = OptisData.CAxisSystemTrajectoryWriter()
    cAxisSystemTrajectoryWriter.Open(pathTrajectoryFile)
    cAxisSystemTrajectoryWriter.Write(dmTrajectory)

    cAxisSystemTrajectoryWriter.Close()
```

8.3.10.3. Generating a Scanning Sequence File

Speos provides you with two ways of generating a Scanning Sequence file (*.OPTScanSequence or *.txt), either with an Excel file or a python script file, to help you define the parameters of the emission pattern of the LiDAR and generate the *.OPTScanSequence / *.txt file to be used in the LiDAR Speos definition.

8.3.10.3.1. Generating a Scanning Sequence File with Excel

Speos provides you with an Excel file to help you define the parameters of the emission pattern of the LiDAR and generate the *.OPTScanSequence / *.txt file to be used in the LiDAR Speos definition.

To generate the *.OPTScanSequence / *.txt:

Make sure your regional settings use a "." as decimal separator and not a ",".

Download the Firing_Sequence_Files_Examples.zip file.

1. Open the **LiDAR_scanning_sequence_generator.xlsm** file.
2. In **File**, define the path and name of the *.OPTScanSequence file to be generated.
3. In **Type of pulses**, define the type of the pulse of the beams.
   The **Type of pulses** is applied to each beam defined as **From pulses** in the **Pulse type** definition.
4. In **Width of pulses (s)**, define the duration of one pulse.
5. In the table, define the different parameters. Each line you add represents a beam.
6. Click **Generate**.

A *.OPTScanSequence file has been generated and is ready to be used in a LiDAR sensor definition.

8.3.10.3.2. Generating a Scanning Sequence File with Python

Speos provides you with an Python script file to help you understand how to create and generate the *.OPTScanSequence / *.txt file to be used in the LiDAR Speos definition.

To generate the *.OPTScanSequence / *.txt:

Make sure your regional settings use a "." as decimal separator and not a ",".

Download the Firing_Sequence_Files_Examples.zip file.

1. Open the **LiDAR_scanning_sequence_generator.py** python script file.

DRAFT

> **Note:** From the Line 156, you will define the parameters to set as indicated by the comments. However the line numbering can vary according to the modifications you apply.

2. From Line 162, define the **PulseShape** (Rectangular, Triangular, Gaussian, Dirac) and **PulseWidth** corresponding to the default width of pulses in seconds.

```
#setup default pulse shape
scan_sequence_data_model.EPulseShape =
OptisCore.Enum_PulseShape(OptisCore.PulseShapeGaussian)
scan_sequence_data_model.PulseWidth = 0.001
```

3. From Line 166, if you want to apply an unique wavelength, define the default monochromatic spectrum in nm.

```
# add a monochromatic spectrum
spectrum_Monochromatic = OptisCore.DSpectrumMonochromatic.Make()
spectrum_Monochromatic.Ptr().Wavelength = 950.0
scan_sequence_data_model.SpectrumList.Push_back(OptisCore.SharedPtr_Cast_DISpectrum_From_DSpectrumMonochromatic(spectrum_Monochromatic))
```

4. From Line 171, ff you want to apply a spectrum with multiple wavelengths, define the sampled spectrum in nm.

```
# add a sampled spectrum
spectrum_Sampled = OptisCore.DSpectrumSampled.Make()

# Define the number of wavelength of the spectrum
numberOfWavelengths = 11
spectrum_Sampled.Ptr().Wavelengths.Resize(numberOfWavelengths)
spectrum_Sampled.Ptr().Values.Resize(numberOfWavelengths)

# Define the value of each wavelength
for i in range(numberOfWavelengths):
w = 945.0 + i * 10.0 / (numberOfWavelengths - 1)
spectrum_Sampled.Ptr().Wavelengths.Set(i, w)
spectrum_Sampled.Ptr().Values.Set(i, 25.0 - (w - 950.0) ** 2)

scan_sequence_data_model.SpectrumList.Push_back(OptisCore.SharedPtr_Cast_DISpectrum_From_DSpectrumSampled(spectrum_Sampled))
```

5. From Line 184, define the intensity distribution of the beams.

```
# add an intensity distribution
numberOfPhiAngles = 5
numberOfThetaAngles = 13
intensity = OptisCore.DDistribution2Double.Make()
intensity.Ptr().Distribution.Allocate(numberOfPhiAngles, numberOfThetaAngles)

for p in range(numberOfPhiAngles):
intensity.Ptr().Distribution.m_X.Set(p, p * 360.0 / (numberOfPhiAngles - 1))

pos = OptisCore.Extent_uint_2()

for t in range(numberOfThetaAngles):
theta = 90.0 * t / (numberOfThetaAngles - 1)
intensity.Ptr().Distribution.m_Y.Set(t, theta)
pos.Set(1, t)

for p in range(numberOfPhiAngles):
pos.Set(0, p)
```

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

```
intensity.Ptr().Distribution.m_Value.Set(pos, math.cos(math.radians(theta)))
 # lambertian (no need to normalize)

scan_sequence_data_model.SampledIntensityList.Push_back(intensity)
```

6. From Line 206, add as many beams as you want:

```
# add few beams:
scanParameters1 = OptisCore.DScanParam()
scanParameters1.Position.Init(0.1, 0.2, 0.3) # mm

# Choose between Rectangular and Elliptic
# Or for a Point shape, do not add the parameters ESurfaceShape, SizeX, SizeY
scanParameters1.ESurfaceShape =
OptisCore.Enum_LidarEmissionSurfaceShape(OptisCore.LidarEmissionSurfaceShapeRectangular)
scanParameters1.SizeX = 3.0 # mm
scanParameters1.SizeY = 2.0 # mm

scanParameters1.ShootingTime = 0.001 # s

scanParameters1.PulseEnergy = 0.05 # J

# Choose between Triangular, Rectangular, Gaussian, Dirac
# Or use ShapeUndefined to use the PulseShape defined in Step 3
scanParameters1.EPulseShape =
OptisCore.Enum_PulseShape(OptisCore.PulseShapeTriangular)

# If you want to use the PulseWidth defined in Step 3, do not add the parameter
 line for PulseWidth
scanParameters1.PulseWidth = 0.00002 # s

# If you do not want to apply an angle, do not add the corresponding parameter
 line
scanParameters1.RotationAngle = 0.0132 # azimuth angle in degrees
scanParameters1.TiltAngle = 0.0276 # elevation angle degrees
scanParameters1.PsiAngle = 0.234 # degrees

# Choose between Gaussian and Sampled
# Or reomove the parameter line to use the Intensity defined in the Source -
 Intensity section's parameters from the LiDAR sensor definition
scanParameters1.EIntensityType =
OptisCore.Enum_LidarIntensityType(OptisCore.LidarIntensityTypeSampled)

# If you selected Sampled as EIntensityType, add the following parameter line
 IntensityIndex and define its index
scanParameters1.IntensityIndex = 0

#If you selected Gaussian as EIntensityType, add the following parameter lines
scanParameters1.GaussianXWidth = 0.5 # FWHM in degrees
scanParameters1.GaussianYHeight = 0.5 # FWHM in degrees
scanParameters1.GaussianTotalAngle = 10.0 # degrees

scanParameters1.SpectrumIndex = 1

scan_sequence_data_model.ScanParamList.Push_back(scanParameters1)
```

For each beam you add, define its parameters using the following template:

```
scanParameters[N] = OptisCore.DScanParam()
scanParameters[N].PARAMETER_TO_DEFINE
```

```
scanParametersN.PARAMETER_TO_DEFINE
...
...
scan_sequence_data_model.ScanParamList.Push_back(scanParametersN)
```

7. From Line 249, define the *OPTScanSequence file name.

```
# serialization file
currentdirectory = os.path.dirname(os.path.realpath(__file__))
sequence_file_name = OptisCore.String(os.path.join(currentdirectory,
"sampleFile.OPTScanSequence"))
sequence_filepath = OptisCore.Path(sequence_file_name)
```

8. Close the Python script file and run it.

A *.OPTScanSequence file has been generated and is ready to be used in a LiDAR sensor definition.

8.3.10.4. Creating a LiDAR Sensor

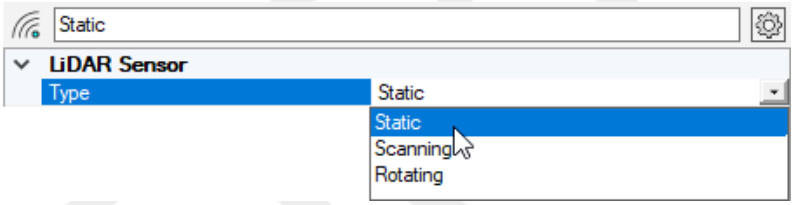
This section gathers the LiDAR sensor creation procedures for the three LiDAR sensor types available in Speos

8.3.10.4.1. Creating a Static LiDAR Sensor

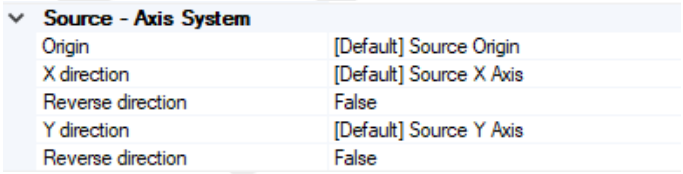
This page shows how to create a static LiDAR sensor that will be used for LiDAR simulation.

To create a Static LiDAR Sensor:

1. From the **Light Simulation** tab, click **System** > **LiDAR**.
2. From the **Type** drop-down list, select **Static**.



3. Define the axis system of the source (emitter channel) by setting its position and orientation.



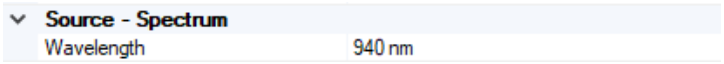
> **Note:** To correctly orient the beam, first verify the orientation of the IES file.

- to select one point for the origin (point where the pulse is emitted).
- to select a line to define the horizontal axis (corresponding to the X axis of the IES file).

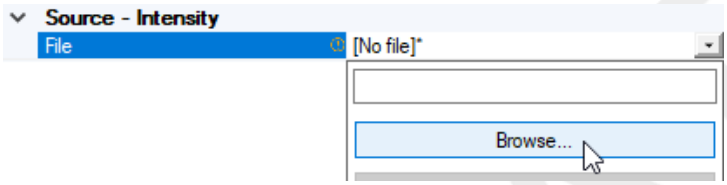
- 
  to select a line to define the vertical axis (corresponding to the Y axis of the IES file).
- or click
  
  and select a coordinate system to autofill the Axis System.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

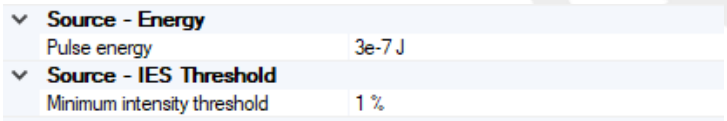
4. Define the monochromatic spectrum of the source.



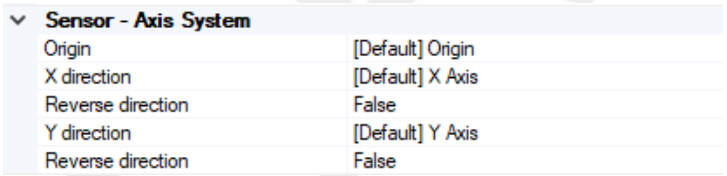
5. To define the source intensity distribution, in **Source-Intensity**, click **Browse** to load an IES (.ies) or Eulumdat (.ldt) file.



6. In **Pulse energy**, define the energy of a single pulse emitted by the LiDAR source (in Joule).



7. Set a **Minimum intensity threshold** to define a threshold under which the signal of the LiDAR source is not taken into account. This helps LiDAR differentiate what should be used or ignored to calculate LiDAR's field of view.
8. Define the axis system of the sensor (receiver channel):



- 
  to select one point for the origin (point from which the pulse is received).
- 
  to select a line to define the horizontal axis of the sensor.
- 
  to select a line to define the vertical axis of the sensor.
- or click
  
  and select a coordinate system to autofill the Axis System.

9. In the **Optics** section, define the properties of the sensor's objective:

Sensor - Optics	
Distortion file	[No file]*
Transmittance	85 %
Focal length	15 mm
Pupil	10 mm
Horizontal FOV	0 °
Vertical FOV	0 °

- In **Distortion file**, click **Browse** to load an .OPTDistortion file.

> **Note:** The *.OPTDistortion file is used to introduce/replicate the optical distortion of the lens. Every lens has varying degrees of distortion.

- Define the **Transmittance** (capacity to allow the light to pass through) of the lens.
- In **Focal length**, define the distance between the center of the optical system to the focus.
- In **Pupil**, define the diameter of the objective aperture.

  The Horizontal and Vertical Field of Views (FOV) are calculated according to the distortion, transmittance, focal length and pupil of the objective.

10. Define the size and resolution of the imager (the sensor) that is placed behind the objective.

Sensor - Imager	
Width	3.2 mm
Height	3.2 mm
Resolution (beta)	True
Horizontal pixels	64
Vertical pixels	64

- Define the Width and Height of the LiDAR sensor.
- Define the number of horizontal and vertical pixels to determine the LiDAR resolution.

11. Define the sensor range and accuracy:

Sensor - Range and accuracy	
Start	0 mm
End	100000 mm
Spatial accuracy	100 mm

- In **Start**, type the minimum distance from which the LiDAR is able to operate and integrate rays.
- In **End**, type the maximum distance up to which the LiDAR is able to operate and integrate rays.
- In **Spatial accuracy**, define the sampling used to save the Raw time of flight.

  The time of flight is the time taken by the light to travel from the LiDAR to the object.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

12. If you want to define an **Aiming area** for the sensor:

DRAFT

Sensor - Aiming area	
User defined	True
Origin	[Default] Sensor Origin
X Direction	[Default] Sensor X Direction
Y Direction	[Default] Sensor Y Direction
Type	Elliptic
Width	10 mm
Height	10 mm

a) Set **User defined** to **True**.
In the 3D view, a preview of the aiming area is displayed to help position it to the desired location.
b) If you want to adjust the axis system (that is, by default, the same as the sensor's), select one point for the **Origin** and two lines for X and Y directions or click  and select a coordinate system to autofill the Axis System.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

c) From the **Type** drop-down list, select the shape you want to use for the aiming area:
- Select **Rectangle**
- Select **Elliptic**

d) Define the **Width** and **Height** of the cover lens.

The LiDAR sensor is created and is visible both in Speos tree and in the 3D view.

You can now create a LiDAR Simulation.

Related concepts

LiDAR Sensor Parameters on page 226
The following section provides more information on static, scanning or rotating LiDAR sensor parameters.

Related information

Creating a LiDAR Simulation on page 446
Creating a LiDAR simulation allows you to generate output data and files that enable to analyze a LiDAR system and configuration. The LiDAR simulation supports several sensors at a time.

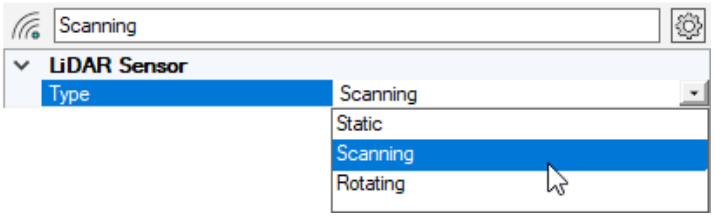
8.3.10.4.2. Creating a Scanning LiDAR Sensor

This page shows you how to create a scanning LiDAR sensor that will be used for LiDAR simulation.

To create a Scanning LiDAR Sensor:

1. From the **Light Simulation** tab, click **System**  > **LiDAR**  .
2. From the **Type** drop-down list, select **Scanning**.

DRAFT



3. Define the axis system of the LiDAR physical module by selecting an origin point , a line for the horizontal direction and a line for the vertical direction or click and select a coordinate system to autofill the Axis System.

System - Axis System	
Origin	[Default] Sensor Origin
X direction	[Default] Sensor X Axis
Reverse direction	False
Y direction	[Default] Sensor Y Axis
Reverse direction	False

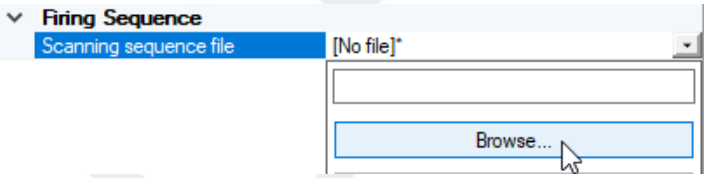
> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

4. If you want to create a dynamic LiDAR sensor, in the **Trajectory file** field, click **Browse** to load a trajectory file (.json).

> **Note:** For more information on trajectory files, refer to Trajectory Files.

When a trajectory file is assigned to a LiDAR sensor and the feature is edited, the trajectory is displayed in the 3D view.

5. In **Firing Sequence**, click **Browse** to load a *.txt file describing the scanning pattern of the LiDAR.



> **Note:** If you need more information on scanning pattern files, see **Firing Sequence Files**.

6. Define the axis system of the source (emitter channel):

Source - Axis System	
Origin	[Default] Source Origin
X direction	[Default] Source X Axis
Reverse direction	False
Y direction	[Default] Source Y Axis
Reverse direction	False

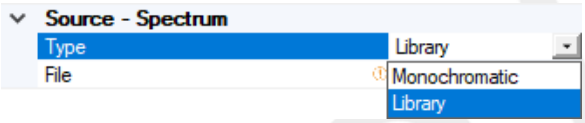
Note: When using an intensity distribution file as source, first verify the orientation of the IES file to correctly orient the beam.

- 
to select one point for the origin (point where the pulse is received).
- 
to select a line to define the horizontal axis (corresponding to the X axis of the IES file).
- 
to select a line to define the vertical axis (corresponding to the Y axis of the IES file).
- or click

and select a coordinate system to autofill the Axis System.

Note: If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

7. Define the spectrum of the source.

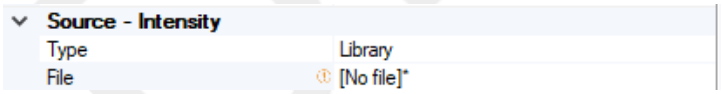


- Select **Monochromatic** and specify the wavelength in nanometers.
- Select **Library** and from the drop-down list click **Browse** to load a .spectrum file.

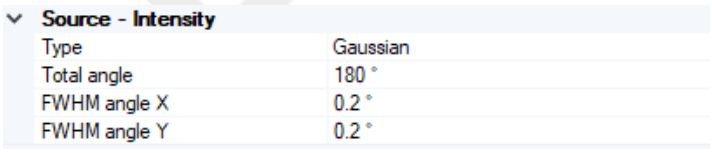
  If you want to see the file properties or edit the file, click **Open file** to open the **Spectrum Editor**  .

8. Define the source intensity distribution of the LiDAR:

- Select **Library** to load an IES (.ies) or Eulumdat (.ldt) file.



- Select **Gaussian** to manually define the intensity distribution profile of the source.



  - Set the total angle of emission of the source.
  - Set the **FWHM angle** for X and Y.
  - In the 3D view, click two lines to define X direction and Y direction.

9. Define the axis system of the sensor (receiver channel):

Sensor - Axis System	
Origin	[Default] Origin
X direction	[Default] X Axis
Reverse direction	False
Y direction	[Default] Y Axis
Reverse direction	False

- 
to select one point for the origin (point from which the pulse is emitted).
- 
to select a line to define the horizontal axis of the sensor.
- 
to select a line to define the vertical axis of the sensor.
- or click

and select a coordinate system to autofill the Axis System.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

10. In the **Optics** section, define the properties of the sensor's objective:

Sensor - Optics	
Distortion file	[No file]*
Transmittance	85 %
Focal length	15 mm
Pupil	10 mm
Horizontal FOV	0 °
Vertical FOV	0 °

- In **Distortion file**, click **Browse** to load an .OPTDistortion file.

> **Note:** The *.OPTDistortion file is used to introduce/replicate the optical distortion of the lens. Every lens has varying degrees of distortion.

- Define the **Transmittance** (capacity to allow the light to pass through) of the lens:

  For a monochromatic source, define a constant Transmittance.

  For a source using a *.spectrum file, click **Browse** to load a **Transmittance spectrum file**.
- In **Focal length**, define the distance between the center of the optical system to the focus.
- In **Pupil**, define the diameter of the objective aperture.

  The Horizontal and Vertical Field of Views (FOV) are automatically calculated according to the parameters of the objective.

> **Note:** The fields of view are only indicative and will not be used for LiDAR simulation.

11. Define the **Width** and **Height** to define the size of the imager (the sensor) that is placed behind the objective.

Sensor - Imager	
Width	3.2 mm
Height	3.2 mm
Resolution (beta)	True
Horizontal pixels	64
Vertical pixels	64

12. When Beta features are enabled, if you want to define the imager resolution, set **Resolution (beta)** to **True** and define the number of **Horizontal pixels** and **Vertical pixels**.
13. Define the sensor range and accuracy:

Sensor - Range and accuracy	
Start	0 mm
End	100000 mm
Spatial accuracy	100 mm

- In **Start**, type the minimum distance from which the LiDAR is able to operate and integrate rays.
- In **End**, type the maximum distance up to which the LiDAR is able to operate and integrate rays.
- In **Spatial accuracy**, define the sampling used to save the Raw time of flight.

  The time of flight is the time taken by the light to travel from the LiDAR to the object.

14. If you want to define an Aiming area for the sensor:

Sensor - Aiming area	
User defined	True
Origin	[Default] Sensor Origin
X Direction	[Default] Sensor X Direction
Y Direction	[Default] Sensor Y Direction
Type	Elliptic
Width	10 mm
Height	10 mm

a) Set **User defined** to **True**.
In the 3D view, a preview of the aiming area is displayed to help position it to the desired location.
b) If you want to adjust the axis system (that is, by default, the same as the sensor's), select one point for the **Origin** and two lines for X and Y directions or click  and select a coordinate system to autofill the Axis System.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

c) From the **Type** drop-down list, select the shape you want to use for the aiming area:
- Select **Rectangle**
- Select **Elliptic**

d) Define the **Width** and **Height** of the cover lens.

The LiDAR sensor is created and is visible both in Speos tree and in the 3D view. This type of LiDAR generates only one simulation result (*.OPTTimeOfFlight).

You can now create a LiDAR Simulation.

DRAFT

Related concepts

LiDAR Sensor Parameters on page 226
The following section provides more information on static, scanning or rotating LiDAR sensor parameters.

Related information

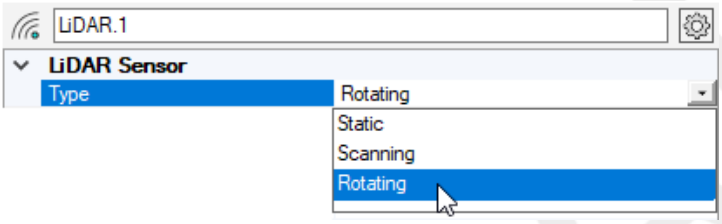
Creating a LiDAR Simulation on page 446
Creating a LiDAR simulation allows you to generate output data and files that enable to analyze a LiDAR system and configuration. The LiDAR simulation supports several sensors at a time.

8.3.10.4.3. Creating a Rotating LiDAR Sensor

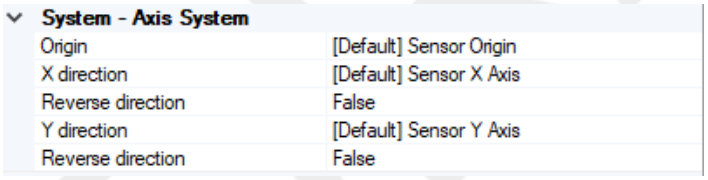
This page shows you how to create a rotating LiDAR sensor that will be used for LiDAR simulation.

To create a Rotating LiDAR Sensor:

1. From the **Light Simulation** tab, click **System** > **LiDAR**.
2. From the **Type** drop-down list, select **Rotating**.



3. Define the axis system of the LiDAR physical module by selecting an origin point, a line for the horizontal direction and a line for the vertical direction or click and select a coordinate system to autofill the Axis System.



> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

4. If you want to create a dynamic LiDAR sensor, in the **Trajectory file** field, click **Browse** to load a trajectory file (.json).

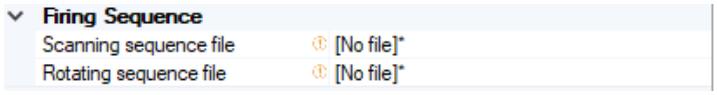
> **Note:** For more information on trajectory files, refer to Trajectory Files.

When a trajectory file is assigned to a LiDAR sensor and the feature is edited, the trajectory is displayed in the 3D view.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

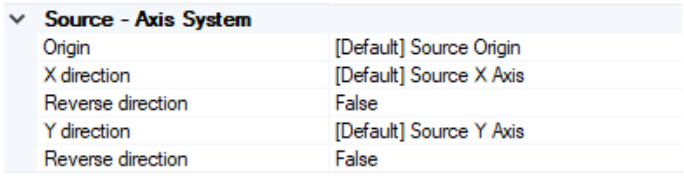
Published: 2023-10-23T14:08:14.04-04:00

5. In **Firing Sequence**, click **Browse** to load two *.txt files respectively describing the scanning pattern and the rotation pattern of the LiDAR.



> **Note:** If you need more information on scanning pattern files, see **Firing Sequence Files**.

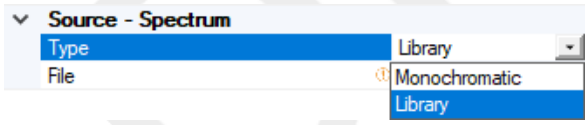
6. Define the axis system of the source (emitter channel):



> **Note:** When using an intensity distribution file as source, first verify the orientation of the IES file to correctly orient the beam.

- 
  to select one point for the origin (point where the pulse is received).
- 
  to select a line to define the horizontal axis (corresponding to the X axis of the IES file).
- 
  to select a line to define the vertical axis (corresponding to the Y axis of the IES file).
- or click
  
  and select a coordinate system to autofill the Axis System.

7. Define the spectrum of the source.

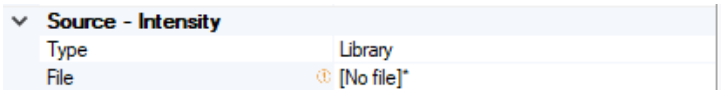


- Select **Monochromatic** and specify the wavelength in nanometers.
- Select **Library** and from the drop-down list click **Browse** to load a .spectrum file.

  If you want to see the file properties or edit the file, click **Open file** to open the **Spectrum Editor**  .

8. Define the source intensity distribution of the LiDAR:

- Select **Library** to load an IES (.ies) or Eulumdat (.ldt) file.



- Select **Gaussian** to manually define the intensity distribution profile of the source.

Source - Intensity	
Type	Gaussian
Total angle	180 °
FWHM angle X	0.2 °
FWHM angle Y	0.2 °

- Set the total angle of emission of the source.
- Set the **FWHM angle** for X and Y.
- In the 3D view, click two lines to define X direction and Y direction.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

9. Define the axis system of the sensor (receiver channel):

Sensor - Axis System	
Origin	[Default] Origin
X direction	[Default] X Axis
Reverse direction	False
Y direction	[Default] Y Axis
Reverse direction	False

- 
to select one point for the origin (point from which the pulse is emitted).
- 
to select a line to define the horizontal axis of the sensor.
- 
to select a line to define the vertical axis of the sensor.
- or click

and select a coordinate system to autofill the Axis System.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

10. In the **Optics** section, define the properties of the sensor's objective:

Sensor - Optics	
Distortion file	[No file]*
Transmittance	85 %
Focal length	15 mm
Pupil	10 mm
Horizontal FOV	0 °
Vertical FOV	0 °

- In **Distortion file**, click **Browse** to load an .OPTDistortion file.

DRAFT

> **Note:** The *.OPTDistortion file is used to introduce/replicate the optical distortion of the lens. Every lens has varying degrees of distortion.

- Define the **Transmittance** (capacity to allow the light to pass through) of the lens:

  For a monochromatic source, define a constant Transmittance.

  For a source using a *.spectrum file, click **Browse** to load a **Transmittance spectrum file**.
- In **Focal length**, define the distance between the center of the optical system to the focus.
- In **Pupil**, define the diameter of the objective aperture.

  The Horizontal and Vertical Field of Views (FOV) are automatically calculated according to the parameters of the objective.

> **Note:** The fields of view are only indicative and will not be used for LiDAR simulation.

11. Define the **Width** and **Height** to define the size of the imager (the sensor) that is placed behind the objective.

Sensor - Imager	
Width	3.2 mm
Height	3.2 mm
Resolution (beta)	True
Horizontal pixels	64
Vertical pixels	64

12. When Beta features are enabled, if you want to define the imager resolution, set **Resolution (beta)** to **True** and define the number of **Horizontal pixels** and **Vertical pixels**.
13. Define the sensor range and accuracy:

Sensor - Range and accuracy	
Start	0 mm
End	100000 mm
Spatial accuracy	100 mm

- In **Start**, type the minimum distance from which the LiDAR is able to operate and integrate rays.
- In **End**, type the maximum distance up to which the LiDAR is able to operate and integrate rays.
- In **Spatial accuracy**, define the sampling used to save the Raw time of flight.

  The time of flight is the time taken by the light to travel from the LiDAR to the object.

14. If you want to define an **Aiming area** for the sensor:

Sensor - Aiming area	
User defined	True
Origin	[Default] Sensor Origin
X Direction	[Default] Sensor X Direction
Y Direction	[Default] Sensor Y Direction
Type	Elliptic
Width	10 mm
Height	10 mm

a) Set **User defined** to **True**.
In the 3D view, a preview of the aiming area is displayed to help position it to the desired location.

b) If you want to adjust the axis system (that is, by default, the same as the sensor's), select one point for the **Origin** and two lines for X and Y directions or click  and select a coordinate system to autofill the Axis System.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

c) From the **Type** drop-down list, select the shape you want to use for the aiming area:

- Select **Rectangle**
- Select **Elliptic**

d) Define the **Width** and **Height** of the cover lens.

The LiDAR sensor is created and is visible both in Speos tree and in the 3D view. This type of LiDAR generates only one simulation result (*.OPTTimeOfFlight).

You can now create a LiDAR Simulation.

Related concepts

LiDAR Sensor Parameters on page 226
The following section provides more information on static, scanning or rotating LiDAR sensor parameters.

Related information

Creating a LiDAR Simulation on page 446
Creating a LiDAR simulation allows you to generate output data and files that enable to analyze a LiDAR system and configuration. The LiDAR simulation supports several sensors at a time.

8.3.11. Camera Sensor

The Camera Sensor allows you to simulate rays integration as in a real camera according to the camera parameters set.

> **Note:** This feature is only available with the **Speos Optical Sensor Test** add on.

8.3.11.1. Camera Sensor General View

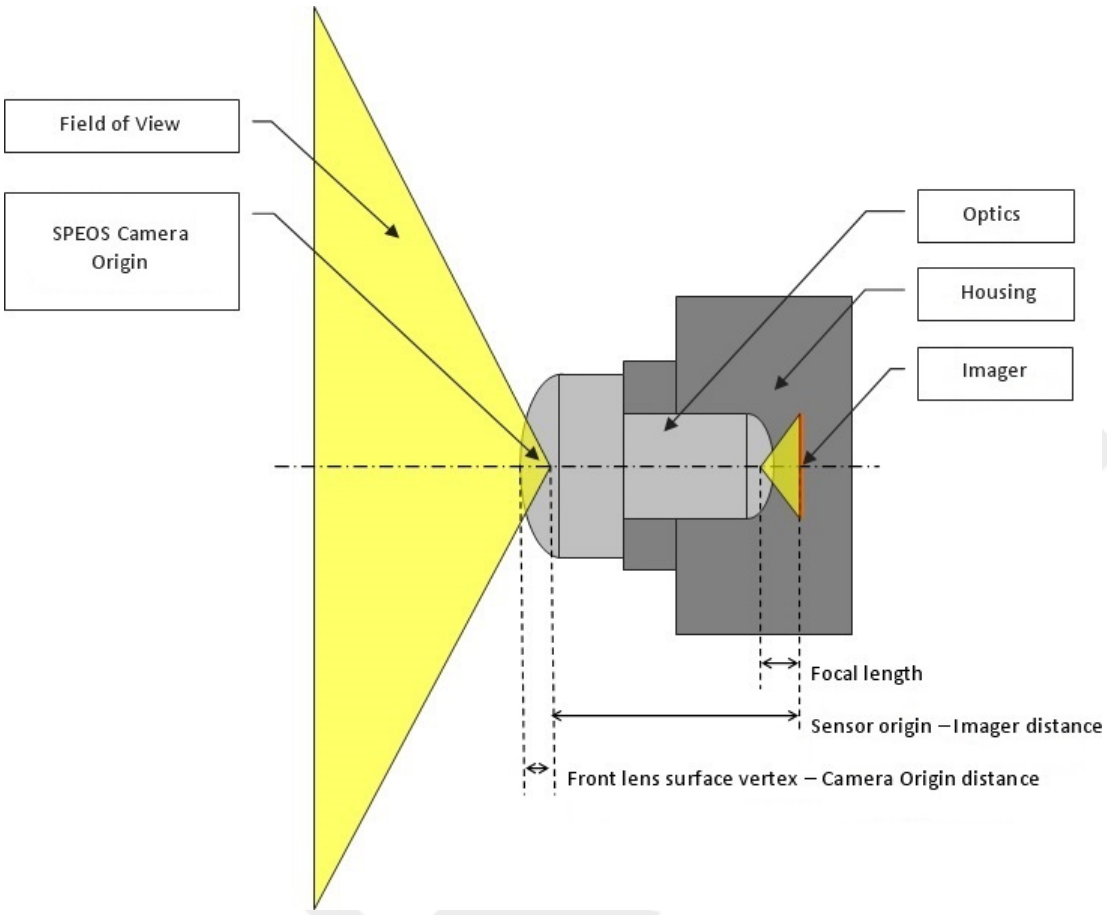
This page illustrates the representation of a sketched camera.

The illustration below represents the main parameters to set when creating a camera sensor.

> **Tip:** For the Distortion Curve (V1), the origin corresponds to the entrance pupil point (EPP) (Speos Camera Origin).

DRAFT

For the Speos Lens System (V2), the origin corresponds to the center of the imager.



8.3.11.2. Camera Sensor Parameters

This section allows you to better apprehend the camera sensor definition as it describes key settings of the sensor.

8.3.11.2.1. Camera Models

This section describes the camera models and their associated file formats available in Speos.

8.3.11.2.1.1. Camera Models Description

This page describes the different camera models that can be used to render the camera behavior in Speos.

To simulate the behavior of a camera, you can create an irradiance simulation containing the complete optical system of the camera, or you can create a camera simulation containing only the camera model.

The camera model allows you to render the behavior of the complete camera system, that is how light is going to propagate through the lens system.

In Speos, three models with different levels of complexity are available to imitate the behavior of a camera. The complexity of the models is based on the number and variety of inputs they consider/contain.

The models are described in an ***.OPTDistortion** file, used as input of the camera sensor.

The goal of using an *.OPTDistortion file into a simulation is to:

- perform fast simulations of the camera behavior (compared to simulating the complete camera system)
- reproduce most of the real lens properties
- perform simulation without exposing the manufacturer intellectual properties

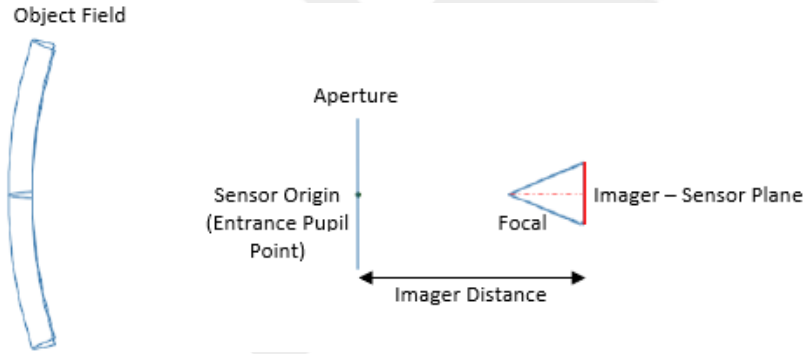
Model / Capabilities	Model V1	Model V2 & V2.1	Model Binary
Distortion			
Entrance pupil position			
Other geometric aberrations			
Depth of field			
Chromatic aberrations			
Physical aberrations (diffraction)			
Radiometry			

Basic Distortion Curve (V1)

The first model describes the angular relationship between the incoming ray's angle and the output ray's angle. This relationship is called Distortion Curve.

The basic model assumes that the origin (entrance pupil point) is fixed and the lens has a revolution symmetry.

Figure 40. Object to Image Angle relationship



Speos Lens System (V2)

This model allows you to simulate more complex behaviors as it can take into account origin (center of the imager) variation and asymmetry.

Using an ***.OPTDistortion** file based on this model as input of your camera sensor definition also allows you to generate a spectral irradiance map. Spectral irradiance maps reflect the photon flux received by the sensor's surface in W/m^2 and can be easily post-processed.

Note: These maps (.irradiance.xmp), generated as a result of a simulation, are available only in your **SPEOS output files folder**. They are not displayed in Speos tree.

The output of a Camera sensor using a Speos Lens System provides a 180°-rotated image which corresponds to the reality.

Figure 41. Variable origin on optical axis

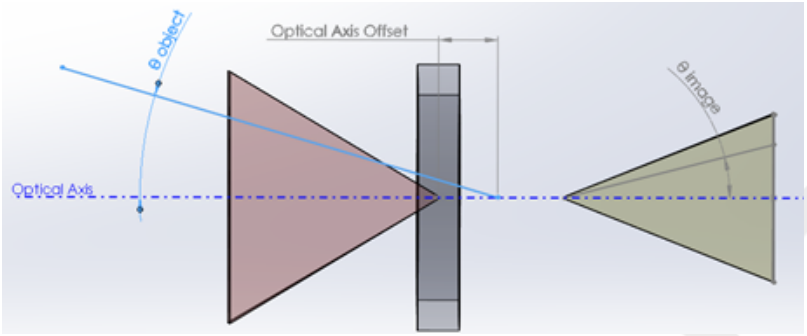
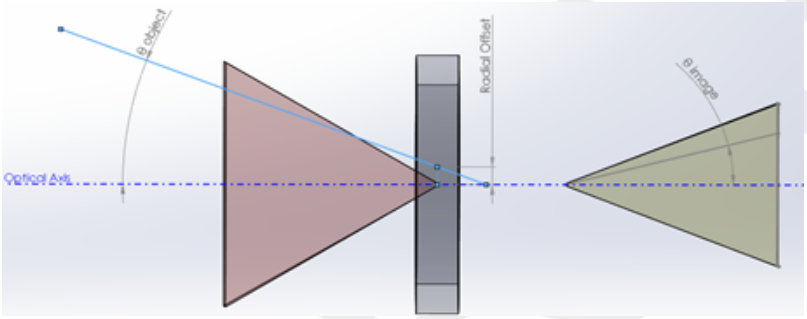


Figure 42. Origin shifted from optical axis



*.OPTDistortion Binary File

Important: The *.OPTDistortion Binary File has been introduced in Speos 2024 R1. Every time improvements are released on the file, you will have to use at least the Speos version for which improvements have been applied.

The *.OPTDistortion binary file is the more complex distortion file as it also consider chromatic and geometric aberrations. The lens data model described in the *.OPTDistortion file is generated from a complete camera system created in Zemax OpticStudio.

Note: This is a binary file, so it is not readable and can only be used as input of a Camera sensor.

Related concepts

Distortion Curve on page 262

The Distortion curve describes an angular relationship between the incoming ray's angle and the imager ray's angle. This relationship is described in an *.OPTDistortion file that is interpreted by Speos to render the camera behavior.

Related reference

Speos Lens System on page 264
This page merely describes the data model and file format of the Speos Lens System.

8.3.11.2.1.2. Distortion Curve

The Distortion curve describes an angular relationship between the incoming ray's angle and the imager ray's angle. This relationship is described in an *.OPTDistortion file that is interpreted by Speos to render the camera behavior.

Distortion Curve Principle

Speos algorithm is based on the pinhole camera principle.

In a pinhole camera model, the size of the image is predicted by the distance between the chief ray (ray starting from the object point passing through the center of the aperture stop) and the optical axis in the imager plane.

Knowing the relationship between incoming ray's angle and the imager ray's angle for each object point allows Speos to simulate the camera behavior. This angular relationship is called **Distortion Curve**. The Distortion Curve, basically, provides the output ray direction in relation to the input ray direction.

> **Note:** This curve is also called the "Chief ray angle curve" or "Principal ray angle curve".

Figure 43. Distortion

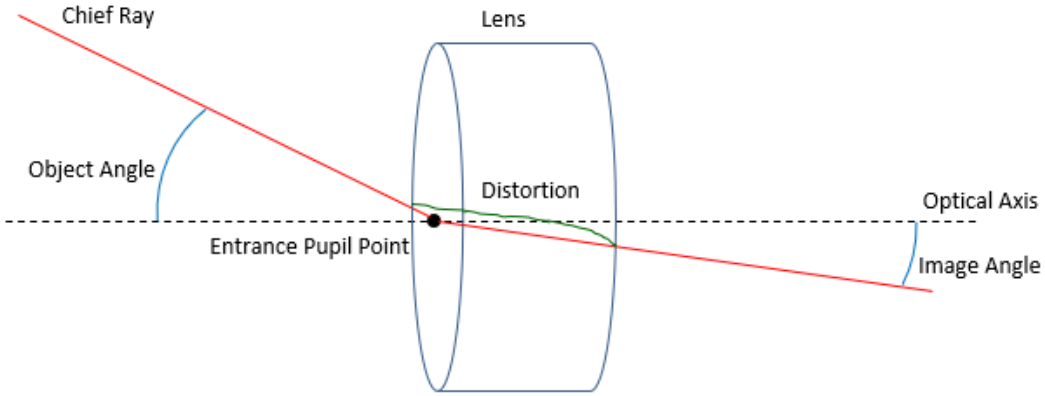
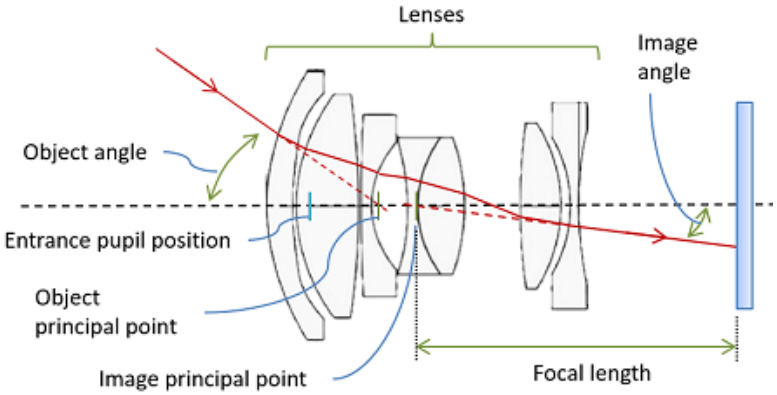


Figure 44. Object Angle (rad) vs. Image Angle (rad).



Distortion Curve File

The distortion curve (.OPTDistortion) file describes the object to image angular relationship and allows Speos to render the camera behavior.

Tip: The information needed to fill this file are usually provided by the supplier. You only need to format it.

Note: Simulation results generated with distortion curve file V2.0 (.xmp, .png, .hdr) are not rotated. Simulation results generated with distortion curve file V1.0 (.xmp, .png, .hdr) are rotated.

Line	Description	Example	Comments
1	Header	OPTIS - Optical distortion file v1.0	Version (v1.0) could be used for further file format update.
2	Comment	Created by x user with DistortionCurveGenerator, 02/12/2011 13:18:32	Users/suppliers comments. For information purposes, without any impact on simulation results.
3	Format	0	0 indicates that the distortion curve is given as a list of sample of object angle vs image angle. No other value will be supported in V1.0 version.
4	Number of samples	91	

DRAFT

Line	Description	Example		Comments
5	Object Angle vs. Image Angle	0	0	Angle values in radian.
		0.017453278	0.0273028387171337	
		0.034906556	0.0537954641261183	
		0.052359333	0.0795296582853432	
		0.069813111	0.1045521483695	
		0.087266389	0.128905211037813	
		0.104719667	0.152627193841727	
		0.122172944	0.175752965965063	
		...	...	
		...	...	
		0.261799167	0.342599269505361	
		0.279252444	0.361513076284991	
		0.296705722	0.3800535557331	

8.3.11.2.1.3. Speos Lens System

This page merely describes the data model and file format of the Speos Lens System.

The Speos Lens System allows you to describe a camera model that takes into account distortion asymmetry and variable origin (default origin is the center of the imager).

To describe such behaviors, several inputs need to be referenced in an OPTDistortion (*.txt) file.

Function Principle

The camera sensor (using the Speos Lens System model as an input to define the camera behavior) works in a reverse propagation principle. This means rays are emitted from the sensor to reach targets (object contained in the scene).

As rays are emitted from the sensor, the sensor itself is considered as the entry point of the Speos Lens System. Rays are then handled by the function and are propagated in the optomechanical system.

Each sample on the sensor is identified by its coordinates:

- For a 2.0 version file, the sensor only considers polar coordinates (r, theta)
- For a 2.1 version file, the sensor can use polar coordinates (r, theta) or cartesian coordinates (x, y).

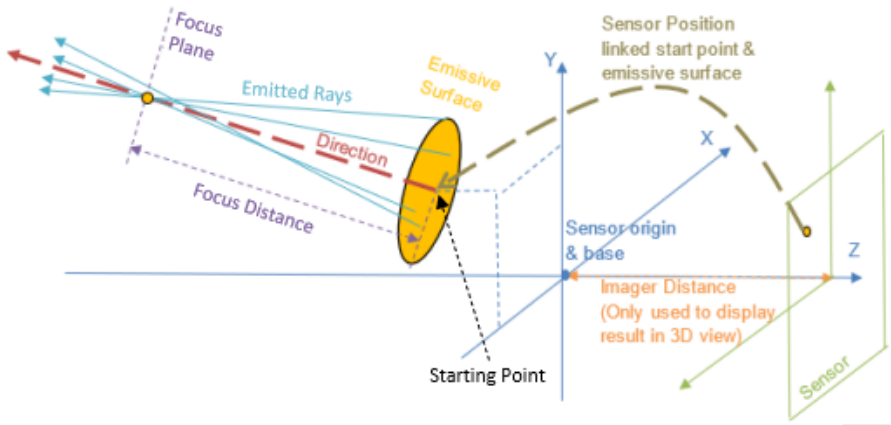
For each sample on the sensor, you need to specify:

- a ray direction
- a starting point
- an emissivity
- a focus distance
- an emissive surface description

The sampling described in the file is way inferior to the sensor's resolution. Only some pixels (points of the sensor) are described, the rest of the sensor's pixels are calculated by data interpolation.

DRAFT

Figure 45. Global Definition



2.0 Version File Format

2.0 Version Structure

The OPTDistortion file v2.0 defining the camera behavior can be divided into two parts:

- A Header containing:
  - information on the file (name, version etc.), the type of coordinate system used to define several parameters.
  - the sampling of the sensor according to polar coordinates (to a r and a theta ɵ).
- A "body" or data bloc containing the data corresponding to each sample on the sensor. Each line of data corresponds to one sensor point.

> **Note:** The Speos Lens System data model v2.0 works only with a ray direction defined as **Spherical** and a **Starting point** defined as **cartesian**.

Line	Description
Line 1	Header indicating filename and version
Line 2	Comment
Line 3	Ray direction coordinate system (spherical)
Line 4	Start point coordinate system (cartesian)
Line 5	Emissive surface type (disk)
Line 6	Number of radii r samples on sensor plane Distances from the optical center (integer value)
Line 7	List of radii r values The number of values should correspond to the value defined at line 6
Line 8	Number of theta angles on sensor plane This number is fixed to consider symmetrical lens systems.

DRAFT

Line	Description
Line 9	List of theta angle values The number of values should correspond to the value defined at line 8. These values are always the same to consider symmetrical lens systems.
Line 10	Blocks of data. Each block corresponds to one r sample, with the order defined at line 7. Each line of a block corresponds to one theta sample, with the order defined at line 9.

2.0 Version Format Example

In the following example, in the data block:

- the five first lines correspond to r=0mm.
- the five next lines correspond to r=4.65mm
- The first line corresponds to theta = 0deg, the second line corresponds to theta = 90deg, etc.
- Each line of the data block includes the values of the different parameters in the following order:

  "Ray direction theta" "Ray direction phi" "Starting point x" "Starting point y" "Starting point z" "Emissivity" "Focus Distance" "Divergence" "Emissive disk radius"

```
OPTIS - Light Transfer Function file v2.0
Comment
Spherical
Cartesian
Disk
2
0   4.65
5
0   90   180   270   360
0.000    0.000    0.000   0.000   0   1   10000   0   7.56
0.000    90.000   0.000   0.000   0   1   10000   0   7.56
0.000    180.000  0.000   0.000   0   1   10000   0   7.56
0.000    270.000  0.000   0.000   0   1   10000   0   7.56
0.000    360.000  0.000   0.000   0   1   10000   0   7.56
15       0.000    0.000   0.000   0   1   10000   0   7.56
15       90.000   0.000   0.000   0   1   10000   0   7.56
15       180.000  0.000   0.000   0   1   10000   0   7.56
15       270.000  0.000   0.000   0   1   10000   0   7.56
15       360.000  0.000   0.000   0   1   10000   0   7.56
```

2.1 Version File Format

2. Version Structure

The OPTDistortion file v2.1 defining the camera behavior can be divided into two parts:

- A Header containing:
  - information on the file (name, version etc.), the type of coordinate system used to define several parameters.
  - the sampling of the sensor according to polar coordinates (to a r and a theta ɵ).
- A "body" or data bloc containing the data corresponding to each sensor point. Each line of data corresponds to one sensor point.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Line	Description
Line 1	Header indicating filename and version
Line 2	Comment
Line 3	Sensor coordinate system (Polar or Cartesian) • Polar: equivalent to v2.0 format definition and describes symmetrical optical systems. • Cartesian: describes asymmetrical optical systems.
Line 4	Ray direction coordinate system (Spherical or Cartesian)
Line 5	Start point coordinate system (Spherical or Cartesian)
Line 6	Emissive surface type (Disk)
Line 7	• Number of radii r samples on sensor plane Distances from the optical center when the sensor coordinate system is Polar at Line 3 (integer value) • Number of x samples on sensor plane When the sensor coordinate system is Cartesian at Line 3 (integer value)
Line 8	List of radii r values (in case of Polar sensor coordinate system) or list of x values (in case of Cartesian sensor coordinate system) The number of values should correspond to the value defined at line 7
Line 9	• Number of theta samples on sensor plane, in case of Polar sensor coordinate system. This number is fixed to consider symmetrical lens systems. • Number of y samples on sensor plane, in case of Cartesian sensor coordinate system.
Line 10	List of theta angle values (in case of Polar sensor coordinate system) or list of y values (in case of Cartesian sensor coordinate system) The number of values should correspond to the value defined at line 9.
Line 11	Blocks of data. Each block corresponds to one r or x sample, with the order defined at line 8. Each line of a block corresponds to one theta / y sample, with the order defined at line 10.

2.1 Version Format Example

```
OPTIS - Light Transfer Function file v2.1
Comment
Cartesian
Cartesian
Cartesian
Disk
20
-10.0   -8.9474   -7.8947   -6.8421   -5.7895   -4.7368   -3.6842   -2.6316
 -1.5789   -0.5263   0.5263   1.5789   2.6316   3.6842   4.7368   5.7895
6.8421   7.8947   8.9474   10.0
25
-12.5   -11.4583   -10.4167   -9.375   -8.3333   -7.2917   -6.25   -5.2083
-4.1667   -3.125   -2.0833   -1.0417   0.0   1.0417   2.0833   3.125   4.1667
```

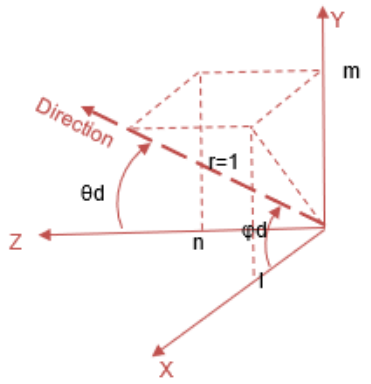
DRAFT

```
   5.2083   6.25   7.2917   8.3333   9.375   10.4167   11.4583   12.5
[DATA BLOCK]
```

Elements

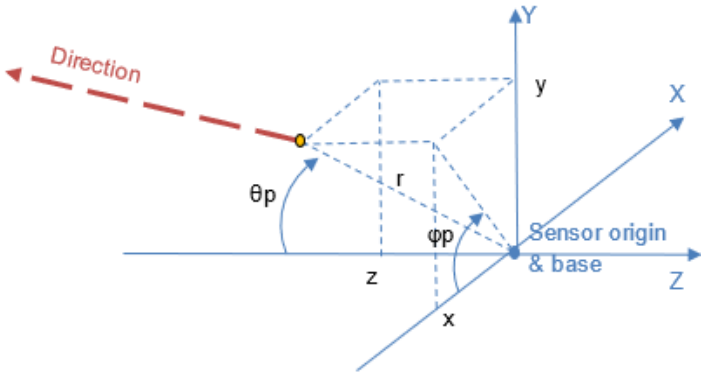
Ray Direction

- With a cartesian system, it is defined by three values (l, m, n)
- With a spherical coordinate system, it is defined by two values (ɵd, ϕd)



Starting Point

- With a cartesian system, it is defined by three values (x, y, z)
- With a spherical coordinate system, it is defined by three values (r, ɵp, ϕp)



Emissivity

Emissivity is used to model vignetting and is defined by a floating value between 0 and 1. It models the potential loss of energy of rays when going through the optical system.

- When the Emissivity is set to 0, the sample on the sensor is not used. That means the irradiance will be null. (for example in case of a fisheye sensor)
- When the Emissivity is set to 1, the sample on the sensor is used

> **Note:** The field of view calculated corresponds to the overlapping between the sensor and the emissive surface. areas with no overlap are not taken into account in the field of view.

DRAFT

Note: Emissivity is no longer used since the version 2022 R1 and is generally set to 1.

Focus Distance

Focus Distance is defined by a floating value:

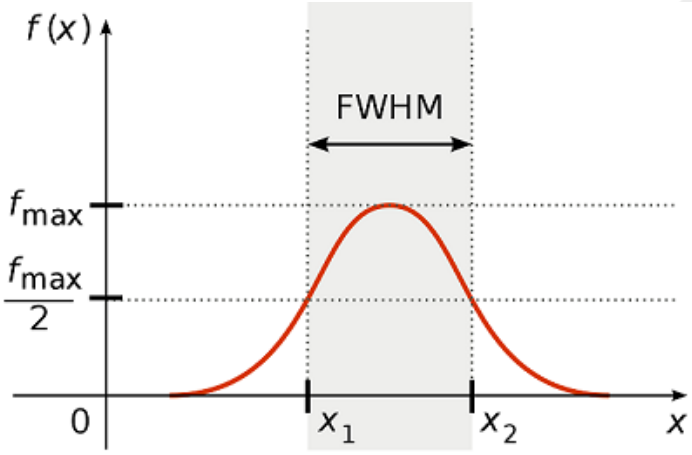
- Real focus (>0)
- Virtual focus (<0)

Divergence (d)

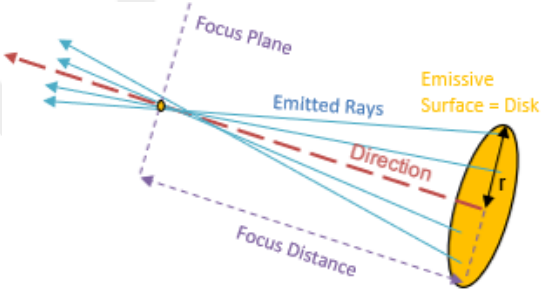
Divergence is defined by a positive floating value that allows you to specify a small statistical deviation for each ray.

With Divergence, you are not targeting an exact single point in the focus plane, but a solid angle around it. This allows you to consider the lens design quality as it is a simple way to model the Point Spread Function.

The statistical distribution follows a 2D gaussian. The Divergence parameter represents the Full Width Half Maximum (FWHM) of the gaussian.



Emissive Surface Description

Disk	Disk type is defined by (d,r) with r as radius and d defining the divergence (both are positive floating values). 

8.3.11.2.2. Understanding Camera Sensor Parameters

General

Mode

- The **Geometric** mode is a simplified version of the Camera Sensor definition parameters. The rendering properties are used by default during the mesh generation.

  The Geometric camera sensor can be used only in inverse simulations without sources and other sensors.

  When you enable the Geometric mode, the parameters relative to spectrum or spectral data are disabled.

- The **Photometric / Colorimetric** mode allows you to set every Camera Sensor parameters, including the photometric definition parameters.

> **Note:** In case of a photometric result generation, the International Commission on Illumination (CIE) defines the visible spectrum as follows: "There are no precise limits for the spectral range of visible radiation since they depend upon the amount of radiant flux reaching the retina and the responsivity of the observer. The lower limit is generally taken between 360 nm and 400 nm and the upper limit between 760 nm and 830 nm".

Layer

> **Note:** The Layer is available only in Photometric / Colorimetric Mode.

Layers allows you to store all photometric results in the same XMP layer or not:

- **None** includes the simulation's results in one layer.
- **Data Separated By Source** includes one layer per active source in the result.

Optics

According to the *.OPTDistortion file model used as input of the Camera, the **Optics** section provides you with different parameters.

Parameter	Description	V1	V2 & 2.1	Binary
Focal length (mm)	Distance between the center of the optical system and the focus. For more information, click here.	✓	✗	✗
Imager distance (mm)	The imager is located at the focal point. The Imager distance has no impact on the result.	✓	✗	✗
F-number	F-number represent the aperture of the front lens. F number has no impact on the result. For more information, click here.	✓	✗	✗

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Parameter	Description	V1	V2 & 2.1	Binary
Transmittance	The Transmittance file is available only in **Photometric / Colorimetric** Mode. Amount of light of the source that passes through the lens and reaches the sensor. The transmittance is expressed in a .spectrum file. For more information, click here.	✓	✓	✗
Horizontal field of view (deg)	Horizontal field of view calculated using the focal length, distortion file, width and height of the sensor.	✓	✓	✓
Vertical field of view (deg)	Vertical field of view calculated using the focal length, distortion file, width and height of the sensor.	✓	✓	✓
Distortion	Optical aberration that deforms and bend straight lines. The distortion is expressed in a .OPTDistortion file. For more information on the .OPTDistortion file, see Camera Models. For more information, click here.	✓	✓	✓

Sensor

Parameter	Description
Horizontal pixels	Defines the horizontal pixels number corresponding to the camera resolution.
Vertical pixels	Defines the vertical pixels number corresponding to the camera resolution.
Width	Defines the sensor's width.
Height	Defines the sensor's height.
Color mode	The Color mode is available only in **Photometric / Colorimetric** Mode. **Color**: simulation results are available in color according to the White Balance mode. **Monochrome:** simulation results are available in grey scale.

DRAFT

Parameter	Description
White balance mode	**Note:** The White balance mode is useful in case of a visible spectrum. The White balance mode will not work in the Infrared and the UV spectrum. The White balance mode is available only in **Photometric / Colorimetric** Mode. **None**: The spectral transmittance of the optical system and the spectral sensitivity for each channel are applied to the detected spectral image before the conversion in a three channel results. This method is referred to as the basic conversion. **Grey world**: The grey world assumption states that the content of the image is grey on average. This method converts spectral results in a three channel results with the basic conversion. Then it computes and applies coefficients to the red, green and blue images to make sure their averages are equal. **User white balance**: In addition to the basic treatment, it allows you to apply your own coefficients to the red, green, blue images. **Display primaries**: Spectral results are converted in a three channels result. Then a post-treatment is realized to take the distortion induced by the display devices into account. With this method, displayed results are similar to what the camera really gets.
Gamma correction	The Gamma correction is available only in **Photometric / Colorimetric** Mode. Compensation of the curve before the display on the screen. For more information, see **Monitor**.
PNG bits	The PNG bits is available only in **Photometric / Colorimetric** Mode. Choose between 8, 10, 12 and 16-bit.

Sensitivity

The Sensor sensitivity is available only in **Photometric / Colorimetric** Mode.

Sensor sensitivity allows you define the spectral sensitivity of the camera sensor according to the color mode selected.

Wavelength

The Wavelength is available only in **Photometric / Colorimetric** Mode.

Wavelength allows you to define the spectral range in which the inverse simulation propagates rays from the sensor.

- **Start** (nm) defines the minimum wavelength.
- **End** (nm) defines the maximum wavelength.
- **Sampling** defines the number of wavelength to be taken into account between the minimum and minimum wavelengths set.

Note: In case of a photometric result generation, the International Commission on Illumination (CIE) defines the visible spectrum as follows: "There are no precise limits for the spectral range of visible radiation since they depend upon the amount of radiant flux reaching the retina and the responsivity of the observer. The

DRAFT

lower limit is generally taken between 360 nm and 400 nm and the upper limit between 760 nm and 830 nm".

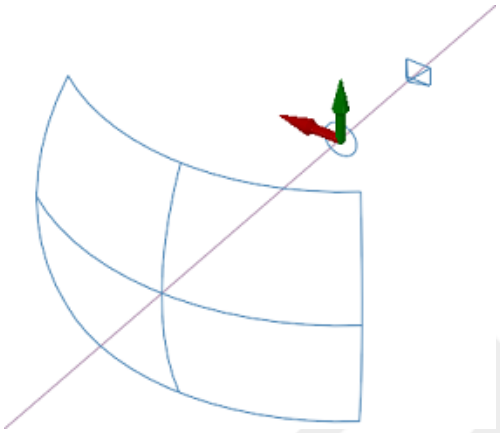
Visualization

Visualization allows you to define the elements (Camera field, Object field, Aperture) of the camera to display in the 3D view.

Visualization Radius changes the radius of the Object field of the camera.

Note: Visualization Radius has no impact on the result.

Figure 46. Display of the Camera field, Aperture and Object field.



8.3.11.2.3. Trajectory File

This page describes the format of trajectory files, used to define the position and orientations of a Speos Light Box, a LiDAR sensor, or a Camera sensor in time.

Description

Positions and orientations are expressed with the respect to a reference coordinate system, so the trajectory is relative to this coordinate system.

For instance, the same trajectory file can be used to describe a translation movement of a car as well as the LiDAR sensor placed on it.

Trajectory is described in a *.json file that contains each chronological sample:

DRAFT

Format Description	Example

• "Time" in second.
• "Origin" with "X", "Y", "Z" coordinates in mm, that corresponds to the relative position with the respect to reference coordinate system.
• "Direction_X" with "X", "Y", "Z" coordinates that corresponds to the relative X direction with the respect to reference coordinate system.
• "Direction_Y" with "X", "Y", "Z" coordinates that corresponds to the relative Y direction with the respect to reference coordinate system.
	<pre>"Trajectory": {
	{
	"Time": 0.0,
	"Origin": {
	"X": 0.0,
	"Y": 0.0,
	"Z": 0.0
	},
	"Direction_X": {
	"X": 1.0,
	"Y": 0.0,
	"Z": 0.0
	},
	"Direction_Y": {
	"X": 0.0,
	"Y": 1.0,
	"Z": 0.0
	}
	},
	"Time": 0.033333333333,
	"Origin": {
	"X": 0.0,
	"Y": -0.0,
	"Z": 900.0
	},
	"Direction_X": {
	"X": 1.0,
	"Y": 0.0,
	"Z": 0.0
	},
	...</pre>

Script Example

Trajectory file can be easily accessed (read or write) using dedicated scripting interfaces available in IronPython and Python.

> **Note:** Make sure to use the **3.9 version** of IronPython or Python language to write your scripts.

IronPython Example

```
import sys
import clr

sys.path += [R"C:\Program Files\ANSYS Inc\v2XX\Optical Products\SPEOS\Bin"]

clr.AddReferenceToFile("Optis.Core_net.dll")
clr.AddReferenceToFile("Optis.Data_net.dll")

# References to Optis.Core and Optis.Data
import Optis.Core as OptisCore
import Optis.Data as OptisData
```

```
try:
  firstData = OptisCore.DAxisSystemData()
  firstData.Time = 0.0
  firstData.Origin.Init(0, 0, 0)
  firstData.Direction_X.Init(1, 0, 0)
  firstData.Direction_Y.Init(0, 1, 0)

  lastData = OptisCore.DAxisSystemData()
  lastData.Time = 7.0
  lastData.Origin.Init(0, 0, 3000)
  lastData.Direction_X.Init(1, 0, 0)
  lastData.Direction_Y.Init(0, 1, 0)

  dmTrajectory = OptisCore.DAxisSystemTrajectory()
  dmTrajectory.Trajectory.Resize(2)
  dmTrajectory.Trajectory.Set(0, firstData)
  dmTrajectory.Trajectory.Set(1, lastData)

  strPathTrajectoryFile = OptisCore.String.From(R"C:\trajectory.json")
  pathTrajectoryFile = OptisCore.Path(strPathTrajectoryFile)

  cAxisSystemTrajectoryWriter = OptisData.CAxisSystemTrajectoryWriter()
  cAxisSystemTrajectoryWriter.Open(pathTrajectoryFile)
  cAxisSystemTrajectoryWriter.Write(dmTrajectory)

  cAxisSystemTrajectoryReader = OptisData.CAxisSystemTrajectoryReader()
  cAxisSystemTrajectoryReader.Open(pathTrajectoryFile)
  dmTrajectory = cAxisSystemTrajectoryReader.Read()

  cAxisSystemTrajectoryReader.Close()

  print "Done"

except:
  print "Exception raised"
```

Python Example

```
import sys

sys.path += [R" C:\Program Files\ANSYS Inc\v2XX\Optical Products\SPEOS\Bin"]

# References to Optis.Core and Optis.Data
import IllumineCore_pywrap as OptisCore
import IllumineData_pywrap as OptisData

try:
  firstData = OptisCore.DAxisSystemData()
  firstData.Time = 0.0
  firstData.Origin.Init(0, 0, 0)
  firstData.Direction_X.Init(1, 0, 0)
  firstData.Direction_Y.Init(0, 1, 0)

  lastData = OptisCore.DAxisSystemData()
  lastData.Time = 7.0
  lastData.Origin.Init(0, 0, 3000)
  lastData.Direction_X.Init(1, 0, 0)
```

```
  lastData.Direction_Y.Init(0, 1, 0)

  dmTrajectory = OptisCore.DAxisSystemTrajectory()
  dmTrajectory.Trajectory.Resize(2)
  dmTrajectory.Trajectory.Set(0, firstData)
  dmTrajectory.Trajectory.Set(1, lastData)

  strPathTrajectoryFile = OptisCore.String(R"C:\trajectory.json")
  pathTrajectoryFile = OptisCore.Path(strPathTrajectoryFile)

  cAxisSystemTrajectoryWriter = OptisData.CAxisSystemTrajectoryWriter()
  cAxisSystemTrajectoryWriter.Open(pathTrajectoryFile)
  cAxisSystemTrajectoryWriter.Write(dmTrajectory)

  cAxisSystemTrajectoryWriter.Close()

  print("Done")

except:
  print("Exception raised")
```

8.3.11.2.4. Trajectory Script Example

The following page gives you a full script example to create a trajectory in Speos.

> **Note:** Make sure to use the **3.9 version** of IronPython or Python language to write your scripts.

```
# © 2012-2021 ANSYS, Inc. All rights reserved. Unauthorized use, distribution,
 or duplication is
prohibited.

# THIS ANSYS SOFTWARE PRODUCT AND PROGRAM DOCUMENTATION INCLUDE TRADE SECRETS
AND ARE CONFIDENTIAL AND PROPRIETARY PRODUCTS OF ANSYS, INC., ITS SUBSIDIARIES,
 OR
LICENSORS.  The software products and documentation are furnished by ANSYS,
Inc., its subsidiaries,
or affiliates under a software license agreement that contains provisions
concerning non-disclosure,
copying, length and nature of use, compliance with exporting laws, warranties,
 disclaimers,
limitations of liability, and remedies, and other provisions.  The software
products and
documentation may be used, disclosed, transferred, or copied only in accordance
 with the terms and
conditions of that software license agreement.

import sys
import clr
import os
from os import path

sys.path += [R"C:\Program Files\ANSYS Inc\v211\Optical Products\SPEOS\Bin"]

clr.AddReferenceToFile("Optis.Core_net.dll")
clr.AddReferenceToFile("Optis.Data_net.dll")
```

```
# References to Optis.Core and Optis.Data
import Optis.Core as OptisCore
import Optis.Data as OptisData

def GetAxisSystemData(iTime, iFrame):
    dAxisSystemData = OptisCore.DAxisSystemData()
    dAxisSystemData.Time = iTime
    dAxisSystemData.Origin.Init(1000 * iFrame.Origin.X,
                                1000 * iFrame.Origin.Y,
                                1000 * iFrame.Origin.Z)

    tmpVector = OptisCore.Vector3_double()
    tmpVector.Init(iFrame.DirX.X,
                   iFrame.DirX.Y,
                   iFrame.DirX.Z)
    tmpVector.Normalize()

    dAxisSystemData.Direction_X.Init(tmpVector.Get(0),
                                     tmpVector.Get(1),
                                     tmpVector.Get(2))

    tmpVector.Init(iFrame.DirY.X,
                   iFrame.DirY.Y,
                   iFrame.DirY.Z)
    tmpVector.Normalize()

    dAxisSystemData.Direction_Y.Init(tmpVector.Get(0),
                                     tmpVector.Get(1),
                                     tmpVector.Get(2))

    return dAxisSystemData

# Working directory
workingDirectory = path.dirname(GetRootPart().Document.Path.ToString())

# SpaceClaim InputHelper
ihTrajectoryName = InputHelper.CreateTextBox("Trajectory.1", 'Trajectory name:
 ', 'Enter the name of the trajectory')
ihFrameFrequency = InputHelper.CreateTextBox(30, 'Timeline frame rate:', 'Set
 timeline frame rate (s-1)', ValueType.PositiveInteger)
ihReverseDirection = InputHelper.CreateCheckBox(False, "Reverse direction",
"Reverse direction on trajectory")
ihObjectSpeed = InputHelper.CreateTextBox(50, 'Object speed:', 'Set the moving
 object speed (km/h)', ValueType.PositiveDouble)

ihCoordinateSystem = InputHelper.CreateSelection("Reference coordinate system",
 SelectionFilter.CoordinateSystem, 1, True)
ihTrajectory = InputHelper.CreateSelection("Trajectory curve",
SelectionFilter.Curve, 1, True)

InputHelper.PauseAndGetInput('Trajectory parameters', ihTrajectoryName,
ihFrameFrequency, ihObjectSpeed, ihReverseDirection, ihCoordinateSystem,
ihTrajectory)

# Animation frame rate (s-1)
frameFrequency = float(ihFrameFrequency.Value)

# Object speed (km/h)
objectSpeed = float(ihObjectSpeed.Value)
```

```
# Trajectory file
trajectoryName = str(ihTrajectoryName.Value)
trajectoryAcquisitionFile = workingDirectory + "\\" + trajectoryName

# Reference coordinate system
coordSysSelection = ihCoordinateSystem.Value
trajectoryCoordSys = coordSysSelection.Items[0]

# Trajectory curve
trajCurveSelection = ihTrajectory.Value
trajectoryCurve = trajCurveSelection.Items[0]

# Reversed trajectory (Boolean)
isReversedTrajectory = ihReverseDirection.Value

# Acquisition of positions
def GetPositionOrientation(i_CoordSys, i_ReferenceCoordSys):
    # change base of current position
    newMatric = Matrix.CreateMapping(i_ReferenceCoordSys.Frame)

    currentPosition = newMatric.Inverse  * i_CoordSys.Frame.Origin
    currentPosition = Point.Create(round(currentPosition.X, 6),
round(currentPosition.Y, 6), round(currentPosition.Z, 6))

    # change base of current iDirection
    iDirectionX = newMatric.Inverse  * i_CoordSys.Frame.DirX

    # change base of current iDirection
    jDirectionX = newMatric.Inverse  *i_CoordSys.Frame.DirY

    # Return new frame
    return Frame.Create(currentPosition, iDirectionX, jDirectionX)

def MoveAlongTrajectory(i_trajectoryCoordSys, i_trajectoryCurve,
i_isReversedTrajectory, i_frameFrequency, i_objectSpeed):
    selectedCoordSys = Selection.Create(i_trajectoryCoordSys)
    newselectedCoordSys = i_trajectoryCoordSys

    pathLength = i_trajectoryCurve.Shape.Length
    selectedCurve = Selection.Create(i_trajectoryCurve)

    # Convert speed in m/s
    convObjectSpeed = float(i_objectSpeed * 1000 / 3600)

    currentPosition = 0.0
    timeStamp = 0.0
    positionTable = []
    timeTable = []

    while currentPosition < 1:
        options = MoveOptions()

        if currentPosition == 0:
            options.Copy = True
        else:
            options.Copy = False

        if i_isReversedTrajectory:
            result = Move.AlongTrajectory(selectedCoordSys, selectedCurve,
1-currentPosition, options)
```

```
            if currentPosition == 0:
              newselectedCoordSys = result.GetCreated[ICoordinateSystem]()[0]

                selectedCoordSys = Selection.Create(newselectedCoordSys)
                if newselectedCoordSys.Frame.Origin !=
i_trajectoryCoordSys.Frame.Origin:
                    options.Copy = False
                    result = Move.AlongTrajectory(selectedCoordSys,
selectedCurve, currentPosition, options)
        else:
            result = Move.AlongTrajectory(selectedCoordSys, selectedCurve,
currentPosition, options)

            if currentPosition == 0:
              newselectedCoordSys = result.GetCreated[ICoordinateSystem]()[0]

                selectedCoordSys = Selection.Create(newselectedCoordSys)
                if newselectedCoordSys.Frame.Origin !=
i_trajectoryCoordSys.Frame.Origin:
                    options.Copy = False
                    result = Move.AlongTrajectory(selectedCoordSys,
selectedCurve, currentPosition, options)

        if(result):
            movedFrame = GetPositionOrientation(newselectedCoordSys,
i_trajectoryCoordSys)
            positionTable.append(movedFrame)
            timeTable.append(timeStamp)

        currentPosition += (convObjectSpeed / i_frameFrequency)/pathLength
        timeStamp +=  1/float(i_frameFrequency)

    result = Delete.Execute(selectedCoordSys)

    return timeTable, positionTable

# Length of path (m)
pathLength = trajectoryCurve.Shape.Length

# Get time stamps and relative coordinate systems
timeTable, positionTable = MoveAlongTrajectory(trajectoryCoordSys,
trajectoryCurve, isReversedTrajectory,  frameFrequency, objectSpeed)

dAxisSystemData_Table = []
for time in timeTable:
    timeIndex = timeTable.index(time)
    fFrame = positionTable[timeIndex]

    dAxisSystemData = GetAxisSystemData(time, fFrame)
    dAxisSystemData_Table.append(dAxisSystemData)

if len(dAxisSystemData_Table) > 0:
    dmTrajectory = OptisCore.DAxisSystemTrajectory()
    dmTrajectory.Trajectory.Resize(len(dAxisSystemData_Table))

    for dAxisSystemData in dAxisSystemData_Table:

dmTrajectory.Trajectory.Set(dAxisSystemData_Table.index(dAxisSystemData),
dAxisSystemData)

    pathTrajectoryFile = str(workingDirectory + "\\" +
```

```
str(ihTrajectoryName.Value) + ".json")
    strPathTrajectoryFile = OptisCore.String.From(pathTrajectoryFile)
    pathTrajectoryFile = OptisCore.Path(strPathTrajectoryFile)

    cAxisSystemTrajectoryWriter = OptisData.CAxisSystemTrajectoryWriter()
    cAxisSystemTrajectoryWriter.Open(pathTrajectoryFile)
    cAxisSystemTrajectoryWriter.Write(dmTrajectory)

    cAxisSystemTrajectoryWriter.Close()
```

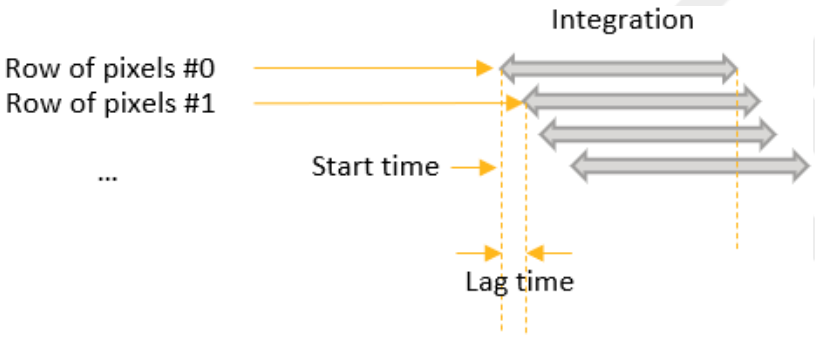
8.3.11.2.5. Acquisition Parameters

The Acquisition parameters are used to define how the Camera sensor captures frames in an inverse simulation.

Acquisition Parameters

Integration corresponds to the time needed to get the data acquired by one row of pixels.

Lag time corresponds to the time difference between two rows of pixels to start the integration.



Camera Effect

The acquisition parameters influence the effects captured by the Camera sensor during the inverse simulation.

When the Lag time is null, the effect produced is the motion blur. When some Lag time is defined, the effect produced is the rolling shutter.

Reference • Inverse Simulation Timeline: False • Integration: 0ms • Lag time: 0ns	
Motion Blur • Inverse Simulation Timeline: True • Integration: 10ms • Lag time: 0ns	

Rolling Shutter • Inverse Simulation Timeline: True • Integration: 1ms • Lag time: 92592ns	

8.3.11.3. Camera Sensor Creation

Two modes are available to create the Camera Sensor. A simplified mode (**Geometric** mode) and an advanced mode (**Photometric/Colorimetric** mode).

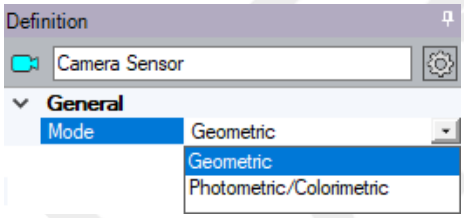
8.3.11.3.1. Creating a Camera Sensor in Geometric Mode

This page shows how to create a Camera Sensor using the Geometric mode. The Geometric mode is a simplified mode that allows you to define a camera sensor without having to set every parameter of the sensor.

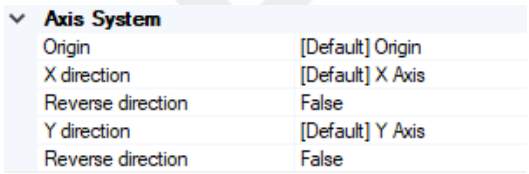
> **Note:** The geometric camera sensor can only be used in inverse simulation without sources or other sensors. In Geometric mode, all parameters relative to spectrum or spectral data are disabled.

To create a Camera Sensor using the Geometric Mode:

1. From the **Light Simulation** tab, click **System** > **Camera** .
2. From the **Mode** drop-down list, select **Geometric** to enable the simplified definition.



3. Define the **Axis System** of the camera sensor in the 3D view by clicking to select an origin, X to select a line and Y to select a line or click and select a coordinate system to autofill the Axis System.



> **Note:** Make sure the sensor is not tangent to a geometry.

> **Note:** Depending on which camera model is described in the .OPTDistortion input file, the origin of the sensor is different.
>
> - If the .OPTDistortion file is based on the Basic Distortion Curve model (v1 version), the origin corresponds to the Entrance Pupil Point (EPP) of the camera.
> - If the .OPTDistortion file is based on the Speos Lens System model (v2 version), the origin corresponds to the center of the sensor.

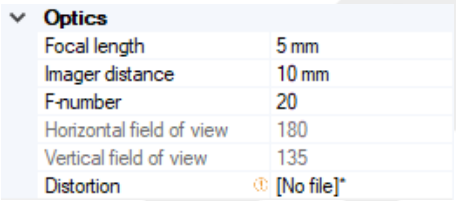
> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

If you need to adjust the axes orientation, use **Reverse direction** on one or both axes.

4. In the **Optics** section, define all the parameters of the camera:

> **Note:** Only the *.OPTDistortion V1 model is compatible with the geometric mode.

> **Note:** Horizontal/vertical fields of view set to 0 are not supported and may generate incorrect result. If the values are set to 0, refresh the Camera sensor feature.



a) In **Focal length**, adjust the distance between the center of the optical system and the sensor (in mm).
b) In **Imager distance**, you can adjust the distance between the aperture and the sensor.

> **Note: Imager distance** does not affect the results. Changing it is only used for visualization purposes and does not represent the real sensor.

c) In **F-number**, type the size of the aperture of the camera's front lens.

> **Note:** The smaller the number, the larger the aperture.
>
> The irradiance is calculated from the radiance by using an acceptance cone for the light (the cone base is the pupil).
>
> More details about the **F-number** can be found here.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

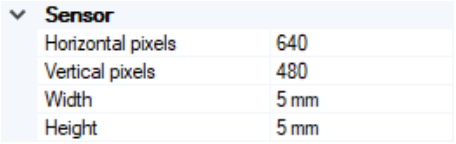
Published: 2023-10-23T14:08:14.04-04:00

d) In **Distortion**, click **Browse** to load an .OPTDistortion file V1.

The *.OPTDistortion file is a file that contains information on the camera and is used to introduce/replicate the behavior of the camera lens.

A preview of the camera sensor system appears in the 3D view.

5. In the **Sensor** section, define the size and resolution of the sensor:



a) Define the number of horizontal and vertical pixels corresponding to the camera resolution.

> **Note:** Speos supports a maximum resolution of 23170 * 23170 pixels.

b) Define the sensor's height and width.

6. If you want to adjust the preview of the sensor, click **Optional or advanced settings**:



a) Activate or deactivate the preview of certain parts of the system by setting them to **True**/**False**.

b) Adjust the **Visualization radius**.

The Camera Sensor is created and visible in Speos tree and in the 3D view.

Related concepts

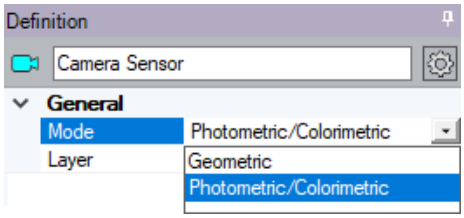
Camera Sensor Parameters on page 259
This section allows you to better apprehend the camera sensor definition as it describes key settings of the sensor.

8.3.11.3.2. Creating a Camera Sensor in Photometric/Colorimetric Mode

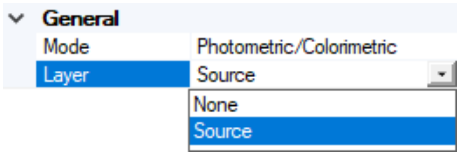
This page shows how to create a Camera Sensor using the Photometric/Colorimetric mode. This mode allows you to set every parameter of the camera sensor including spectral definition.

To create a Camera Sensor using the Photometric/Colorimetric Mode:

1. From the **Light Simulation** tab, click **System** > **Camera**.
2. From the **Mode** drop-down list, select **Photometric/Colorimetric** to enable the advanced definition.



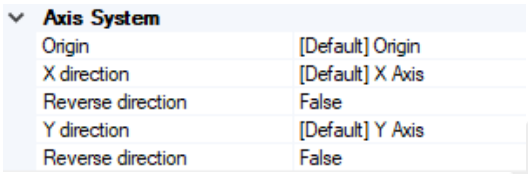
3. From the **Layer** drop-down list:



- Select **None** to get the simulation's results in one layer.
- Select **Source** if you have created more than one source and want to include one layer per active source in the result.

> **Tip:** You can change the source's power or spectrum with the Virtual Lighting Controller in the **Virtual Photometric Lab** or in the **Virtual Human Vision Lab**.

4. Define the **Axis System** of the camera sensor in the 3D view by clicking to select an origin, X to select a line and Y to select a line or click  and select a coordinate system to autofill the Axis System.



> **Note:** Make sure the sensor is not tangent to a geometry.

> **Note:** Depending on which camera model is described in the .OPTDistortion input file, the origin of the sensor is different.
>
> - If the .OPTDistortion file is based on the Basic Distortion Curve model (v1 version), the origin corresponds to the Entrance Pupil Point (EPP) of the camera.
> - If the .OPTDistortion file is based on the Speos Lens System model (v2 version), the origin corresponds to the center of the sensor.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

If you need to adjust the axes orientation, use **Reverse direction** on one or both axes.

5. If you want to create a dynamic Camera sensor, in the **Trajectory file** field, click **Browse** to load a trajectory file (.json).

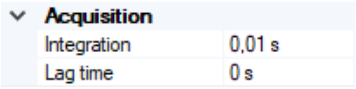
The trajectory describes the different positions and orientations of features in time. The Trajectory file is used to simulate dynamic objects.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

> **Note:** For more information on trajectory files, refer to Trajectory File on page 273.

When a trajectory file is assigned to a Camera sensor and the feature is edited, the trajectory is displayed in the 3D view.

6. If you want to create a dynamic Camera sensor, in the **Acquisition** section, define:



- the **Integration** time needed to get the data acquired by one row of pixels.
- the **Lag time**, if you want to create a rolling shutter effect in the result.

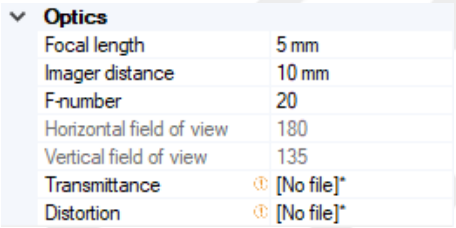
> **Note:** For more information on the acquisition parameters, refer to Acquisition Parameters.

7. In the **Optics** section, define all the parameters of the camera:

> **Note:** According to the *.OPTDistortion file model used as input of the Camera, the **Optics** section provides you with different parameters.

For more information on which parameter is available, refer to Understanding Camera Sensor Parameters.

> **Note:** Horizontal/vertical fields of view set to 0 are not supported and may generate incorrect result. If the values are set to 0, refresh the Camera sensor feature.



a) In **Focal length**, adjust the distance between the center of the optical system and the sensor (in mm).
b) In **Imager distance**, you can adjust the distance between the aperture and the sensor.

> **Note: Imager distance** does not affect the results. Changing it is only used for visualization purposes and does not represent the real sensor.

c) In **F-number**, type the size of the aperture of the camera's front lens.

> **Note:** The smaller the number, the larger the aperture.
>
> The irradiance is calculated from the radiance by using an acceptance cone for the light (the cone base is the pupil).

> More details about the **F-number** can be found here.

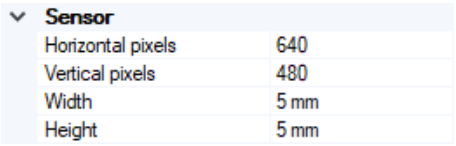
d) In **Transmittance**, click **Browse** to load a .spectrum file. The spectrum file defines the amount of light that passes through the lens to reach the sensor

e) In **Distortion**, click **Browse** to load an .OPTDistortion file.

The *.OPTDistortion file is a file that contains information on the camera and is used to introduce/replicate the behavior of the camera lens.

A preview of the camera sensor system appears in the 3D view.

8. Define the sensor's size and resolution:

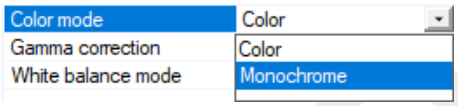


a) Define the number of horizontal and vertical pixels corresponding to the camera resolution.

> **Note:** Speos supports a maximum resolution of 23170 * 23170 pixels.

b) Define the sensor's height and width.

9. From the **Color mode** drop-down list, define the sensor's color management:



- Select **Monochrome** to define a sensor with one channel.

  The sensitivity section is displayed and allows you to define a spectrum file for the sensor's channel.

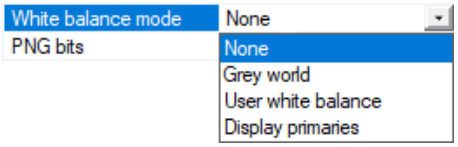
- Select **Color** to define the sensor as a RGBcamera.

  The sensitivity section is displayed and allows you to define a spectrum file for each of the three sensing channels.

> **Note:** With the **Color** mode, the simulations results are available in color according to the White Balance Mode.

10. If needed, adjust the Gamma correction.

11. From the **White balance mode** drop-down list, choose which correction to apply to get true whites:



- Select **None** to apply no correction and realize a basic conversion from spectral results to RGB picture.
- Select **Grey world** to apply a coefficient on each channel to get the same average for the three channels.
- Select **User white balance** to use the grey world method and manually type the coefficients to apply.

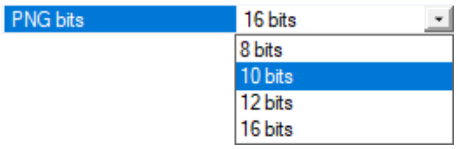
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

In the **Sensor white balance** section, adjust the coefficients for Red, Green and Blue colors.

- Select **Display primaries** to display the results and the colors as they are perceived by the camera.

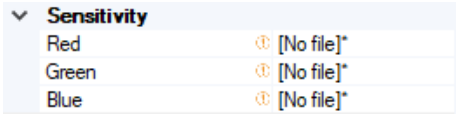
  In the **Sensor white balance** section, click **Browse** to load a .spectrum file for each color.

12. In **PNG bits**, select the number of bits used to encode a pixel.

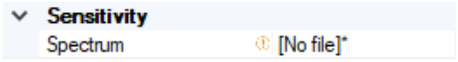


13. Adjust the sensor sensitivity:

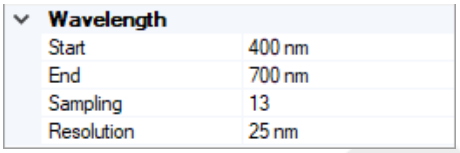
- If you selected the **Color** mode, import a .spectrum files per color.



- If you selected the **Monochrome** mode, import one .spectrum file.



14. In Wavelength, define the spectral range of the sensor:



- Edit the **Start** (minimum wavelength) and **End** (maximum wavelength) values to determine the wavelength range to be considered by the sensor.
- If needed, in **Sampling**, adjust the number of wavelengths to be computed during simulation.

  The Resolution is automatically computed according to the sampling and wavelength start and end values.

15. If you want to adjust the preview of the sensor, click **Optional or advanced settings** :



a) Activate or deactivate the preview of certain parts of the system by setting them to **True**/**False**.

   Display pupil is available only for the model V1.

   Display object field is not supported in case of a binary distortion file.

b) Adjust the **Visualization radius**.

The Camera Sensor is created and visible in Speos tree and in the 3D view.

Related concepts

Camera Sensor Parameters on page 259
This section allows you to better apprehend the camera sensor definition as it describes key settings of the sensor.

DRAFT

8.3.12. Physical Camera

The following section helps you understand the goal of a Physical Camera and how to create one.

> **Note:** The Physical Camera is in BETA mode for the current release.

8.3.12.1. Physical Camera Overview

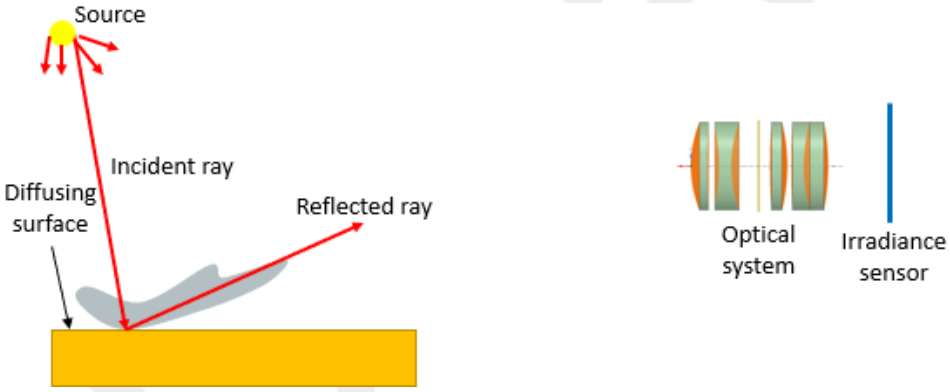
The Physical Camera allows you to create a lens system with its own optimized propagation engine, to be included in a bigger system, which improves the convergence when propagating rays and considers ghost images.

> **Note:** The Physical Camera is in BETA mode for the current release.

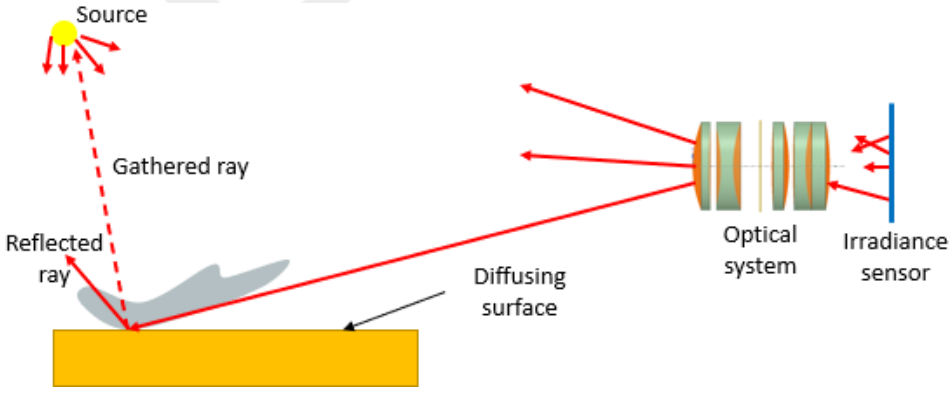
Context

Simulating lens systems in Speos is a real challenge due to the complexity of the lens geometries, and can be achieved using different ways:

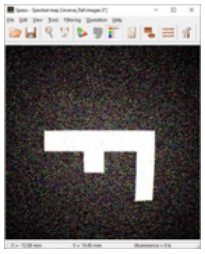
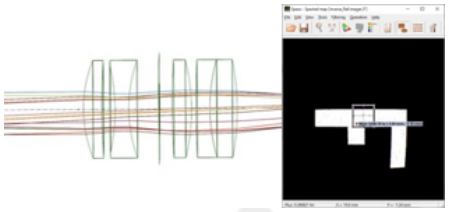
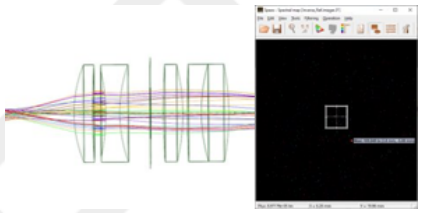
- A Direct simulation is not practical. The rays propagates from the light source to the scene, then trough the lens system to the Irradiance sensor. However, the probability for rays to reach the aperture of the lens system is very low.



- An Inverse simulation will allow you to simulate the complete lens system. However, it takes many passes (rays per pixel) to solve most of the ghost images. In most cases, it is not possible to solve the low contribution ones.



- A Simulation including a Camera Sensor using a distortion file (Speos Lens System). The Speos Lens System is a ROM that describes the light transfer function inside a lens system. This format is generated by OpticStudio and used by Speos to speed up acquisitions of the camera sensor (as opposed to the complete lens system). However, the Speos Lens System only describes the propagation of chief rays and does not take into account ghost images.

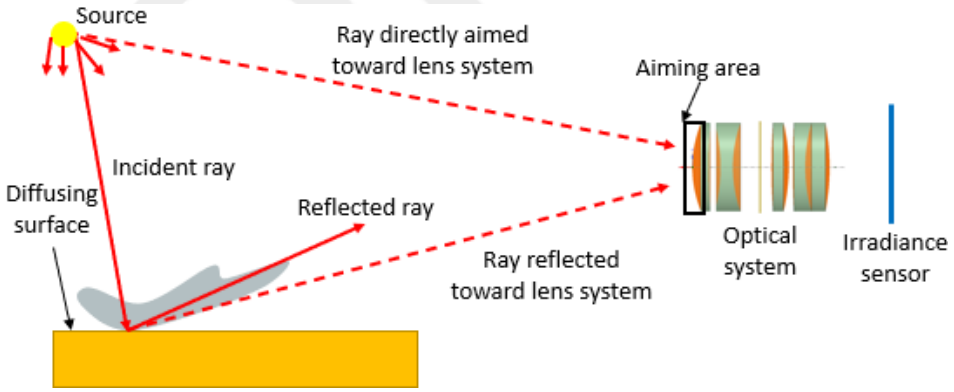
<table>
<tr>
<td rowspan="3"> 
Inverse Simulation result (20 passes - 3 min).
Too noisy to be used.</td>
<td> 
Main sequence converges fast, contributing to nearly 90% of the result.</td>
</tr>
<tr>
<td> 
Secondary sequences take time to converge.
This sequence contributes to nearly 1% of the result.</td>
</tr>
</table>

Simulations using real lens geometries, compared with camera sensor based on distortion file (Speos Lens System), take more time, or more rays to achieve the same level of signal, but Speos Lens System does not take into account ghost images as it only considers the one sequence.

Physical Camera

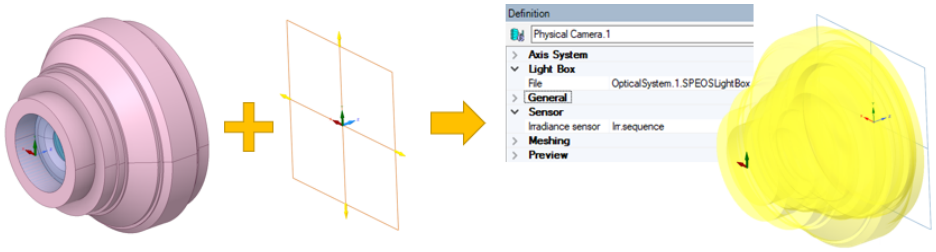
The goal is to use a specific propagation engine to simulate the lens system part of an optical system that will improve the convergence when propagating rays and consider ghost images.

Figure 47. Direct Simulation using the Aiming Area Principle through the Physical Camera



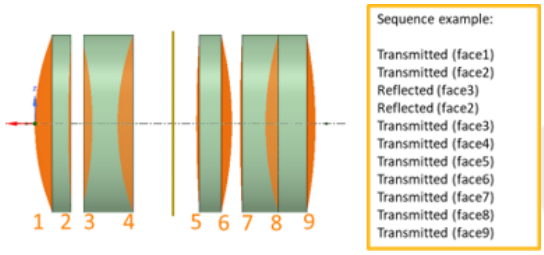
Note: The Aiming Area corresponds to the first face of the lens system surrounded by a bounding box. For more information, refer to Aiming Area.

In this context, the Physical Camera allows you to define a real lens system through a Speos Light Box combined with an Irradiance Sensor that will use the optimized propagation engine specific to the physical camera when simulated.



Therefore, in simulation, two propagation engines will be used according to the part of the system:

- The Speos standard propagation engine will be used for the scene
- The optimized propagation engine will be used for physical camera (lens system): Optimized propagation engine will rely on knowing which surface is in front of one impact or behind, that means on predefined sequences. Rays are not propagated anymore using the Monte Carlo algorithm, on the contrary, rays strictly follow sequences of interaction. For each impact along the sequences, Fresnel specular reflection or transmission are used to reduce the power of the ray according to the type of interaction.



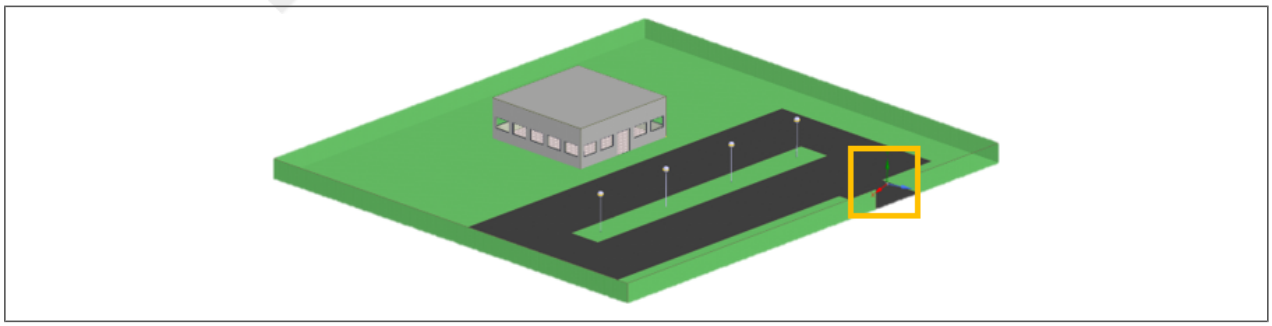
Physical Camera in Simulation

During an Inverse simulation, rays start from each pixel of the Irradiance sensor and are propagated through the Physical Camera sensor, eventually using the sequence file. Once rays quit the sensor, they continue their optical path towards the sources, considering the geometries of the scene.

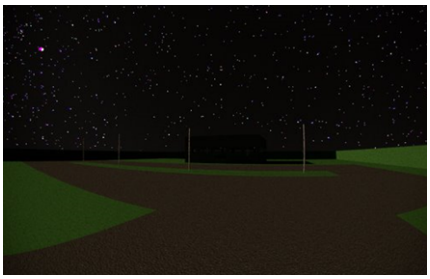
Therefore, splitting the simulation in two part is more efficient:

- For individual or "non ambient" light sources, Direct simulation should be preferred to get ghost effects for instance.
- For ambient light sources, Inverse simulation should be preferred to get the ambient contribution (without Sun) that does not introduce ghost.

> **Note:** Since each pass of an inverse simulation corresponds to 1 ray per pixel, per sequence, per wavelength, it is strongly recommended to focus on the main optical path to get the ambient contribution.



DRAFT

	
Inverse simulation with 1 sequence night sky ambient source, max illuminance at 0,001lx	Direct simulation with 100 sequences 4 luminaires, max illuminance at 1lx

Generic Workflow

1. Create a sequence file.

   The goal of creating a sequence file is to provide optical paths to be followed by rays to the Physical Camera sensor. This sequence file will then be used as input of the Physical Camera sensor to solve optical paths.

2. Once the sequence file is created, add it to the Physical Camera sensor.
3. Modify the Irradiance sensor according to the data separation you want to analyze.

   If you set the Irradiance sensor with the data separation by **Sequence**, another *.OPTSequence file will be generated which will consider only the **Number of sequences** (X) set in the Physical Camera. That means, you will only have the X first sequences in the Sequence Detection tool during a Stray Light analysis.

4. Run the simulation.

   > **Note:** When an *.OPTSequence file is used in a Physical Camera, every generated simulation result filename (*.xmp, *.lpf, etc.) is suffixed with "_sequential".

   Make sure the simulation type is the same (Direct or Inverse) as the simulation type used to generate the *.OPTSequence file.

   Make sure the **Dispersion** is activated.

8.3.12.2. Understanding the Physical Camera Parameters

The following page helps you understanding how to define and use the different components of a Physical Camera sensor.

> **Note:** The Physical Camera is in BETA mode for the current release.

Speos Light Box

The Speos Light Box contains all the geometries related to the lens system (lenses, opto-mechanics, etc.) of the camera to be simulated.

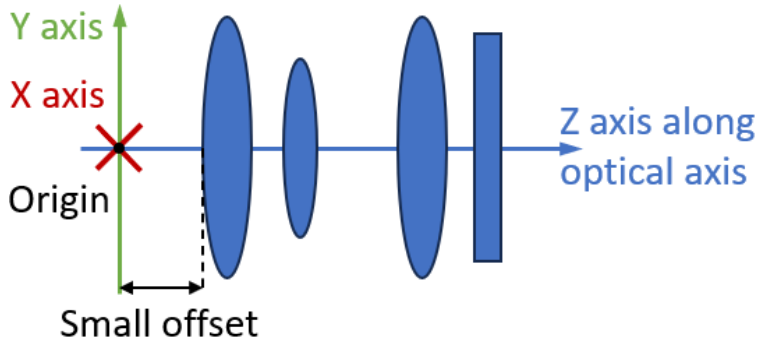
The Light Box must follow three important rules to be correctly used by the Physical Camera sensor:

DRAFT

- The axis system of the Light Box must be defined so the optical axis of the lens system follows the Z axis.

  The axis system is defined during the export of the Light Box.

- The origin of the axis system must be located with a small offset (around 10 times the Geometrical Distance Tolerance from the first face geometry).



- The Light Box cannot include:
  - 3D textures
  - geometries with the optical properties: fluorescent materials, materials with volumic scattering properties, grating surface properties, doe/thin lens surface properties.

Lens geometries imported from OpticStudio are compatible.

> **Warning:** A black-boxed Light Box is not compatible with the Physical Camera sensor.

Sequence File

The *.OPTSequence file corresponds to the file that determines the different paths available that rays can follow.

In the Physical Camera sensor, you have the possibility to add or not a Sequence file.

> **Important:** The Physical Camera sensor is compatible with the Direct and Inverse simulations. However, the *.OPTSequence file depends on the simulation type. An *OPTSequence file generated with a Direct simulation can only be used as input in a Direct simulation. The same rule goes for the Inverse simulation.

Without Sequence File

It is possible not to add a sequence file. The Physical Camera sensor will only benefit from the Aiming Area. The optimized propagation engine makes sure that rays launched aim to this area. Once rays interfered with the aiming area the propagation is done following the Monte Carlo algorithm.

When no sequence file is used, you can only set the **Layer** separation type to **Sequence** in the Irradiance sensor.

> **Note:** Not adding a sequence file to a Physical Camera sensor allows you to generate the future sequence file to be added to the Physical Camera sensor afterwards.

For more details on how to create this sequence file, refer to Creating an *.OPTSequene File for a Physical Camera Sensor.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

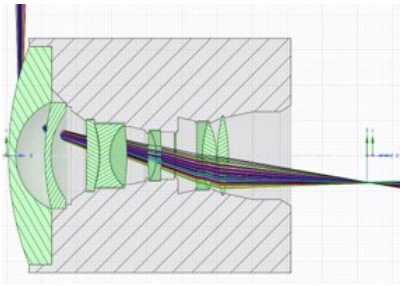
With Sequence File

To get the most out of the Physical Camera sensor simulation performance, you may need to add a Sequence file previously generated. When a sequence file is added to a Physical Camera, rays aim to the Aiming Area, then propagate following the sequences saved into the sequence file (and not following the Monte Carlo algorithm).

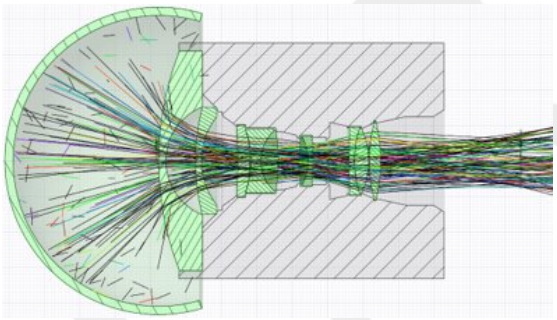
> **Important:** When inputting an *.OPTSequence file, the Physical Camera only keeps the sequences that enter the lens system from the first face of the system and exit the lens system from the last face of the system without exiting the lens system between the first and last faces.
>
> For a better readability, in the Sequence Detection tool, the interactions before the lens system are not displayed.

The number of sequences followed is determined using the **Number of sequences** parameter.

- You can create a sequence file for a one-source configuration. However, the drawback of generating a sequence file based on one source is that it highly depends on the light source and its position.



- You can create a universal sequence file covering as many optical paths as possible. This way, you do not depend on a light source and its position, on the contrary, you can place light sources at different positions.



For more details on how to create this sequence file, refer to Creating an *.OPTSequene File for a Physical Camera Sensor.

> **Note:** When an *.OPTSequence file is used in a Physical Camera, every generated simulation result filename (*.xmp, *.lpf, etc.) is suffixed with "_sequential".

Number of Sequences

Number of Sequences determines the number of sequences to play and provide into the *.xmp result file.

Number of Sequences is useful only if you add a sequence file as input of the Physical Camera.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Irradiance Sensor

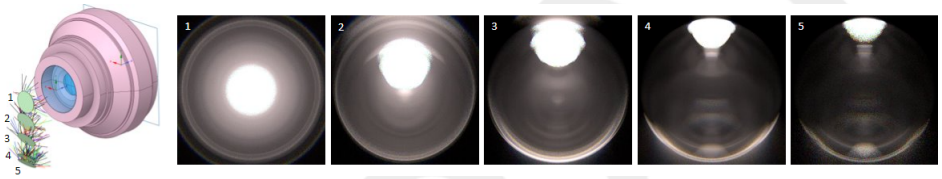
The Irradiance sensor is a mandatory part of the Physical Camera.

The Irradiance sensor must follow three important rules to be correctly used by the Physical Camera sensor:

- The Irradiance sensor must be positioned in the scene considering the lens system defined in the Light Box, which is usually placed in front of it.
- The Integration type must be Planar.
- Depending on whether you add a sequence file or not, the Layer type choice differs:
  - If you add no sequence file, you can only set the **Layer** separation type to **Sequence** in the Irradiance sensor.
  - If you add a sequence file, you can set any **Layer** separation type in the Irradiance sensor.

> **Important:** If you add a sequence file and you set the Irradiance sensor with the Layer separation type by **Sequence**, another *.OPTSequence file will be generated which will consider only the Number of sequences (X) set in the Physical Camera. That means, you will only have the X first sequences in the Sequence Detection tool during a Stray Light analysis.

Figure 48. Example: Physical Camera sensor with Sequence file + Irradiance Sensor with data separated by source



8.3.12.3. Aiming Area

The following page helps you understand the purpose of the Aiming Area in a Physical Camera.

The Aiming Area is an important notion of the Physical Camera. The Aiming Area is not a parameter to be set but you need to understand its impact in order to correctly define your system and simulations.

Description

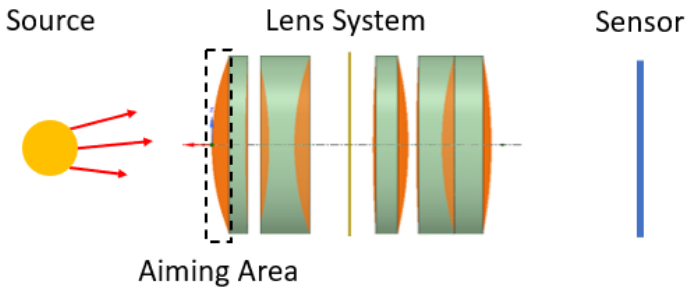
The Aiming Area corresponds to the first face of the lens system surrounded by a bounding box. The optimized propagation engine makes sure that rays launched aim to this area. Once rays interfered with the aiming area, the propagation is done following the Monte Carlo algorithm from the Aiming Area.

According to the simulation, the aiming area is different:

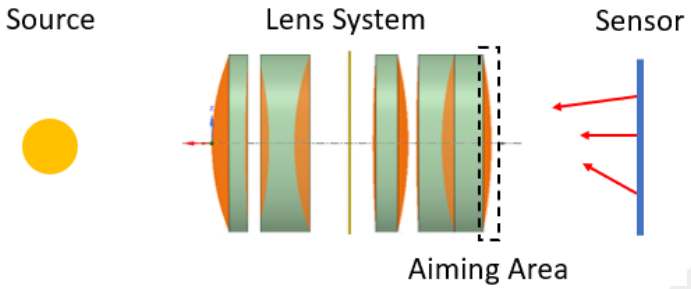
- In case of a Direct Simulation the Aiming Area corresponds to the first face of the lens from the source.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00



- In case of an inverse simulation the Aiming Area corresponds to the first face of the lens from the sensor.



Direct Simulation VS Inverse Simulation

According to the source to simulate you will not use the same simulation type.

- To simulate an Ambient Source, use the Inverse Simulation.
- To simulate all other sources, use the Direct Simulation.

When rays generated from the sensor enter the Aiming Area, the propagation is mainly done specularly. That means Speos has no control over the propagation (direction) of the rays afterthat. Therefore, when rays exit the lens system (after last face of the light box used), the convergence will be correct in case of Ambient Sources, but very slow in case of other sources.

Emission Angle of Sources

The sources with an emission angle correspond to the source for which you can define an intensity diagram (Surface, Luminaire, Display).

If a ray launched from the Aiming Area is outside of the emission angle of the source, the ray is lost. If the emission angle is small, the convergence is very slow. We recommend you to convert your source in a Ray File Source as rays are hard-coded.

8.3.12.4. Creating a Physical Camera

The following procedure helps you create a Physical Camera to simulate a lens system.

> **Note:** The Physical Camera is in BETA mode for the current release.

To create a Physical Camera:

You must have exported a **Speos Light Box** containing a lens system following specific rules.

You must have created an **Irradiance sensor** following specific rules.

DRAFT

Make sure the Physical Camera is in the ambient material. That is to say, avoid including the Physical Camera in a volumic body.

1. From the **Light Simulation** tab, click **System** > **Physical Camera (beta)** .
2. Define the **Axis System** of the Physical Camera in the 3D view by clicking to select one point for the origin, to select a line for the X axis, to select a line for the Y axis or click and select a coordinate system to autofill the Axis System.

Axis System	
Origin	[Default] Origin
X direction	[Default] X Axis
Reverse direction	False
Y direction	[Default] Y Axis
Reverse direction	False

The Axis System is only used to position the optical system included in the Light Box.

> **Note:** Make sure the sensor is not tangent to a geometry.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

If you need to adjust the axes orientation, use **Reverse direction** on one or both axes.

3. In the **Light Box** section, browse and select a *.speoslightbox that contains all the geometries of a lens system.

Light Box	
File	[No file]*

4. In the **General** section:

General	
Number of sequences	10
Sequence file	[No file]

a) Define the **Number of sequences** to be provided in the *.xmp result.

Number of Sequences determines the number of sequences to play and provide into the *.xmp result file.

Number of Sequences is useful only if you add a sequence file as input of the Physical Camera.

b) In the **Sequence file** field, you can add or not a Sequence file.

> **Note:** When an *.OPTSequence file is used in a Physical Camera, every generated simulation result filename (*.xmp, *.lpf, etc.) is suffixed with "_sequential".

For more details on the purpose of adding a sequence file or not, refer to Understanding the Physical Camera Parameters.

5. In the **Sensor** section, select an **Irradiance sensor**. Make sure the sensor follow the rules.

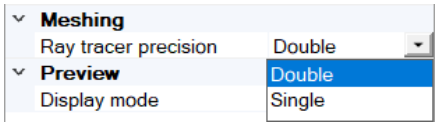
Sensor	
Irradiance sensor	[No selection]*

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Depending on whether you add a Sequence file or not, the Layer type choice will differ. For more information refer to Understanding the Physical Camera Parameters.

6. In the **Meshing** section, select the **Ray tracer precision** algorithm to be used by the propagation engine of the Physical Camera:



The Ray tracer precision is dedicated to calculate the intersections between rays and the geometries contained in the Light Box used in the Physical Camera.

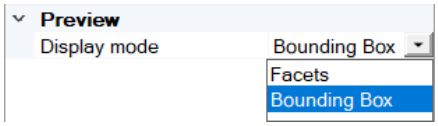
- Select **Single**

  The Single Precision mode allows you to use a fast ray tracing technique that provides a standard level of precision.

- Select **Double**

  The **Double Precision** mode uses Smart Engine, a ray tracing technique that provides a high level of precision.

7. In the **Preview** section, with **Display mode**, select how should the Physical Camera be displayed in the 3D view.



8. In the 3D view, click  to compute the Physical Camera.

The Physical Camera is created and visible in Speos tree and in the 3D view.

8.3.12.5. Creating an *.OPTSequence File for a Physical Camera Sensor

The following page presents you the workflow on how to create an *.OPTSequence file to be used in a Physical Camera sensor.

> **Note:** The Physical Camera is in BETA mode for the current release.

To create an *.OPTSequence file:

1. Create a Speos Light Box Export containing all the geometries related to the lens system (lenses, opto-mechanics, etc.)

   You can also do an OpticStudio Import to define the geometries of the Light Box.

   a) Define the Axis System of the Light Box so that the optical axis of the lens system follows the Z axis.

      The origin of the axis system must be located with a small offset from the first geometry face. For more details, refer to Understanding the Physical Camera Parameters.

   b) Define if the Light Box is black-boxed or not.

      > **Note:** A black-boxed Light Box will not be supported with a Light Expert Analysis.

   c) Compute the Light Box Export.

2. Create an Irradiance sensor:
   a) Position the Irradiance sensor in the scene considering the lens system defined in the Light Box, which is usually placed in front of it.
   b) Select the **Planar** integration type.
   c) Select **Sequence** as **Layer** type and define sequences per **Faces**.
      This will allow to generate the *.OPTSequence file to be used later in the Physical Camera sensor.
3. Create a Physical Camera in which you will add the Light Box containing the lens system, and the Irradiance sensor (with Layer defined by Sequences and Sequences defined per Faces).
4. Compute the Physical Camera.
5. Create a Direct or an Inverse simulation:
   a) Make sure the **Dispersion** is activated.
   b) Add the Physical Camera.

      Note: Light Expert is not compatible when the Physical Camera uses a black-boxed Light Box.

   c) Add geometries and sources.
      The ability of the *.OPTSequence file to be generated to solve optical paths highly depends on the source setup you will add here. For more information, refer to Understanding the Physical Camera Parameters.

   Important: The *.OPTSequence file depends on the simulation type. An *OPTSequence file generated with a Direct simulation can only be used as input in a Direct simulation. The same rule goes for the Inverse simulation.

6. Run the simulation.

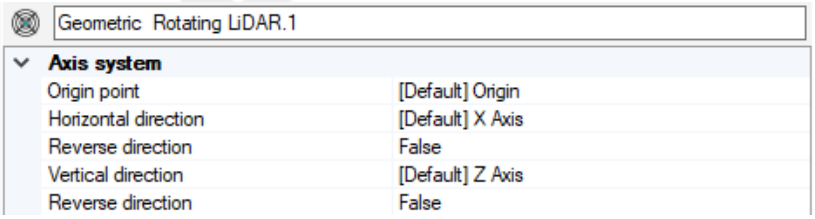
The *.OPTSequence file is automatically generated and can be then added as input to the Physical Camera sensor.

8.3.13. Creating a Geometric Rotating LiDAR Sensor

This page shows how to create a Geometric Rotating LiDAR sensor that will be used for LiDAR simulation.

To create a Geometric Rotating LiDAR sensor:

1. From the **Light Simulation** tab, click **System** > **Geometric Rotating LiDAR**.
2. Define the **Axis system** of the sensor by setting its position and orientation:



- In the 3D view, click to select one point for the origin (this point places the sensor in the scene and defines from where the pulses are sent).

  Two arrows appear in the 3D view and indicate the sensor's orientation.

- If you need to adjust the **Horizontal direction** (LiDAR's line of sight), in the 3D view, click  and select a line.

  The horizontal direction is used as a reference for the horizontal and vertical fields of view.

- If you need to adjust the **Vertical direction** (the rotation axis), in the 3D view, click  and select a line.

  The vertical direction is normal to the horizontal axis and is considered as a rotation axis.

- or click

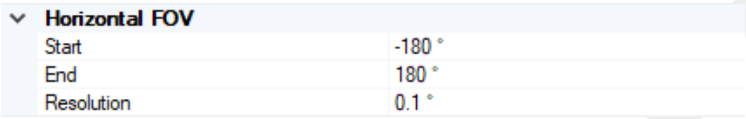
  

  and select a coordinate system to autofill the Axis System.

3. From the **Type** drop-down list, select one of the predefined types of color scale.

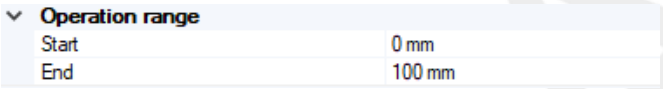
   The scale is used to display the results in the 3D view. The distance from the object is illustrated by color variation.

4. Define the sensor's field of view:

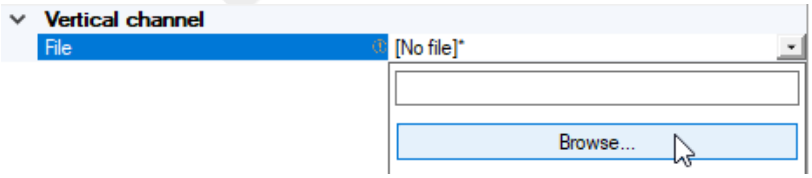
   a) If needed, edit the **Start** and **End** points of the horizontal field of view.
   b) Adjust the **Resolution** of the sensor (in degrees).

5. In **Operation range**, bound the sensor's detection range by editing the **Start** and **End** values.

6. If needed, adjust the **Point thickness** to be used for 3D view visualization.
7. In **Vertical channel**, click in the file field and click **Browse** to load a vertical channel (.txt) file.

   > **Note:** The vertical channel basically defines the vertical resolution of the rotating LiDAR by listing each channel's angular direction. The angles range from -90° to 90°. file
   >
   > The file is structured as follows:
   >
   > Line 1: Total number of vertical channels
   >
   > Line 2 : empty
   >
   > Line 3: 45 (first channel's angular direction)
   >
   > Line 4: 30 (second channel's angular direction) etc.

8. Press **F4** to leave the edition mode.

The Geometric Rotating LiDAR sensor is created.

Now, create a Geometric Rotating LiDAR simulation.

DRAFT

Related information

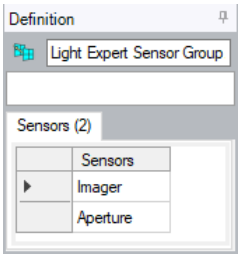
Creating a Geometric Rotating LiDAR Simulation on page 464
Creating a Geometric Rotating LiDAR simulation allows you to perform field of view studies. A field of view study allows you to quickly identify what can or must be optimized (for example, the number, position and direction of sensors) in a LiDAR system.

8.3.14. Creating a Light Expert Group

The Light Expert Group is dedicated to perform a Light Expert analysis for several sensors at the same time.

To create a light expert group:

1. From the **Light Simulation** tab, click **Light Expert Group** .
2. In the 3D view, click and select one or more Irradiance sensors and/or one Intensity sensor at maximum.



Make sure all sensors have the same Layer type: **None**, **Data Separated by Source** or **Data separated by Sequence**.

> **Tip:** You can also select a **Named selection** composed of sensors. For more information, refer to the Grouping section.

3. Click **Validate** .

The sensors are added to the Light Expert Group.

You can now create and run a Direct Simulation containing the Light Expert Group.

Click here for more information on how to perform a multi-sensors light expert analysis.

8.4. Stray Light Analysis

The following section helps you set up and analysis the stray light in an optical system.

8.4.1. Stray Light Analysis Overview

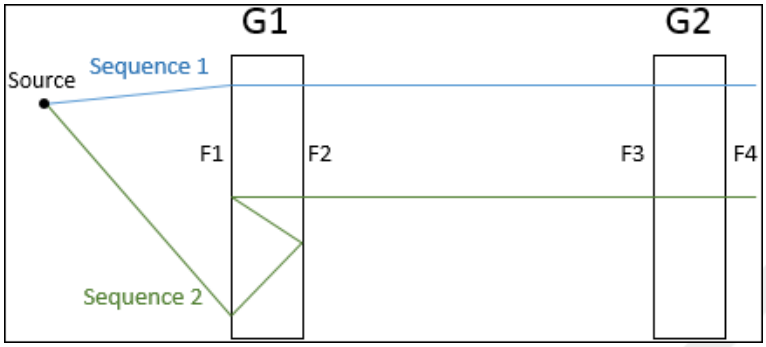
The Stray Light corresponds to the contribution of the photons that you can visualize according to the sensor defined.

Description

The Stray Light Analysis allows you to visualize the contribution of the photons, separating them by sequence in the simulation results.

A sequence is the path taken by the rays, calculated either according to the faces (F) or the geometries (G) the rays hit.

Example of Sequences



Understanding the example:

- Defining the sequence 1 per faces: the rays hit the face F1, then the face F2, the face F3 and the face F4. (F1 F2 F3 F4)
- Defining the sequence 1 per geometries: the rays hit twice the geometry G1, then twice the geometry G2. (G1 G1 G2 G2)
- Defining the sequence 2 per faces: the rays hit the face F1, then the face F2, then hit the faces F1 and F2 again, and finally the faces F3 and F4. (F1 F2 F1 F2 F3 F4)
- Defining the sequence 2 per geometries: the rays hit the geometry G1 four times, then the geometry 2 twice. (G1 G1 G1 G1 G2 G2)

> **Note:** Defining the sequences per faces is useful when you have only one geometry. Defining the sequences per geometries is useful when you have more than one geometry.

> **Note:** In case of an optical system with one or more lenses, we recommend you to define the sequences per geometries.

The Stray Light Analysis is compatible with the Light Expert to visualize the interactive ray tracing of each sequence.

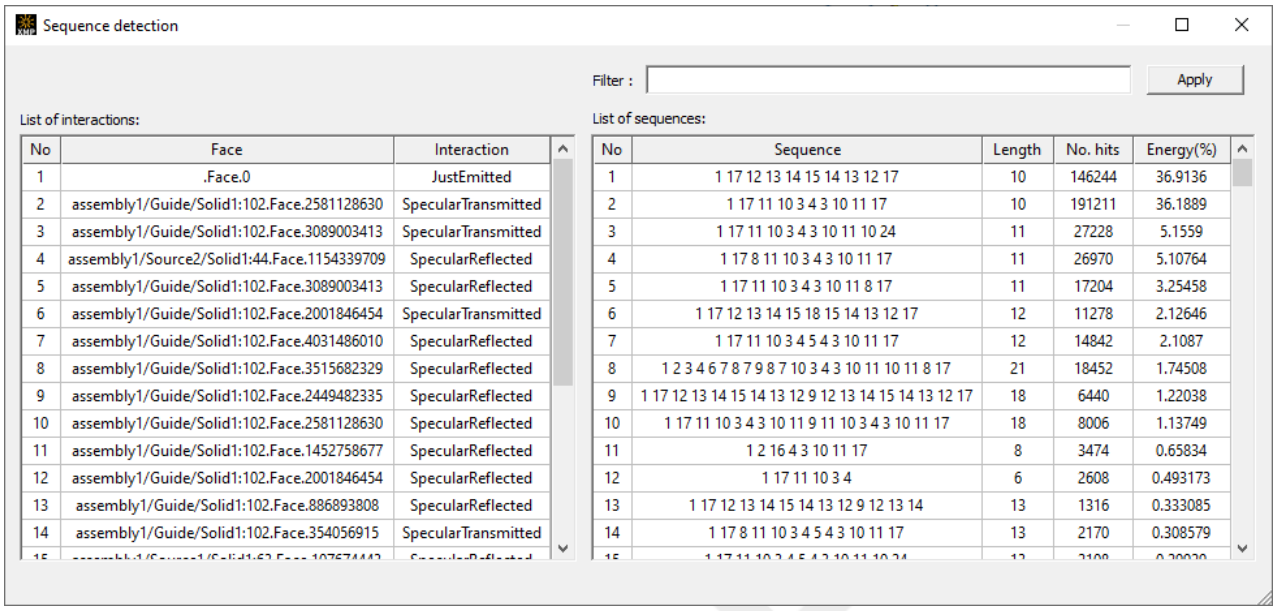
8.4.2. Understanding the Sequence Detection Tool

The Sequence Detection tool helps you see the interactions of each sequence with the elements hit by the rays as well as the order in which elements have been hit.

> **Note:** The Sequence Detection Tool is not compatible with Isolated Simulations.

The Sequence Detection tool provides you with two lists:

- The list of interactions
- The list of sequences



No	Face	Interaction
1	.Face.0	JustEmitted
2	assembly1/Guide/Solid1:102.Face.2581128630	SpecularTransmitted
3	assembly1/Guide/Solid1:102.Face.3089003413	SpecularTransmitted
4	assembly1/Source2/Solid1:44.Face.1154339709	SpecularReflected
5	assembly1/Guide/Solid1:102.Face.3089003413	SpecularReflected
6	assembly1/Guide/Solid1:102.Face.2001846454	SpecularTransmitted
7	assembly1/Guide/Solid1:102.Face.4031486010	SpecularReflected
8	assembly1/Guide/Solid1:102.Face.3515682329	SpecularReflected
9	assembly1/Guide/Solid1:102.Face.2449482335	SpecularReflected
10	assembly1/Guide/Solid1:102.Face.2581128630	SpecularReflected
11	assembly1/Guide/Solid1:102.Face.1452758677	SpecularReflected
12	assembly1/Guide/Solid1:102.Face.2001846454	SpecularReflected
13	assembly1/Guide/Solid1:102.Face.886893808	SpecularReflected
14	assembly1/Guide/Solid1:102.Face.354056915	SpecularTransmitted

No	Sequence	Length	No. hits	Energy(%)
1	1 17 12 13 14 15 14 13 12 17	10	146244	36.9136
2	1 17 11 10 3 4 3 10 11 17	10	191211	36.1889
3	1 17 11 10 3 4 3 10 11 10 24	11	27228	5.1559
4	1 17 8 11 10 3 4 3 10 11 17	11	26970	5.10764
5	1 17 11 10 3 4 3 10 11 8 17	11	17204	3.25458
6	1 17 12 13 14 15 18 15 14 13 12 17	12	11278	2.12646
7	1 17 11 10 3 4 5 4 3 10 11 17	12	14842	2.1087
8	1 2 3 4 6 7 8 7 9 8 7 10 3 4 3 10 11 10 11 8 17	21	18452	1.74508
9	1 17 12 13 14 15 14 13 12 9 12 13 14 15 14 13 12 17	18	6440	1.22038
10	1 17 11 10 3 4 3 10 11 9 11 10 3 4 3 10 11 17	18	8006	1.13749
11	1 2 16 4 3 10 11 17	8	3474	0.65834
12	1 17 11 10 3 4	6	2608	0.493173
13	1 17 12 13 14 15 14 13 12 9 12 13 14	13	1316	0.333085
14	1 17 8 11 10 3 4 5 4 3 10 11 17	13	2170	0.308579

List of Interactions

The list of interactions lists the interactions of the rays with the elements of the optical system.

- When a result has been generated using "faces" sequence layer parameter, the list of interactions provide the different faces that rays have interacted with.

  Example: Optical Surface Rectangular.1:3176.Face.1 for the first face of "Optical Surface Rectangular.1:3176" body

  A face name can be listed several times in the list of interactions. For instance, when the face is considered for transmission and reflection.

  > **Note:** Face names defined in the Speos are not retrieved by the Sequence Detection tool in case of sequences by faces. The Sequence Detection tool only knows the name of the body on which the faces belong. Each face of a body is identified by a unique number (integer): .Face.1 for the first face of the body found in the list, .Face2, for the second one, etc.

- When a result has been generated using "geometries" layer parameter, the list of interactions provide the different bodies that rays have interacted with.

  Example: Optical Surface Rectangular.1:3174

- When an interaction has no face name, that means it corresponds to the interaction with a sensor.

List of Sequence

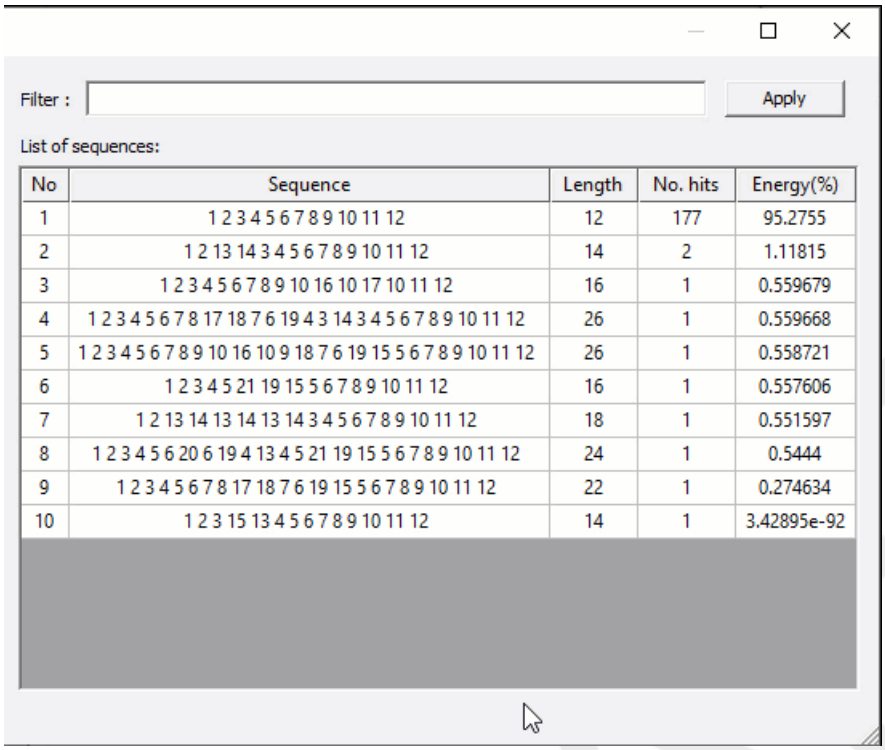
The list of sequences provides the different paths taken by the rays, calculated either according to the faces (F) or the geometries (G) the rays hit.

For more information on the sequence, see Stray Light Analysis Overview.

By default, the sequences of the List of sequences are sorted by descending order of Energy(%) values.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

The List of sequences can be sorted according to the following columns content (No., Length, No. hits, Energy(%)) by clicking on the column header by ascending or descending order.



No	Sequence	Length	No. hits	Energy(%)
1	1 2 3 4 5 6 7 8 9 10 11 12	12	177	95.2755
2	1 2 13 14 3 4 5 6 7 8 9 10 11 12	14	2	1.11815
3	1 2 3 4 5 6 7 8 9 10 16 10 17 10 11 12	16	1	0.559679
4	1 2 3 4 5 6 7 8 17 18 7 6 19 4 3 14 3 4 5 6 7 8 9 10 11 12	26	1	0.559668
5	1 2 3 4 5 6 7 8 9 10 16 10 9 18 7 6 19 15 5 6 7 8 9 10 11 12	26	1	0.558721
6	1 2 3 4 5 21 19 15 5 6 7 8 9 10 11 12	16	1	0.557606
7	1 2 13 14 13 14 13 14 3 4 5 6 7 8 9 10 11 12	18	1	0.551597
8	1 2 3 4 5 6 20 6 19 4 13 4 5 21 19 15 5 6 7 8 9 10 11 12	24	1	0.5444
9	1 2 3 4 5 6 7 8 17 18 7 6 19 15 5 6 7 8 9 10 11 12	22	1	0.274634
10	1 2 3 15 13 4 5 6 7 8 9 10 11 12	14	1	3.42895e-92

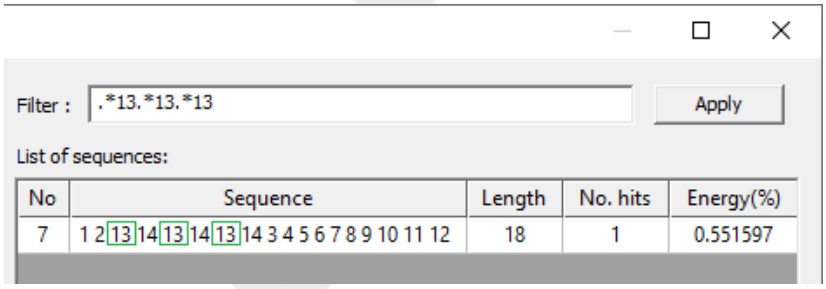
Sequence Detection Filter

The Sequence Detection Filter helps you find the sequences with specific interactions or list of interactions.

The Sequence Detection Filter uses regular expressions (regex).

Example: You want to display the sequences that include at least 3 times the geometry 13: **.*13.*13.*13**

As a result, sequences that include at least 3 times the geometry 13 will appear in the list of sequences.



No	Sequence	Length	No. hits	Energy(%)
7	1 2 13 14 13 14 13 14 3 4 5 6 7 8 9 10 11 12	18	1	0.551597

8.4.3. Making Stray Light Analysis

The following procedure helps you set up a Stray Light Analysis.

To make a Stray Light Analysis:

1. Create an Irradiance, a Radiance or an Intensity sensor.
2. In the **General** section, in **Layers**, select **Sequences**
3. Define the **Maximum number of sequences** to calculate.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

4. Define the sequences per **Geometries** or **Faces**.
5. Create a Direct or Inverse simulation.
6. For the Inverse simulation, in the Simulation Options, activate the **Monte Carlo algorithm**, the **Dispersion**, and deactivate the **Splitting**.
7. In the **General** section, set the **Light Expert** to **True**.

> **Note:** Whether Light Expert is activated or not during the simulation definition, it is automatically used as a background process as soon as data are separated by sequence. The advantage of activating Light Expert manually is that it allows you to activate the option only for the sensors you want to analyze and to determine the number of rays you want to embed in the results.

8. In the 3D view, click **Compute**  to launch the simulation.

   At the end of the simulation, under the Simulation node, two files are generated:

   - a XMP map including layers displaying the contribution of each sequence.
   - a .lpf file.

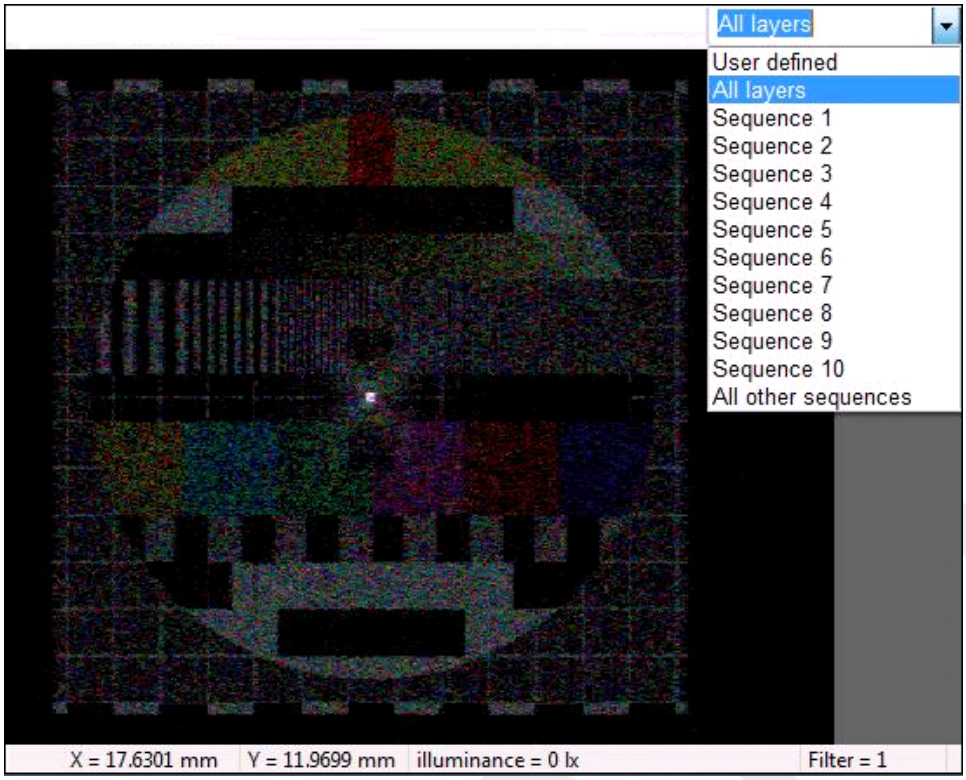
8.4.4. Analyzing Stray Light

The following procedure helps you analyze the stray light thanks to Virtual Photometric Lab and Light Expert.

To analyze Stray Light:

1. Right-click the *.lpf result file to open the Light Expert and the XMP map in Virtual Photometric Lab.
2. In Virtual Photometric Lab, select the sequences and define the measures to make the analysis.
   - To display the contribution of a sequence on the map: in the **Layer** list, select a sequence.
   - To display the contribution of a sequence in the 3D view: use the Light Expert and select a sequence in the **Layer** list.

DRAFT



> **Note:** When the number of sequences found and displayed in the Layer list is lower than the maximum number of sequences asked, the **All other sequences** layer is empty.

- To display the rays in the 3D view, open the **Measure** tool and specify measure areas on the XMP map.
- To combine sequences in the XMP map, use the **Virtual Lighting controller**.
- In **Tools**, select **Sequence Detection** to see the interactions of each sequence with the elements.

> **Note:** The Sequence Detection Tool is not compatible with faceted geometries as the multiple faces of the faceted geometries are not detected.

> **Note:** When elements (faces or geometries) are hidden behind other elements in the 3D view, when using **Sequence Detection**, they can be highlighted only when the rendering mode manages transparency. Click the face in the **List of interactions** to highlight it in the 3D view.
>
> Make sure to activate the component (or sub-component) in which the simulation is located if you want to highlight faces or geometries with the Sequence Detection tool.

> **Note:** From Speos 2018, separating data by sequence generates a .OPTSequence file containing the sequences data used to display sequences in the xmp map.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

9: Components

The following section comprises several optical components such as 3D texture and Speos Light Boxes.

9.1. Speos Light Box

The Speos Light box allows you to export data between CAD platforms. Speos Light Boxes can be used for data exchange between suppliers and customers.

9.1.1. Speos Light Box Overview

The Speos Light box allows you to export and import geometries along with the associated optical data (sources, Optical Properties, meshing properties).

The Speos Light Box can be used for data exchange between suppliers and customers. It is compatible with multi-CAD platform where Ansys software are integrated.

The Light Box can also be encrypted with passwords to reinforce security and black-boxed to restrict the access to all the sources/geometries/optical properties details and protect the intellectual property.

In Speos, you can:

- Export a Speos Light Box file,
- Import a Speos Light Box file,
- Re-import an exported Speos Light Box file.

> **Note:** Ansys software can only read and use data from a Speos Light Box. Nothing can be modified from a Speos Light Box.

What does a Speos Light Box contain?

Geometries

Compatible geometries along with associated optical and meshing properties.

All geometries integrated to the Speos Light Box are stored as mesh (with meshing properties related to the Speos properties).

> **CAUTION:** You cannot add faceted geometries from the 3D view into a Speos Light Box. Add them from the Structure tree.

Sources

In the **Sources** list, from the specification tree or 3D view, you can click the sources to add to the exported component.

DRAFT

You can add the following sources:

- Source group
- Surface source
- Ray file source
- Display source
- Light Field source
- Sources coming from imported Speos Light Boxes.

Note: Speos Light Box Export does not support Speos Patterns.

Note: The **Length of rays** of a source is not taken into account in a Speos Light Box. After the Speos Light Box import, the length will be different from the length set in the original source.

Files

A Speos Light Box can include the following file types:

- *.scattering (advanced scattering files)
- *.simplescattering (simple scattering files)
- *.brdf (complete scattering files)
- *.unpolished (unpolished files)
- *.BSDF180 (BSDF180 files)
- *.anisoptropicbsdf (anisotropic bsdf files)
- .material (material files)
- *.ies (IES files)
- *.ldt (Eulumdat files)
- *.spectrum (Spectrum file)

Note: All other file types not included in the list above are not encapsulated in the Speos Light Boxes. However they are physically located next to the file as dependency files.

9.1.2. Understanding Speos Light Box Parameters

This section describes the parameters to set when exporting or importing a Speos Light Box.

Black Box Option (Export)

Description

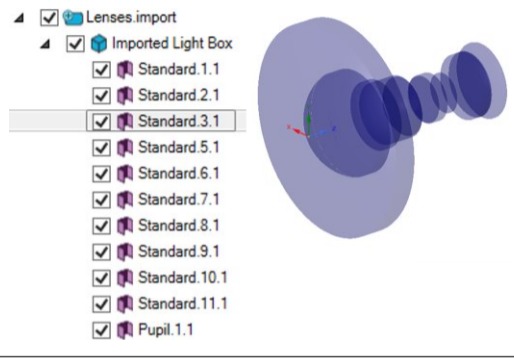
A Speos Light Box can be black-boxed to restrict the access to all its sources/geometries/optical properties details and protect the intellectual property.

When the Black Box option is activated, the *.SPEOSLightBox file and all the non-packaged dependencies are placed in an encrypted container, when computing the Speos Light Box Export feature.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

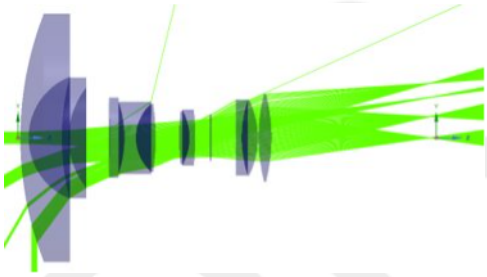
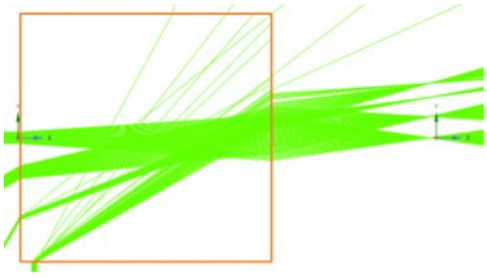
Published: 2023-10-23T14:08:14.04-04:00

Then, when importing a black-boxed Speos Light Box, the Speos Light Box Import feature in the tree does not display the included geometries and sources, and only the bounding box of the Speos Light Box is displayed in the 3D view.

Black Box Deactivated	Black Box Activated
	

Impacts

- In case of an Interactive Simulation or a Light Expert analysis:
  - When simulating a system with a black-boxed Speos Light Box, the ray tracing does not show the optical path inside the black-boxed Speos Light Box. A simple line is displayed inside the bounding box between the input impact and the output impact on the bounding box.
  - Impact report does not mention impacts inside the black-boxed Speos Light Box, it only records impacts on the bounding box.

Black Box Deactivated	Black Box Activated
	

- In case you input a black-boxed light box in a Speos Pattern, or a Camera, the 3D view will not display the content of the light box.
- The HTML report provides limited information related to the black-boxed light box (example: number of bodies contained in the light box is not counted).
- Results can be impacted. Example: If you separated the data by source, in the XMP result you cannot see the layers corresponding to the sources included in a black-boxed light box.

> **Note:** When using a black-boxed light box in a Speos pattern, the bounding box of the light box can appear larger and sometimes incorrectly oriented in the 3D view.

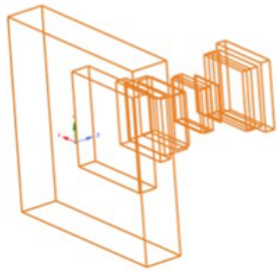
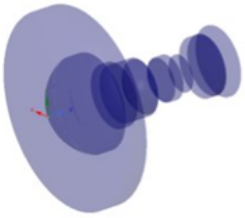
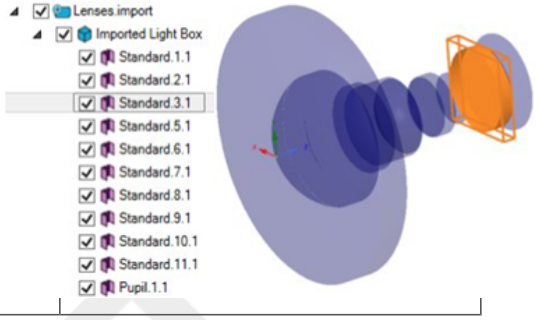
Display Mode (Import)

The Display Mode parameter provide you with to modes to visualize your imported Speos Light Box:

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

- Facets
- Bounding Box

Bouding Box	Facets	Facets with a selected element
		

Important: Each time you change the Display mode, you need to compute the Speos Light Box.

Trajectory File (Import)

The Trajectory File (.json) allows you to create a dynamic group of geometries. When a trajectory file is assigned to a Speos Light Box Import and the feature is edited, the trajectory is displayed in the 3D view.

To understand how to create a Trajectory File, refer to Trajectory File and Trajectory Script Example.

9.1.2.1. Trajectory File

This page describes the format of trajectory files, used to define the position and orientations of a Speos Light Box, a LiDAR sensor, or a Camera sensor in time.

Description

Positions and orientations are expressed with the respect to a reference coordinate system, so the trajectory is relative to this coordinate system.

For instance, the same trajectory file can be used to describe a translation movement of a car as well as the LiDAR sensor placed on it.

Trajectory is described in a *.json file that contains each chronological sample:

DRAFT

Format Description	Example

• "Time" in second.
• "Origin" with "X", "Y", "Z" coordinates in mm, that corresponds to the relative position with the respect to reference coordinate system.
• "Direction_X" with "X", "Y", "Z" coordinates that corresponds to the relative X direction with the respect to reference coordinate system.
• "Direction_Y" with "X", "Y", "Z" coordinates that corresponds to the relative Y direction with the respect to reference coordinate system.
	<pre>"Trajectory": {
	{
	"Time": 0.0,
	"Origin": {
	"X": 0.0,
	"Y": 0.0,
	"Z": 0.0
	},
	"Direction_X": {
	"X": 1.0,
	"Y": 0.0,
	"Z": 0.0
	},
	"Direction_Y": {
	"X": 0.0,
	"Y": 1.0,
	"Z": 0.0
	}
	},
	"Time": 0.033333333333,
	"Origin": {
	"X": 0.0,
	"Y": -0.0,
	"Z": 900.0
	},
	"Direction_X": {
	"X": 1.0,
	"Y": 0.0,
	"Z": 0.0
	},
	...</pre>

Script Example

Trajectory file can be easily accessed (read or write) using dedicated scripting interfaces available in IronPython and Python.

> **Note:** Make sure to use the **3.9 version** of IronPython or Python language to write your scripts.

IronPython Example

```
import sys
import clr

sys.path += [R"C:\Program Files\ANSYS Inc\v2XX\Optical Products\SPEOS\Bin"]

clr.AddReferenceToFile("Optis.Core_net.dll")
clr.AddReferenceToFile("Optis.Data_net.dll")

# References to Optis.Core and Optis.Data
import Optis.Core as OptisCore
import Optis.Data as OptisData
```

```
try:
  firstData = OptisCore.DAxisSystemData()
  firstData.Time = 0.0
  firstData.Origin.Init(0, 0, 0)
  firstData.Direction_X.Init(1, 0, 0)
  firstData.Direction_Y.Init(0, 1, 0)

  lastData = OptisCore.DAxisSystemData()
  lastData.Time = 7.0
  lastData.Origin.Init(0, 0, 3000)
  lastData.Direction_X.Init(1, 0, 0)
  lastData.Direction_Y.Init(0, 1, 0)

  dmTrajectory = OptisCore.DAxisSystemTrajectory()
  dmTrajectory.Trajectory.Resize(2)
  dmTrajectory.Trajectory.Set(0, firstData)
  dmTrajectory.Trajectory.Set(1, lastData)

  strPathTrajectoryFile = OptisCore.String.From(R"C:\trajectory.json")
  pathTrajectoryFile = OptisCore.Path(strPathTrajectoryFile)

  cAxisSystemTrajectoryWriter = OptisData.CAxisSystemTrajectoryWriter()
  cAxisSystemTrajectoryWriter.Open(pathTrajectoryFile)
  cAxisSystemTrajectoryWriter.Write(dmTrajectory)

  cAxisSystemTrajectoryReader = OptisData.CAxisSystemTrajectoryReader()
  cAxisSystemTrajectoryReader.Open(pathTrajectoryFile)
  dmTrajectory = cAxisSystemTrajectoryReader.Read()

  cAxisSystemTrajectoryReader.Close()

  print "Done"

except:
  print "Exception raised"
```

Python Example

```
import sys

sys.path += [R" C:\Program Files\ANSYS Inc\v2XX\Optical Products\SPEOS\Bin"]

# References to Optis.Core and Optis.Data
import IllumineCore_pywrap as OptisCore
import IllumineData_pywrap as OptisData

try:
  firstData = OptisCore.DAxisSystemData()
  firstData.Time = 0.0
  firstData.Origin.Init(0, 0, 0)
  firstData.Direction_X.Init(1, 0, 0)
  firstData.Direction_Y.Init(0, 1, 0)

  lastData = OptisCore.DAxisSystemData()
  lastData.Time = 7.0
  lastData.Origin.Init(0, 0, 3000)
  lastData.Direction_X.Init(1, 0, 0)
```

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

```
  lastData.Direction_Y.Init(0, 1, 0)

  dmTrajectory = OptisCore.DAxisSystemTrajectory()
  dmTrajectory.Trajectory.Resize(2)
  dmTrajectory.Trajectory.Set(0, firstData)
  dmTrajectory.Trajectory.Set(1, lastData)

  strPathTrajectoryFile = OptisCore.String(R"C:\trajectory.json")
  pathTrajectoryFile = OptisCore.Path(strPathTrajectoryFile)

  cAxisSystemTrajectoryWriter = OptisData.CAxisSystemTrajectoryWriter()
  cAxisSystemTrajectoryWriter.Open(pathTrajectoryFile)
  cAxisSystemTrajectoryWriter.Write(dmTrajectory)

  cAxisSystemTrajectoryWriter.Close()

  print("Done")

except:
  print("Exception raised")
```

9.1.2.2. Trajectory Script Example

The following page gives you a full script example to create a trajectory in Speos.

> **Note:** Make sure to use the **3.9 version** of IronPython or Python language to write your scripts.

```
# © 2012-2021 ANSYS, Inc. All rights reserved. Unauthorized use, distribution,
 or duplication is
prohibited.

# THIS ANSYS SOFTWARE PRODUCT AND PROGRAM DOCUMENTATION INCLUDE TRADE SECRETS
AND ARE CONFIDENTIAL AND PROPRIETARY PRODUCTS OF ANSYS, INC., ITS SUBSIDIARIES,
 OR
LICENSORS.  The software products and documentation are furnished by ANSYS,
Inc., its subsidiaries,
or affiliates under a software license agreement that contains provisions
concerning non-disclosure,
copying, length and nature of use, compliance with exporting laws, warranties,
 disclaimers,
limitations of liability, and remedies, and other provisions.  The software
products and
documentation may be used, disclosed, transferred, or copied only in accordance
 with the terms and
conditions of that software license agreement.

import sys
import clr
import os
from os import path

sys.path += [R"C:\Program Files\ANSYS Inc\v211\Optical Products\SPEOS\Bin"]

clr.AddReferenceToFile("Optis.Core_net.dll")
clr.AddReferenceToFile("Optis.Data_net.dll")
```

```
# References to Optis.Core and Optis.Data
import Optis.Core as OptisCore
import Optis.Data as OptisData

def GetAxisSystemData(iTime, iFrame):
    dAxisSystemData = OptisCore.DAxisSystemData()
    dAxisSystemData.Time = iTime
    dAxisSystemData.Origin.Init(1000 * iFrame.Origin.X,
                                1000 * iFrame.Origin.Y,
                                1000 * iFrame.Origin.Z)

    tmpVector = OptisCore.Vector3_double()
    tmpVector.Init(iFrame.DirX.X,
                   iFrame.DirX.Y,
                   iFrame.DirX.Z)
    tmpVector.Normalize()

    dAxisSystemData.Direction_X.Init(tmpVector.Get(0),
                                     tmpVector.Get(1),
                                     tmpVector.Get(2))

    tmpVector.Init(iFrame.DirY.X,
                   iFrame.DirY.Y,
                   iFrame.DirY.Z)
    tmpVector.Normalize()

    dAxisSystemData.Direction_Y.Init(tmpVector.Get(0),
                                     tmpVector.Get(1),
                                     tmpVector.Get(2))

    return dAxisSystemData

# Working directory
workingDirectory = path.dirname(GetRootPart().Document.Path.ToString())

# SpaceClaim InputHelper
ihTrajectoryName = InputHelper.CreateTextBox("Trajectory.1", 'Trajectory name:
 ', 'Enter the name of the trajectory')
ihFrameFrequency = InputHelper.CreateTextBox(30, 'Timeline frame rate:', 'Set
 timeline frame rate (s-1)', ValueType.PositiveInteger)
ihReverseDirection = InputHelper.CreateCheckBox(False, "Reverse direction",
"Reverse direction on trajectory")
ihObjectSpeed = InputHelper.CreateTextBox(50, 'Object speed:', 'Set the moving
 object speed (km/h)', ValueType.PositiveDouble)

ihCoordinateSystem = InputHelper.CreateSelection("Reference coordinate system",
 SelectionFilter.CoordinateSystem, 1, True)
ihTrajectory = InputHelper.CreateSelection("Trajectory curve",
SelectionFilter.Curve, 1, True)

InputHelper.PauseAndGetInput('Trajectory parameters', ihTrajectoryName,
ihFrameFrequency, ihObjectSpeed, ihReverseDirection, ihCoordinateSystem,
ihTrajectory)

# Animation frame rate (s-1)
frameFrequency = float(ihFrameFrequency.Value)

# Object speed (km/h)
objectSpeed = float(ihObjectSpeed.Value)
```

```
# Trajectory file
trajectoryName = str(ihTrajectoryName.Value)
trajectoryAcquisitionFile = workingDirectory + "\\" + trajectoryName

# Reference coordinate system
coordSysSelection = ihCoordinateSystem.Value
trajectoryCoordSys = coordSysSelection.Items[0]

# Trajectory curve
trajCurveSelection = ihTrajectory.Value
trajectoryCurve = trajCurveSelection.Items[0]

# Reversed trajectory (Boolean)
isReversedTrajectory = ihReverseDirection.Value

# Acquisition of positions
def GetPositionOrientation(i_CoordSys, i_ReferenceCoordSys):
    # change base of current position
    newMatric = Matrix.CreateMapping(i_ReferenceCoordSys.Frame)

    currentPosition = newMatric.Inverse  * i_CoordSys.Frame.Origin
    currentPosition = Point.Create(round(currentPosition.X, 6),
round(currentPosition.Y, 6), round(currentPosition.Z, 6))

    # change base of current iDirection
    iDirectionX = newMatric.Inverse  * i_CoordSys.Frame.DirX

    # change base of current iDirection
    jDirectionX = newMatric.Inverse  *i_CoordSys.Frame.DirY

    # Return new frame
    return Frame.Create(currentPosition, iDirectionX, jDirectionX)

def MoveAlongTrajectory(i_trajectoryCoordSys, i_trajectoryCurve,
i_isReversedTrajectory, i_frameFrequency, i_objectSpeed):
    selectedCoordSys = Selection.Create(i_trajectoryCoordSys)
    newselectedCoordSys = i_trajectoryCoordSys

    pathLength = i_trajectoryCurve.Shape.Length
    selectedCurve = Selection.Create(i_trajectoryCurve)

    # Convert speed in m/s
    convObjectSpeed = float(i_objectSpeed * 1000 / 3600)

    currentPosition = 0.0
    timeStamp = 0.0
    positionTable = []
    timeTable = []

    while currentPosition < 1:
        options = MoveOptions()

        if currentPosition == 0:
            options.Copy = True
        else:
            options.Copy = False

        if i_isReversedTrajectory:
            result = Move.AlongTrajectory(selectedCoordSys, selectedCurve,
1-currentPosition, options)
```

```
            if currentPosition == 0:
              newselectedCoordSys = result.GetCreated[ICoordinateSystem]()[0]

                selectedCoordSys = Selection.Create(newselectedCoordSys)
                if newselectedCoordSys.Frame.Origin !=
i_trajectoryCoordSys.Frame.Origin:
                    options.Copy = False
                    result = Move.AlongTrajectory(selectedCoordSys,
selectedCurve, currentPosition, options)
        else:
            result = Move.AlongTrajectory(selectedCoordSys, selectedCurve,
currentPosition, options)

            if currentPosition == 0:
              newselectedCoordSys = result.GetCreated[ICoordinateSystem]()[0]

                selectedCoordSys = Selection.Create(newselectedCoordSys)
                if newselectedCoordSys.Frame.Origin !=
i_trajectoryCoordSys.Frame.Origin:
                    options.Copy = False
                    result = Move.AlongTrajectory(selectedCoordSys,
selectedCurve, currentPosition, options)

        if(result):
            movedFrame = GetPositionOrientation(newselectedCoordSys,
i_trajectoryCoordSys)
            positionTable.append(movedFrame)
            timeTable.append(timeStamp)

        currentPosition += (convObjectSpeed / i_frameFrequency)/pathLength
        timeStamp +=  1/float(i_frameFrequency)

    result = Delete.Execute(selectedCoordSys)

    return timeTable, positionTable

# Length of path (m)
pathLength = trajectoryCurve.Shape.Length

# Get time stamps and relative coordinate systems
timeTable, positionTable = MoveAlongTrajectory(trajectoryCoordSys,
trajectoryCurve, isReversedTrajectory,  frameFrequency, objectSpeed)

dAxisSystemData_Table = []
for time in timeTable:
    timeIndex = timeTable.index(time)
    fFrame = positionTable[timeIndex]

    dAxisSystemData = GetAxisSystemData(time, fFrame)
    dAxisSystemData_Table.append(dAxisSystemData)

if len(dAxisSystemData_Table) > 0:
    dmTrajectory = OptisCore.DAxisSystemTrajectory()
    dmTrajectory.Trajectory.Resize(len(dAxisSystemData_Table))

    for dAxisSystemData in dAxisSystemData_Table:

dmTrajectory.Trajectory.Set(dAxisSystemData_Table.index(dAxisSystemData),
dAxisSystemData)

    pathTrajectoryFile = str(workingDirectory + "\\" +
```

```
str(ihTrajectoryName.Value) + ".json")
    strPathTrajectoryFile = OptisCore.String.From(pathTrajectoryFile)
    pathTrajectoryFile = OptisCore.Path(strPathTrajectoryFile)

    cAxisSystemTrajectoryWriter = OptisData.CAxisSystemTrajectoryWriter()
    cAxisSystemTrajectoryWriter.Open(pathTrajectoryFile)
    cAxisSystemTrajectoryWriter.Write(dmTrajectory)

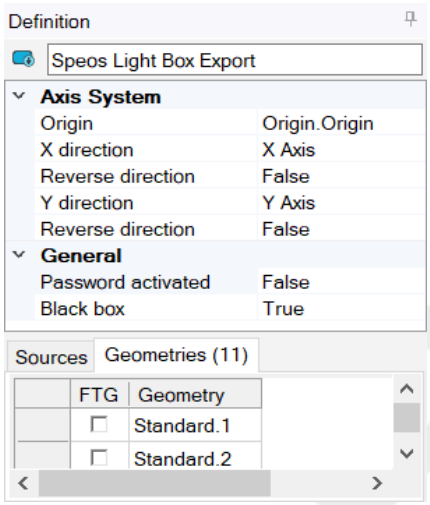
    cAxisSystemTrajectoryWriter.Close()
```

9.1.3. Exporting a Speos Light Box

This page shows how to export a Speos Light Box. Exporting a light box enables you to safely exchange data.

To export a Speos Light Box:

1. From the **Light Simulation** tab, click **Light Box** > **Export Speos Light Box** .



2. Rename the **Speos Light Box**.
3. If you want to set an **Axis System** (different from the axis system of the assembly), click one point for the origin point and two lines for X and Y directions or click and select a coordinate system to autofill the Axis System.

   Setting an axis system is optional but ensures that the exported component keeps the right orientation and origin point. If the axis system is kept empty, the reference is the origin of the main assembly.

   > **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

4. In the 3D view click to select the source(s) you want to include in the export.

   > **Note:** You cannot select a source included in a Speos Light Box Import.

DRAFT

Your selection appears in the **Sources** list.

5. In the 3D view click  to select the geometries you want to include in the export.

   Make sure a material is applied on all surface bodies that you want to include in a Speos Light Box Export.

   > **Note:** You cannot select a geometry included in a Speos Light Box Import.

   Your selection appears in the **Geometries** list. Optical Properties (VOP, SOP, FOP) related to the selected geometries are included in the export.

   > **Tip:** You can select elements from the 3D view or directly from the tree. To deselect an element, click back on it.

6. To edit the meshing properties of the exported geometries, right-click the light box from the **Simulation** panel and click **Options**.
7. Set the **Fast Transmission Gathering** to **True** if you want to neglect the light transmission with transparent surfaces.

   > **Note:** FTG does not apply to a 3D Texture, Polarization Plate and Speos Light Box Import.

8. If you want to encrypt the light box, set **Password activated** to **True**.
9. To define the password:

   > **Warning:** In version prior to 2023 R1, passwords were not hidden. As soon as you open a project containing passwords, they are hidden and you cannot retrieve them. Make sure to save your passwords in a safe place before opening your project in 2023 R1 or subsequent versions.

   - Enter a custom password manually.

     The password is directly hidden. Make sure to save it in a safe place as you will not be able to retrieve it from Speos afterwards.

   - Click in the **Password** field and click **Generate Password** to automatically generate a strong password.

     A prompt message appears to warn you that the password is directly hidden and copied into your clipboard. Click **OK** and make sure to save it in a safe place as you will not be able to retrieve it from Speos afterwards.

10. If you want to restrict the access to all the sources/geometries/optical properties details of the Speos Light Box and protect the intellectual property, check **Black box**.

The **Speos Light Box** is now created. A *.SPEOSLightBox file is exported in a subfolder located in **SPEOS output files** directory. The file dependencies are included in the *.SPEOSLightBox file.

> **Tip:** To include modifications done after the export, right-click the Speos Light Box export and click **Compute**  to regenerate the light box.

DRAFT

Related concepts

Speos Light Box Overview on page 306
The Speos Light box allows you to export and import geometries along with the associated optical data (sources, Optical Properties, meshing properties).

Related tasks

Importing a Speos Light Box on page 318
This page shows how to import .SPEOSLightBox, and .speos files into the CAD platform. Importing a light box enables you to retrieve exported geometries, sources and optical properties.

9.1.4. Importing a Speos Light Box

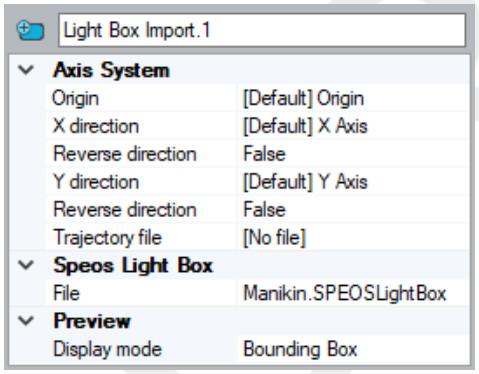
This page shows how to import .SPEOSLightBox, and .speos files into the CAD platform. Importing a light box enables you to retrieve exported geometries, sources and optical properties.

> **Note:** An imported light box cannot be modified.
>
> The imported elements, such as sources and geometries can be used in interactive, direct or inverse simulations only.

To import a Speos Light Box:

In the case of a *.speos import, make sure all dependencies are available.

1. From the **Light Simulation** tab, click **Light Box** > **Import Speos Light Box**  .



2. Set the **Axis System** by clicking one point for the origin and two lines for X and Y directions or click  and select a coordinate system to autofill the Axis System.

   The axis system is needed to correctly place the Speos Light Box into the scene according to the axis system of the assembly in which you import it.

   > **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

3. If you want to create a dynamic group of geometries, in the **Trajectory file** field, click **Browse** to load a trajectory file (.json).

   > **Note:** For more information on trajectory files, refer to Trajectory File on page 309.

   When a trajectory file is assigned to a Speos Light Box Import and the feature is edited, the trajectory is displayed in the 3D view.
4. Click in the **File** field and click **Browse** to load a Speos component (*.speos) or Speos Light Box (*.SPEOSLightBox) file.
5. If the exported Light Box is encrypted, a **Password** field appears.

   > **Warning:** In version prior to 2023 R1, passwords were not hidden. As soon as you open a project containing passwords, they are hidden and you cannot retrieve them. Make sure to save your passwords in a safe place before opening your project in 2023 R1 or subsequent versions.

6. Enter the password.
7. In the **Preview** section, define the **Display mode** of the imported Speos light box between **Bounding Box** and **Facet**.

   > **Note:** Each time you change the Display mode, you need to compute the Speos Light Box.

   > **Note:** In case of a black-boxed Speos Light Box, the **Display mode** is automatically set to **Bounding Box**.

8. Click **Compute**  to import the file in the current assembly.

   > **Note:** Once the import is completed, the components inherit the information defined during the export.
   >
   > If the axis system set for the Speos Light Box Import is edited, you must regenerate the import file manually to get the change displayed in the 3D view.

The Speos Light Box is imported in the current assembly. Sources and geometries are displayed in the 3D view according to the parameters set during the export or hidden in a bounding box and non-accessible in case of black-boxed light box.

Related concepts

Speos Light Box Overview on page 306
The Speos Light box allows you to export and import geometries along with the associated optical data (sources, Optical Properties, meshing properties).

Related tasks

Exporting a Speos Light Box on page 316
This page shows how to export a Speos Light Box. Exporting a light box enables you to safely exchange data.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

9.2. Optical Design Exchange

The Optical Design Exchange feature allows you to import an optical design system created in OpticStudio, and to use and analyze this optical design system through the Speos capabilities.

> **Note:** The Optical Design Exchange feature is in BETA mode for the current release.

9.2.1. Optical Design Exchange Overview

The following page provide you with an overview of the Optical Design Exchange to help you understand how to use it in the context of Speos.

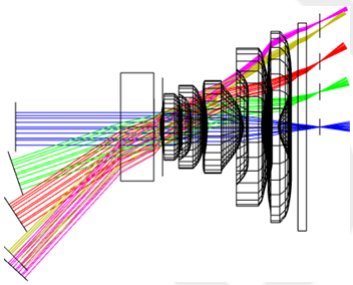
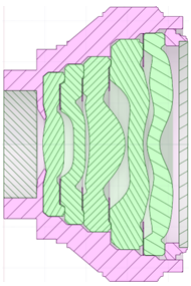
> **Note:** The Optical Design Exchange feature is in BETA mode for the current release.

Context

In the design of imaging systems, analyzing stray light and assessing image quality require to consider the lens geometries and the opto-mechanical elements that constrain them.

Ghost analysis is typically performed in Ansys Zemax OpticStudio during the design of the free-floating lenses. Then, the optical design created with Ansys Zemax OpticStudio can be imported into Speosto perform stray light analysis using the Speos capabilities:

- Using the CAD in which Speos is integrated you can add the opto-mechanical elements or modify the lenses to apply mechanical constraints.
- You can perform the stray light analysis while considering the opto-mechanical elements.

Stage 1	Stage 2	Stage 3
		
Optical design and optimization Ghost analysis in Ansys Zemax OpticStudio	Optical design import to Speos (geometries, optical properties and sensors)	Optomechanical design Stray light analysis of the complete system in a virtual scene

In Speos

Speos provides you with the Optical Design Exchange feature which allows you to import a complete optical lens design created in Ansys Zemax OpticStudio.

The Optical Design Exchange feature imports the following elements from Ansys Zemax OpticStudio:

- Geometries
- Optical properties
- Imager

A new component corresponding to the elements imported using the Optical Design Exchange feature is added to the SpaceClaim Structure tree:

- One geometry is created per body
- One origin is created per imager position.

A list of objects and surfaces corresponding to the Optical Properties definition and lens characteristics of each geometry imported is added as children of the Optical Design Exchange feature of the Speos Simulation tree.

Irradiance sensors are created in the Speos Simulation tree to correspond to the imagers of the system.

Characteristics

- The Optical Design Exchange feature can be included:
  - in a Light Box Export

    The Light Box can be black-boxed.
  - directly in a simulation as geometry.

Limitations

- The Optical Design Exchange feature (or a Light box that includes it) is not compatible with GPU simulations.
- Speos applies an Optical Polished surface optical property when no surface optical property (coating) is specified on a face in Ansys Zemax OpticStudio. This may lead to unwanted Speos simulation results. Make sure optical properties are applied in Ansys Zemax OpticStudio to avoid such issues.
- Speos may generate unexpected simulation results if integrated rays are outside of the wavelength range defined. Therefore, when defining sources and sensors in a simulation containing the imported optical design system, make sure to consider the spectral range of the Optical Design Exchange feature.

  This warning can be activated/deactivated.

Generic Workflow

Design Creation Stage

1. You create the optical design system using Ansys Zemax OpticStudio.

   When creating the system, you must apply surface properties to Annulus surface. We recommend you to use the "ABSORB" material. Otherwise Speos will not be able to read the surface correctly.

   The STOP surface has no optical property and is considered as absorbent.
2. You export the optical design system in a readable file for Speos (*.odx).

   The *.odx file includes the geometries of the system with their position and orientation, the position and orientation of the imager, the materials and coatings used in the system.

Import and Checking Stage

1. In Speos, you import and position the optical design system contained in the *.odx file using the Optical Design Exchange feature.

DRAFT

Ansys Zemax OpticStudio is not needed to import the *.odx file.

2. You compute the Optical Design Exchange feature.

   A list of elements are created in the Speos Simulation tree and in the SpaceClaim Structure tree.

3. You must verify that each SpaceClaim geometry (Structure tree) is correctly associated to their Speos Objects and Surfaces (Speos Simulation tree).
4. You must verify the optical properties of Speos Objects and Surfaces are associated to the correct geometry.

Design Use and Analysis Stage

Once the Optical Design Exchange feature is created and all the elements imported:

1. According to the goal of the project:
   a. You start designing the opto-mechanical parts, or, import opto-mechanical parts designed in other CAD platforms.

      -and/or-

   b. You modify the lenses geometry of the system to apply mechanical constraints.
2. You define a specific Meshing directly on the Optical Design Exchange feature.
3. You create a Simulation (Interactive, Direct, Inverse):
   a. Add Sources.
   b. Add the Irradiance Sensors imported from *.odx file and other sensors if necessary.
   c. Add the Optical Design Exchange feature to the Geometries list and other geometries if necessary.

      Adding the Optical Design Exchange feature only adds the geometries (modified or not) imported from the *.odx file. If opto-mechanical geometries have to be added, they need to be selected from the SpaceClaim Structure tree.

4. Once the simulation is defined, you run the simulation.
5. You analyze the result using the Speos tools and workflow (Light Expert, Stray Light Analysis, Virtual Photometric Lab, Virtual Human Vision Lab).

9.2.2. Understanding What Can Be Exported from Ansys Zemax OpticStudio

The following page provides you information on what can be exported from Ansys Zemax OpticStudio to a *.odx file, to be then imported in Speos.

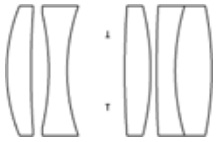
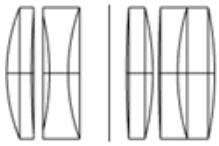
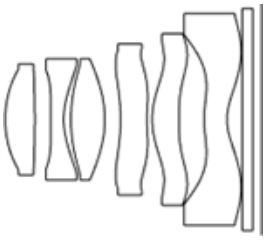
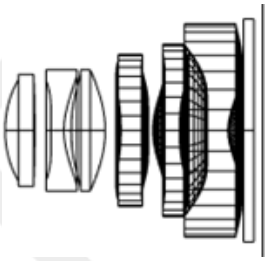
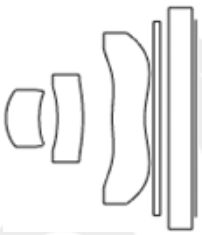
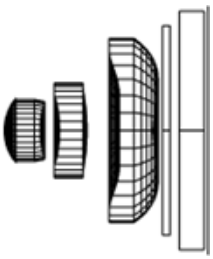
Geometries

In Ansys Zemax OpticStudio you need to define the geometries that will compose the optical system.

Two types of geometries can be used to define a lens in Speos:

- Sequential geometry: corresponds to an assembly of surfaces.
- Non-sequential geometry: corresponds to an object.

DRAFT

<table>
<tr><th colspan="2">Sequential Geometries</th><th colspan="2">Non-Sequential Geometries</th></tr>
<tr><td rowspan="3">• Even asphere
• Extended asphere
• Q-type asphere
• Standard
• Extended Odd Asphere
• Odd asphere
• STOP</td><td> 
Standard surfaces</td><td rowspan="3">• Annulus
• Even asphere lens
• Extended asphere lens
• Extended odd asphere lens
• Odd asphere lens
• Q-Type asphere lens surface
• Standard lens</td><td> 
Standard lenses</td></tr>
<tr><td> 
Standard, Even Asphere and Extended Asphere surfaces</td><td> 
Standard lenses, Even Asphere lens, Compound lenses made of Asphere surfaces</td></tr>
<tr><td> 
Standard, Q-Type Asphere surfaces</td><td> 
Standard lenses, Compound lenses made of Q-Type Asphere surfaces</td></tr>
</table>

Optical Properties

In Ansys Zemax OpticStudio you need to define the optical properties to be associated to the lenses of the optical system.

Material

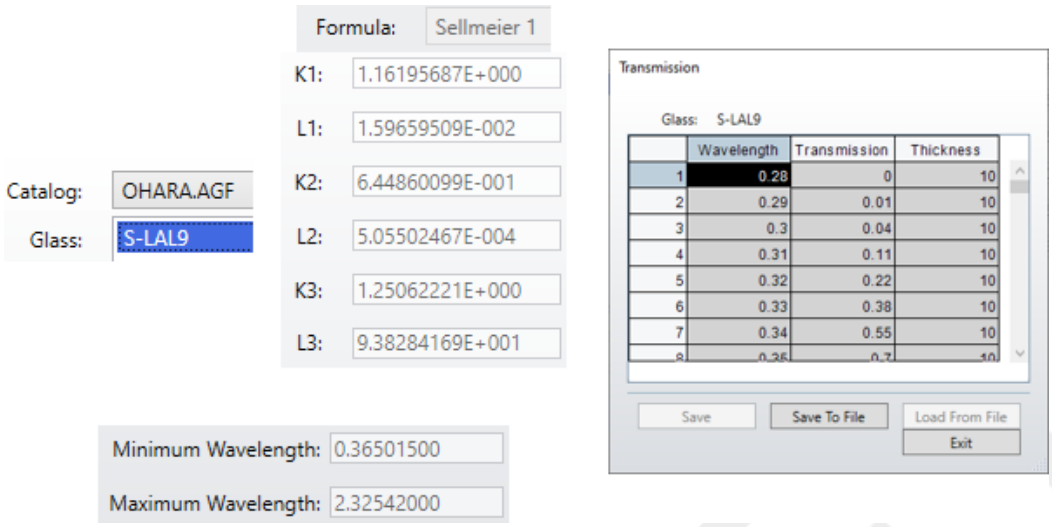
The Material corresponds to the Volume Optical Properties of the lens. Ansys Zemax OpticStudio provides an open catalog of glasses and their reference.

The following material information are exported in the *.odx file. After the *.odx file import in Speos, some information are displayed or not in the Speos Object information of the Optical Design Exchange feature:

- The **catalog** used and the **glass reference**. Displayed in Speos.
- The **dispersion formula** used. Not displayed in Speos.
- The related **coefficients**. Not displayed in Speos.
- The **minimum** and **maximum wavelengths**. Displayed in Speos.
- The **spectral absorption**. Not displayed in Speos.

DRAFT

Ansys Zemax OpticStudio defines the spectral transmission. During the *.odx import, Speos deduces the spectral absorption.

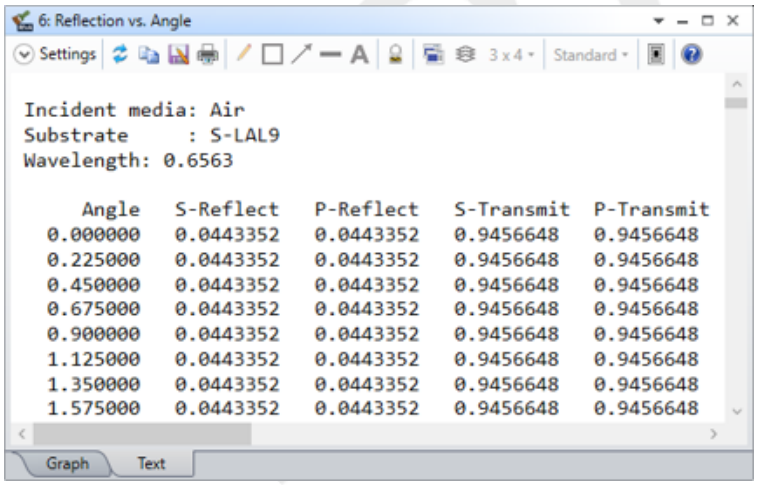


Coating

The Coating corresponds to the Surface Optical Properties of the lens.

The reflection and transmission as a function of incident angle and wavelength are exported in the *.odx file.

> **Important:** Coating data depend on the materials that define the diopter.



Sensors

In Ansys Zemax OpticStudio you need to define the sensors that will compose the optical system.

In case of sequential component, the sensors are defined using the size of an image surface.

In case of non-sequential component, the sensors are defined using the size and number of pixels of any Detector Rectange object.

9.2.3. Understanding the Optical Design Exchange Parameters

The following page provides you with the list of the different parameters of an Optical Design Exchange feature.

> **Note:** The Optical Design Exchange feature is in BETA mode for the current release.

Axis System

The **Axis System** of the Optical Design Exchange feature allows you to position and orientate the optical design system to be imported in the CAD scene.

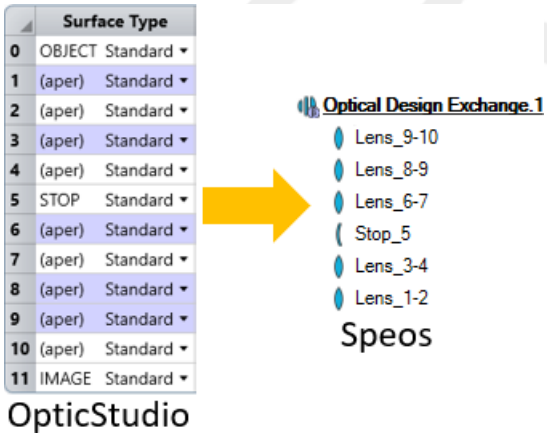
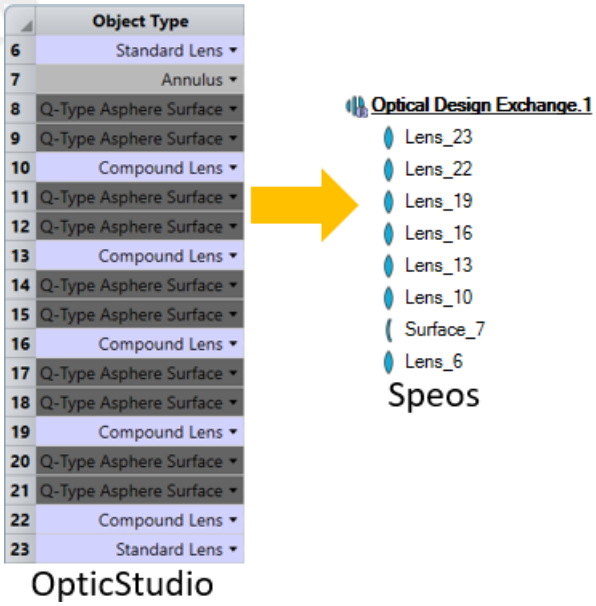
The Axis System does not interfer with the imager and geometry positions inside the optical design system.

Global - Geometries

Geometries is an informational and read-only parameter in the interface that counts the number of geometries imported after computing the Optical Design Exchange feature.

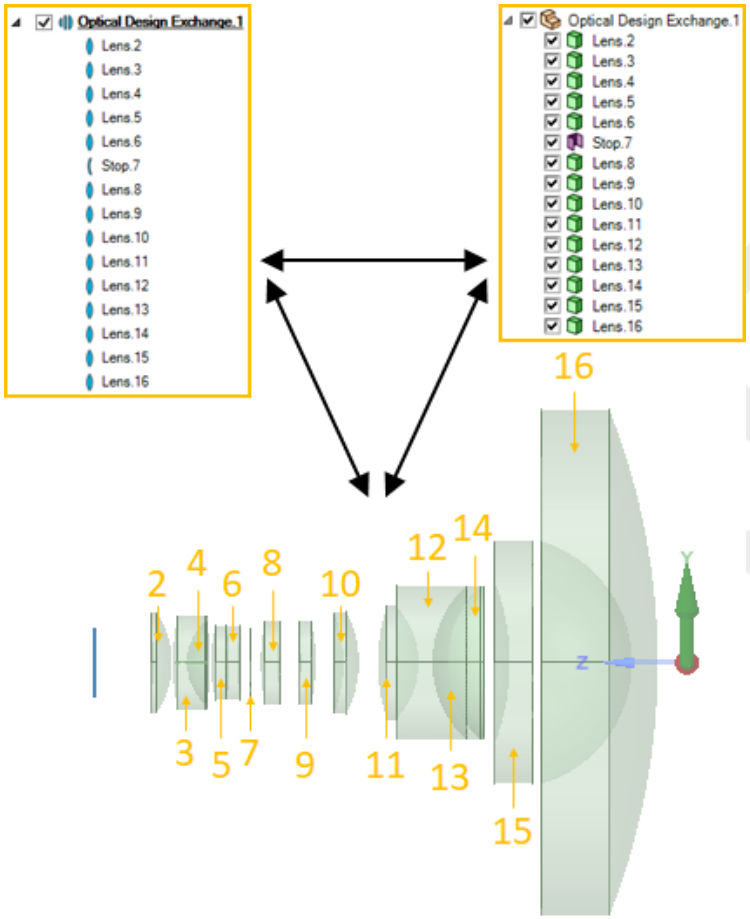
Object/Surface

Objects and Surfaces are the reflection of what has been exported from Ansys Zemax OpticStudio. They provide information on the definition of each imported lens according to how they have been defined in Ansys Zemax OpticStudio.

Sequential Component Export	Non-Sequention Component Export
Geometries indicate the type of object (lens or stop) and the corresponding line indexes from Ansys Zemax OpticStudio.	Geometries indicate the type of object (lens or surface) and the corresponding line index from Ansys Zemax OpticStudio.
	

General Characteristics

- Objects and Surfaces appear as children of the Optical Design Exchange feature node after computing the feature.
- All displayed parameters are read-only.
- All dimensions are expressed in millimeters.
- Wavelengths are expressed in nanometers.
- Infinite radius is expressed with a value of 0mm.
- The order of the imported elements is respected between the SpaceClaim Structure tree, the Speos Simulation tree and the 3D view.



Object

Objects are represented with this icon  . Each object correspond to a lens of the imported optical system.

As Speos element it includes:

- the Back Face and Front Faces dimensions
- the volume body dimension
- the Surface Optical Properties definition of the volume body faces. Surface Optical Properties appear only when a coating is applied to at least one face.
- the Volume Optical Properties definition of the volume body

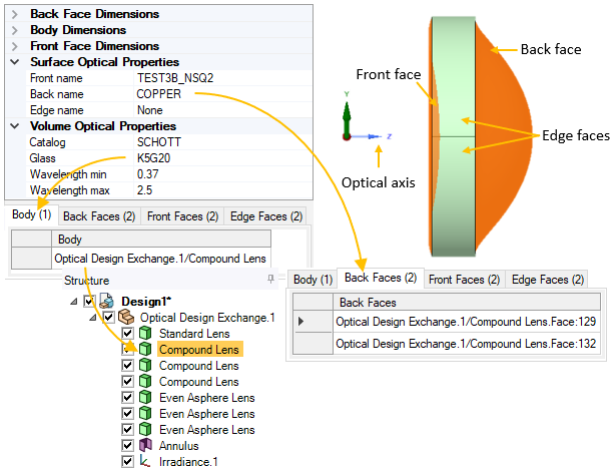
> **Important:** When defining sources and sensors in a simulation containing the imported optical design system, make sure to consider the spectral range of the Optical Design Exchange feature.

Back Face Dimensions	
Radius	7.38424
Conic	0
Norm radius	1
Max aperture X	0
Min aperture X	0
Max aperture Y	0
Min aperture Y	0
Decenter X	0
Decenter Y	0
Is rectangle	False
Clear	1.68422
Edge	1.7
Body Dimensions	
Thickness	0.5271
Front Face Dimensions	
Radius	1.51741
Conic	-1
Norm radius	1
Max aperture X	0
Min aperture X	0
Max aperture Y	0
Min aperture Y	0
Decenter X	0
Decenter Y	0
Is rectangle	False
Clear	1.58435
Edge	1.7
Surface Optical Properties	
Front name	TEST3B_NSQ2
Back name	COPPER
Edge name	None
Volume Optical Properties	
Catalog	SCHOTT
Glass	K5G20
Wavelength min	0.37
Wavelength max	2.5

- the link to the volume body in the SpaceClaim Structure tree. The Volume Optical Properties are applied to the body listed.
- the list of the different faces composing each volume body face (Back Faces, Front Faces, Edge Faces).

  Each list is associated to an optical property. That means, the optical property is applied on all elements of a same list.

  Example: The Surface Optical Property name COPPER is applied on all faces of the Back Faces list.

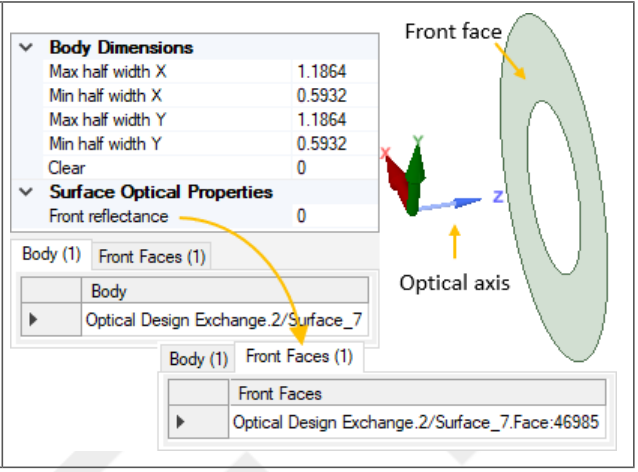


Surface

Surfaces are represented with this icon  . A surface correspond to an annulus/stop of the imported optical system.

The Annulus/Stop is the aperture in the system that limits the size of the beam transmitted through the system. The physical stop in the system usually joins the housing or other mechanical parts so that light cannot leak around the outside edge. The outer diameter of the annulus is set to 1.5 times the inner diameter.

As Speos element it includes:

• the body dimensions • the Surface Optical Properties definition of the face. It shows a reflectance of 0% applied on the surface. • the link to the surface body in the Structure tree • the link to the face composing the body	

Options

The Options correspond to specific settings of the Optical Design Exchange feature that are applied when running a simulation.

For more information on the Options, refer to Specific Simulation Options.

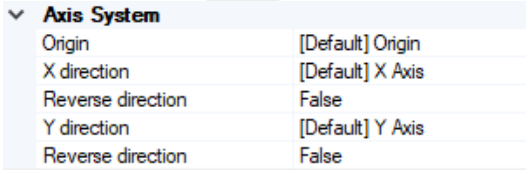
9.2.4. Importing an Optical Design System Created in Ansys Zemax OpticStudio

The following procedure helps you import an optical design system created in Ansys Zemax OpticStudio and set it in the Speos environment.

> **Note:** The Optical Design Exchange feature is in BETA mode for the current release.

To import the optical design system:

1. From the **Light Simulation** tab, in the **Component** click **Optical Design Exchange**  .
2. In the 3D view, define the **Axis system** of the Optical Design Exchange feature.

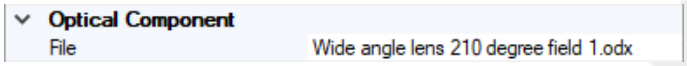


-  to select an origin point.

- 
  to select a line defining the X axis.
- 
  to select a line defining the Y axis.
- or click
  
  and select a coordinate system to autofill the Axis System.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

3. In the **Optical Component** section, from the **File** field, browse and select a *.odx file containing the optical design system exported from Ansys Zemax OpticStudio.



You do not need to have Ansys Zemax OpticStudio installed to import an *.odx file into Speos.

4. Click **Compute**  to build the feature.

At the end of the feature compute the optical design system appears in the 3D view, together with the list of the geometries in the SpaceClaim Structure tree, and the list of Objects and Surfaces containing the optical properties information and geometry characteristics in the Speos Simulation tree.

Verify that all imported elements are correctly imported and that geometries and Objects/Surfaces are correctly associated.

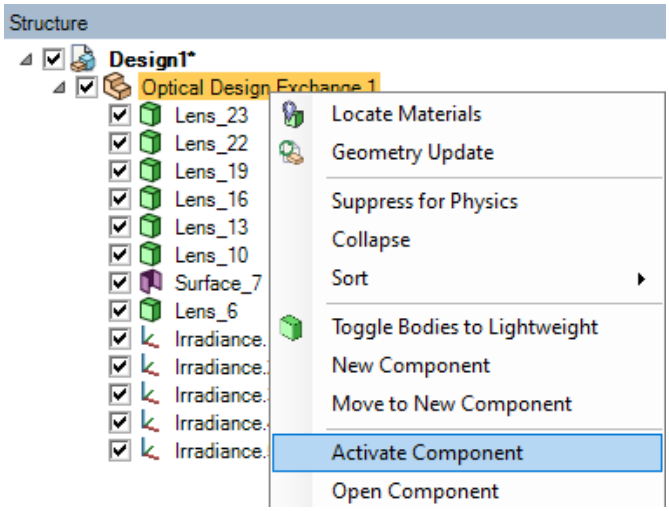
9.2.5. Modifying Imported Geometries

According to the final design expected, you may need to modify the lenses directly to apply mechanical constraints in your system. Then you will need to assign the new created faces/surfaces to a volume body face.

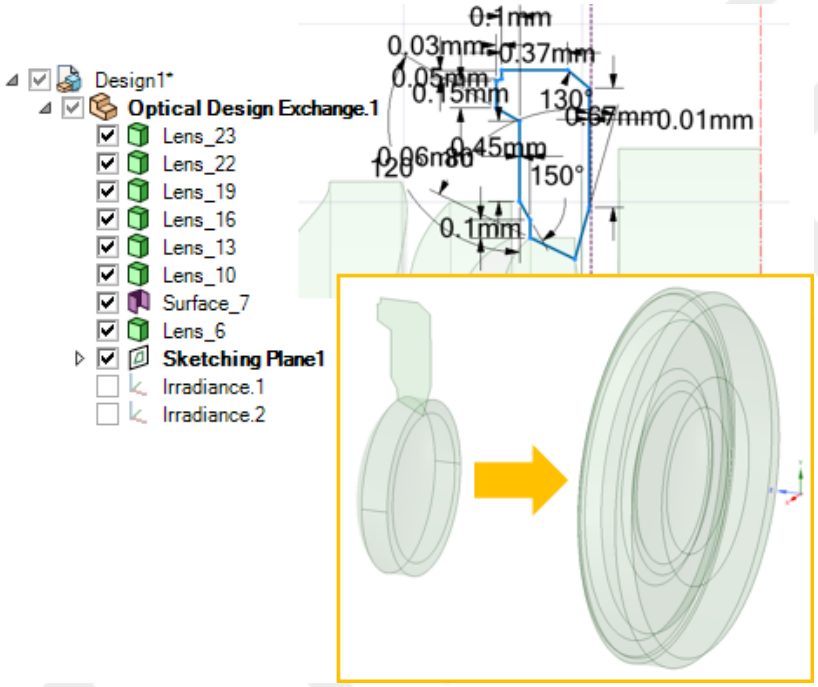
> **Note:** The Optical Design Exchange feature is in BETA mode for the current release.

1. In the **Structure** tree, activate the component corresponding to the Optical Design Exchange.

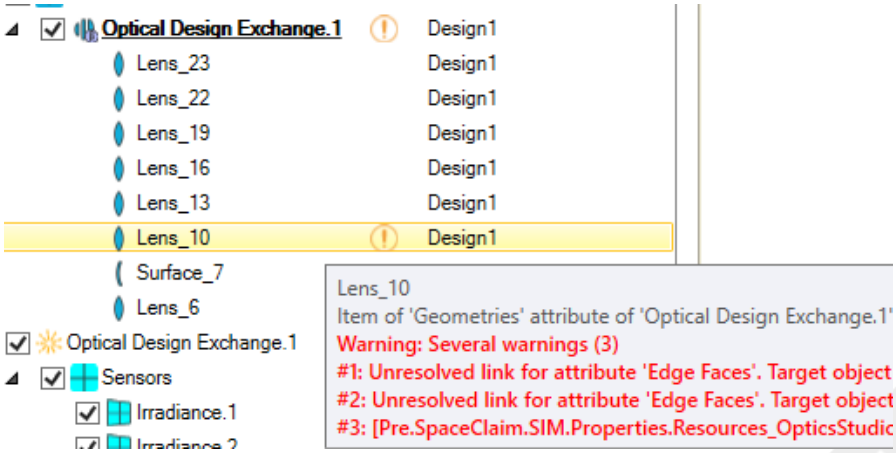
DRAFT



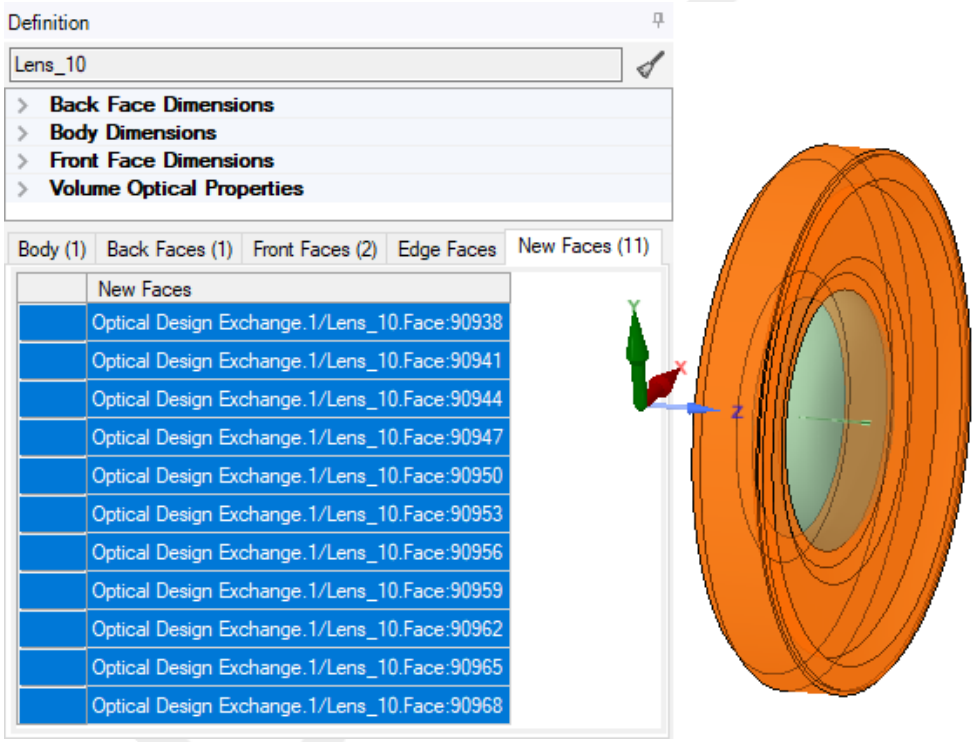
2. Modify the lens geometry (Structure tree) according to your needs using the SpaceClaim tools.



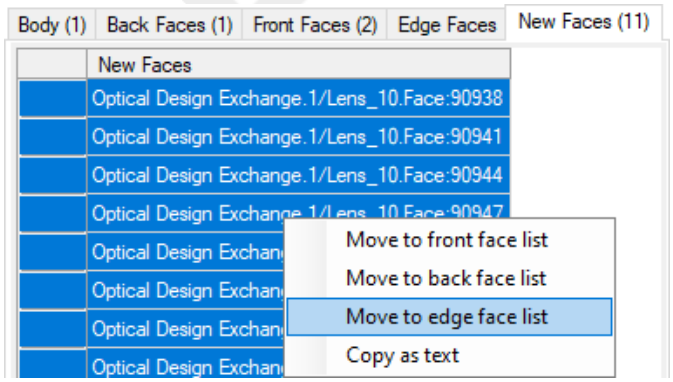
3. In the **Structure** tree, activate the parent component to be able to acces to the Speos Optical Design Exchange feature in the Simulation tree.
   The Optical Design Exchange feature automatically detects changes that happened on the body associated to the lens.



4. In the Speos Simulation tree, under the Optical Design Exchange feature, open the Object definition corresponding to the modified lens geometry.
   You can see a new list next to the body and faces lists. This list corresponds to the new added faces on the lens geometry that Speos has detected, which are not integrated yet in a dedicated list.



5. Add each new face in the appropriate list.



Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Once in the appropriate list, the new faces inherit the optical properties of the corresponding face (Back, Front or Edge).

6. Repeat from **Step 1** to **Step 5** for every lens geometry to be modified.

Now you can define a simulation with the Optical Design Feature feature and the opto-mechanical element if any, then analyze using the speos tools.

9.2.6. Specific Simulation Options

This section helps you set the simulation settings that will be specific to the Optical Design Exchange feature during a simulation.

> **Note:** The Optical Design Exchange feature is in BETA mode for the current release.

9.2.6.1. Setting the Specific Simulation Options

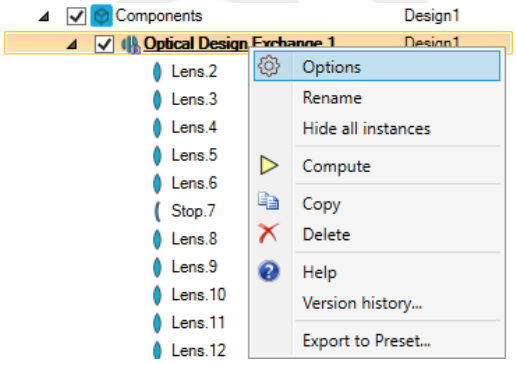
The following procedure helps you set the simulation settings that will be specific to the Optics Design Exchange feature during a simulation.

> **Note:** The Optical Design Exchange feature is in BETA mode for the current release.

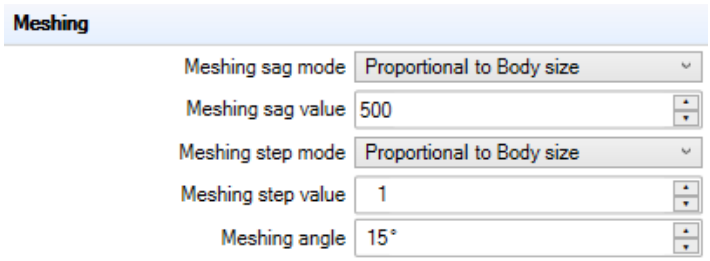
> **Note:** Texture options are not supported by the Optical Design Exchange Feature.

To set the simulation options:

1. From the **Simulation** tree, right-click the Optical Design Exchange feature.
2. Click **Options**.



3. In the **Meshing** section, define the Meshing properties of the geometries.



The Meshing allows you to mobilize and concentrate computing power on one or certain areas of a geometry to obtain a better level of detail in your results. With the meshing settings, you can lighten memory resources and accelerate simulation in specific cases.

For more information on the Meshing settings, refer to Understanding the Specific Simulation Options.

The simulation options are set for the Optical Design Exchange feature and will be taken into account in the simulation.

9.2.6.2. Understanding the Specific Simulation Options for an Optical Design Exchange

The following page describes the specific settings of the Optical Design Exchange feature to be applied when running a simulation containing the feature.

Note: The Optical Design Exchange feature is in BETA mode for the current release.

Note: Texture options are not supported by the Optical Design Exchange Feature.

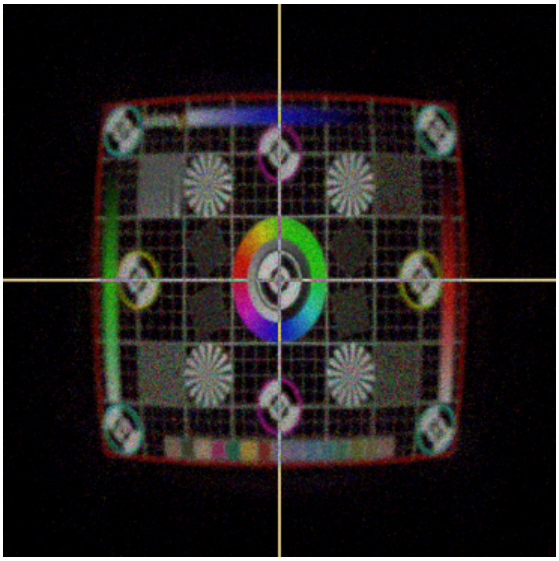
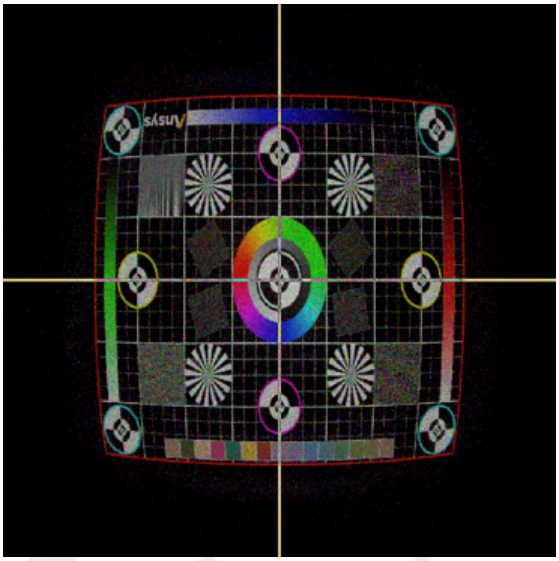
Meshing

With the meshing settings, you can lighten memory resources and accelerate simulation in specific cases. Meshing quality affects the simulation performances and the quality of the produced results, particularly when simulating optical systems.

Important: We recommend you to define at least the meshing parameters as follows. However, be aware that the meshing values need to be adapted to the optical system to be meshed.

- Meshing sag mode = Fixed
- Meshing sag value = 0.001 mm
- Meshing step mode = Fixed
- Meshing step value = 1 mm
- Meshing angle = 12°

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

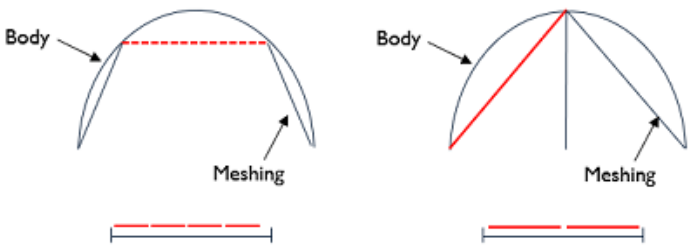
Default Meshing Definition	Fine Meshing Definition
	

- **Meshing Sag and Step Mode**
  - **Proportional to Face size**: create a mesh of triangles that are proportional to the size of each face of the object. The sag and step value therefore depend on the size of each face.
  - **Proportional to Body size**: create a mesh of triangles that are proportional to the size of the object. The sag and step value therefore depend on the size of the body.
  - **Fixed**: creates a mesh of triangles fixed in size regardless of the size of the body or faces. The mesh of triangles is forced on the object.
- **Meshing sag value**: defines the maximum distance between the object and the geometry.



> **Note:** If the **Meshing sag value** is too large compared to the body size, Speos recalculate with a **Meshing sag value 128** to better correspond to the body size.

- **Meshing step value**: defines the maximum length of a segment (in mm).



- **Meshing angle**: adjusts the angle in the mesh triangle.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

If you need more information, see Understanding Meshing Properties.

9.3. 3D Texture

3D Texture allows you to design and simulate millions of micro-patterns bypassing CAD system limitations.

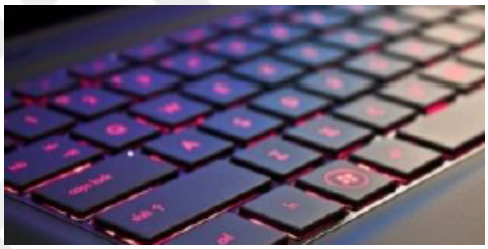
9.3.1. 3D Texture Overview

3D Texture allows you to simulate micro texture by modeling and projecting millions of geometrical items on a geometry.

3D Texture







Duplications of a geometrical item (**Patterns**) are projected on your geometrical base (**Support**) according to a specific distribution (**Mapping**) to simulate micro texture.

These patterns have volume and surface optical properties, and can be added, removed etc. (**Boolean Operation**).

As Speos is not able to model the patterns geometrically, the 3D Texture tool interest is to model them for an optical simulation without having to create millions of small geometries in the CAD model.

Uses and Applications

3D textures can be used to design lighting systems such as light guides, Brightness Enhancement Films (BEF) and back-lighting units that are composed of millions of geometrical elements.

Main Capabilities

- Patterns can be applied on any body.
- 3D Texture can be applied on any CAD shapes (flat or freeform, rectangular or not).

> **Note:** You cannot apply a 3D texture on an element having a surface that is tangent to another element.

> **Important:** 3D texture does not support patterns composed of a material using a custom made *.sop plug-in (surface optical properties).

- High generation capability (from hundreds to billions of elements).
- Low memory usage (about 150 Mb for 1 million patterns).

DRAFT

- 3D Texture is compatible with scripts and can be optimized through it.
- Driving capabilities of each element's position, orientation and scale.

In Speos:

- Create 3D Texture using a custom mapping file (a .txt file containing the coordinates of your patterns).
- Control every parameter of your pattern (scale factor, distribution, offset, shape etc.).

Related concepts

Boolean Operation on page 336
The following page lists all boolean operations available with the 3D Texture tool. The boolean operation is executed between the support and the pattern.

Related information

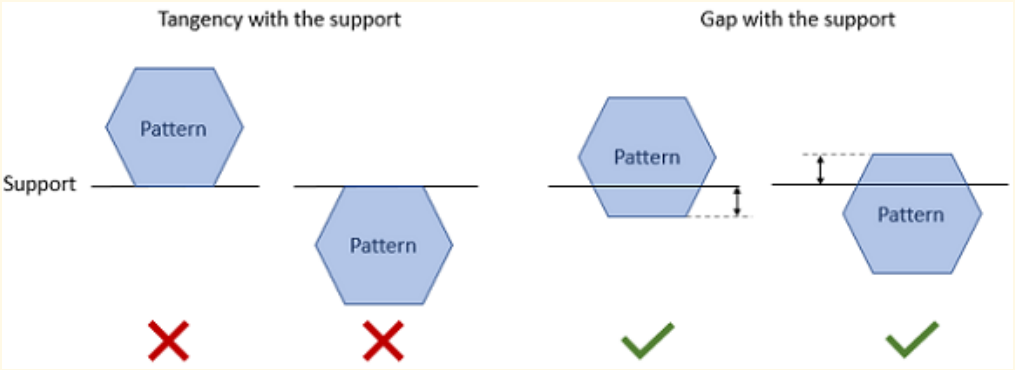
Creating a 3D Texture on page 342
This page shows the first steps of a 3D texture creation.

9.3.2. Boolean Operation

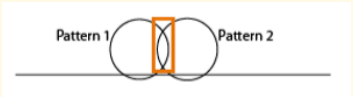
The following page lists all boolean operations available with the 3D Texture tool. The boolean operation is executed between the support and the pattern.

Note:

- You cannot set tangent surfaces between patterns and a support. A gap is needed and must be larger or equal to ten times the **Geometrical Distance Tolerance**.



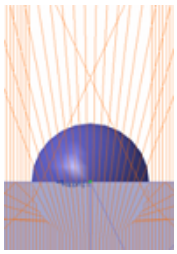
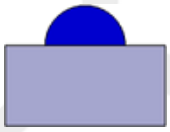
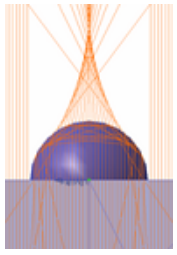
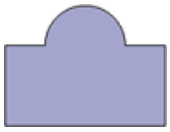
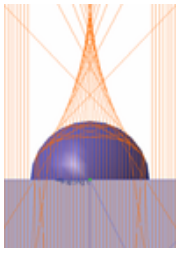
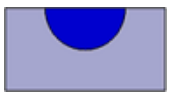
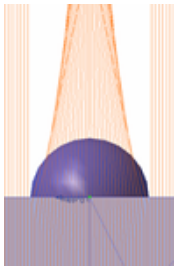
- You cannot set tangent surfaces between patterns. A gap is needed and must be larger or equal to ten times the **Geometrical Distance Tolerance**.
- Patterns cannot intersect.



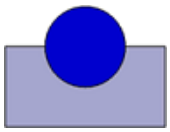
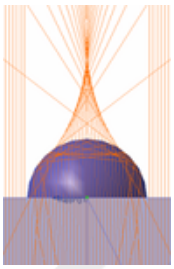
Compatibility between Boolean Operation and Type of Support

	Diffuse Support	Diffuse Pattern
Remove	✓	Not applicable
Add on Same Material	✗	✗
Add on Different Material	✗	✓
Add In	✓	✓
Insert	✓	✗

Operations

	Actual behavior of the pattern	Preview of the rays distribution to visualize pattern behavior
Remove		
Add on different material		
Add on same material		
Add In		

DRAFT

	Actual behavior of the pattern	Preview of the rays distribution to visualize pattern behavior
Insert		

Note: Set the Geometrical Distance Tolerance to G / 100 in the assembly preferences (ex: if G=1e^{-5} then Geometrical Distance Tolerance=1e^{-7}). This gives fewer errors in the propagation of the photons. Also note that the texture width cannot be larger than the material width.

Related information

Creating a 3D Texture on page 342
This page shows the first steps of a 3D texture creation.

9.3.3. Mapping File

The Mapping file basically defines the position, orientation and scale of each elements in the texture over the support. The mapping file is a text file format with an *.OPT3DMapping extension.

Mapping Parameters

The Mapping file describes the number of patterns used in a 3D texture, their coordinates, orientation and scale values with respect to the axis system.

The *.OPT3DMapping file is built according to the following structure:

- First line: number of patterns to be built in the texture.
- Following lines: **x y z ix iy iz jx jy jz kx ky kz**
  - **x y z**: coordinates of the pattern's origin according to the axis system.
  - **ix iy iz**: orientation of the pattern with the respect of X direction of the axis system.
  - **jx jy jz**: orientation of the pattern with the respect of Y direction of the axis system.
  - **kx ky kz**: pattern scale values for x y and z directions.

A value of "1" means 100% of the original pattern size.

DRAFT

Figure 49. Parameters to set for each pattern

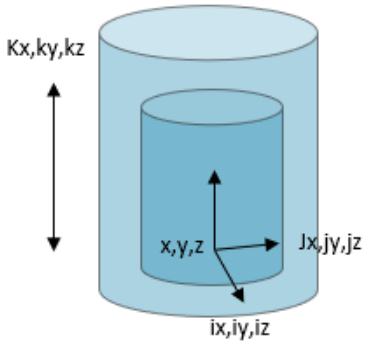
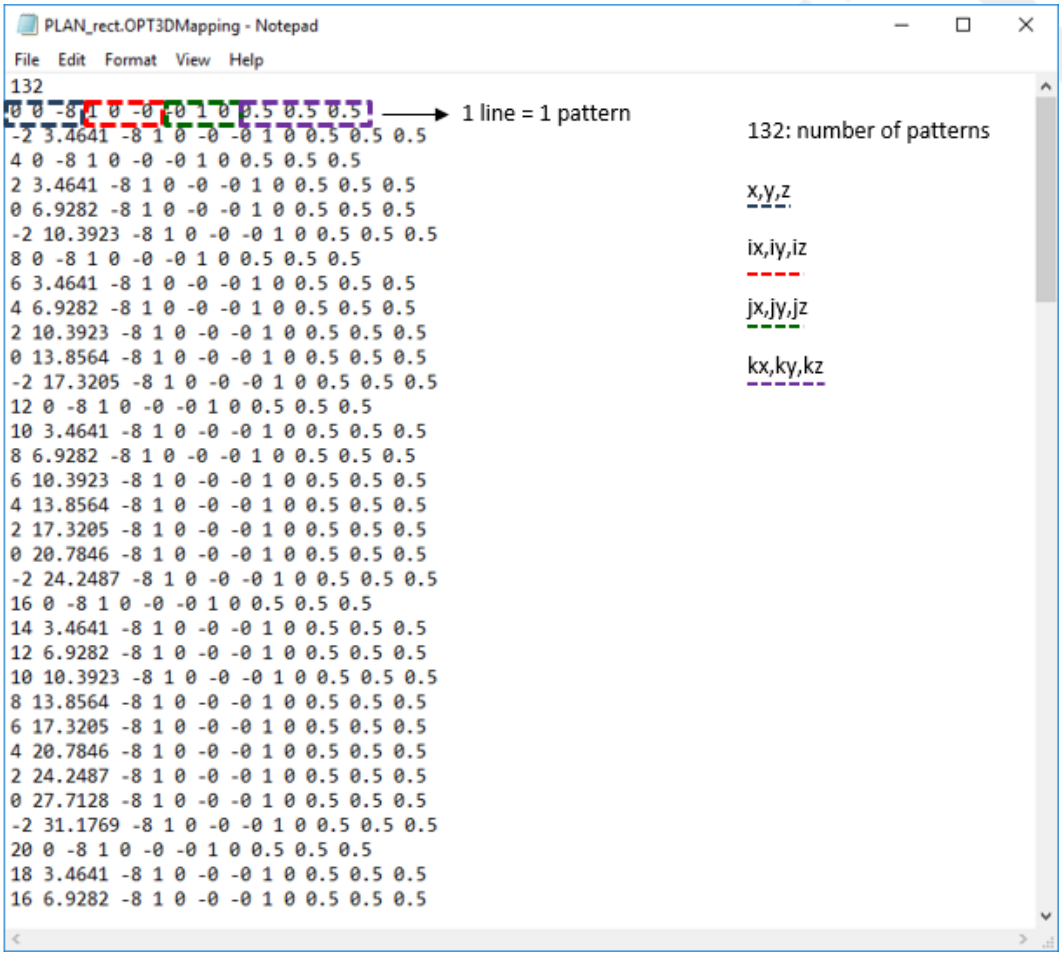


Figure 50. Mapping file example



Related concepts

Scale Factors on page 340

A pattern can have a global scale factor and 3 independent pattern scale factors for X,Y and Z. A scale ratio is used to model small textures (1/100 mm as an example).

9.3.4. Scale Factors

A pattern can have a global scale factor and 3 independent pattern scale factors for X,Y and Z. A scale ratio is used to model small textures (1/100 mm as an example).

> **Note:** The pattern scale factors are cumulative. Final pattern scale factor = global scale factor * pattern scale factor.

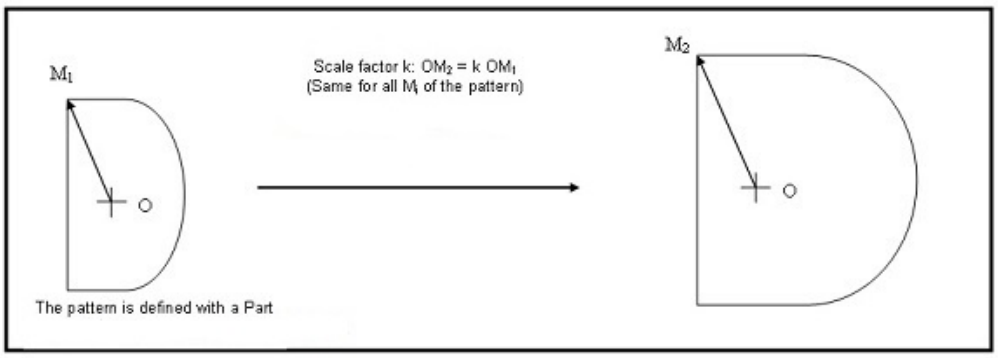
Global Scale Factor

The global scale factor is a I scale factor and is cumulative to each pattern scale factor.

A global scale factor of "1" means 100% of the original pattern size.

The value must be different from 0.

1 Scale Factor



Then, the **.txt** mapping file is:

- x1, y1, z1, i1x, i1y, i1z, j1x, j1y, j1z, k1
- x2, y2, z2, i2x, i2y, i2z, j2x, j2y, j2z, k2
- x3, y3, z3, i3x, i3y, i3z, j3x, j3y, j3z, k3 ...

3 Scale Factors

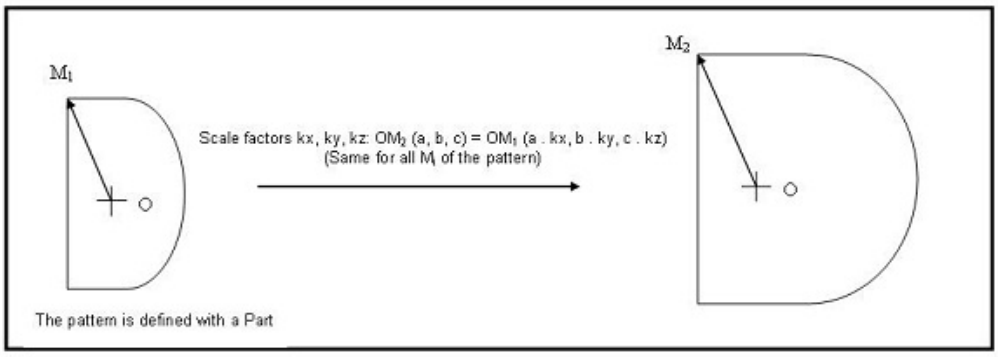
Each pattern can have three scale factors. These factors are used to set the size of each pattern independently on the 3 axes (X, Y, Z).

Figure 51. Pattern scale factor (x1,y1,z1) (x2, y1, z1) (x1, y1, z2)



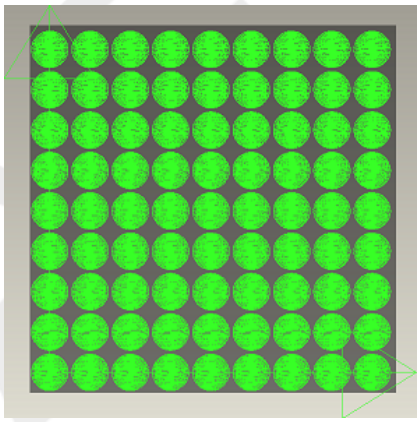
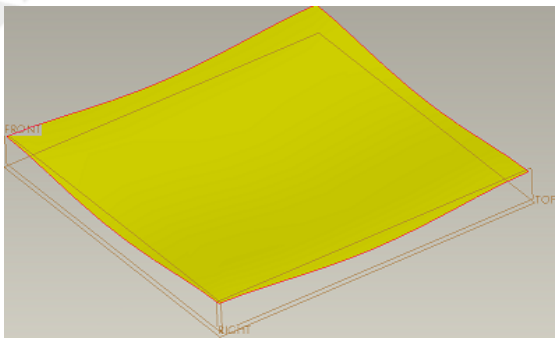
The following scheme describes the modification of the geometry of a pattern.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

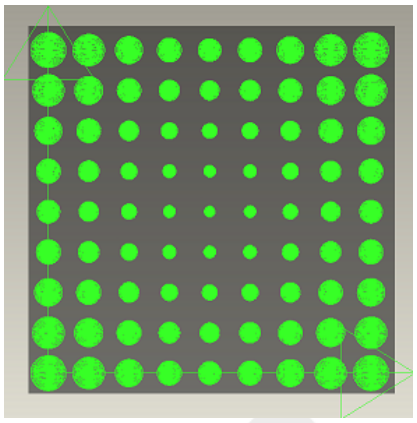


Then the **.txt** mapping file is:

- x1, y1, z1, i1x, i1y, i1z, j1x, j1y, j1z, k1x, k1y, k1z
- x2, y2, z2, i2x, i2y, i2z, j2x, j2y, j2z, k2x, k2y, k2z
- x3, y3, z3, i3x, i3y, i3z, j3x, j3y, j3z, k3x, k3y, k3z

3D texture mapping without variable scale	
Variable scale	

DRAFT

3D texture mapping with variable scale	

Related information

Mapping File on page 338
The Mapping file basically defines the position, orientation and scale of each elements in the texture over the support. The mapping file is a text file format with an *.OPT3DMapping extension.

Creating a 3D Texture on page 342
This page shows the first steps of a 3D texture creation.

9.3.5. Creating a 3D Texture

This page shows the first steps of a 3D texture creation.

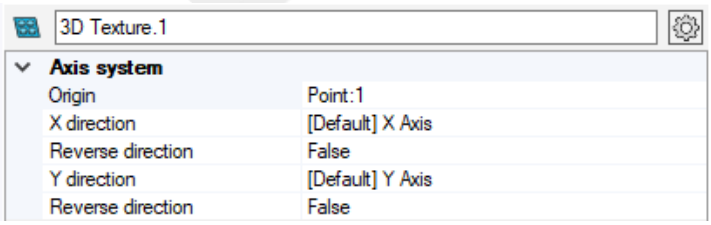
To create a 3D Texture:

A pattern must already be created.

The pattern is a geometrical item (a part) that can be duplicated to generate a mapping. A pattern can have its own optical properties and scaling factors.

> **Tip:** If you import a pattern from another CAD software, save the geometry as an .scdoc file to be able to use in Speos 3D texture.

1. From the **Light Simulation** tab, click **3DTexture** .



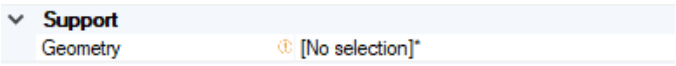
2. Rename the feature.

3. Set the Axis system by clicking a point for the origin point and two lines for X and Y directions or click  and select a coordinate system to autofill the Axis System.

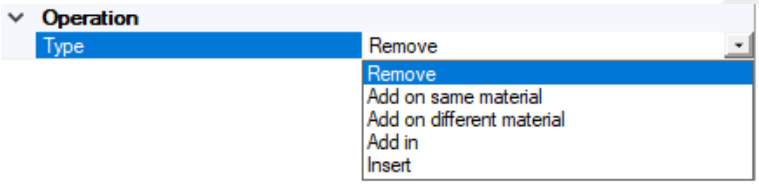
   The Axis System determines where does the mapping begins and in what direction it propagates. It is used as a reference for the position, orientation and scaling of each pattern.

   > **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

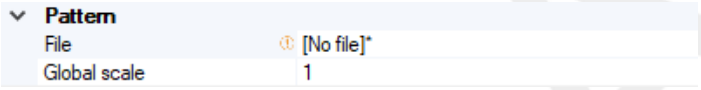
4. In the 3D view, click to select the **Support** to pattern.



5. In **Operation**, from the **Type** drop-down list, select the boolean operation to be executed on the patterns.



6. In **Pattern**, click in the file field and click **Browse** to load a pattern part.



   > **Note:** The pattern file is not compatible with the assembly external component. That means you cannot retrieve a file that references another file containing the pattern geometry.

   > **Important:** 3D texture does not support patterns composed of a material using a custom made *.sop plug-in (surface optical properties).

7. Define the Global scale of the pattern.

   Now set your Mapping type: Rectangular , Circular , Hexagonal , Variable Pitches , From File .

Related concepts

Scale Factors on page 340
A pattern can have a global scale factor and 3 independent pattern scale factors for X,Y and Z. A scale ratio is used to model small textures (1/100 mm as an example).

Related information

3D Texture Overview on page 335
3D Texture allows you to simulate micro texture by modeling and projecting millions of geometrical items on a geometry.

Mapping File on page 338

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

The Mapping file basically defines the position, orientation and scale of each elements in the texture over the support. The mapping file is a text file format with an *.OPT3DMapping extension.

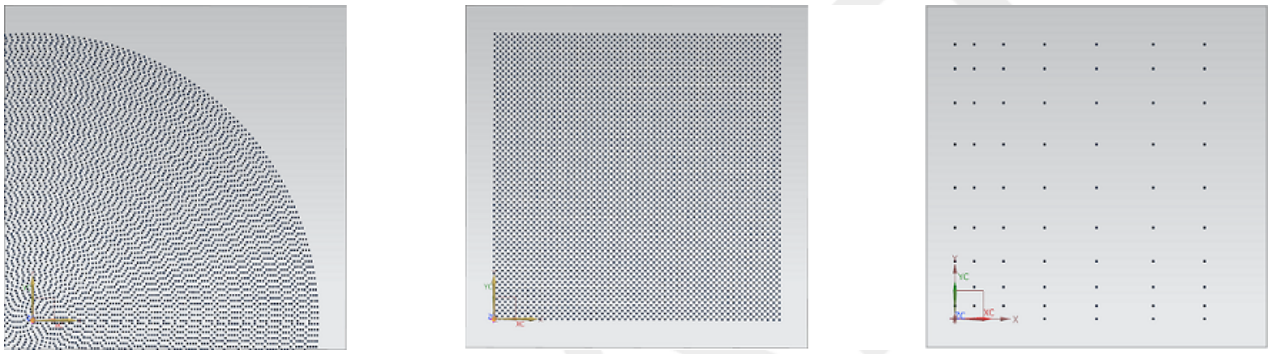
9.3.6. Mapping

The type of mapping you use determines the patterns' distribution on the support.

9.3.6.1. Understanding Mapping

This page describes the different types of mappings available and the mapping process.

The mapping determines the distribution of the patterns on the support.

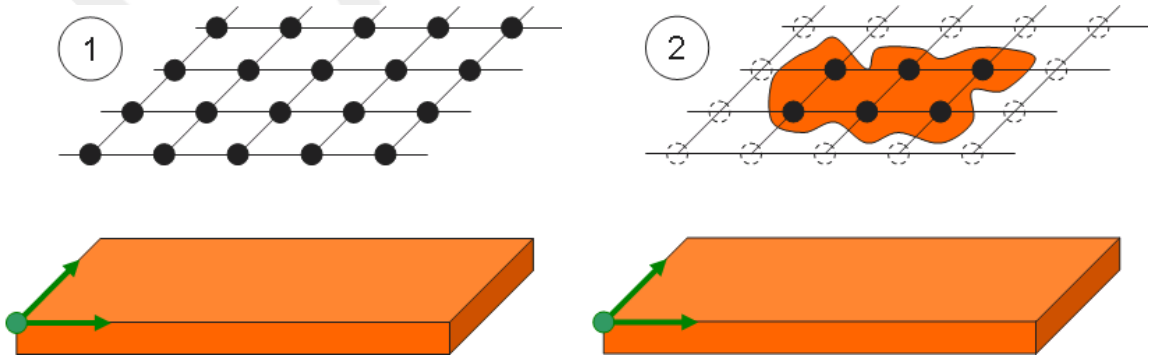


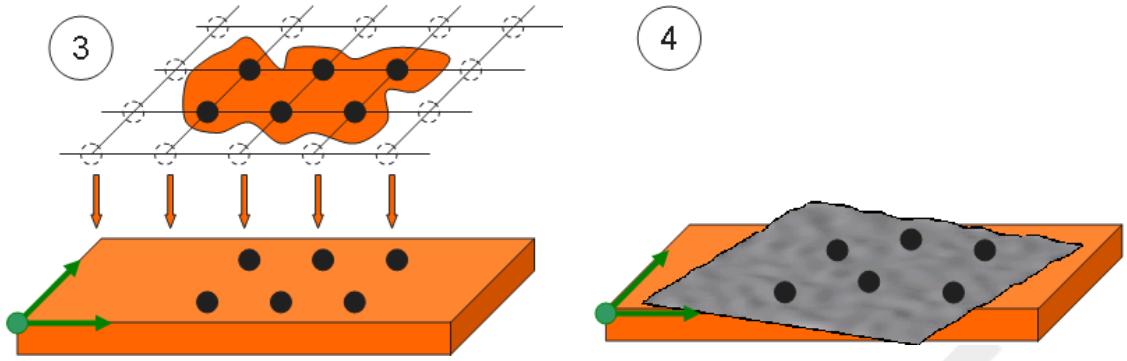
Different mapping types are available. Each type describes a way to create a virtual grid that is going to be projected on a part's surface.

When creating a 3D texture you can choose to use:

- Automatic mappings: Rectangular, Circular, Hexagonal, Variable Pitches. These mappings are automatically calculated and you only have to set a few parameters to obtain the desired distribution.
- Mapping Files : Mapping files are .txt files containing all the information needed to generate the mapping. It allows you to generate a completely customized mapping and/or save a mapping you designed with the automatic mappings and use it in another CAD system where Ansys software is integrated.

Example of the mapping process using various mapping options:





1. Mapping: a virtual grid is created using standard parameters (distance between patterns, mapping length, etc.).
2. Filtering: a quilt or a face is used to define the grid limitations.
3. All the patterns included in the limited grid are projected along the Z direction on the first encountered surface of the selected part.
4. Shift: an offset (shift along Z) can be applied on the projected patterns.

Limiting Surface

The **Limiting Surface** allows you to apply a surface on the geometry on which you apply the 3D texture to limit the 3D texture to that specific surface.

Important: The Limiting Surface must be composed of only one face. If the Limiting Surface is multi-faces, the 3D Texture will be applied on only one face.

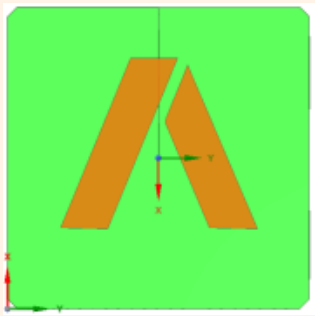
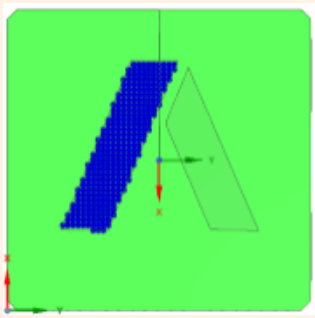
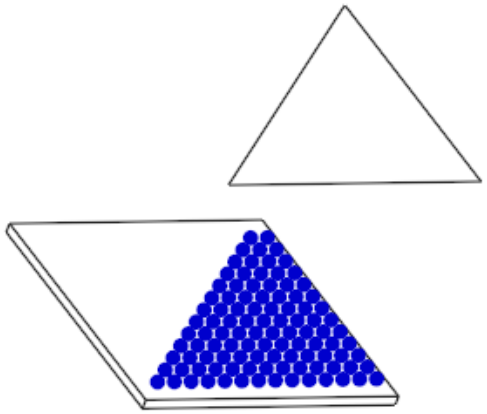
	
Limiting Surface composed of two faces	3D Texture applied on only one face

Figure 52. 3D texture filtering



Offset Surface

The **Offset Surface** is a shift surface that allows you to apply an offset to the projected patterns along the Z direction according to the origin of the 3D texture's axis system.

The **Shift scale** helps you define the offset.

Related information

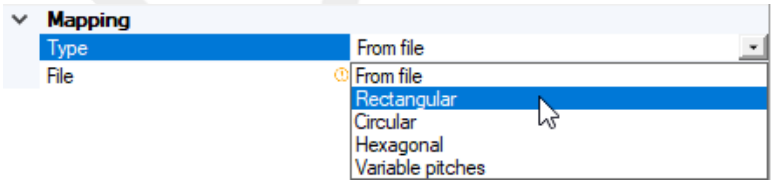
Creating a 3D Texture on page 342
This page shows the first steps of a 3D texture creation.

3D Texture Overview on page 335
3D Texture allows you to simulate micro texture by modeling and projecting millions of geometrical items on a geometry.

9.3.6.2. Creating a Rectangular Mapping

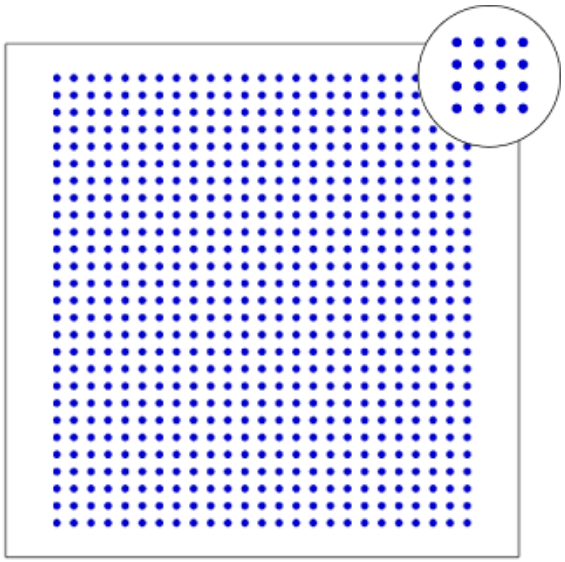
A 3D texture must be created . A pattern and support to pattern must already be selected. An axis system must be set.

1. From the **Mapping Type** drop-down list, select **Rectangular**.



2. Set the following parameters:

DRAFT

Distance between patterns along X		
Distance between patterns along Y		
Mapping Length along X		
Mapping length along Y		
X angular offset		
Y angular offset		

3. If you want to limit the mapping to a specific area, shape:

Limiting surface	[No selection]

   a) If not already done, import or create a face in the current assembly.

   b) In the 3D view, click  and select the imported/created object to limit the 3D texture distribution to that specific face.

4. If you want to define an offset along Z direction on the projected patterns:

Offset surface	[No selection]
Offset surface ratio	1

   a) In the 3D view, click  and select the support on which to apply the offset.

   b) Type a value in **Offset surface ratio** to determine the offset of your patterns.

5. In the **Pattern** section, define the patterns' orientation.

6. If you want to define three scale factors to set the size of each pattern independently on the 3 axes (X, Y, Z):

X scale surface	[No selection]
X scale	1
Y scale surface	[No selection]
Y scale	1
Z scale surface	[No selection]
Z scale	1

   a) Click  to select an **X scale surface**,  to select a **Y scale surface** and  to select an **Z scale surface**.

   The scaling factor to apply to a specific mapping point is defined by the height of the point of this surface, at the corresponding coordinates (X,Y).

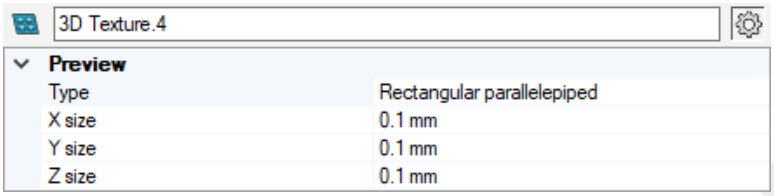
   > **Note:** You can define a specific scale on all three independent axes (X,Y,Z axes) or on one axis only (only on X for example).

   b) Define a pattern scale value.

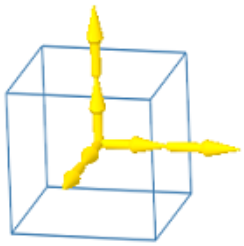
DRAFT

The scale value is applied to all the patterns of a direction. The pattern scale factors are cumulative to the global scale factor (global scale factor * pattern scale factor = final pattern scale factor).

7. Click **Compute** to verify the 3D texture distribution.
   A progress bar appears and allows you to stop the compute if necessary. At the end of the compute, the Dots appear on the support indicating the patterns' location on the surface.
8. To fully compute and visualize the patterns on the support:
   a) Click **Optional or advanced settings** .



   b) Adjust the **X size**, **Y size** and **Z size** of the preview box to see it appear in the 3D view.



   c) Using the 3D view manipulators, drag the box onto the 3D texture support to compute the patterns.

The rectangular mapping is created and an OPT3D Mapping file is automatically generated and stored in the SPEOS inputs files folder.

Related information

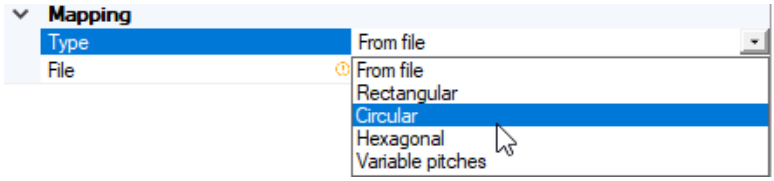
Creating a 3D Texture on page 342
This page shows the first steps of a 3D texture creation.

3D Texture Overview on page 335
3D Texture allows you to simulate micro texture by modeling and projecting millions of geometrical items on a geometry.

9.3.6.3. Creating a Circular Mapping

A 3D texture must be created . A pattern and support to pattern must already be selected. An axis system must be set.

1. From the **Mapping Type** drop-down list, select **Circular**.



2. Set the following parameters:

DRAFT

Radial distance between patterns



Mapping radius

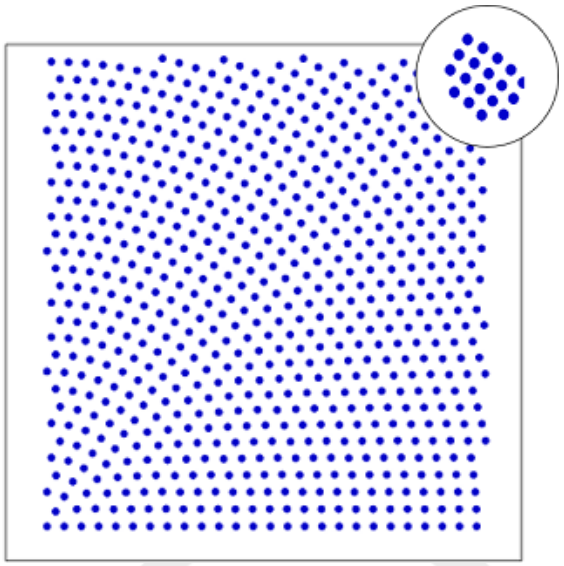


Distance between rings



X angular offset





3. If you want to limit the mapping to a specific area, shape:

Limiting surface	[No selection]

   a) If not already done, import or create a face in the current assembly.

   b) In the 3D view, click  and select the imported/created object to limit the 3D texture distribution to that specific face.

4. If you want to define an offset along Z direction on the projected patterns:

Offset surface	[No selection]
Offset surface ratio	1

   a) In the 3D view, click  and select the support on which to apply the offset.

   b) Type a value in **Offset surface ratio** to determine the offset of your patterns.

5. In the **Pattern** section, define the patterns' orientation.

6. If you want to define three scale factors to set the size of each pattern independently on the 3 axes (X, Y, Z):

X scale surface	[No selection]
X scale	1
Y scale surface	[No selection]
Y scale	1
Z scale surface	[No selection]
Z scale	1

   a) Click  to select an **X scale surface**,  to select a **Y scale surface** and  to select an **Z scale surface**.

   The scaling factor to apply to a specific mapping point is defined by the height of the point of this surface, at the corresponding coordinates (X,Y).

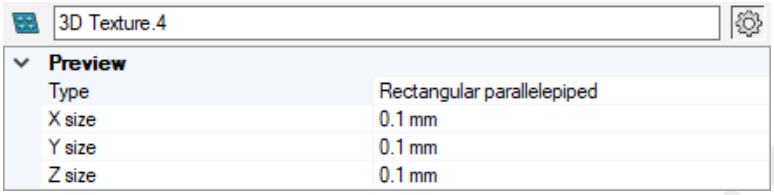
> **Note:** You can define a specific scale on all three independent axes (X,Y,Z axes) or on one axis only (only on X for example).

   b) Define a pattern scale value.

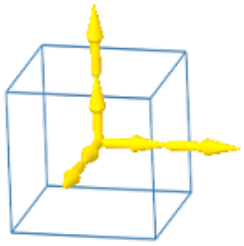
The scale value is applied to all the patterns of a direction. The pattern scale factors are cumulative to the global scale factor (global scale factor * pattern scale factor = final pattern scale factor).

7. Click **Compute** to verify the 3D texture distribution.
   A progress bar appears and allows you to stop the compute if necessary. At the end of the compute, the Dots appear on the support indicating the patterns' location on the surface.
8. To fully compute and visualize the patterns on the support:

   a) Click **Optional or advanced settings** .



   b) Adjust the **X size**, **Y size** and **Z size** of the preview box to see it appear in the 3D view.



   c) Using the 3D view manipulators, drag the box onto the 3D texture support to compute the patterns.

The ciruclar mapping is created and an OPT3D Mapping file is automatically generated and stored in the Speos inputs files folder.

Related information

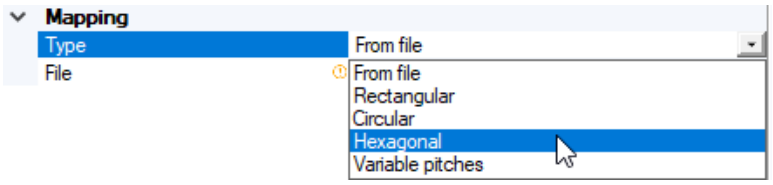
Creating a 3D Texture on page 342
This page shows the first steps of a 3D texture creation.

3D Texture Overview on page 335
3D Texture allows you to simulate micro texture by modeling and projecting millions of geometrical items on a geometry.

9.3.6.4. Creating a Hexagonal Mapping

A 3D texture must be created . A pattern and support to pattern must already be selected. An axis system must be set.

1. From the **Mapping Type** drop-down list, select **Hexagonal**.



2. Set the following parameters:

Hexagon length along X

Hexagon edge length along X

Distance between hexagon along X

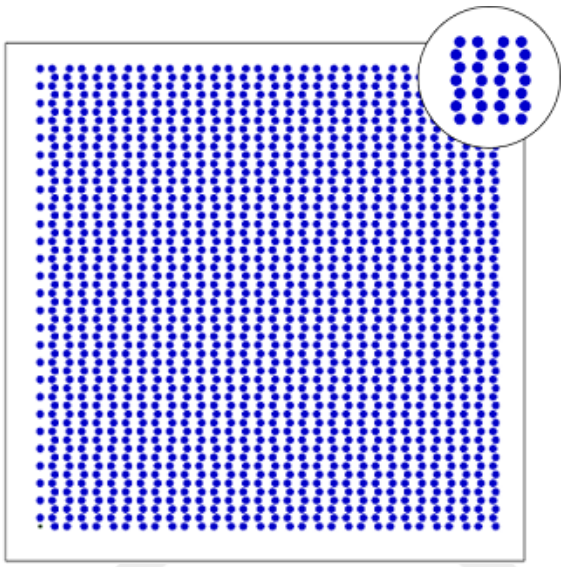
Hexagon length along Y

Mapping length along X

Mapping length along Y

X angular offset

Y angular offset



3. If you want to limit the mapping to a specific area, shape:

Limiting surface	[No selection]

   a) If not already done, import or create a face in the current assembly.

   b) In the 3D view, click and select the imported/created object to limit the 3D texture distribution to that specific face.

4. If you want to define an offset along Z direction on the projected patterns:

Offset surface	[No selection]
Offset surface ratio	1

   a) In the 3D view, click and select the support on which to apply the offset.

   b) Type a value in **Offset surface ratio** to determine the offset of your patterns.

5. In the **Pattern** section, define the patterns' orientation.

6. If you want to define three scale factors to set the size of each pattern independently on the 3 axes (X, Y, Z):

X scale surface	[No selection]
X scale	1
Y scale surface	[No selection]
Y scale	1
Z scale surface	[No selection]
Z scale	1

   a) Click to select an **X scale surface**, to select a **Y scale surface** and to select an **Z scale surface**.

   The scaling factor to apply to a specific mapping point is defined by the height of the point of this surface, at the corresponding coordinates (X,Y).

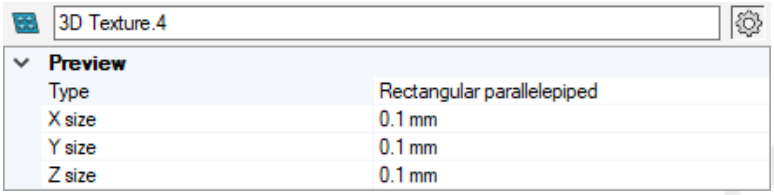
   > **Note:** You can define a specific scale on all three independent axes (X,Y,Z axes) or on one axis only (only on X for example).

   b) Define a pattern scale value.

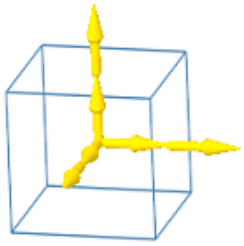
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

The scale value is applied to all the patterns of a direction. The pattern scale factors are cumulative to the global scale factor (global scale factor * pattern scale factor = final pattern scale factor).

7. Click **Compute** to verify the 3D texture distribution.
   A progress bar appears and allows you to stop the compute if necessary. At the end of the compute, the Dots appear on the support indicating the patterns' location on the surface.
8. To fully compute and visualize the patterns on the support:
   a) Click **Optional or advanced settings** .



   b) Adjust the **X size**, **Y size** and **Z size** of the preview box to see it appear in the 3D view.



   c) Using the 3D view manipulators, drag the box onto the 3D texture support to compute the patterns.

The hexagonal mapping is created and an OPT3D Mapping file is automatically generated and stored in the SPEOS inputs files folder.

Related information

Creating a 3D Texture on page 342
This page shows the first steps of a 3D texture creation.

3D Texture Overview on page 335
3D Texture allows you to simulate micro texture by modeling and projecting millions of geometrical items on a geometry.

9.3.6.5. Variable Pitches Mapping

Creating a variable pitches mapping allows you to control and customize the distribution of the patterns on the support.

9.3.6.5.1. Understanding Variable Pitches Mapping

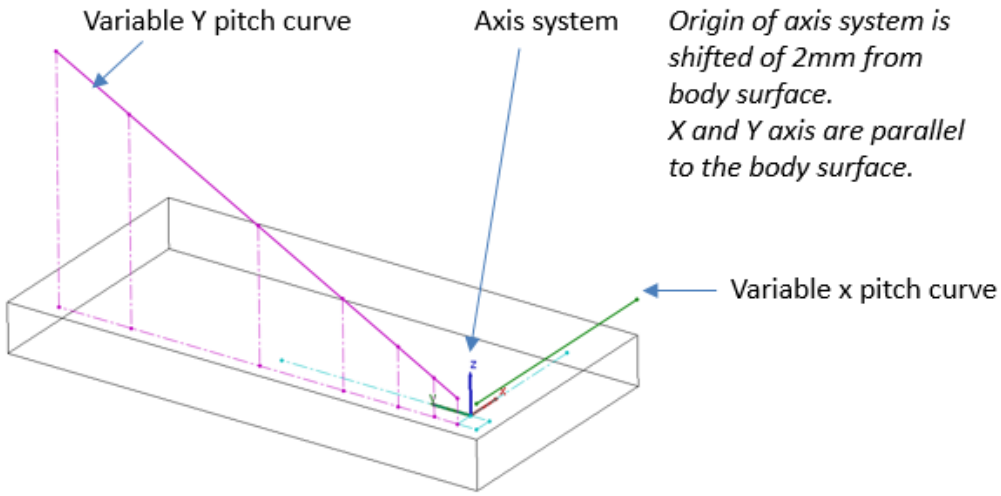
This page describes how to create variable pitches curves for a good calculation.

The Variable Pitch Mapping uses two curves sketched in the CAD environment (one along the X axis, one along the Y axis). These curves will be used to determine the distribution of the mapping along both axes.

DRAFT

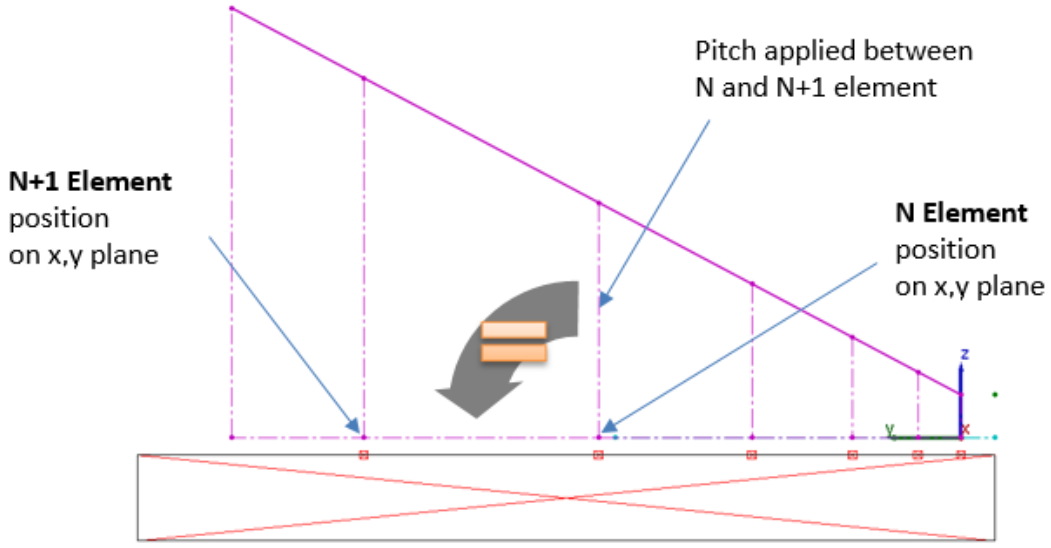
Positioning and Calculation

For a correct construction of the 3D Texture, position the start point of the curve defining a variable pitch on the side of the origin of the axis system:



The 3D Texture algorithm reference is the axis system. It defines a projection direction (Z axis) used to:

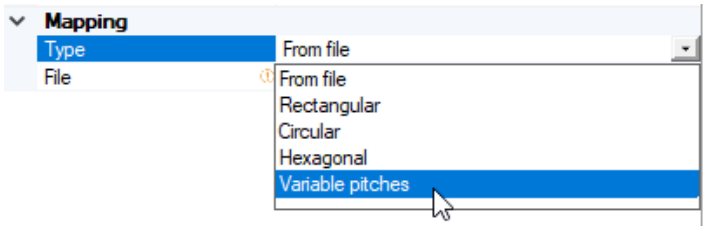
- Position new element: the position is first computed in the plan, then projected on the body along projection direction.
- Calculate distance between two variable pitches: the distance between two elements (**N** and **N+1**) is the distance between the curve and the position of the first element (**N**) in the axis plane along projection direction.



9.3.6.5.2. Creating a Variable Pitches Mapping

A 3D texture must be created . A pattern and support to pattern must already be selected. An axis system must be set.

1. From the **Mapping Type** drop-down list, select **Variable Pitches**.



2. If not already done, draw two curves or lines that will determine the distribution of the mapping.
3. In the 3D view, click to select a pitch curve along X and to select a pitch curve along Y.

> **Note:** X pitch curve must cut the yOz plane of the 3D texture. Y pitch curve must cut the xOz plane of the 3D texture.

4. To define the spacing of the patterns along X and/or Y, set the pitch ratio of each curve.
5. Set the following parameters:

X pitch curve ratio

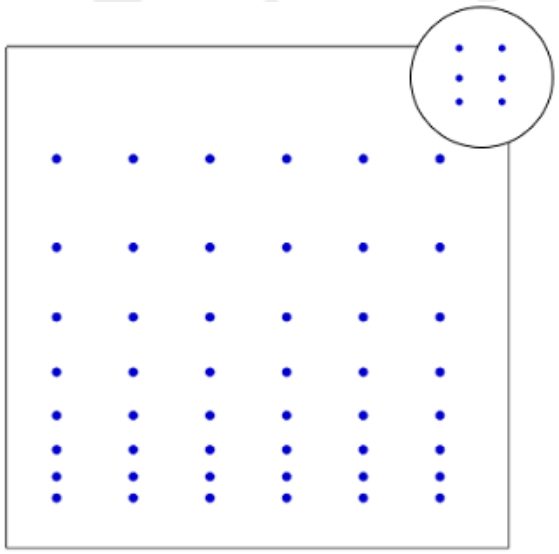
Y pitch curve ratio

Mapping length along X

Mapping length along Y

X angular offset

Y angular offset



6. If you want to limit the mapping to a specific area, shape:

Limiting surface	[No selection]

   a) If not already done, import or create a face in the current assembly.
   b) In the 3D view, click and select the imported/created object to limit the 3D texture distribution to that specific face.

7. If you want to define an offset along Z direction on the projected patterns:

Offset surface	[No selection]
Offset surface ratio	1

   a) In the 3D view, click and select the support on which to apply the offset.
   b) Type a value in **Offset surface ratio** to determine the offset of your patterns.

8. In the **Pattern** section, define the patterns' orientation.
9. If you want to define three scale factors to set the size of each pattern independently on the 3 axes (X, Y, Z):

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

X scale surface	[No selection]
X scale	1
Y scale surface	[No selection]
Y scale	1
Z scale surface	[No selection]
Z scale	1

a) Click  to select an **X scale surface**,  to select a **Y scale surface** and  to select an **Z scale surface**.

The scaling factor to apply to a specific mapping point is defined by the height of the point of this surface, at the corresponding coordinates (X,Y).

> **Note:** You can define a specific scale on all three independent axes (X,Y,Z axes) or on one axis only (only on X for example).

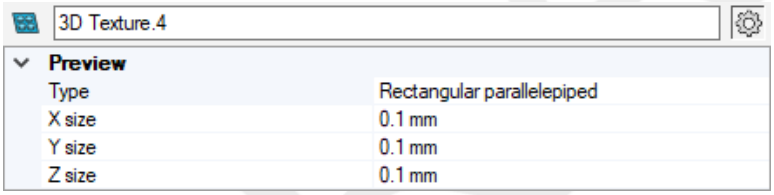
b) Define a pattern scale value.

The scale value is applied to all the patterns of a direction. The pattern scale factors are cumulative to the global scale factor (global scale factor * pattern scale factor = final pattern scale factor).

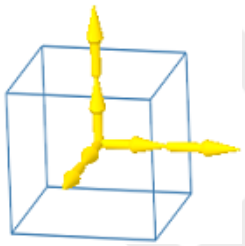
10. Click **Compute**  to verify the 3D texture distribution.
A progress bar appears and allows you to stop the compute if necessary. At the end of the compute, the Dots appear on the support indicating the patterns' location on the surface.

11. To fully compute and visualize the patterns on the support:

a) Click **Optional or advanced settings**  .



b) Adjust the **X size**, **Y size** and **Z size** of the preview box to see it appear in the 3D view.



c) Using the 3D view manipulators, drag the box onto the 3D texture support to compute the patterns.

The variable pitches mapping is created and an OPT3D Mapping file is automatically generated and stored in the SPEOS inputs files folder.

Related information

Creating a 3D Texture on page 342
This page shows the first steps of a 3D texture creation.

3D Texture Overview on page 335

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

3D Texture allows you to simulate micro texture by modeling and projecting millions of geometrical items on a geometry.

9.3.6.6. Using Mapping Files

Mapping files are .txt files containing the coordinates of the patterns.

9.3.6.6.1. Mapping File

The Mapping file basically defines the position, orientation and scale of each elements in the texture over the support. The mapping file is a text file format with an *.OPT3DMapping extension.

Mapping Parameters

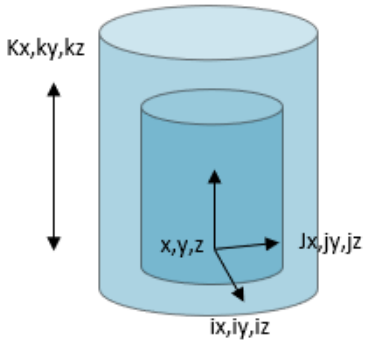
The Mapping file describes the number of patterns used in a 3D texture, their coordinates, orientation and scale values with respect to the axis system.

The *.OPT3DMapping file is built according to the following structure:

- First line: number of patterns to be built in the texture.
- Following lines: **x y z ix iy iz jx jy jz kx ky kz**
  - **x y z**: coordinates of the pattern's origin according to the axis system.
  - **ix iy iz**: orientation of the pattern with the respect of X direction of the axis system.
  - **jx jy jz**: orientation of the pattern with the respect of Y direction of the axis system.
  - **kx ky kz**: pattern scale values for x y and z directions.

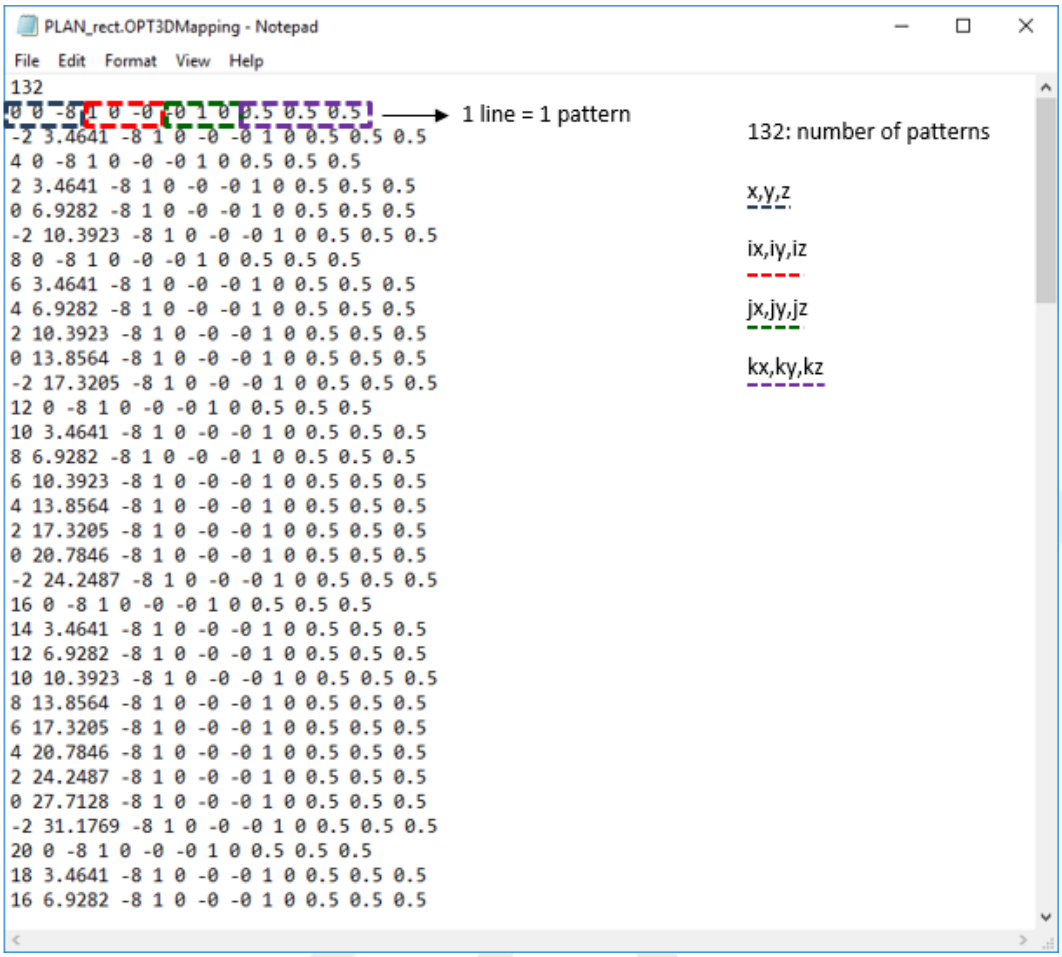
A value of "1" means 100% of the original pattern size.

Figure 53. Parameters to set for each pattern



RAFT

Figure 54. Mapping file example



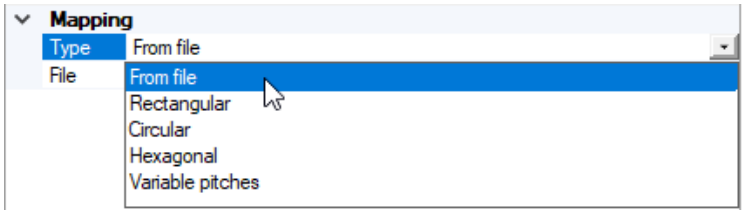
Related concepts

Scale Factors on page 340
A pattern can have a global scale factor and 3 independent pattern scale factors for X,Y and Z. A scale ratio is used to model small textures (1/100 mm as an example).

9.3.6.6.2. Creating a Mapping from a File

A 3D texture must be created . A pattern and support to pattern must already be selected. An axis system must be set.

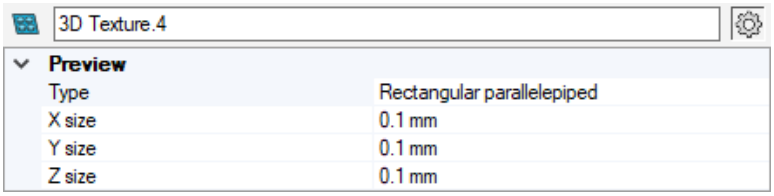
1. From the **Mapping Type** drop-down list, select **From file**.



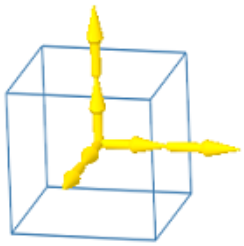
2. Double-click in the file field to browse to load an .OPT3DMapping file.

3. Click **Compute** to verify the 3D texture distribution.
   A progress bar appears and allows you to stop the compute if necessary. At the end of the compute, the Dots appear on the support indicating the patterns' location on the surface.
4. To fully compute and visualize the patterns on the support:

   a) Click **Optional or advanced settings** .



   b) Adjust the **X size**, **Y size** and **Z size** of the preview box to see it appear in the 3D view.



   c) Using the 3D view manipulators, drag the box onto the 3D texture support to compute the patterns.

The mapping is created. If you want to modify the 3D texture distribution, open and edit the OPT3DMapping (.txt) file and re-import it to update the modifications.

Related information

Mapping File on page 338
The Mapping file basically defines the position, orientation and scale of each elements in the texture over the support. The mapping file is a text file format with an *.OPT3DMapping extension.

Creating a 3D Texture on page 342
This page shows the first steps of a 3D texture creation.

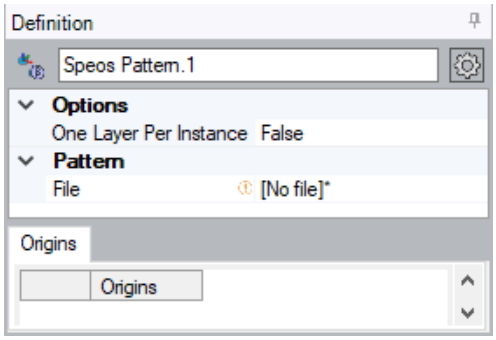
9.4. Creating a Speos Pattern

The Speos Pattern allows you to create and orientate multiple instances of a same source file (Ray file source *.ray, *.tm25ray or Speos Lightbox *.SPEOSLightBox) so that you do not have to create the same source multiple times in Speos and overload the Speos tree and so its readability.

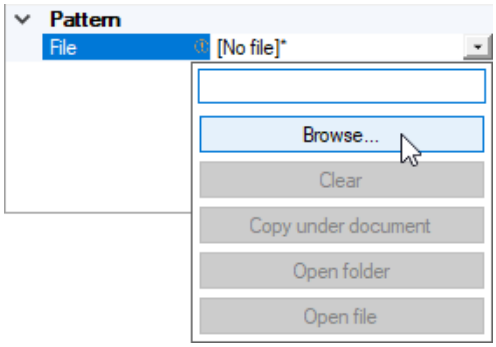
To create a Speos Pattern:

> **Note:** We recommend you to create first the origins on which you want to position the instances of the source pattern.

1. From the **Light Simulation** tab, in the **Component** section, click **Speos Pattern** .

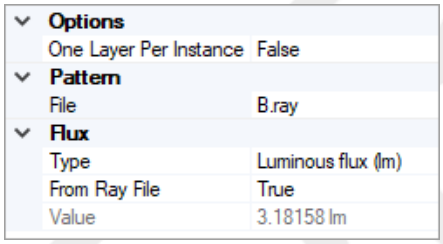


2. In the **Pattern** section, browse and select a Ray File source (*.ray *.tm25ray) or a Speos Light Box file (*.SPEOSLightBox) from which you want to create multiple instances.



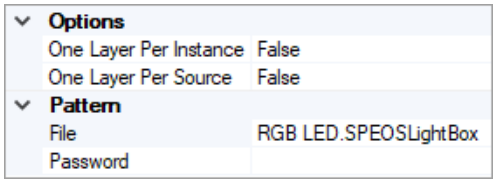
> **Note:** When using a black-boxed light box in a Speos pattern, the bounding box of the light box can appear larger and sometimes incorrectly oriented in the 3D view.

3. If you selected a Ray File Source as pattern:



a) Define the **Type** of the flux between **Luminous flux (lm)** and **Radiant flux (W)**.
b) Define the flux value of the instances:
   - Set **From Ray File** to **True** to use the flux from the ray file.
   - Set **From Ray File** to **False** and manually define a value for a custom flux.
c) In **One Layer Per Instance**, define if you want to separate the data per instance in layers in the XMP result. One layer will represent one instance in the result:
   - Set to **True** if you want to create one layer for each instance in the XMP result.
   - Set to **False** if you want to create only one layer for all instances in the XMP result.

4. If you selected a Speos Light Box as pattern:



a) If the file is encrypted, enter the password to use it.

> **Warning:** In version prior to 2023 R1, passwords were not hidden. As soon as you open a project containing passwords, they are hidden and you cannot retrieve them. Make sure to save your passwords in a safe place before opening your project in 2023 R1 or subsequent versions.

b) In **One Layer Per Instance** define if you want to separate the data per instance in layers in the XMP result. One layer will represent one instance in the result:

- Set to **True** if you want to create one layer for each instance in the XMP result.
- Set to **False** if you want to create only one layer for all instances in the XMP result.

c) In **One Layer Per Source** define if you want to separate the data per source in the XMP result. One layer will represent one source in the result:

- Set to **True** if you want to create one layer for each source in the XMP result.
- Set to **False** if you want to create only one layer for all sources in the XMP result.

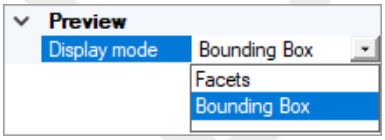
> **Note:** If you set both **One Layer Per Instance** and **One Layer Per Source** to **True**, one layer per source per instance will be created.

5. In **Optional or advanced settings** :

- In case of a Ray File Source, you can adjust the **Number of rays** per instance and the **Ray length** to display in the 3D view.



- In case of Speos Light Box, you can define how to preview the Light Box in the 3D view:



  - Select **Facets** to display the shape of the Light Box.
  - Select **Bounding box** to display a rectangular box containing the Light Box.

> **Important:** Make sure the bounding boxes of the Speos patterns do not overlap. Speos does not support bounding boxes overlap.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

> **Note:** In case of a complex Speos Light Box, Facets may take long to display whereas Bounding box may be faster.

6. In the 3D view, click  and select several origins.

   Each instance of the pattern will be centered and oriented on each origin system listed.

> **Note:** The same origin can be used in other Speos Patterns.

7. Click **Validate**  .

The source instances are automatically displayed in the 3D view.

Now you can add the Speos Pattern in a simulation either in the sources list or in the geometry list. If the Speos Pattern is added in the geometry list, then only the Speos Pattern geometry will be used and not the source.

DRAFT

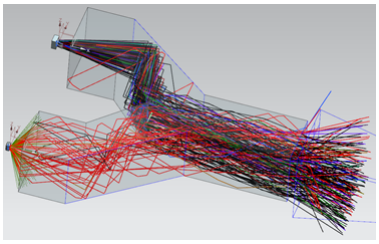
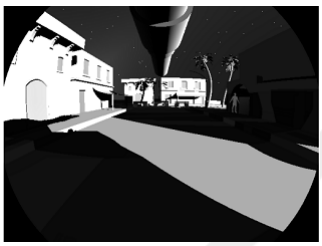
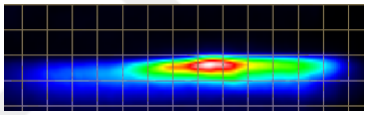
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

10: Simulations

Simulations allow you to give life to the optical system in order to generate results.

10.1. Simulations Overview

Simulations allow you to give life to the optical system in order to generate results.

		
Interactive Simulation	PNG Result of an Inverse Simulation	Xmp result of a Direct Simulation

Simulations allow to materialize and test out an optical system by propagating rays between the key components of optical simulation (materials, sources, sensors).

Different types of simulations are available to cover the different needs that might appear along the entire design process.

From dynamic to deterministic simulation, you can visualize how the light behaves in a given optical system and analyze the results of such simulation.

Types of Simulations

- The Interactive Simulation allows you to visualize in the CAD the behavior of light rays in an optical system. This simulation can be really useful to quickly understand how a design modification can impact the optical behavior.
- The Direct Simulation allows you to propagate a large number of rays from sources to sensors through an optical system (a geometry).
- The Inverse Simulation allows you to propagate a large number of rays from sensors (a camera or an eye) to sources through an optical system.
- The LiDAR Simulation allows you to generate output data and files that enable to analyze a LiDAR system and configuration.

In Speos

You can create any kind of simulation depending on your configuration.

Different simulation outputs may be generated depending on the type of simulation performed:

- HMTL reports that provide detailed information about the simulation.
- Extended Maps (XMP), useful to analyze the results or test a measure according to specific regulations.
- PNG results
- .ldt, .ies files (intensity files generated when creating an interactive or direct simulation including an intensity sensor).
- HDRIs

DRAFT

- Speos360 files
- Ray files (when the generation is enabled during simulation definition).
- LPF, LP3 files (generated when Light Expert is activated).

Some tools are available to manage the simulations results like Light Expert or the LABS viewers allowing you to analyze and perform measures on the results.

Related information

Inverse Simulation on page 405
The Inverse Simulation allows you to propagate a large number of rays from a camera or a sensor to the sources through an optical system.

Interactive Simulation on page 391
The Interactive Simulation allows you to visualize the behavior of light rays in an optical system.

Direct Simulation on page 398
The Direct Simulation allows you to propagate a large number of rays from sources to sensors through an optical system.

LiDAR on page 441
LIDAR is a remote sensing technology using pulsed laser light to collect data and measure the distance to a target. LIDAR sensors are used to develop autonomous driving vehicles.

10.2. Simulation Management

This section presents the different modes of simulation computation available in Speos and describes simulation compatibilities.

10.2.1. Simulation Compatibility

This page describes which sources and sensors are compatible with the different types of simulation.

CPU Simulations

Simulation	Source	Sensor
Interactive Simulation	Interactive Source Ray File Source Surface Source Luminaire Source Display Source Light Field source	Irradiance Sensor Intensity Sensor Radiance Sensor 3D Irradiance Sensor Camera Sensor

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Simulation	Source	Sensor
Direct Simulation	Ray File Source[8] Surface Source Luminaire Source Ambient Source[1] Display Source Light Field source	Irradiance Sensor Intensity Sensor Radiance Sensor[7] 3D Irradiance Sensor 3D Energy Density Sensor VR Immersive Sensor[7] VR Observer sensor[7] Human Eye Sensor[7] Light Field sensor Light Expert Group[6] Physical Camera
Inverse Simulation	Surface Source Luminaire Source Ambient Source Display Source[5] Light Field source	Irradiance Sensor Radiance Sensor[2] VR Observer sensor Human Eye Sensor Camera Sensor[3,4] Physical Camera

Warning: During an Inverse simulation only, the first pixel of a sensor determines the medium in which the sensor is. Make sure the sensor does not overlap with two different media at the same time, otherwise you may generate propagation errors and wrong result.

Warning: Issue when sensor pixel size is larger than the size of geometry, in Inverse simulation only:

- At the beginning of the simulation, a visibility map is computed. To compute it, a ray is launched at the center of each pixel of the sensor to know if the ray has intersected geometries or not.
- During the simulation, rays are randomly emitted on all the surface of the pixel. If a ray emitted (during the visibility map computation) in the center of a pixel intersects no geometry, all rays emitted in that pixel will never intersect geometries.

This may then generate propagation errors and wrong result.

To solve this issue, you may use a sensor with smaller pixels.

[1]When ambient and/or environment sources are enabled for direct simulation, only 2D and 3D irradiance sensors are taken into account.

[2]Only for colorimetric and spectral radiance sensors.

[3] The SPEOS Lens System model (.OPTDistortion v2 version) is not compatible with a deterministic algorithm in inverse simulation.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

[4]An Inverse Simulation, with Timeline deactivated and using a Camera Sensor, only generates a HDRI file (*.hdr).

[5]Inverse Simulations using Display Sources are not compatible with the FTG option activated (Fast Transmission Gathering).

> **Note:** Geometries are embedded in the simulation even if not selected from the geometries list. For example, if a 3D Texture element is selected for simulation, its associated support body is also embedded in the simulation even if the support body was not selected as geometry.

[6]Only one Light Expert Group can be added to a Direct simulation.

[7]When a Direct simulation is composed of a sensor using the Gathering algorithm (Radiance, Human Eye, Observer, Immersive), and a polarizing surface state (unpolished, coated, polarizer, polar plate, optical polished, plugin, polar anisotropic surface), simulation results might not be accurate due to the fact that gathering does not take into account the polarization of the ray, acting as if the ray is unpolarized.

[8]When using a Ray File source in simulation, make sure all rays start from the same medium. Otherwise you will have an unrealistic behavior and may face differences between GPU and CPU simulations.

GPU Simulations

For an exhaustive list of GPU Solver limitations and non-compatibility, see GPU Simulation Limitations on page 367.

- Files, components, sources or sensors that are not listed in the following table are not compatible with GPU Simulations.
- GPU Simulations use the Monte Carlo algorithm.
- GPU Simulations simulate all sensors at once.

> **CAUTION:** As all sensors are loaded into memory at the same time, the Video RAM might become saturated when using many sensors.

> **Warning:** Propagation errors are not managed by GPU simulations. This may lead to inconsistent results between CPU and GPU simulations. For instance, when a Surface Source is considered as tangent to a body.

Simulation	Source	Sensor
Direct Simulation	Ray File Source[7] Surface Source Luminaire Source Ambient Source[1] Display Source	Irradiance Sensor Intensity Sensor Radiance Sensor[6] 3D Irradiance Sensor VR Immersive Sensor[6] VR Observer sensor[6] Human Eye Sensor[6]

Simulation	Source	Sensor
Inverse Simulation	Surface Source Luminaire Source Ambient Source Display Source[5]	Irradiance Sensor Radiance Sensor[2] VR Immersive Sensor VR Observer sensor Camera Sensor[3,4] Human Eye Sensor

[1]When ambient and/or environment sources are enabled for direct simulation, only 2D and 3D irradiance sensors are taken into account.

[2]Only for colorimetric and spectral radiance sensors.

[3]Camera Sensors using dynamic parameters (Trajectory file and Acquisition parameters) in an Inverse Simulation using the Timeline is compatible with the GPU Compute.

[4]An Inverse Simulation, with Timeline deactivated and using a Camera Sensor, only generates a HDRI file (*.hdr).

[5]Inverse Simulations using Display Sources are not compatible with the FTG option activated (Fast Transmission Gathering).

[6]When a Direct simulation is composed of a sensor using the Gathering algorithm (Radiance, Human Eye, Observer, Immersive), and a polarizing surface state (unpolished, coated, polarizer, polar plate, optical polished, plugin, polar anisotropic surface), simulation results might not be accurate due to the fact that gathering does not take into account the polarization of the ray, acting as if the ray is unpolarized.

[7]When using a Ray File source in simulation, make sure all rays start from the same medium. Otherwise you will have an unrealistic behavior and may face differences between GPU and CPU simulations.

> **Note:** Geometries are embedded in the simulation even if not selected from the geometries list. For example, if a 3D Texture element is selected for simulation, its associated support body is also embedded in the simulation even if the support body was not selected as geometry.

Speos Live Preview

For an exhaustive list of GPU Solver limitations and non-compatibility, see GPU Simulation Limitations on page 367.

> **Note:**
>
> - Files, components, sources or sensors that are not listed in the following table are not compatible with Speos Live Preview.
> - Speos Live Preview is not compatible with propagation in ultraviolet or infrared.
> - Speos Live Preview simulations use the Monte Carlo algorithm.

DRAFT

Simulation	Source	Sensor	Materials and Components
Direct Simulation	Ray File Source Surface Source Luminaire Source Display Source	Irradiance Sensor Intensity Sensor[1] Radiance Sensor Human Eye Sensor	Speos Light Box, BSDF 180 file (*.bsdf180), Unpolished file (*.unpolished), Perfect/rough colored mirror files (*.mirror), Anisotropic BSDF file (*.anisotropicbsdf), spectral intensity maps (when used as input of a surface source definition) Complete scattering file - BRDF (*.brdf), Coating file (*.coated), Advanced Scattering file (*.scattering), Simple Scattering file (*.simplescattering), Mirror, Lambertian and Optical Polished built-in models, Non-fluorescent material (*.material) files
Inverse Simulation	Surface Source Luminaire Source Ambient Source Display Source[3]	Irradiance Sensor Radiance Sensor[1] Camera Sensor[2] Human Eye Sensor	

[1]Intensity Sensors with Near Field activated are not supported.

[2]Camera Sensors using dynamic parameters (Trajectory file and Acquisition parameters) in an Inverse Simulation using the Timeline is compatible with the Live Preview.

[3]Inverse Simulations using Display Sources are not compatible with the FTG option activated (Fast Transmission Gathering).

10.2.2. GPU Simulation Limitations

The following page presents you an exhaustive list of the GPU Solver usage limitations and non-compatibility.

The GPU Solver corresponds to the GPU Compute and the Speos Live Preview.

Material Definition

Surface Optical Properties do not support:

- Texture normalization: Color from BSDF and Color from texture
- *.fluorescent file format
- White Specular option in *.anisotropicbsdf file format
- *.retroreflecting file format
- SOP Plugin

Volume Optical Properties do not support:

- Fluorescence
- Birefringence
- Index Gradient
- Metallic
- Non-homogeneous volume

Light Sources

Thermic Surface Source are not supported

Ambient Sources:

- Are not supported in Direct Simulation

DRAFT

- Natural Light Ambient Sources do not support Night Sky model (Moon and Stars)
- US Standard Atmosphere 1976 Ambient Sources are not supported
- MODTRAN Sources are not supported

Lightfield Sources are not supported

Sensors

The GPU Solver does not support:

- 3D Energy Density Sensors
- LiDAR Sensors
- Geometric Rotating LiDAR Sensors
- Lightfield Sensors

Irradiance Sensors:

- Only support the Planar Integration type

3D Irradiance Sensors:

- Only support the Planar Integration type

Intensity Sensors do not support:

- Polar Intensity (IESNA / Eulumdat)
- Near Field in Conoscopic (XMP):

Camera Sensors support the SPEOS Lens System model v1.0 and v2.0. The v2.1 version is not supported

GPU hardware memory shall be adequate to the VR sensor use cases. Otherwise, possible memory issues can occur. Example: a low GPU hardware with a very high sensor resolution.

Simulation

The GPU Solver does not support:

- Interactive Simulations
- HUD Optical Analysis
- LiDAR Simulations
- Geometric Rotating LiDAR Simulations

General Options:

- You can only set a GPU simulation to **Single precision**: GPU Solver cannot handle **Ray tracer precision** set to **Double precision** and will always run as **Single precision**.
- GPU simulations with a **Geometrical Distance Tolerance** to 0.0001 or less may generate errors. We recommend you to set the **Geometrical Distance Tolerance** to a minimum of 0.01mm.

Inverse Simulations do not support:

- Deterministic algorithm
- Optimized Propagation
- Splitting
- Number of gathering rays per source
- Maximum gathering error

DRAFT

Result

The **Depth of field** parameter from the **Virtual Human Vision Lab** is not supported in XMP map generated with a GPU Simulation.

Light Expert

The GPU Solver does not support:

- Light Expert
- Result data separated by sequence
- Result data separated by polarization
- Generation of *.lpf and *.lp3

The result data separated by surface are supported only in Direct Simulations

Components

The GPU Solver does not support:

- 3D Textures
- Polarization plates
- Optical Design Exchange features

Reporting

The Simulation Report does not display the Error Tracking.

*.Speos360 results do not contain simulation statistics in the **Speos 360 properties** window.

The detailed simulation report is available in the XMP results

10.2.3. GPU/CPU Differences

When running a same simulation, you may face differences between GPU and CPU results. This may be due to different reasons. This page is dedicated to present the behavior difference between CPU and GPU, and the possible good practice to get equivalent results if there are any.

Propagation Errors Case

Behavior: Irradiance Sensor

For more information on the different propagation errors, refer to Understanding the Propagation Errors.

Summary: A ray intersecting the sensor is always integrated on GPU result, whereas a ray intersecting the sensor is never integrated on CPU result if it has encountered a propagation error.

On **GPU**, a ray is integrated into the Irradiance Sensor as soon as it hits the sensor.

- If the ray has encountered a propagation error (before or after the sensor), the ray is integrated into the sensor.
- If the ray has not encountered a propagation error, the ray is integrated into the sensor.

On **CPU**, the full ray is propagated, then Speos parses the ray trace to know if it has intersected the sensor:

- If the ray has encountered a propagation error (before or after the sensor), the ray is not integrated into the sensor.
- If the ray has not encountered a propagation error, the ray is integrated into the sensor.

DRAFT

Good Practice

Reduce the propagation errors for the CPU Simulation to get nearly the same result as the GPU Simulation, by reducing the Geometrical Distance ToleranceGeometrical Distance Tolerance.

Maximum Number of Surface Interaction Case

- On **GPU**, when a ray is stopped by the Maximum number of surface interaction option, if the ray is stopped after the intersection with the sensor, it will be integrated on the sensor.
- On **CPU**, when a ray is stopped by the Maximum number of surface interaction option, if the ray is stopped after the intersection with the sensor, it will not be integrated on the sensor.

Project Origin Case

In small systems shifted from the Speos origin, some differences can appear between CPU and GPU results.

To reduce the differences, make sure to position the system as close as possible to the Speos origin.

Output Faces Case

In simulation:

- In CPU, for each pixel per pass, if the ray emitted by the CPU does not intersect an output face, the CPU will emit again a ray until the ray intersects an output face.
- In GPU, for each pixel per pass, the GPU emits one ray whatever it intersects or not an output face. GPU does not emit again if the ray does not intersect an output face.

That means, for a same number of pass, CPU does converge better than GPU. To get the same result on GPU, you need to increase the number of pass.

Stop Conditions: On Number of Passes Limit

For more information on the option, refer to Creating an Inverse Simulation.

In case of a GPU simulation, the number of passes defined corresponds to the minimum threshold of rays received per pixel. That means while a pixel has not received this minimum number of rays, the simulation still runs.

If you activated the dispersion, this minimum threshold is multiplied by the sensor sampling. That means for a number of pass of 100 and a sampling of 13, each pixel will need to receive at least 1300 rays (the progress of the number of rays per pixel is indicated in the Simulation Progress Bar).

In case of a CPU simulation, the number of pass, there is no minimum threshold of rays to be received per pixel, the simulation stops when the number of pass is reached.

Dispersion Calculation Case

For information on the Dispersion case, refer directly to Dispersion.

Ray File Reading Algorithm

In case of a GPU Simulation: read rays from file are checked at simulation initialization. The Ray file validity is checked and you are notified if the ray file is corrupted with a warning message. If a corrupted ray file is used, it could lead to biased results because rays are not rechecked at simulation run.

In case of a CPU Simulation: rays are checked when simulation is running. No warning message is displayed if the Ray file is corrupted. If a corrupted ray file is used, it could lead to biased results.

DRAFT

Surface Source with Exit Geometries

In case of a CPU Simulation: rays from the Exit Geometries are only propagated forward according to the emitted rays from the Surface Source.

In case of a GPU Simulation: rays from the Exit Geometries can be propagated forward and backwards

10.2.4. Computing Simulations

This page describes the different ways to run a simulation inSpeos.

Estimated RAM to Compute Simulation

In the Simulation definition an estimation of the RAM required to compute the simulation is displayed.

This estimation is based on the sensors and their number of layers and the sources defined in the Direct or Inverse simulation.

> **Note:** The meshing applied on geometries is not considered in the RAM estimation.
>
> The following sensors are not considered in the RAM estimation:
>
> - 3D Energy Density sensor
> - 3D Irradiance sensor
> - LiDAR sensor
> - Light Field sensor

> **Tip:** If a XML template is used in sensors, deactivating **Filtering** options may divide by around 2 the required memory.

Automatic Compute

> **Note:** The **Automatic compute** is only available from the feature contextual menu.

Depending on the feature, the computation is done manually or automatically.

In Speos, certain features (usually features that do not require heavy calculation resources) are updated automatically, like certain optical design parts.

The **Automatic Compute** is useful when coupled with an Interactive Simulation to generate rays automatically without having to manually compute the simulation.

> **Note:** This action replaces the Live Trace.

However, when the feature requires a certain amount of memory resources, the **Compute**  option is used to generate the feature or launch the simulation.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Interactive Simulation can be switched from one mode to another depending on your needs:

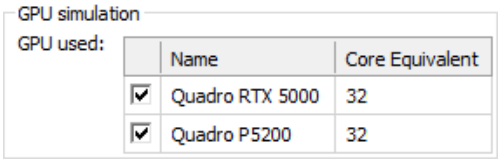
- Activate the **Automatic compute** when wanting the simulation to be updated every time an input is modified (best suited for light simulations).
- Deactivate this mode when working with heavy simulations to avoid unwanted updates when modifying an input.

GPU Compute

The **GPU Compute** option runs simulations on your computer using the cores of your GPU. The GPU compute offers a faster alternative than classic CPU simulation.

To define the GPU to use for the compute:

1. Click **File** > **Speos options**
2. Click the **Light Simulation** section.
3. In the **GPU** tab, in the **GPU simulation** section check the GPUs to use.



> **Note:** If you select multiple GPUs, simulations will consume the sum of the equivalent cores per GPU.
>
> If you select no GPU, Speos will automatically select the most powerful available.

To get a list of simulations compatible with GPU Compute, see Simulation Compatibility.

Live Preview

The Live Preview allows you to compute the simulation using progressive rendering with the most powerful GPU available on your computer. The result is displayed in a dedicated window and calculated in real-time.

For more information on the Interactive Live Preview, refer directly to the dedicated chapter Speos Live Preview.

Interactive Live Preview

The Interactive Live Preview tool permits you to directly see, in the current running Live Preview window, the changes applied on your project without having to launch a simulation to see the result.

For more information on the Interactive Live Preview, refer directly to the dedicated chapter Interactive Live Preview.

External Simulations

External Simulations are simulations run through Speos Core.

Thanks to Speos Core, exported simulations can be run locally or on a network while keeping Speos available.

For more information, see Speos Core.

Related tasks

Using the Feature Contextual Menu on page 29

DRAFT

This page lists all the operations that can be performed from the features' contextual menu.

Related information

Speos Core on page 385
Speos Core is used to run External Simulations. Thanks to Speos Core, exported simulations can be run locally or on a network while keeping Speos available.

10.2.5. Speos Live Preview

Speos Live Preview allows you to compute the simulation using progressive rendering with the most powerful GPU available on your computer, and displays the result in a dedicated window calculated in real-time.

10.2.5.1. Understanding the Live Preview

The following page presents you an overview of the Speos Live Preview.

Description

Speos Live Preview allows you to compute the simulation using progressive rendering with the most powerful GPU available on your computer, and displays the result in a dedicated window calculated in real-time.

Therefore:

- you can preview the result of a simulation without having to run it the usual way.
- you save time which allows you to be more effective and to focus more on the design.

The Live Preview is very useful when you do not know exactly the result you want to obtain, as it provides you with a workflow more oriented toward the exploration. It helps you experiment and apprehend what happens optically upon any change you operate. However, be aware that the Live Preview should not be used for validation purposes.

Technical Information

The Live Preview is compatible with NVIDIA GPUs only. GPUs supported are NVIDIA Quadro P5200 or higher.

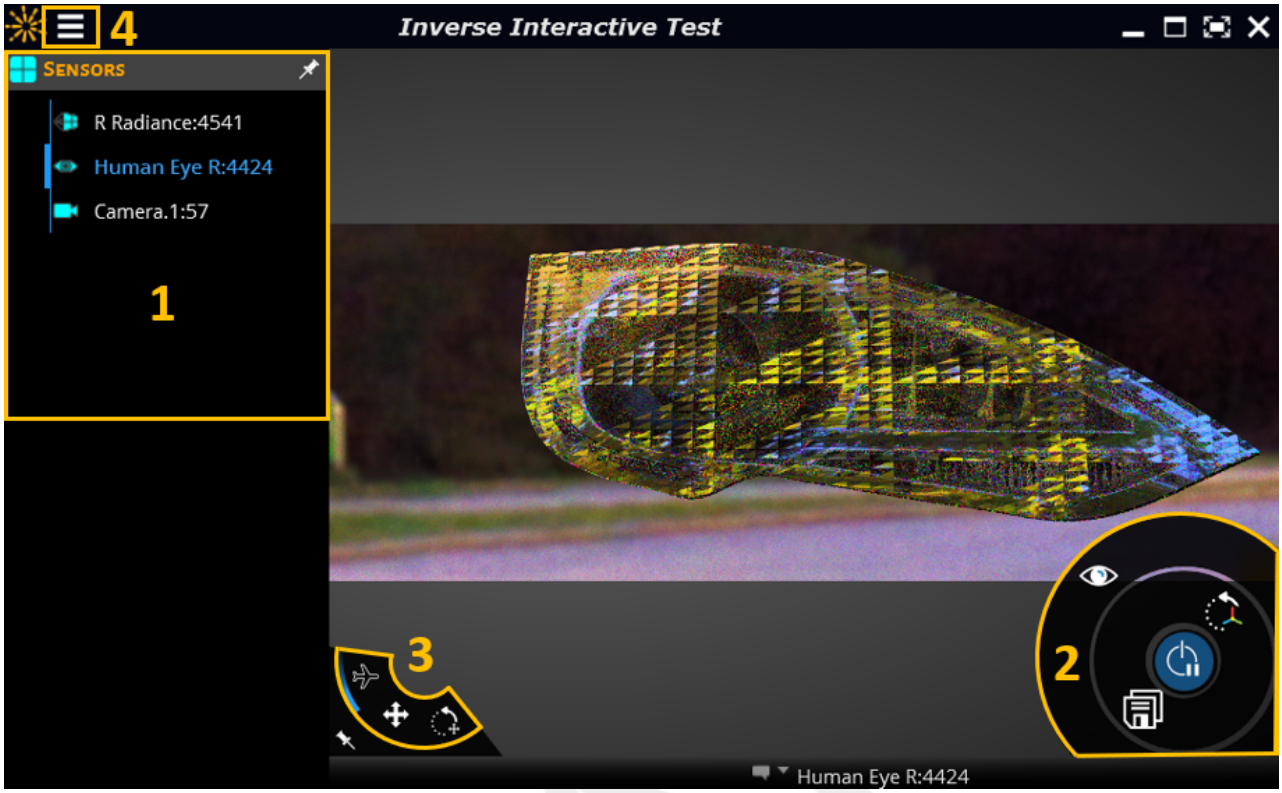
The Live Preview is 4K compatible.

To get a list of the files that are compatible with the Live Preview, see Simulation Compatibility.

Interface

The Live Preview provides you with an intuitive interface and navigation in the project.

DRAFT



1. The Sensor panel provides you with the simulated sensors that you can select individually to change the preview window.
2. The bottom right ring provides you with:
   - Save and Reset preview buttons
   - Result viewing options corresponding to the different Display modes
   - A play/pause button
3. The bottom left ring controls the navigation modes.
4. The top left menu provides you with customizable interface and navigation settings to better fit your use.

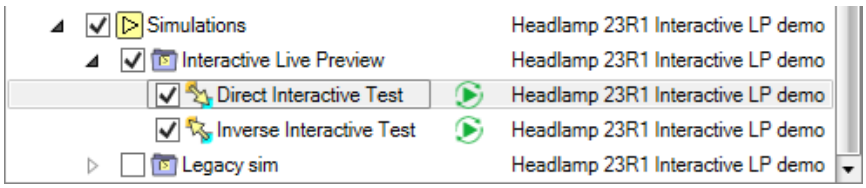
10.2.5.2. Running a Live Preview

The following procedure helps you launch and run a Live Preview out of a simulation.

To run a Live Preview:

You must have created a Direct or Inverse simulation.

1. In the Speos tree, select a Direct or an Inverse simulation.



2. In the **Light Simulation** tab, in the **Tools** section, click **Preview**  .

DRAFT

The Live Preview window opens and you can explore and analyze the preview according to the sensors you added to the simulation.

If you want to go further into the exploration, you can modify parameters of your system and update the preview. You will then directly see in the current running Live Preview window the changes applied on your project without having to launch a simulation to see the result.

Related reference

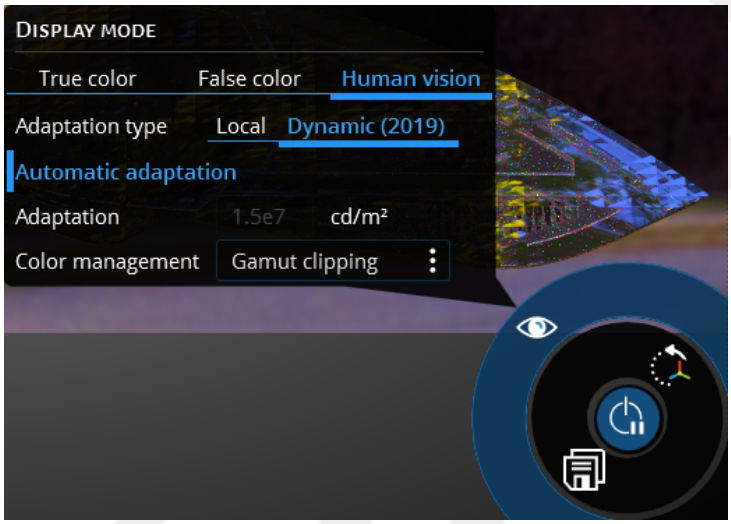
Live Preview Parameters on page 375
The following page presents you the different parameters available to control the Live Preview display.

Preference Settings on page 377
The following page presents you the customizable interface and navigation settings to better fit your use.

10.2.5.3. Live Preview Parameters

The following page presents you the different parameters available to control the Live Preview display.

Display Options



True Color

True Color allows you to preview the result close to the reality with the color information.

False Color

False Color displays a preview easier to read with a false color scale.

Human Vision

- The **Human Vision, Local Adaptation** mode defines the accommodation on a fixed value of the luminance map.
- The **Human Vision, Dynamic Adaptation 2019** mode enables the adaptation of the human eye.

  It models the fact that the eye adapts locally as the viewer scans the different areas of the luminance map.

  Fore more information, refer to **Parameters of Human Vision**.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Automatic Adaptation

With Automatic Adaptation you can:

- activate **Automatic Adaptation** to base the adaptation on the maximum value of the result.
- deactivate **Automatic Adaptation** to set a custom adaptation value.

Color management

Gamut Clipping: colors outside the gamut of the result displayed are clipped along the gamut borders towards the center of the lightness axis in order to make potential details appear.

Maintain lightness and hue keeps the lightness and hue of color inside the monitor gamut. Otherwise colors outside the monitor gamut will be saturated. Colors are closer to the white point luminance than the gamut borders: this favors better luminance variation over color rendering.

> **Note:** The **Color management** options are not available in **False color**.

> **Note:** When computing simulations containing camera sensors, the color layout cannot be changed and the level controller is unavailable.

Save



You can save the current live preview result as XMP (*.xmp) or as picture (*.png, *.jpg, *.hdr, *.exr) when you reach the targeted design performance.

> **Note:** Live Preview options (Human Visions and Color management) are not saved in the export result.

> **Note:** If you do not see the save result option, refer to Setting Speos Preferences to allow the result generation.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Restart



The **Restart** button allows you to reset the preview to its initial state.

Play/Pause



Pausing the preview prevents any change you could apply (sensor change, navigation, display options, etc.) to keep the preview steady.

Playing the preview allows all possible change to the result displayed.

Navigation Modes



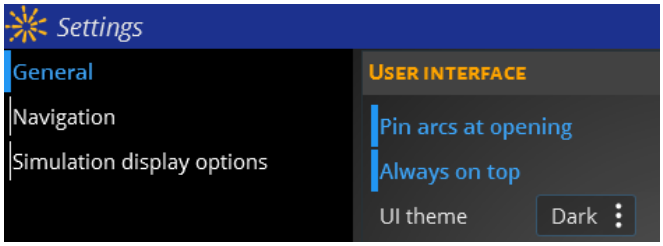
- 3D navigation switches between Third person
  
  and First person navigation
  
  when moving your Illuminance sensor.
- 2D navigation
  
  llows to zoom into details of your result.
- Reset 2D position
  
  fits the result size to window.

10.2.5.4. Preference Settings

The following page presents you the customizable interface and navigation settings to better fit your use.

DRAFT

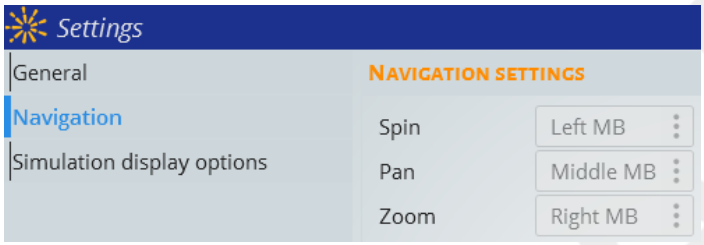
General



- **Pin arcs at opening** holds still the navigation and parameters arcs at the next opening of the Live Preview to prevent them to slide out of the Live Preview window.
- Activating **Always on top** keeps the Live Preview window to front. No other window can be in front of the Live Preview window

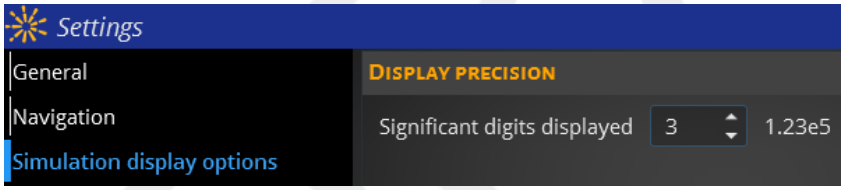
  Deactivating **Always on top** allows you to a window to be in front of the Live Preview window.
- **UI theme** allows you to switch the color theme of the Live Preview window between **Dark** and **Light**.

Navigation



The **Navigation** settings describe you how to use the different navigation commands.

Simulation display options



Significant digit displayed allows you to tune the number of digits to display in the interface values. A preview is displayed next to the parameter.

10.2.6. Interactive Live Preview

The Interactive Live Preview tool permits you to directly see in the current running Live Preview window the changes applied on your project without having to launch a simulation to see the result.

> **Note:** The Interactive Live Preview feature is in BETA mode for the current release.

DRAFT

10.2.6.1. Interactive Live Preview Overview

The following page gives you an overview of the Interactive Live Preview and its common use.

Description

The Interactive Live Preview tool permits you to directly see in the current running Live Preview window the changes applied on your project without having to launch a simulation to see the result. Indeed, this save you from launching a full simulation to see the changes as it does not mesh the optical system. This way you can understand quickly which modification has an impact and what is not supported.

You can:

- Modify most numerical values in sources, sensors and material definitions, as well as file path (spectrum, IES, etc.).
- Move an object as long as it is not bound to a geometry. When you have an object oriented by an axis system, moving the axis system will be considered in Live Preview Update.

> **Note:** Refer to Parameters Compatibility with Interactive Live Preview to make sure that the parameters you change are compatible with the Interactive Live Preview. Otherwise, if you modify an incompatible parameter a warning will be raised and **Update Preview** button will not be available.

Interactive Live Preview Common Workflow

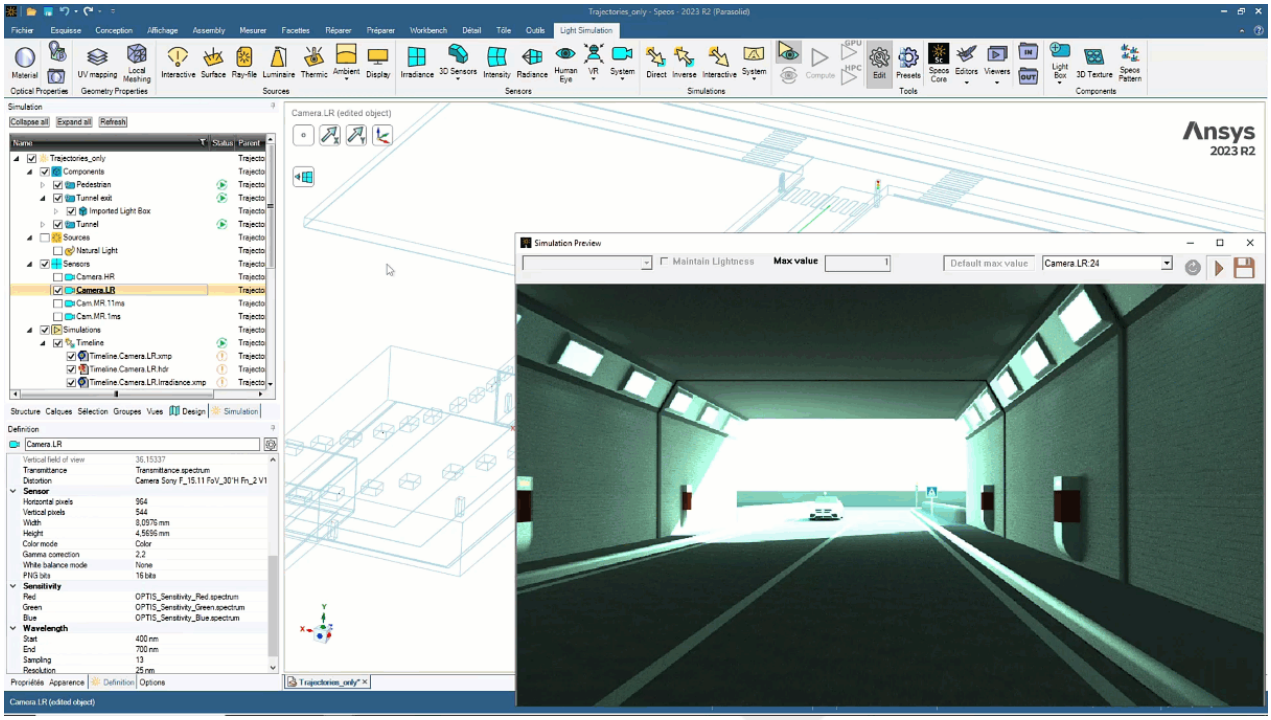
1. Create a Direct or Inverse Simulation of your optical system.
2. Click **Preview** to run the Live Preview.
3. Modify one or several parameter in features used in the simulation.
4. Click **Update Preview** to run the Interactive Live Preview.
5. See the change in the Live Preview window.
6. Iterate again the process to quickly see impact of the modifications.

Example

The following video shows you an example on how to use the Interactive Live Preview in the case of parameters change and also in the case of Timeline parameter change:

- You can see that parameters are applied to Live Preview rendering upon click of Interactive Live Preview.
- In the case of Timeline parameter change, the change is automatically applied to the Live Preview rendering without having to click Interactive Live Preview.

DRAFT



10.2.6.2. Using the Interactive Live Preview

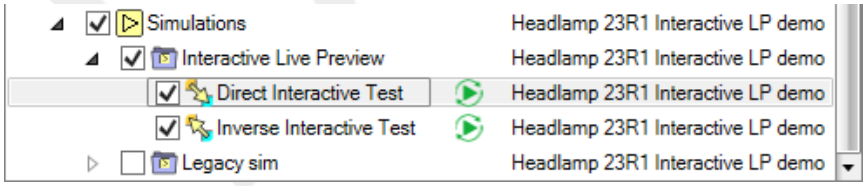
The following procedure helps you use the Interactive Live Preview in the Live Preview window to see the changes applied on features used in simulation.

You must have created a Direct or Inverse simulation.

You must have run first a Preview of the simulation.

1. After running the **Preview**, leave the **Live Preview** window opened.
2. Modify one or several compatible parameters of features used in the selected simulation.

   Refer to Parameters Compatibility with Interactive Live Preview to make sure that the parameters you change are compatible with the Interactive Live Preview. Otherwise, if you modify an incompatible parameter a warning will be raised and **Update Preview** button will not be available.
3. Select the simulation.



> **Note:** Do not open the definition of the simulation.

   Speos detects that modifications have been made on the project. The **Update Preview** button is ungreyed.
4. Click **Upate Preview**  .

You can directly see the changes in the Live Preview window according to the parameters you modified.

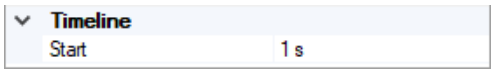
10.2.6.3. Using the Interactive Live Preview with the Timeline Parameter

The following procedure helps you use the Interactive Live Preview in the Live Preview window in case of Timeline parameter change of an Inverse Simulation.

You must have created an Inverse simulation.

You must have run first a Preview of the simulation.

1. After running the **Preview**, leave the Live Preview window opened.
2. In the **Inverse Simulation** definition, modify the **Timeline** parameter.



The Live Preview updates automatically without having to click the **Update Preview** button.

10.2.6.4. Parameters Compatibility with Interactive Live Preview

The following page describes you which features parameters are compatible with the Interactive Live Preview.

Parameters and features that are not listed in the following tables are not compatible with the Interactive Live Preview.

Sources

Source	Parameters
Surface Source	• Flux • Intensity • Spectrum • Emissive
Ray File Source	• Flux • Axis System
Luminaire Source	• Intensity • Flux • Spectrum • Axis System
Uniform Source	• Axis System (Zenith and Sun) • Luminance • Mirror Extent • Sun • Spectrum
CIE Standard Overcast Sky Source	• Axis System (Zenith) • Luminance • Spectrum

DRAFT

Source	Parameters
CIE Standard General Sky Source	• Axis System • CIE Type • Luminance • Sun Automatic/Manual • Time zone and location
Natural Light Source	• Axis System (Zenith and Sun) • Turbidity • Sun Automatic/Manual • Time zone and location • Sky
Environment Source	• Axis System (Zenith and North) • Luminance • Image File • Color Space • White Point
Display Source	• Image • X/Y Range • Flux • Intensity • Color Space • White Point • Axis System

Sensors

Sensor	Parameters
Irradiance Sensor	• Axis System • XMP Template • Integration type (only Planar type supported) • X/Y Range • X/Y Sampling and Resolution • Wavelength • Integration Direction
Intensity Sensor	• Axis System • XMP Template • Format • Orientation • X/Y Range • X/Y Sampling and Resolution • Wavelength

DRAFT

Sensor	Parameters
Radiance Sensor	• Definition from • Observer Type • Focal Length • Axis System • XMP Template • X/Y Range • X/Y Sampling and Resolutiokn • Wavelength • Integration Angle
Human Eye Sensor	• Axis System • XMP Template • Horizontal and Vertical FOV • Wavelength • Pupil Diameter
Camera Sensor	• Mode (only photometric mode supported) • Axis System • Acquisition • Timeline (through Inverse Simulation) • Focal Length • Imager Distance • F-number • Transmittance • Distortion • Pixels • Width/Height • Color Mode • Gamma correction • White balance mode • PNG bits • Sensitivity • Wavelength • Visualization Parameters

Optical Properties

Feature	Parameters
Material	• Volume properties • Surface properties • Use texture
Surface Layer (if Use texture is activated)	• Texture image • Texture normal map

DRAFT

Geometry Properties

Feature	Parameters
UV Mapping	• Geometry
UV Map	• Mapping Type • Axis System • Rotation • U/V Parameters

Component

Feature	Parameters
Speos Light Box Import	• Moving the Light Box • Trajectory file

10.2.7. Exporting a Simulation

The following procedure helps you fully export a simulation.

To export a simulation:

You must have set a simulation but not have run it.

> **Warning:** Exporting a project breaks the link between your project and the exported simulation. The exported simulation is not available in the Speos Simulation tree. If you need to export a simulation and to keep it in the Speos Simulation tree, refer to Linked Export.

1. In the **Speos Simulation** tree, right-click the simulation you want to export.
2. Click **Export**  .
3. Browse and select a path for your ***.speos** folder.
4. Name the file and click **Save**.

The simulation has been exported in the folder of the same name as the *.speos file along with the input files.

Now you can use Speos Core to run the simulation out of Speos.

10.2.8. Linked Exporting a Simulation

The following procedure helps you fully export the simulation keeping a link between the project and the simulation in the SPEOS Simulation tree.

To linked export a simulation:

You must have set a simulation but not have run it.

1. In the **Speos Simulation** tree, right-click the simulation you want to linked export.

DRAFT

2. Click **Linked Export**  .

The simulation has been exported in the **Speos isolated files** folder along with the input files, and a link to the exported simulation has been created in the Speos Simulation tree.

Now you can use Speos Core to run the simulation out of Speos.

10.2.9. Speos Core

Speos Core is used to run External Simulations. Thanks to Speos Core, exported simulations can be run locally or on a network while keeping Speos available.

10.2.9.1. Speos Core Overview

Speos Core allows you to run exported simulations (*.speos file) out of Speos.

Description

The main benefit of externalizing a simulation is the ability to perform parallel processing while still being able to use Speos.

Two main types of external simulations can be used to run exported simulations (*.speos files) using the Speos Core interface:

- Local Update: runs the simulation with Speos Core on your workstation.
- Network Update: runs the simulation with Speos HPC or Ansys Cloud.

Local Update

The Local Update allows you to run a simulation while keeping Speos available. The exported simulation is launched locally on the workstation by using the Speos Core interface.

Depending on the simulation export type used, the simulation results are generated in the .speos directory (Export) or both in the .speos directory and in Speos tree (Linked Export).

Network Update

The Network Update allows you to run a simulation with Speos HPC or the Ansys Cloud service while keeping Speos available.

Whether your network corresponds to a Linux cluster, a Windows cluster or the Ansys Cloud service, the network configuration is performed directly from the Speos Core interface.

Once the network environment is configured, simulations can be exported and run using two different methods:

- Manual method: Export the simulation manually using Export or Linked Export, then run it through Speos Core.

  This option allows you to specify and adjust the settings used for the simulation during the job submission.

- Automatic method: Launch the simulation using the **Speos HPC Compute** command from the Speos interface.

  This command combines a Linked Export and a job submission.

DRAFT

Note: Before using this command, you need to create a first simulation to submit in order to configure the cluster for Speos HPC or Ansys Cloud.

Related concepts

Network Update on page 387
The Network Update runs a simulation with Speos HPC or Ansys Cloud while keeping Speos available.

Related tasks

Running a Simulation Using a Local Update on page 386
The following procedure helps you run a simulation in Speos Core out of Speos on your workstation while keeping Speos available.

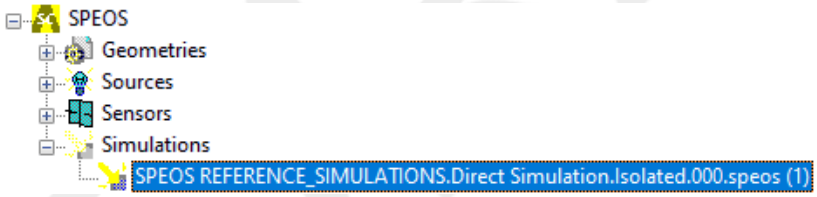
10.2.9.2. Running a Simulation Using a Local Update

The following procedure helps you run a simulation in Speos Core out of Speos on your workstation while keeping Speos available.

To run a simulation using a local update:

You must have exported or linked exported a simulation.

1. In Speos, click  to open **Speos Core**.
2. Click **File** > **Open**  and select the *.speos file.
3. From the **Speos Core** tree, select the simulation to update.



4. Click **Local update**  or **GPU local update**  .

Note: The **GPU local update** considers the cores of the GPU you set in the Speos Options to run the simulation. For more information, refer to the GPU Compute section of the Computing Simulations page.

Note: Error XX% corresponds to the evolution of the total error number during the simulation.

The simulation results are available in the folder for both Export and Linked Export Simulations and from the Speos Simulation tree for Linked Export Simulations.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

10.2.9.3. Network Update

The Network Update runs a simulation with Speos HPC or Ansys Cloud while keeping Speos available.

10.2.9.3.1. Initial Configuration

An initial configuration of the cluster is necessary before launching simulation with Speos HPC or Ansys Cloud.

10.2.9.3.1.1. Configuring the Network for HPC Simulations

The following procedure helps you configure the Speos HPC cluster and the submission job to be able to launch a network update from Speos afterwards.

To configure the network for HPC simulations:

1. Export a Simulation.
2. Open Speos Core  .
3. Click **File** > **Open**  .
4. Browse and select the *.speos file corresponding to the exported simulation.
5. Click **Speos HPC cluster configuration**  and configure the cluster.

> **Note:** For more information on how to configure the cluster, refer to the **Linux Configuration** or the **Windows Configuration**.

6. Click **Speos HPC simulation**  to configure and submit the simulation job.

> **Note:** For more information on how to configure and submit a simulation job, refer to **Submitting a Simulation Job from Speos Core**.

The configuration you defined will be the one that will be used when using the **Speos HPC Compute** command.

10.2.9.3.1.2. Ansys Cloud Simulations

The Ansys Cloud service is compatible with Speos and can be used to launch simulations on a pool of dedicated virtual desktops.

> **Note:** For more information on the Cloud principle, licensing conditions or any other information regarding the Ansys Cloud, refer to the *Ansys Cloud guide*.

10.2.9.3.1.2.1. Setting Up the Cloud Environment

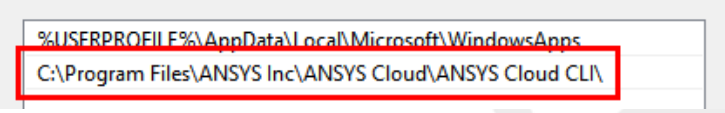
This procedure shows you how to install and configure your environment in order to be able to solve simulations in the cloud.

You must already have an Ansys account, a cloud essentials subscription assigned to you and a set of Ansys Elastic Currencies (AEC).

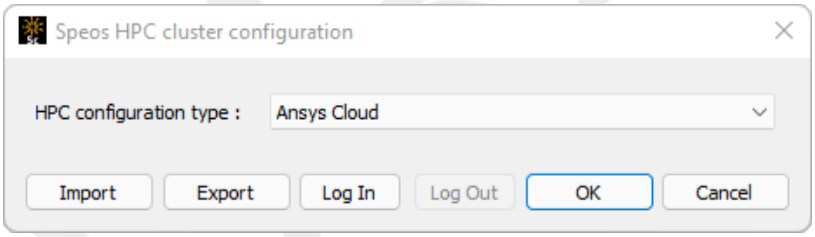
> **Note:** If you need help regarding subscriptions, get in touch with your Ansys representative.

To set up the Ansys Cloud environment:

1. Make sure you are meeting the Ansys Cloud requirements.
2. Install the Ansys Cloud suite.
3. Check your Ansys Cloud installation.
4. Make sure Ansys Command Line Interface (CLI) has been correctly installed with the Ansys Cloud suite installation:
   a) Access your environment variables.
   b) From the **System variables**, edit the **Path** variable and verify that the CLI directory path has been added as a new value of the current environment variable **ANSYSCloudCLI_ROOT**.



5. Configure your Speos Core environment:
   a) From the Speos interface, open **Speos Core**.
   b) Click **Speos HPC cluster configuration**  .



   c) In **HPC configuration type**, select **Ansys Cloud**.
6. Click **Log in** to connect to the Ansys Cloud.
7. When prompted, sign in to your Ansys Cloud account.
8. Click **OK**.

Your Ansys Cloud environment is set up and ready to be used.

Related tasks

Solving Simulations in the Ansys Cloud on page 389
This procedure shows you how to solve simulations in the Cloud using Speos Core to specify the cluster configuration and Ansys Cloud as a job scheduler.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

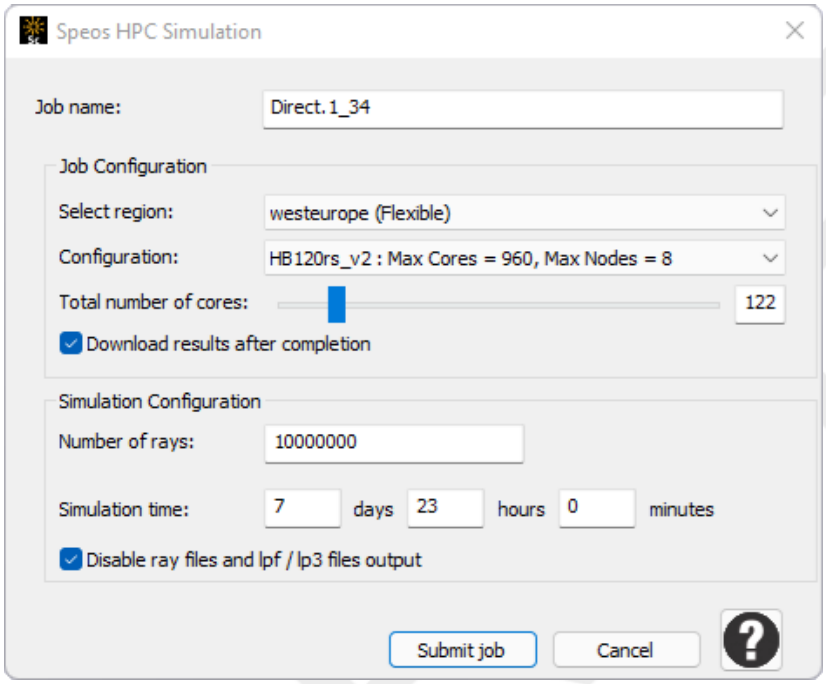
Published: 2023-10-23T14:08:14.04-04:00

10.2.9.3.1.2.2. Solving Simulations in the Ansys Cloud

This procedure shows you how to solve simulations in the Cloud using Speos Core to specify the cluster configuration and Ansys Cloud as a job scheduler.

Ansys Cloud Installation should be complete and your Speos Core environment should be configured.

1. From the Speos interface, open **Speos Core**.
2. Click **File** > **Open** to load an exported *.speos simulation.
3. Click **Speos HPC Simulation**.



4. In **Job name**, type a meaningful job simulation name that will help you identify the job in the job monitor.

> **Note:** By default this name is your user name with a simulation index. The index is incremented at each new simulation.

5. From the **Select Region** drop-down list, select the Cloud region that is the closest to you.

> **Note:** A flexible region allows you to define the **Total number of cores** below.

6. In **Configuration** choose from a list of pre-configured hardware configurations that have been optimized for the solver that you are using. Each configuration has a set number of cores, nodes, and Ansys Elastic Units (AEUs) per hour.
7. With the **Total number of cores** slider, define the maximum number of cores to use for the simulation.

   We recommend you to use a multiple of the number of core available per machine, as job are launched on several nodes with the same number of cores per node. For example, for HB60rs, define a multiple of 60.

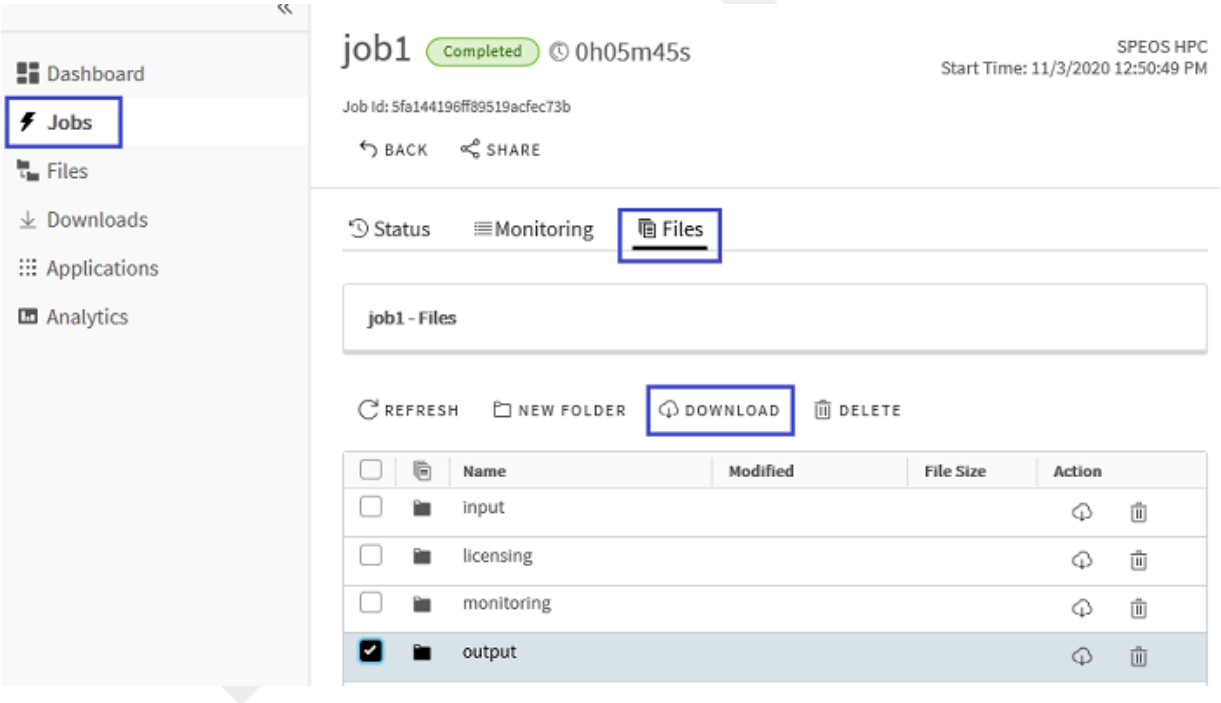
8. Check **Download results after completion** if you want the results to be downloaded in the .speos folder.
9. In **Number of rays**, define the number of rays used for simulation. This number is retrieved from the *.speos data set in Speos but can be adjusted if needed.
10. In **Simulation time**, define maximum time the simulation can run.

> **Note:** This duration cannot exceed the "Maximum scheduler wall clock", that is 1 hour.

11. Tick **Disable ray files and lpf/lp3 output** if you want to disable the generation of these outputs in order to improve simulation time and performance.
12. Click **Submit job**.
    The job input files begin uploading to a storage directory in the Cloud. Once the files have been uploaded, the Cloud service will allocate resources for the job, and the job will start running on the Cloud hardware.

> **Note:** You will receive an email notification when a job starts, completes, fails, or is stopped, enabling you to keep track of jobs even when you are not using the Cloud portal.

13. Once the simulation is complete, from the Ansys Cloud portal, click **Jobs**, select your job from the list, then click **Files** and download your output result files into the **Speos isolated files** folder.



The simulation is solved through Ansys Cloud and can be managed on the Cloud portal.

For more information on file transfers or job monitoring on the Cloud portal, refer to the *Solving in the Cloud* section of the *Ansys Cloud Guide*.

The configuration you defined will be the one that will be used when using the **Speos HPC Compute** command.

Related tasks

Setting Up the Cloud Environment on page 388

This procedure shows you how to install and configure your environment in order to be able to solve simulations in the cloud.

10.2.9.3.2. Running a Simulation Using a Network Update

The following procedure helps you run a simulation in Speos HPC or Ansys Cloud out of Speos without having to export or linked export it beforehand.

To run a simulation using the network update:

You must have proceeded to the initial configuration either for HPC Simulations or Ansys Cloud Simulations.

1. In the **Speos Simulation** tree, right-click a simulation.
2. Click **HPC Compute**  .

The simulation has been exported in the Speos isolated files folder along with the input files, a link to the exported simulation has been created in the Speos Simulation tree, and the simulation is automatically run on Speos HPC or Ansys Cloud according to the configuration you defined.

10.3. Interactive Simulation

The Interactive Simulation allows you to visualize the behavior of light rays in an optical system.

10.3.1. Creating an Interactive Simulation

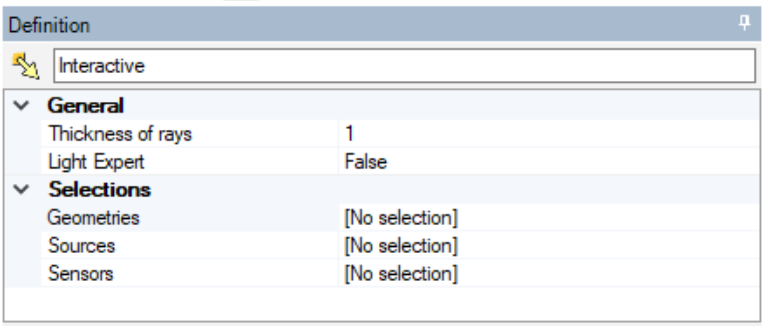
The Interactive Simulation allows you to analyze, visualize and validate the effect of your model on the optical behavior of the light.

> **Tip:** You can update the simulation each time you modify the model to immediately see how your modification impacted the light propagation.

To create an Interactive Simulation:

Make sure the Speos inputs used to define the Simulation (example: sources, geometry, sensors) are located in the same component as the simulation, or in a child component.

1. From the **Light Simulation** tab, click **Interactive**  .

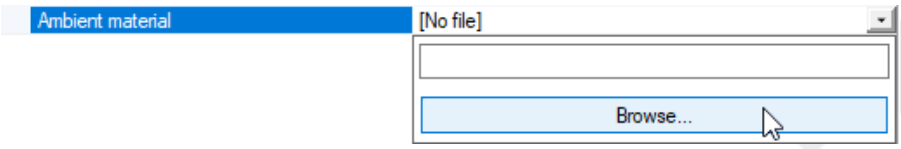


2. Adjust the thickness of the rays to display in the 3D view.
3. If you want the simulation to generate a Light Path Finder file, set **Light Expert** to **True**.

   For more information on Light Expert analyses, refer to Light Expert.
4. In **Ambient material**, browse a .material file if you want to define the environment in which the light will propagate (water, fog, smoke for example).

   The ambient material allows you to specify the media that surrounds the optical system. Including an ambient material in a simulation brings realism to the optical result.

   The .material file is taken into account for simulation.



> **Note:** If you do not have the .material file corresponding to the media you want to use for simulation, use the **User Material Editor**, then load it in the simulation.

5. In the 3D view, click  to select geometries,  to select sources and  to select sensors.

   *The selected geometry, source(s) and sensor(s) appear in the **Linked objects**.*

> **Note:** If you want to customize the interactive simulation advanced settings, see Adjusting Interactive Simulation Settings .

6. Preview the Meshing of the geometries before running a simulation and adjust meshing values if needed to avoid simulation or geometries errors.

The Interactive Simulation is created and the rays are displayed in the 3D view to illustrate the light's optical behavior with respect to the geometry.

Related concepts

Light Expert on page 465
The Light Expert is a tool that allows you to specify what ray path to display in the 3D view.

Related information

Adjusting Interactive Simulation Settings on page 392
This page describes the simulation settings that you can adjust to customize your interactive simulation.

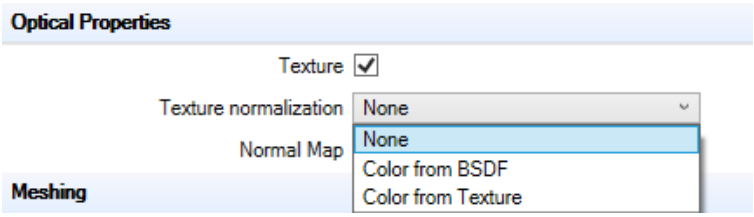
10.3.2. Adjusting Interactive Simulation Settings

This page describes the simulation settings that you can adjust to customize your interactive simulation.

Right-click the interactive simulation and click **Options** to open the advanced settings.

Geometry's Settings

Optical Properties



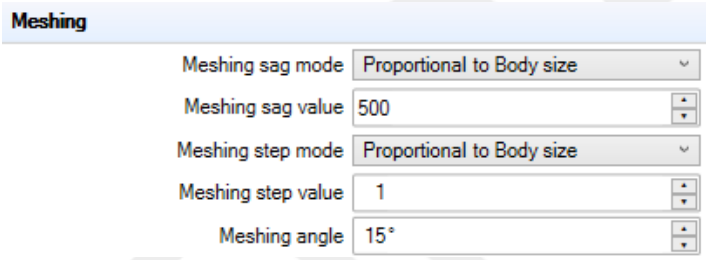
Texture application can have an impact on the simulation results. If texture have been applied in the scene, activate **Texture** and/or **Normal Map** if needed.

The **Texture normalization** determines the rendering of the texture.

- With **None**, the simulation results uses both the image texture and the texture mapping optical properties
- **Color from Texture** means that the simulation result uses the color and the color lightness of the image texture.
- **Color from BSDF** means that the simulation result uses the BSDF information of the texture mapping optical properties.

> **Note:** For more information on texture rendering, see Texture Normalization .

Meshing

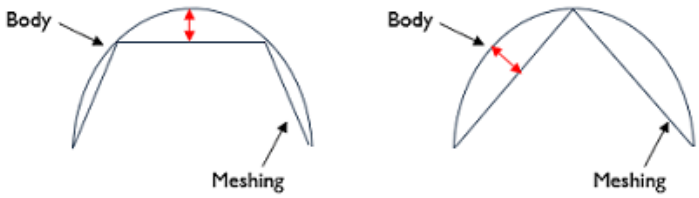


> **Note:** For same values of meshing, meshing results can be different between the CAD platforms in which Speos is integrated.

With the meshing settings, you can lighten memory resources and accelerate simulation in specific cases.

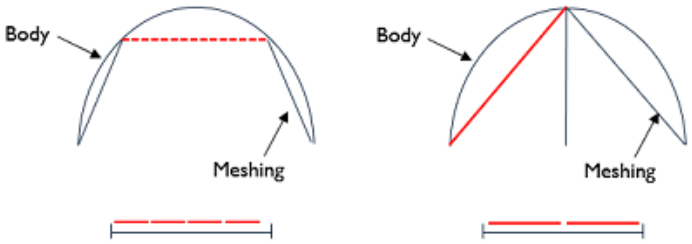
- **Meshing Sag and Step Mode**
  - **Proportional to Face size**: create a mesh of triangles that are proportional to the size of each face of the object. The sag and step value therefore depend on the size of each face.
  - **Proportional to Body size**: create a mesh of triangles that are proportional to the size of the object. The sag and step value therefore depend on the size of the body.
  - **Fixed**: creates a mesh of triangles fixed in size regardless of the size of the body or faces. The mesh of triangles is forced on the object.
- **Meshing sag value**: defines the maximum distance between the object and the geometry.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



> **Note:** If the **Meshing sag value** is too large compared to the body size, Speos recalculate with a **Meshing sag value 128** to better correspond to the body size.

- **Meshing step value**: defines the maximum length of a segment (in mm).



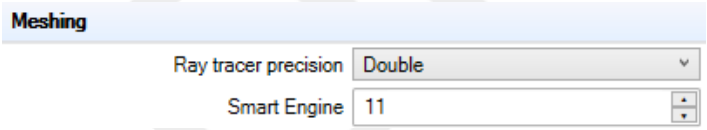
> **Note:** For a Heavyweight body, the **Meshing step value** precision decreases when applying a value below 0.01mm.

- **Meshing angle**: adjusts the angle in the mesh triangle.

If you need more information, see Understanding Meshing Properties.

Simulation

Meshing



Ray tracer precision:

The **Ray tracer precision** is set as **Automatic** by default.

- The **Single Precision** mode allows you to use a fast ray tracing technique that provides a standard level of precision.
- The **Double Precision** mode uses Smart Engine, a ray tracing technique that provides a high level of precision.

  Smart Engine is a pre-calculation method supposed to improve the definition of the rays' impact area in the scene.

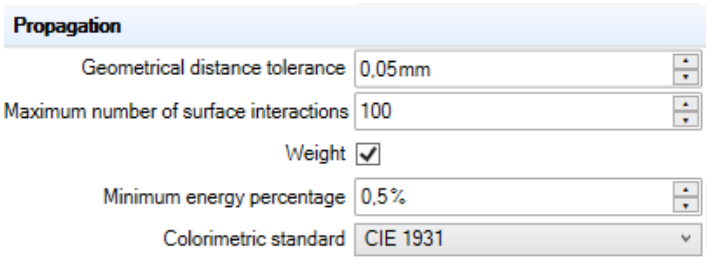
  The scene (the simulation environment comprising the geometries) is subdivided into blocks to help the rays find/locate the elements they need to interact with.

  The Smart Engine value defines a balance between the speed and the memory. The higher the value, the more subdivided the scene becomes.

- The **Automatic** mode lets Speos choose what ray tracing technique to use after a quick analysis of your system. The ray tracing technique is defined according to the bounding box diagonal size.
  - If the bounding box diagonal size is smaller than 10 meters, the **Single Precision** mode is selected.
  - If the bounding box diagonal size is bigger than 10 meters, the **Double Precision** mode is selected.

> **Note:** When choosing the **Automatic** mode, the ray tracing method chosen by Speos is available in the simulation HTML report.

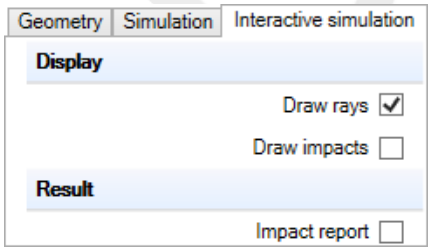
Propagation



- The Geometrical distance tolerance defines the maximum distance to consider two faces as tangent.
- The **Maximum number of surface interactions** allows you to define a value to determine the maximum number of ray impacts during propagation. When a ray has interacted N times with the geometry, the propagation of the ray stops. This option can be useful to stop the propagation of rays in specific optical systems (in an integrated sphere in which a ray is never stopped for example).
- The **Weight** option allows you to activate the consideration of the ray's energy. Each time the rays interacts with a geometry, it loses some energy (weight).
  - The **Minimum energy percentage** value defines the minimum energy ratio to continue to propagate a ray with weight. It helps the solver to better converge according to the simulated lighting system.

> **Note:** For more details, see Setting the Weight Properties.

Interactive Simulation



Display

- **Draw rays**

Allows you to display the rays trajectories in the 3D view. This option is activated by default when creating an interactive simulation.

> **Note:** Deactivate this option when working with Light Expert results to prevent LXP rays and simulation rays from overlapping. For changes to be taken into account, you need to recompute the simulation.

- **Draw impacts**

  Allows you to display the rays impacts in the 3D view.

> **Note:** You can display the rays impact or trajectories alone or together.

Result

Impact report: Allows you to integrate details like number of impacts, position and surface state to the HTML simulation report.

Related information

Creating an Interactive Simulation on page 391
The Interactive Simulation allows you to analyze, visualize and validate the effect of your model on the optical behavior of the light.

Understanding Advanced Simulation Settings on page 429
The following section describes the advanced parameters to set when creating a simulation.

10.3.3. Camera Projected Grid Parameters

The projected grid represents the sensor pixels on the simulation geometry. This projection is done following the camera's distortion.

> **Note:** The projected grid is generated as a result of an interactive simulation containing a camera sensor.

Once a grid is generated, you can edit its parameters. Each time you generate a grid, the grid uses the parameters of the previous generated grid as default parameters.

DRAFT

Connection	
Min distance tolerance	20 mm
Max incidence	85 °
Max distance from camera	5000 mm
Authorize connection between bodies	True
Graduation	
Primary step	10
Secondary step	1
Display graduation text	False
Highlights	
Display first highlighted lines	False
X display first	0
Y display first	0
Display second highlighted lines	False
X display second	0
Y display second	0
Close boundaries	False

Connection

Pixels' Connection

With grid connection parameters, you can connect two adjacent pixels of the grid that do not belong to the same body.

To connect two adjacent pixels, they need to fulfill one of the two parameters **Min distance tolerance** (mm) or **Max incidence** (deg):

- The parameter Min distance tolerance (mm) is having priority over the parameter Max incidence (deg).
- If the two adjacent pixels do not fulfill the parameter Min. distance tolerance (mm), then Speos checks if they fulfill the parameter Max incidence (deg).
- The two adjacent pixels can fulfill both parameters.

Parameters

- **Min distance tolerance** (mm): The distance tolerance for which two adjacent pixels to be connected by a line. Example: for a Min. distance tolerance of 5mm, all adjacent pixels, for which the distance is less than 5mm, are connected by a line.
- **Max incidence**: Maximum angle under which two projected pixels should be connected by a line. Example: for a Max. incidence of 85°, if the angle to the normal (normal of the plane of the two pixels) of the farther pixel from the origin is less than 85°, then the two pixels are connected by a line.



Angle 45°: Connection	Angle 88°: No connection

DRAFT

- **Max distance from camera** (mm): Maximum distance between a pixel and the camera. With maximum distance from camera, you can limit the visualization at a specific distance of the camera.
- **Authorize connection between bodies**: allows you to decide to display the connection between bodies that fulfill one of the parameters (Min. distance tolerance or Max. incidence).

Graduation

With the grid graduations, you can modify the two levels of graduation, **Primary step** (yellow default color) and **Secondary step** (green default color).

To lighten the visualization, we recommend you to increase the graduation step parameters when the grid resolution becomes high.

> **Note:** Setting the graduation steps to zero prevents the display of the grids.

Highlights

These parameters allow to define four lines to highlight on the grid.

10.4. Direct Simulation

The Direct Simulation allows you to propagate a large number of rays from sources to sensors through an optical system.

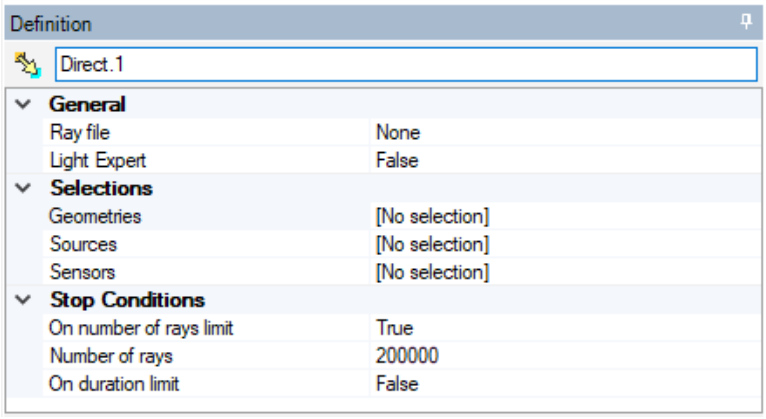
10.4.1. Creating a Direct Simulation

The Direct Simulation is commonly used to analyze standard optical systems.

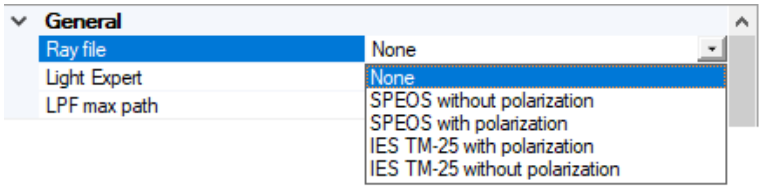
To create a Direct Simulation:

Make sure the Speos inputs used to define the Simulation (example: sources, geometry, sensors) are located in the same component as the simulation, or in a child component.

1. From the **Light Simulation** tab, click **Direct**  .



2. If you want a ray file to be generated at the end of the simulation, from the **Ray File** drop-down list:



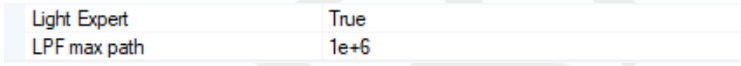
- Select **SPEOS without polarization** to generate a ray file without polarization data.
- Select **SPEOS with polarization** to generate a ray file with the polarization data for each ray.
- Select **IES TM-25 with polarization** to generate a .tm25ray file with polarization data for each ray.
- Select **IES TM-25 without polarization** to generate a .tm25ray file without polarization data.

> **Note:** The size of a ray file approximates 30Mb to 1 Mrays. Consider freeing space on your computer prior to launching the simulation.

> **CAUTION:** If the surface that will be used for the creation of the ray file is fully absorbent (SOP set to Mirror 0%), all rays are absorbed and no ray will be integrated in the ray file. This ray file will be empty.

3. If you want the simulation to generate a Light Path Finder file, set **Light Expert** to **True**.
4. If you activated Light Expert, in **LPF max path**, you can adjust the maximum number of rays the light expert file can contain.

> **Note:** The default value is 1e6 (1 million rays).

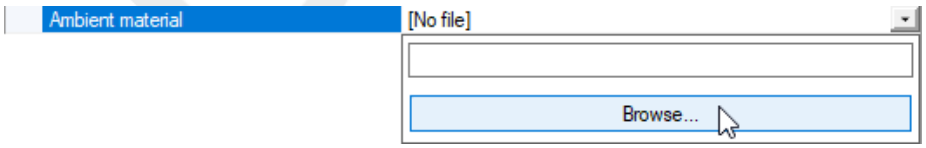


For more information on Light Expert analyses, refer to Light Expert.

5. In **Ambient material**, browse a .material file if you want to define the environment in which the light will propagate (water, fog, smoke etc.).

   The ambient material allows you to specify the media that surrounds the optical system. Including an ambient material in a simulation brings realism to the optical result.

   The .material file is taken into account for simulation.

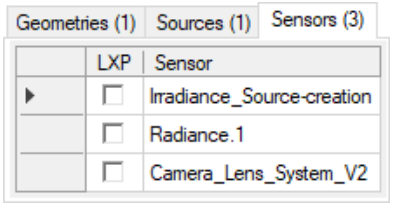


> **Note:** If you do not have the .material file corresponding to the media you want to use for simulation, use the **User Material Editor**, then load it in the simulation.

6. In the 3D view, click  to select geometries,  to select sources and  to select sensors.

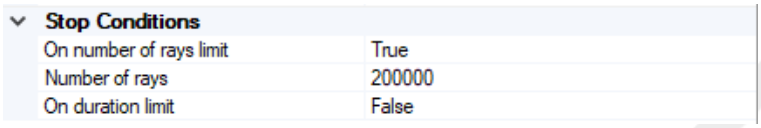
*The selected geometry, source(s) and sensor(s) appear in their respective lists as **Linked objects**.*

7. If Light Expert is activated, you can activate LXP for each sensor contained in the simulation to make a light expert analysis.



In case you add a Light Expert Group for a multi-sensors light expert analysis, the LXP is automatically activated.

8. Define the criteria to reach for the simulation to end:



> **Important:** The maximum amount of rays a sensor's pixel can receive is 16 millions. Beyond this number a saturation effect appears which leads to incorrect results. You can only observe at the end of the simulation if there is a saturation effect. In this case make sure to modify the parameters to avoid such issue.

- To stop the simulation after a certain number of rays were sent, set **On number of rays limit** to **True** and define the number of rays.
- To stop the simulation after a certain duration, set **On duration limit** to **True** and define a duration.

> **Note:** If you activate both criteria, the first condition reached ends the simulation.
>
> If you select none of the criteria, the simulation ends when you stop the process.

> **Note:** If you want to adjust the Direct Simulation advanced settings, see Adjusting Direct Simulation settings.

9. Preview the Meshing of the geometries before running a simulation and adjust meshing values if needed to avoid simulation or geometries errors.

10. In the 3D view, click **Compute**



to launch the simulation.

> **Tip:** To faster compute the simulation using GPU cores, click **GPU Compute** option  .

To compute the simulation in progressive rendering, click **Preview**  . This option opens a new window and displays the result as the simulation is running.

Preview is compatible with NVIDIA GPUs only. GPUs supported are NVIDIA Quadro P5200 or higher.

The Direct Simulation is created along with .xmp results, an HTML report and if Light Expert is activated, a .lpf file. The .xmp result also appears in the 3D view on the sensor(s).

*In case of a Light Field Source used, an Optical Light Field *.olf file is generated.*

If only a Natural Light Ambient Source is used in the Direct Simulation, no power will be noted in the HTML report, as no power is set in the definition of the Source (unlike other sources).

Related concepts

Light Expert on page 465
The Light Expert is a tool that allows you to specify what ray path to display in the 3D view.

Related information

Adjusting Direct Simulation Settings on page 401
This page describes the simulation settings that you can adjust to customize your direct simulation.

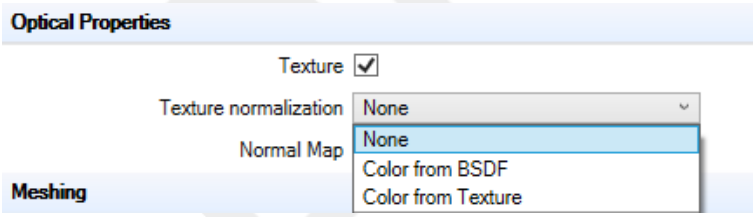
10.4.2. Adjusting Direct Simulation Settings

This page describes the simulation settings that you can adjust to customize your direct simulation.

Right-click the direct simulation and click **Options** to open the advanced settings.

Geometry's Settings

Optical Properties



Texture application can have an impact on the simulation results. If texture have been applied in the scene, activate **Texture** and/or **Normal Map** if needed.

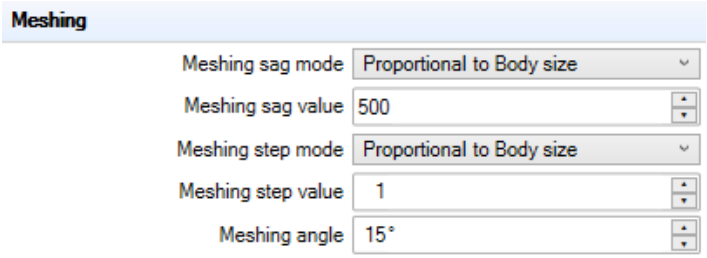
The **Texture normalization** determines the rendering of the texture.

- With **None**, the simulation results uses both the image texture and the texture mapping optical properties.
- **Color from Texture** means that the simulation result uses the color and the color lightness of the image texture.
- **Color from BSDF** means that the simulation result uses the BSDF information of the texture mapping optical properties.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

> **Note:** For more information on texture rendering, see Texture Normalization.

Meshing

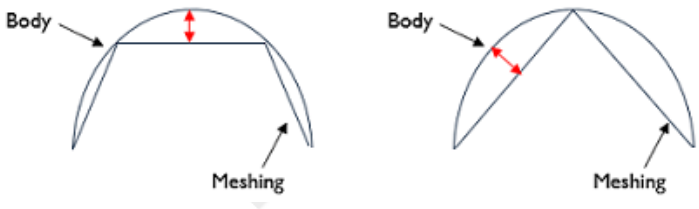


> **Note:** For same values of meshing, meshing results can be different between the CAD platforms in which Speos is integrated.

> **Note:** In case of a thin body, make sure to apply a fixed meshing sag mode and a meshing sag value smaller than the thickness of the body. Otherwise you may generate incorrect results.

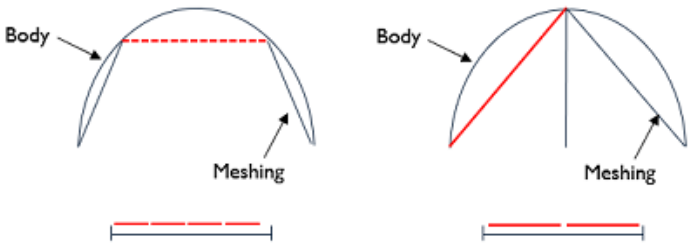
With the meshing settings, you can lighten memory resources and accelerate simulation in specific cases.

- **Meshing Sag and Step Mode**
  - **Proportional to Face size**: create a mesh of triangles that are proportional to the size of each face of the object. The sag and step value therefore depend on the size of each face.
  - **Proportional to Body size**: create a mesh of triangles that are proportional to the size of the object. The sag and step value therefore depend on the size of the body.
  - **Fixed**: creates a mesh of triangles fixed in size regardless of the size of the body or faces. The mesh of triangles is forced on the object.
- **Meshing sag value**: defines the maximum distance between the object and the geometry.



> **Note:** If the **Meshing sag value** is too large compared to the body size, Speos recalculate with a **Meshing sag value 128** to better correspond to the body size.

- **Meshing step value**: defines the maximum length of a segment (in mm).



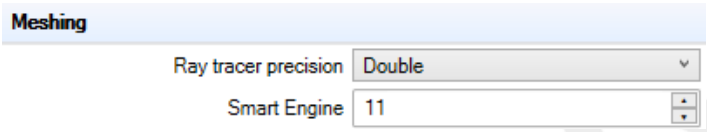
> **Note:** For a Heavyweight body, the **Meshing step value** precision decreases when applying a value below 0.01mm.

- **Meshing angle**: adjusts the angle in the mesh triangle.

If you need more information, see Understanding Meshing Properties.

Simulation

Meshing



Ray tracer precision:

The **Ray tracer precision** is set as **Automatic** by default.

- The **Single Precision** mode allows you to use a fast ray tracing technique that provides a standard level of precision.
- The **Double Precision** mode uses Smart Engine, a ray tracing technique that provides a high level of precision.

  Smart Engine is a pre-calculation method supposed to improve the definition of the rays' impact area in the scene.

  The scene (the simulation environment comprising the geometries) is subdivided into blocks to help the rays find/locate the elements they need to interact with.

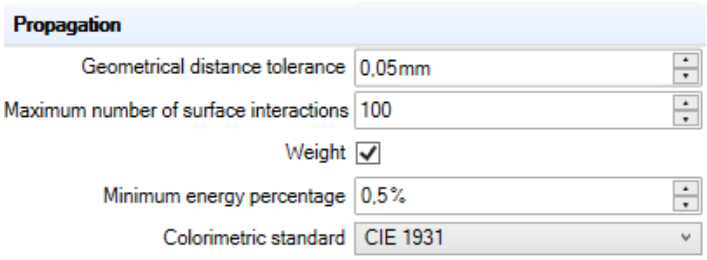
  The Smart Engine value defines a balance between the speed and the memory. The higher the value, the more subdivided the scene becomes.

- The **Automatic** mode lets Speos choose what ray tracing technique to use after a quick analysis of your system. The ray tracing technique is defined according to the bounding box diagonal size.
  - If the bounding box diagonal size is smaller than 10 meters, the **Single Precision** mode is selected.
  - If the bounding box diagonal size is bigger than 10 meters, the **Double Precision** mode is selected.

> **Note:** When choosing the **Automatic** mode, the ray tracing method chosen by Speos is available in the simulation HTML report.

DRAFT

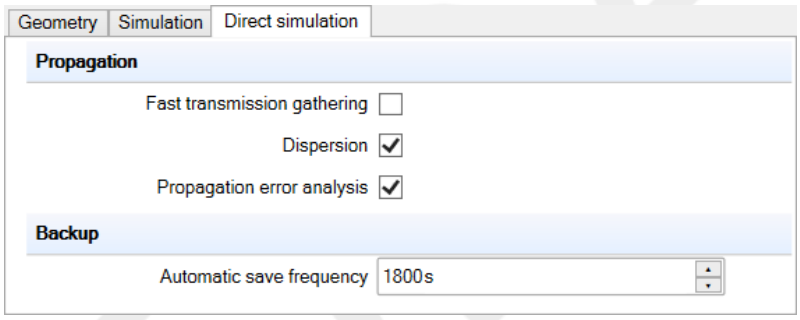
Propagation



- The Geometrical distance tolerance defines the maximum distance to consider two faces as tangent.
- The **Maximum number of surface interactions** allows you to define a value to determine the maximum number of ray impacts during propagation. When a ray has interacted N times with the geometry, the propagation of the ray stops. This option can be useful to stop the propagation of rays in specific optical systems (in an integrated sphere in which a ray is never stopped).
- The **Weight** option allows you to activate the consideration of the ray's energy. Each time the rays interacts with a geometry, it loses some energy (weight).
  - The **Minimum energy percentage** value defines the minimum energy ratio to continue to propagate a ray with weight. It helps the solver to better converge according to the simulated lighting system.

> **Note:** For more details, see Setting the Weight Properties.

Direct Simulation



Propagation

Fast transmission gathering

Fast Transmission Gathering accelerates the simulation by neglecting the light refraction that occurs when the light is being transmitted though a transparent surface.

This option is useful when transparent objects of a scene are flat enough to neglect the refraction effect on the direction of a ray (windows, windshield etc).

> **Note:** Fast Transmission Gathering does not apply to 3D Texture, Polarization Plate and Speos Component Import.

With **Fast transmission gathering** activated:

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- The result is correct only for flat glass (parallel faces).
- The convergence of the results is faster.
- The effect of the refraction on the direction is not taken into account.
- Dispersion is allowed.

Propagation Error Analysis

The Propagation error analysis option helps you identify the root causes and decrease the percentage of errors, by generating a *.lpf file out of the simulation, dedicated to store only the rays in error. Through this file you can display the rays in error in the 3D view and the type of error so that you can correct your optical system.

Backup

Automatic save frequency allows you to define a backup interval. This option is useful when computing long simulations.

> **Note:** A reduced number of save operations naturally increases the simulation performance.

Related information

Creating a Direct Simulation on page 398
The Direct Simulation is commonly used to analyze standard optical systems.

Understanding Advanced Simulation Settings on page 429
The following section describes the advanced parameters to set when creating a simulation.

10.5. Inverse Simulation

The Inverse Simulation allows you to propagate a large number of rays from a camera or a sensor to the sources through an optical system.

10.5.1. Creating an Inverse Simulation

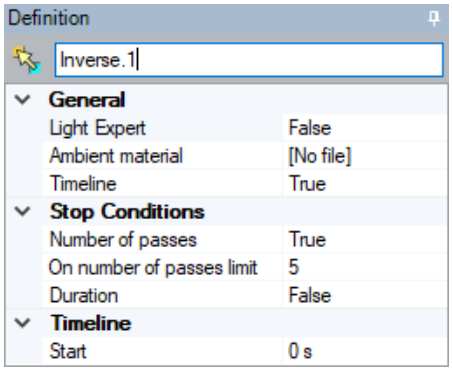
The Inverse Simulation allows you to reverse the light trajectory direction. The propagation is done from the sensors to the sources. It is useful when needing to analyze optical systems where the sensors are small and the sources are diffuse.

To create an Inverse Simulation:

Make sure the Speos inputs used to define the Simulation (example: sources, geometry, sensors) are located in the same component as the simulation, or in a child component.

1. From the **Light Simulation** tab, click **Inverse**  .

DRAFT



2. If you want the simulation to generate a Light Path Finder file, set **Light Expert** to **True**.
3. If you activated Light Expert, in **LPF max path**, you can adjust the maximum number of rays the light expert file can contain.

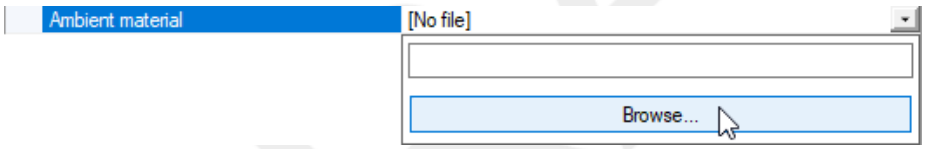
> **Note:** The default value is 1e6 (1 million rays).

For more information on Light Expert analyses, refer to Light Expert.

4. In **Ambient material**, browse a .material file if you want to define the environment in which the light will propagate (water, fog, smoke etc.).

   The ambient material allows you to specify the media that surrounds the optical system. Including an ambient material in a simulation brings realism to the optical result.

   The .material file is taken into account for simulation.



> **Note:** If you do not have the .material file corresponding to the media you want to use for simulation, use the **User Material Editor**, then load it in the simulation.

5. If you want to consider time during the simulation, set **Timeline** to **True**.

> **Note:** This allows dynamic objects, such as Speos Light Box Import and Camera Sensor, to move along their defined trajectories and consider different frames corresponding to the positions and orientations of these objects in simulation.

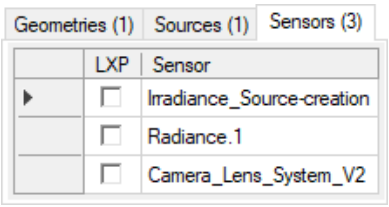
6. In the 3D view, click  to select geometries,  to select sources and  to select sensors.

   *The selected geometry, source(s) and sensor(s) appear in their respective lists as **Linked objects**.*

> **Note:**

- When you include Geometric Camera Sensors, you cannot include another sensor type and you must not select a source. When you include Photometric / Colorimetric Camera Sensors, you must select a source.
- If you include an Irradiance sensor in a non-Monte-Carlo Inverse simulation, geometries are considered as absorbent.

7. If Light Expert is activated, you can activate LXP for each sensor contained in the simulation to make a light expert analysis.



8. If some sources of your scene generate light that needs to pass through specific faces to reach the sensor (for example if you have an ambient source that needs to pass through the windshield), define out path faces/sources:

   a. In the 3D view, click  and select the face(s) to be considered as out path faces.

   The faces are selected and make the interface between the interior and exterior environments.

   b. In the 3D view, click  and select the source(s) to be considered as out sources.

   The sources are selected and are linked to the out path faces selected.

9. If the **Optimized propagation** algorithm is set to **None**, in **Stop Conditions**, define the criteria to reach for the simulation to end:

   For more information on how to set **Optimized propagation**, refer to Monte Carlo Calculation Properties on page 413.

   - To stop the simulation after a certain number of rays were sent, set **On number of passes limit** to **True** and define the number of passes.

     In case of a GPU simulation, the number of passes defined corresponds to the minimum threshold of rays received per pixel. That means while a pixel has not received this minimum number of rays, the simulation still runs.

     Note: If you activated the dispersion, this minimum threshold is multiplied by the sensor sampling. That means for a number of pass of 100 and a sampling of 13, each pixel will need to receive at least 1300 rays (the progress of the number of rays per pixel is indicated in the Simulation Progress Bar).

     In case of a CPU simulation, the number of pass, there is no minimum threshold of rays to be received per pixel, the simulation stops when the number of pass is reached.

   - To stop the simulation after a certain duration, set **On duration limit** to **True** and define a duration.

     Note: If you activate both criteria, the first condition to be reached ends the simulation.

DRAFT

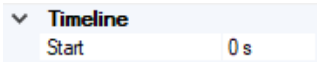
If you select none of the criteria, the simulation ends when you stop the process.

Note: If you want to adjust the Inverse Simulation advanced settings, see Adjusting Inverse Simulation settings.

10. If the **Optimized propagation** algorithm is set to **Relative** or **Absolute**, in **Stop Conditions**, define the **Absolute stop value** or **Relative stop value** to reach for the simulation to end.

    For more information on how to set **Optimized propagation**, refer to Monte Carlo Calculation Properties on page 413.

11. If **Timeline** is set to **True**, from the **Timeline** section, define the **Start** time.



Warning: From the version 2022 R2, the Timeline **Start** have been improved in order to enter time values below 1ms. This change will impact your custom scripts, and will need to be modified consequently, as previous time format is no longer supported.

12. Preview the Meshing of the geometries before running a simulation and adjust meshing values if needed to avoid simulation or geometries errors.

13. In the 3D view, click **Compute** to launch the simulation.

Tip: To faster compute the simulation using GPU cores, click **GPU Compute** option .

To compute the simulation in progressive rendering, click **Preview** . This option opens a new window and displays the result as the simulation is running.

Preview is compatible with NVIDIA GPUs only. GPUs supported are NVIDIA Quadro P5200 or higher.

The Inverse Simulation is created along with .xmp results, an HTML report and if Light Expert is activated, a .lpf file. The .xmp result also appears in the 3D view on the sensor(s).

If Timeline is not activated and the Inverse Simulation uses a Camera sensor, only a HDRI file (.hdr) is generated.*

If Timeline is activated and a Camera sensor is moving, the Inverse Simulation is considered as dynamic and only generates a spectral exposure map for each sensor, even the static sensors. This map corresponds to the acquisition of the camera sensor and expresses the data for each pixel in Joules/m²/nm.

Related concepts

Light Expert on page 465
The Light Expert is a tool that allows you to specify what ray path to display in the 3D view.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Related information

Adjusting Inverse Simulation Settings on page 409
This page describes the simulation settings that you can adjust to customize your inverse simulation.

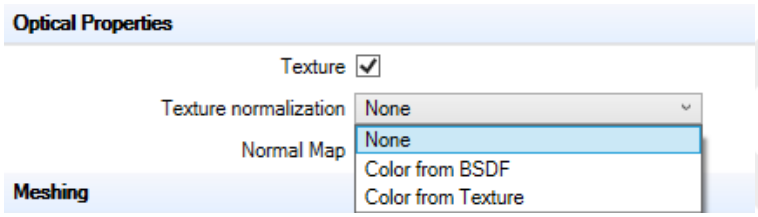
10.5.2. Adjusting Inverse Simulation Settings

This page describes the simulation settings that you can adjust to customize your inverse simulation.

Right-click the Inverse simulation and click **Options** to open the advanced settings.

Geometry's Settings

Optical Properties



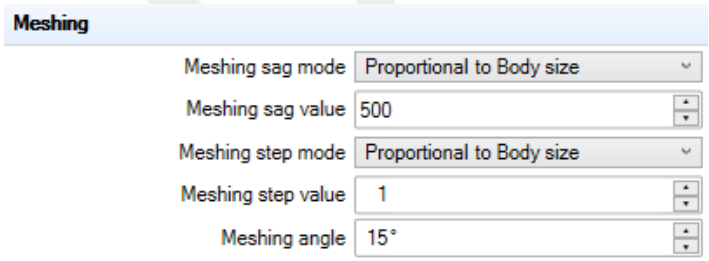
Texture application can have an impact on the simulation results. If texture have been applied in the scene, activate **Texture** and/or **Normal Map** if needed.

The **Texture normalization** determines the rendering of the texture.

- With **None**, the simulation results uses both the image texture and the texture mapping optical properties
- **Color from Texture** means that the simulation result uses the color and the color lightness of the image texture.
- **Color from BSDF** means that the simulation result uses the BSDF information of the texture mapping optical properties.

> **Note:** For more information on texture rendering, see Texture Normalization .

Meshing

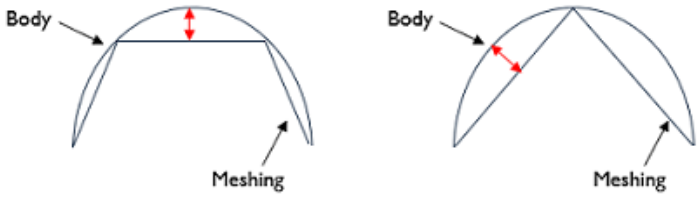


> **Note:** For same values of meshing, meshing results can be different between the CAD platforms in which Speos is integrated.

> **Note:** In case of a thin body, make sure to apply a fixed meshing sag mode and a meshing sag value smaller than the thickness of the body. Otherwise you may generate incorrect results.

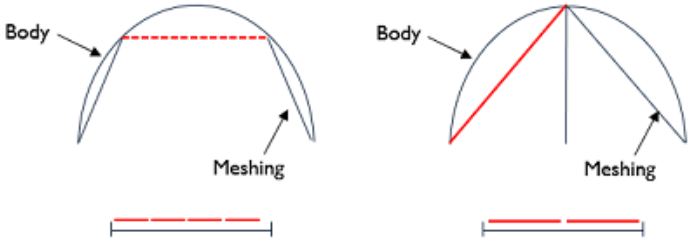
With the meshing settings, you can lighten memory resources and accelerate simulation in specific cases.

- **Meshing Sag and Step Mode**
  - **Proportional to Face size**: create a mesh of triangles that are proportional to the size of each face of the object. The sag and step value therefore depend on the size of each face.
  - **Proportional to Body size**: create a mesh of triangles that are proportional to the size of the object. The sag and step value therefore depend on the size of the body.
  - **Fixed**: creates a mesh of triangles fixed in size regardless of the size of the body or faces. The mesh of triangles is forced on the object.
- **Meshing sag value**: defines the maximum distance between the object and the geometry.



> **Note:** If the **Meshing sag value** is too large compared to the body size, Speos recalculate with a **Meshing sag value 128** to better correspond to the body size.

- **Meshing step value**: defines the maximum length of a segment (in mm).



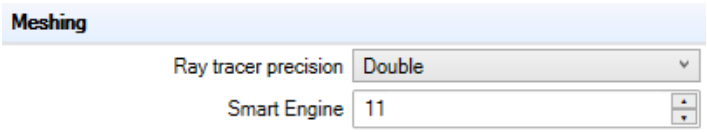
> **Note:** For a Heavyweight body, the **Meshing step value** precision decreases when applying a value below 0.01mm.

- **Meshing angle**: adjusts the angle in the mesh triangle.

If you need more information, see Understanding Meshing Properties.

Simulation

Meshing



Ray tracer precision:

The **Ray tracer precision** is set as **Automatic** by default.

- The **Single Precision** mode allows you to use a fast ray tracing technique that provides a standard level of precision.
- The **Double Precision** mode uses Smart Engine, a ray tracing technique that provides a high level of precision.

  Smart Engine is a pre-calculation method supposed to improve the definition of the rays' impact area in the scene.

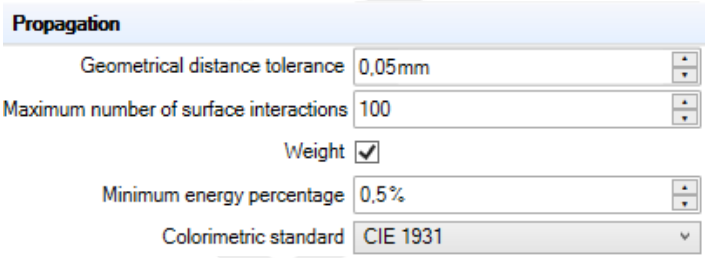
  The scene (the simulation environment comprising the geometries) is subdivided into blocks to help the rays find/locate the elements they need to interact with.

  The Smart Engine value defines a balance between the speed and the memory. The higher the value, the more subdivided the scene becomes.

- The **Automatic** mode lets Speos choose what ray tracing technique to use after a quick analysis of your system. The ray tracing technique is defined according to the bounding box diagonal size.
  - If the bounding box diagonal size is smaller than 10 meters, the **Single Precision** mode is selected.
  - If the bounding box diagonal size is bigger than 10 meters, the **Double Precision** mode is selected.

> **Note:** When choosing the **Automatic** mode, the ray tracing method chosen by Speos is available in the simulation HTML report.

Propagation



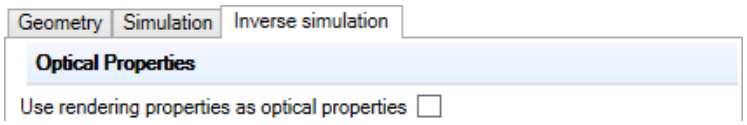
- The Geometrical distance tolerance defines the maximum distance to consider two faces as tangent.
- The **Maximum number of surface interactions** allows you to define a value to determine the maximum number of ray impacts during propagation. When a ray has interacted N times with the geometry, the propagation of the ray stops. This option can be useful to stop the propagation of rays in specific optical systems (in an integrated sphere in which a ray is never stopped).
- The **Weight** option allows you to activate the consideration of the ray's energy. Each time the rays interacts with a geometry, it loses some energy (weight).
  - The **Minimum energy percentage** value defines the minimum energy ratio to continue to propagate a ray with weight. It helps the solver to better converge according to the simulated lighting system.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Note: For more details, see Setting the Weight Properties.

Inverse Simulation

Optical Properties



Activating **Use rendering properties as optical properties** allows you to automatically convert appearance properties into physical parameters according to the following conversion table.

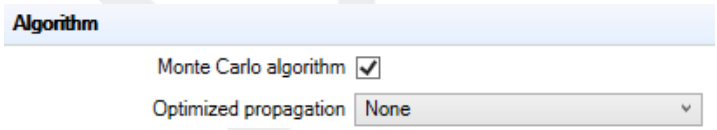
Appearance Parameters	Physical Parameters PP(l)
Intensity + Color[RGB]	Lambertian L(l)
Ambient + Color[RGB]	Lambertian L(l)
Shine	Gaussian Angle a
Highlight + Highlight[RGB]	Gaussian Reflection
Reflection	Specular reflection SR(l)
Transparency + Highlight[RGB]	Specular transmission ST(l)

Algorithm

With the inverse simulation, you can select the calculation algorithm used to interpret your optical system.

You can choose between the Monte Carlo algorithm and a deterministic calculation. This selection impacts the parameters to set in the **Propagation** section.

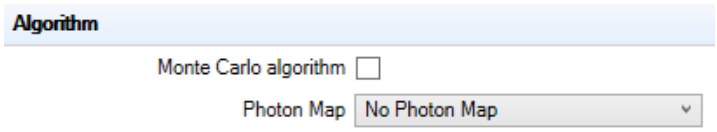
- The **Monte Carlo** algorithm is a randomized algorithm that allows you to perform probabilistic simulations. It allows you to manage dispersion, bulk diffusion, multiple diffuse inter-reflections and supports light expert analysis.



Note: To define the propagation settings, see Monte Carlo Calculation Properties .

- The deterministic algorithm allows you to perform determinist simulations that produce results showing little to no noise but that are considered as biased. This algorithm does not manage dispersion, bulk diffusion or light expert analysis. You can create a deterministic simulation with or without generating a photon map.

DRAFT



Note: To define the propagation settings, see Deterministic Calculation Properties .

Related information

Creating an Inverse Simulation on page 405
The Inverse Simulation allows you to reverse the light trajectory direction. The propagation is done from the sensors to the sources. It is useful when needing to analyze optical systems where the sensors are small and the sources are diffuse.

Understanding Advanced Simulation Settings on page 429
The following section describes the advanced parameters to set when creating a simulation.

10.5.3. Calculation Properties

The following section describes the propagation settings for both the Monte Carlo and Deterministic algorithms.

10.5.3.1. Monte Carlo Calculation Properties

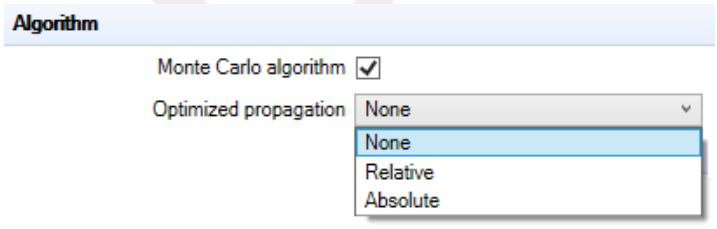
This page describes the Monte Carlo Calculation Properties to set when creating an inverse simulation.

The Monte Carlo algorithm is a randomized algorithm that allows you to perform probabilistic simulations.

This algorithm is reliable, efficient and suits a lot of configurations but can, according to your configuration, take a certain amount of time to compute.

Note: In a Monte Carlo inverse simulation, if the absorption value of a BRDF is negative, it is considered as a null value.

Optimized Propagation



Note: The Optimized propagation algorithm is only compatible with the Radiance sensors.

The **Optimized Propagation** algorithms consist in sending rays from each pixel of the sensor until one of the stopping criteria is reached.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- **None**: the same number of passes is used for each pixel of the image (current and default algorithm).

  This algorithm may generate unbalanced results. Some pixels may have a good signal-to-noise ratio (SNR) whereas some other pixels may show too much noise.

- **Relative** and **Absolute**: the algorithm adapts the number of passes per pixel to send the optimal number of rays according to the signal each pixel needs. As a result, the SNR is adequate in areas where pixels need more rays thus giving a balanced image.

  These two modes are based on the same principle, however the method of calculation is slightly different.

  - In **Relative**, the pixel's standard deviation is compared with a threshold value (error margin expressed in percentage) defined by the user. This value determines the error margin (standard deviation) tolerated. Rays will be launched until the standard deviation of the pixel is lower than the defined threshold value. All the values of the map are then known with the same "relative" precision.

    The standard deviation is normalized (expressed as an average) and compared to the threshold value (percentage).

    The pixel's standard deviation is computed on each pixel according to the following formula:

    $$\frac{\hat{\sigma}_N}{\hat{\theta}} < \sigma_r$$

    σ_N: Estimate of the standard deviation relative to the number of rays (N).

    θ: average signal of the map.

    σ_r: User defined standard deviation.

    The greater N is (the more rays are sent), the more the σ (standard deviation) converges to the threshold value (σ_A). Here the standard deviation is normalized by an average (θ).

  - In **Absolute**, the pixel's value is simply compared with a fixed threshold value (photometric value) defined by the user. This photometric unit determines the error margin (standard deviation) tolerated for each pixel of the map. Rays will be launched until the standard deviation of the pixel is lower than the defined threshold value. All the values are thereby known with the same precision.

    The standard deviation is simply compared to the threshold value.

    The pixel's standard deviation is computed on each pixel according to the following formula:

    $$\hat{\sigma}_N < \sigma_A$$

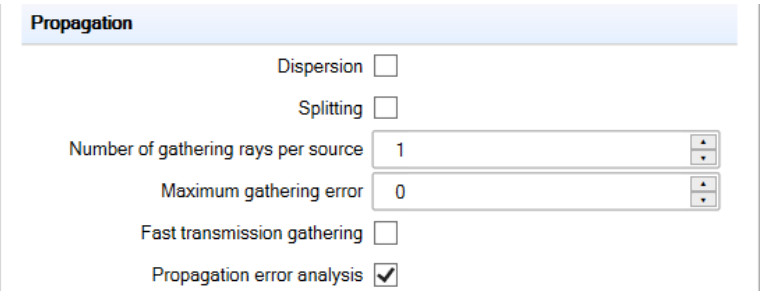
    σ_N: Estimate of the standard deviation relative to the number of rays (N).

    σ_A: User defined standard deviation.

    The greater N is (the more rays are sent), the more the σ (standard deviation) converges to the threshold value (σ_A).

Number of standard passes before optimized passes: corresponds to the minimum number of passes without pass optimization (standard pass : all pixels emit rays. Optimized pass : pixels with standard deviation higher than defined threshold emit rays).

DRAFT



Dispersion

With this parameter, you can activate the dispersion calculation. In optical systems in which the dispersion phenomena can be neglected, the colorimetric noise is canceled by deactivating this parameter.

> **Note:** This parameter should not be used with simulations involving diffusive materials.
>
> For more details, refer to Dispersion.

Splitting

This option is useful when designing tail lamps.

Splitting allows you to split each propagated ray into several paths at their first impact after leaving the observer point. Further impacts along the split paths do not provide further path splitting. This feature is primarily intended to provide a faster noise reduction on scenes with optical polished surface as the first surface state seen from the observer. An observer watching a car rear lamp is a typical example of such a scene.

> **Note:** The split is only done to the first impact. Two rays are split on an optical polished surface. The 2 rays are weighted using Fresnel's law. On other surfaces there may be more or less split rays depending on the surface model.
>
> We are considering either the transmitted or the reflected ray (only one of them, we pick one each time). The choice (R or T) is achieved using Monte Carlo: the probability for reflection is the Fresnel coefficient for reflection. So depending on the generated random number, the ray will be either reflected or transmitted.

Close to the normal incidence, the reflection probability is around 4%, which is low. This low probability makes that when we want to see the reflection of the environment, we observe a lot of noise. The splitting algorithm removes this noise by computing the first interaction without using Monte Carlo.

Number of gathering rays per source

In inverse simulations, each ray is propagated from the observer point through the map and follows a random path through the system.

There is often a very small probability for a ray to hit a light source on its own. To increase this probability, new rays are generated at each impact on diffuse surfaces. These rays are called *shadow rays.* They are targeted to each light source in the system and the program check whether a direct hit on the sources is possible or not. If not, nothing happens. If the program finds a hit, it computes the corresponding radiance to store in the map.

The **Number of gathering rays per source** parameter pilots the number of shadow rays to target at each source.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



Maximum Gathering Error

With this parameter, you can reduce the simulation time for scenes with large number of sources and where each source contributes to illuminate a small area of the scene. The value set here defines the level below which a source can be neglected. For instance a value of 10 means that all sources contributing less than 10% to the illumination of all sources are not taken in consideration. 0, the default value means that no approximation will be done.

> **Note:** You must take some precautions by using layer operations tool of the **Virtual Photometric Lab**. For instance if **maximum gathering error** is defined at 1% for a simulation and if the flux of a source is increased 10 times with the layer operations tool, it means that maximum gathering error is now 10% for this source.

Fast Transmission Gathering

Fast Transmission Gathering accelerates the simulation by neglecting the light refraction that occurs when the light is being transmitted though a transparent surface.

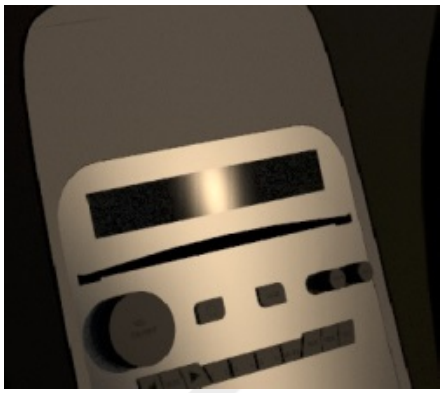
This option is useful when transparent objects of a scene are flat enough to neglect the refraction effect on the direction of a ray (windows, windshield etc).

> **Note:** Fast Transmission Gathering does not apply to 3D Texture, Polarization Plate and Speos Component Import.

With **Fast transmission gathering** activated:

- The result is right only for flat glass (parallel faces).
- The convergence of the results is faster.
- The effect of the refraction on the direction is not taken into account.
- Dispersion is allowed.

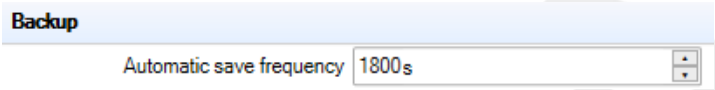
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

	
5 minutes - 25 passes	5 minutes - 35 passes

Propagation Error Analysis

The Propagation error analysis option helps you identify the root causes and decrease the percentage of errors, by generating a *.lpf file out of the simulation, dedicated to store only the rays in error. Through this file you can display the rays in error in the 3D view and the type of error so that you can correct your optical system.

Intermediate Save Frequency



During an inverse simulation, intermediate results can be saved. Setting an intermediate save frequency is useful when computing long simulations and wanting to check intermediate results.

Setting this option to 0 means that the result is saved only at the end of the simulation. If the simulation is stopped without finishing the current pass, no result is available.

> **Note:** A reduced number of save operations naturally increases the simulation performance.

In the case of high sensor sampling the save operation can take the half of the simulation time when automatic save frequency is set to 1.

Related information

Adjusting Inverse Simulation Settings on page 409
This page describes the simulation settings that you can adjust to customize your inverse simulation.

Deterministic Calculation Properties on page 417
This section describes the Deterministic calculation properties to set when creating an inverse simulation.

10.5.3.2. Deterministic Calculation Properties

This section describes the Deterministic calculation properties to set when creating an inverse simulation.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

10.5.3.2.1. Understanding Photon Mapping for a Deterministic Simulation

The Deterministic simulation is fast and targeted but is best suited to analyze simple optical paths. When contributions and optical interactions are multiple, you should use Monte Carlo algorithm or generate a photon map.

Photon Mapping

Photon Mapping is a luminance algorithm that allows you to realistically simulate and render the interaction of light with objects.

This algorithm takes into account diffuse inter reflections, caustics and surface contributions of the optical system.

The Photon Mapping process is the following:

- The first step of photon mapping is a photon propagation phase. A first pass is done using a Monte Carlo direct simulation to send photons from sources into the scene. Photons are then stored in a map.
- The second pass, called the Gathering phase, is a deterministic inverse simulation. The photon map from the first pass is used to compute local radiance.

At the end of the simulation, a noise map is generated and can be reused for future simulations.

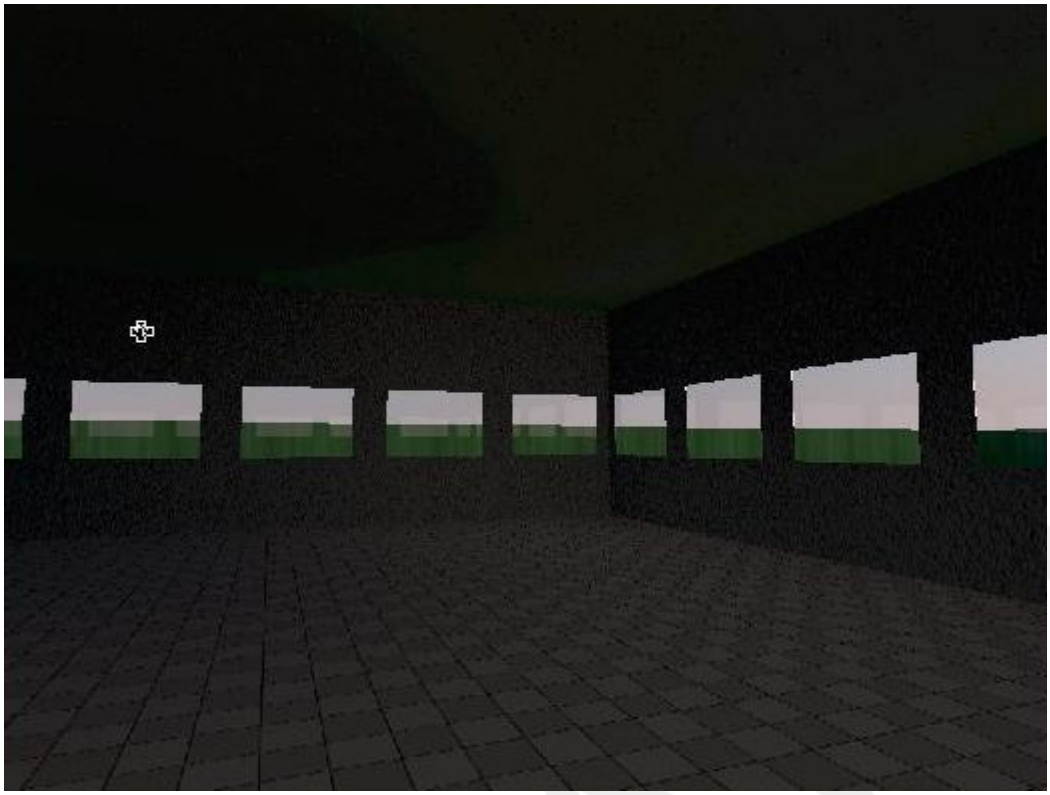
> **Note:** Photon Mapping produces "biased" results.

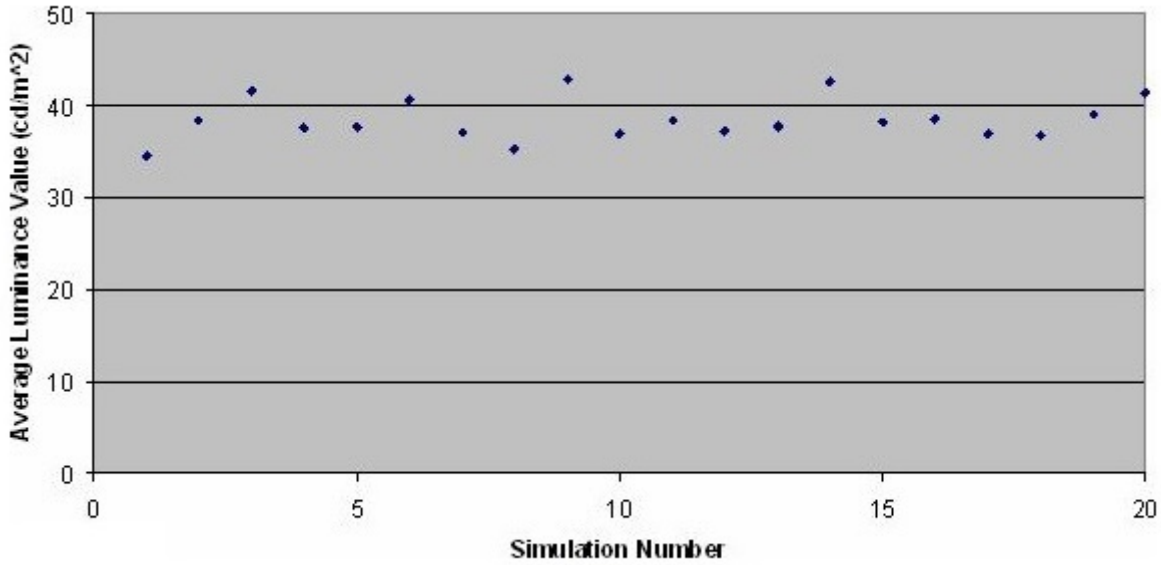
Simulation Results

As the first pass corresponds to a Monte Carlo direct simulation, photons are randomly drawn. As a result, photon deposition on scene parts are different from one simulation to another, implying different photometric results in localized measurements as shown below.

For example, considering a 10x10 mm^2 ellipsoid measurement area, and running 20 simulations, a 2.3 cd/m^2 standard deviation is obtained on this measurement series.

DRAFT





Related information

Deterministic Calculation Properties without Photon Map on page 420
This page describes the parameters to set when wanting to create a deterministic inverse simulation without generating a photon map.

Deterministic Calculation Properties with Photon Map on page 422
This page describes the parameters to set when wanting to create a deterministic inverse simulation while generating a photon map.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

10.5.3.2.2. Deterministic Calculation Properties without Photon Map

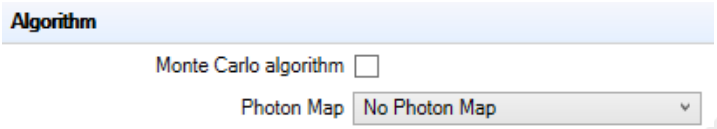
This page describes the parameters to set when wanting to create a deterministic inverse simulation without generating a photon map.

The deterministic algorithm allows you to perform determinist simulations. This type of simulation produces results showing little to no noise but that are considered as biased.

The deterministic simulation is entirely appropriate to analyze and render simple optical paths with unidirectional contributions.

When contributions and optical interactions are multiple, it is recommended to use Monte Carlo algorithm or generate a **photon map**.

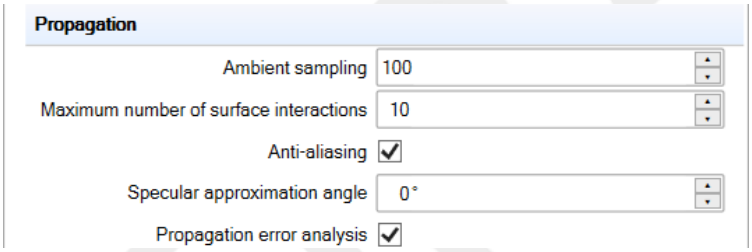
Algorithm



> **Note:** A simulation without photon map avoids noise but does not manage the diffuse inter-reflections. Only color are taken into account.

Example of simulation without photon map

Propagation Properties



Use Rendering Properties as Optical Properties

Activating this option allows you to automatically convert appearance properties into physical parameters according to the following conversion table.

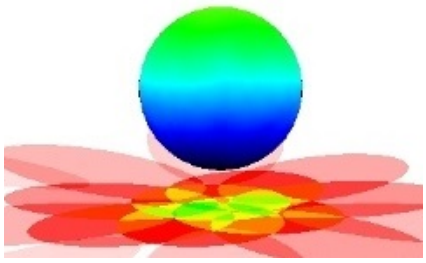
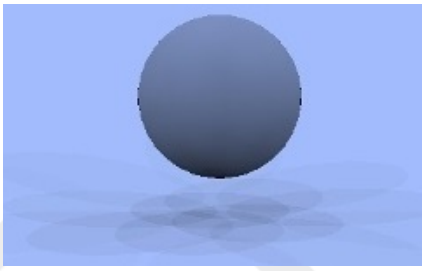
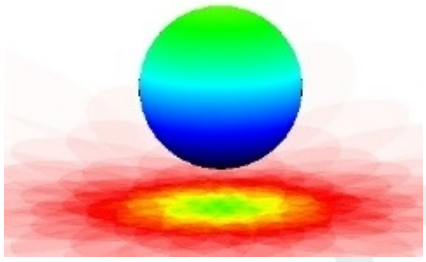
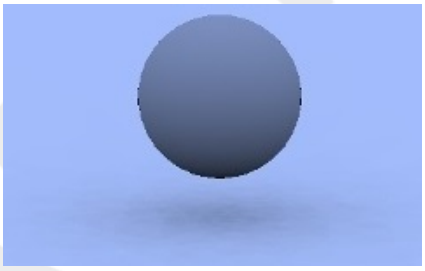
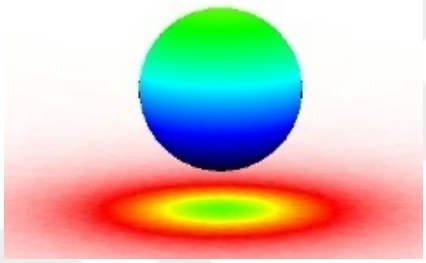
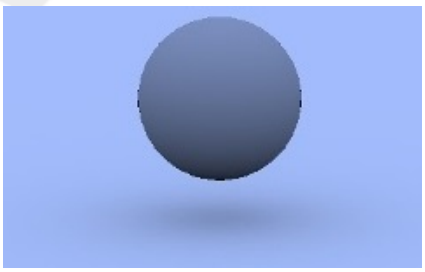
Appearance Parameters	Physical Parameters PP(l)
Intensity + Color[RGB]	Lambertian L(l)
Ambient + Color[RGB]	Lambertian L(l)
Shine	Gaussian Angle a
Highlight + Highlight[RGB]	Gaussian Reflection
Reflection	Specular reflection SR(l)
Transparency + Highlight[RGB]	Specular transmission ST(l)

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Ambient Sampling

This parameter defines the sampling. The sampling corresponds to the quality of the ambient source. The greater the value, the better the quality of the result, but the longer the simulation. The following table gives some ideas of the balance between quality and time.

> **Note:** A default value could be 20 and a value for good results could be 100.

Ambient Sampling = 20 Reference Time / 3		
Default value Ambient Sampling = 100 Reference Time		
Ambient Sampling = 500 Reference Time x 4		

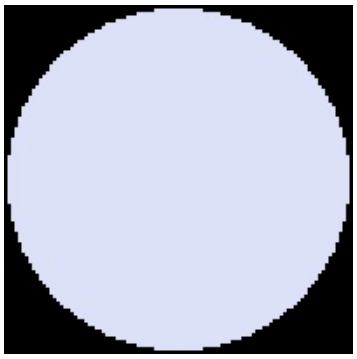
Maximum Number of Surface Interactions

This number defines the maximum number of impacts a ray can make during propagation. Once the ray has interacted N times with a surface, it is stopped.

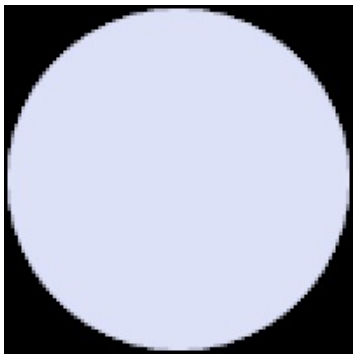
Anti-Aliasing

The anti-aliasing option allows you to reduce artifacts such as jagged profiles and helps refine details. However, this option tends to increase simulation time.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



Anti-aliasing deactivated Reference Time / 2



Default Value Anti-aliasing activated Reference Time

Specular Approximation Angle

The specular approximation angle option allows you to replace the specular reflection by a gaussian reflection to increase the probability of the propagated rays to reach the sources.

This option also allows you to decrease the noise in the simulation's results and improve simulation time.

The typical application is the rendering of automotive tail lamps lit appearance. For this application, a typical value would be 5 to 10 degrees.

> **Note:** For more information, see Specular Approximation Angle.

Related information

Monte Carlo Calculation Properties on page 413
This page describes the Monte Carlo Calculation Properties to set when creating an inverse simulation.

Deterministic Calculation Properties with Photon Map on page 422
This page describes the parameters to set when wanting to create a deterministic inverse simulation while generating a photon map.

10.5.3.2.3. Deterministic Calculation Properties with Photon Map

This page describes the parameters to set when wanting to create a deterministic inverse simulation while generating a photon map.

The deterministic algorithm allows you to perform determinist simulations. This type of simulation produces results showing little to no noise but that are considered as biased.

The deterministic simulation is entirely appropriate to analyze and render simple optical paths with unidirectional contributions.

When contributions and optical interactions are multiple, you should use Monte Carlo algorithm or generate a photon map.

> **Note:** A simulation with build photon map generates map noises and manages diffuse inter-reflections.

DRAFT

It is safe to use photon maps when surfaces are lambertian, diffuse or specular.

It is not safe to use photon maps when surfaces are gaussian with a small FWHM angle.

Photon Mapping

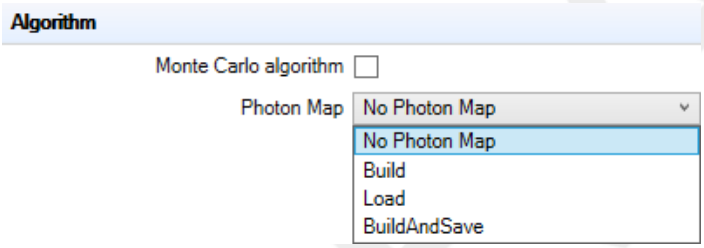
The Photon Mapping is a luminance algorithm used to take into account multiples diffuse inter-reflections. But in the preset case, it is a two pass algorithm.

- The first step of photon mapping is a photon propagation phase. A first pass is done using a Monte Carlo direct simulation to send photons from sources into the scene. Photons are then stored in a map.
- The second pass is a deterministic inverse simulation and is called the Gathering phase. The photon map from the first pass is used to compute local radiance.

Note: If you need more information on Photon Mapping, see Understanding Photon Mapping for a Deterministic Simulation .

Algorithm

Three modes are available with Photon Mapping:



- **Build** allows you to fully generate a photon map.
- **Load** allows you to use a previously generated map.
- **BuildAndSave** allows you to fully generate and store a photon map.

The parameters to set may vary depending on the option selected. When loading an existing map, less parameters need to be set.

Note: The parameters described in the following sections correspond to the **Build** photon map mode.

Propagation Properties

Use Rendering Properties as Optical Properties

Activating this option allows you to automatically convert appearance properties into physical parameters according to the following conversion table.

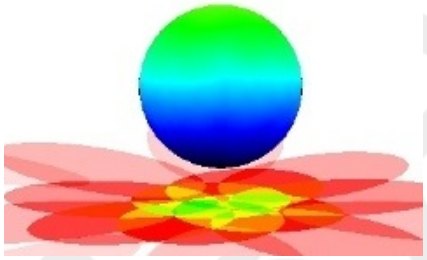
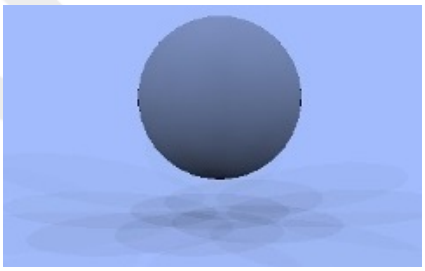
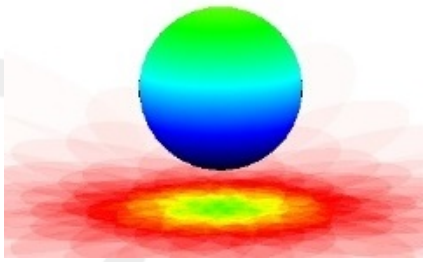
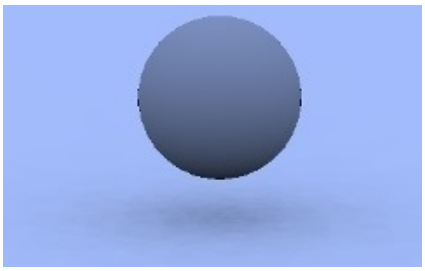
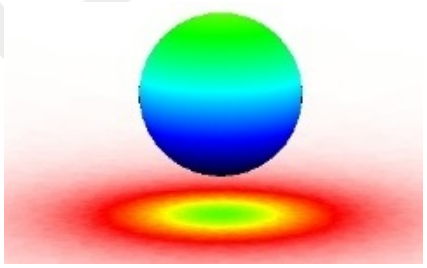
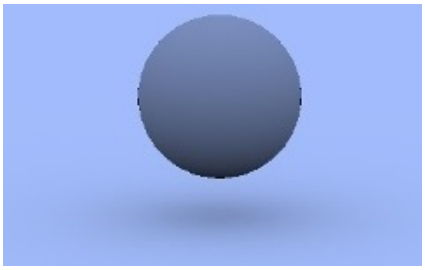
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Appearance Parameters	Physical Parameters PP(l)
Intensity + Color[RGB]	Lambertian L(l)
Ambient + Color[RGB]	Lambertian L(l)
Shine	Gaussian Angle a
Highlight + Highlight[RGB]	Gaussian Reflection
Reflection	Specular reflection SR(l)
Transparency + Highlight[RGB]	Specular transmission ST(l)

Ambient Sampling

This parameter defines the sampling. The sampling corresponds to the quality of the ambient source. The greater the value, the better the quality of the result, but the longer the simulation. The following table gives some ideas of the balance between quality and time.

> **Note:** A default value could be 20 and a value for good results could be 100.

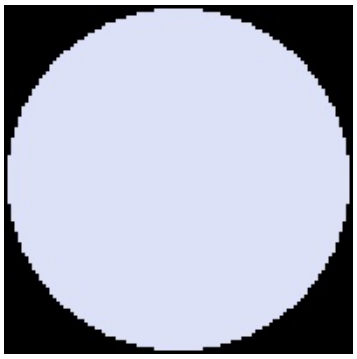
Ambient Sampling = 20 Reference Time / 3		
Default value Ambient Sampling = 100 Reference Time		
Ambient Sampling = 500 Reference Time x 4		

Maximum Number of Surface Interactions

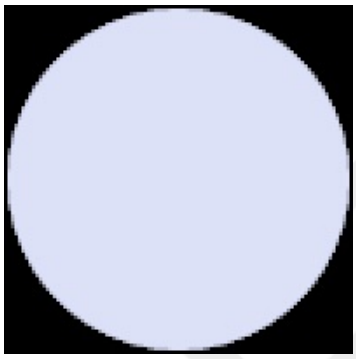
This number defines the maximum number of impacts a ray can make during propagation. Once the ray has interacted N times with a surface, it is stopped.

Anti-Aliasing

The anti-aliasing option allows you to reduce artifacts such as jagged profiles and helps refine details. However, this option tends to increase simulation time.



Anti-aliasing deactivated Reference Time / 2



Default Value Anti-aliasing activated Reference Time

Specular Approximation Angle

The specular approximation angle option allows you to replace the specular reflection by a gaussian reflection to increase the probability of the propagated rays to reach the sources.

This option also allows you to decrease the noise in the simulation's results and improve simulation time.

The typical application is the rendering of automotive tail lamps lit appearance. For this application, a typical value would be 5 to 10 degrees.

> **Note:** For more information, see Specular Approximation Angle.

Numbers of Photons Launched in Direct Phase

This number represents the number of rays sent in the direct phase.

DRAFT



Direct Photon Number = 10000



Direct Photon Number = 5000

Maximum Number of Surface Interactions in Direct Phase

This number defines the maximum number of impacts a ray can make during the propagation phase. Once the ray has interacted N times with a surface, it is stopped.

Max Neighbors

Max neighbors represents the number of photons from the photon map taken into account to calculate the luminance.



Default



Double neighbors number

Max Search Radius

Max search radius represents the maximum distance from the luminance calculation's point to search for neighbors contribution.

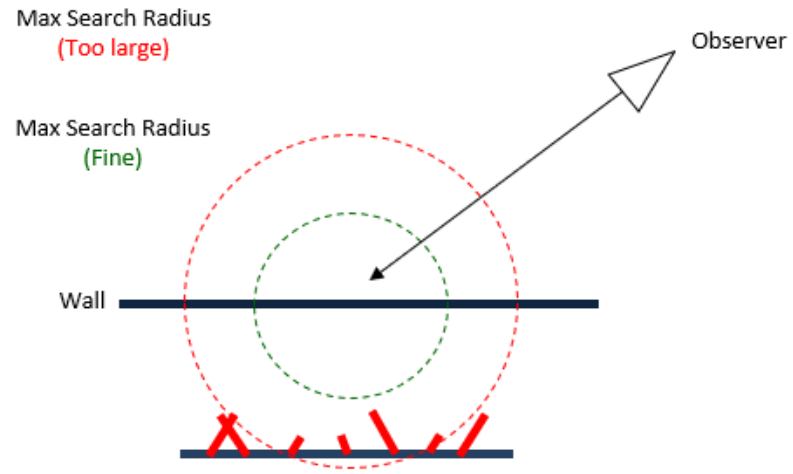
The **Max search radius** parameter could have a strong impact on the results according to the **Max neighbors** parameter setting.

A balance shall be found between these two parameters to ensure good calculation and rendering.

Consider, for example, a wall with one face illuminated and the other face not illuminated and that does not transmit any light.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

In the case of a sensor observing the no transmitting face, if the **Max search radius** is higher than the depth of the wall, the sensor gives some luminance values corresponding to the illuminated side of the wall.





Infinite max search radius

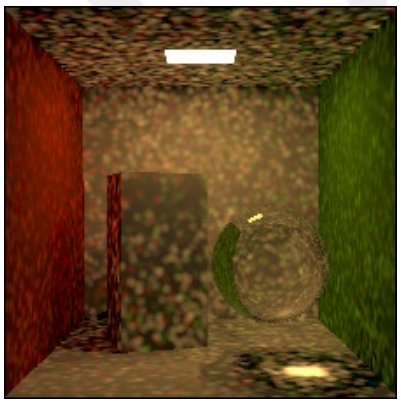
Max search radius equals to the depth of the walls

In the following example, the effect of a too large max search radius in simulation results is described.

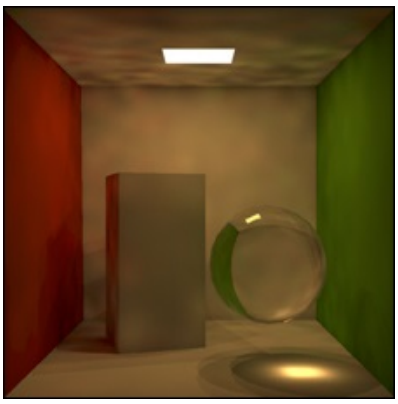
Note the white dots on the right illustration. They result from an unbalanced relationship between **max search radius** and **max neighbors**.

For a given **max neighbors** value, if the **max search radius** is too small, the sensor does not collect enough neighbors and generates noisy result.

And vice versa if the max search radius is fixed and the **max neighbors** value is too high, the sensor gathers all the neighbors but there are not enough information on the defined area of research.



Max search radius = 10



Max search radius = 100

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

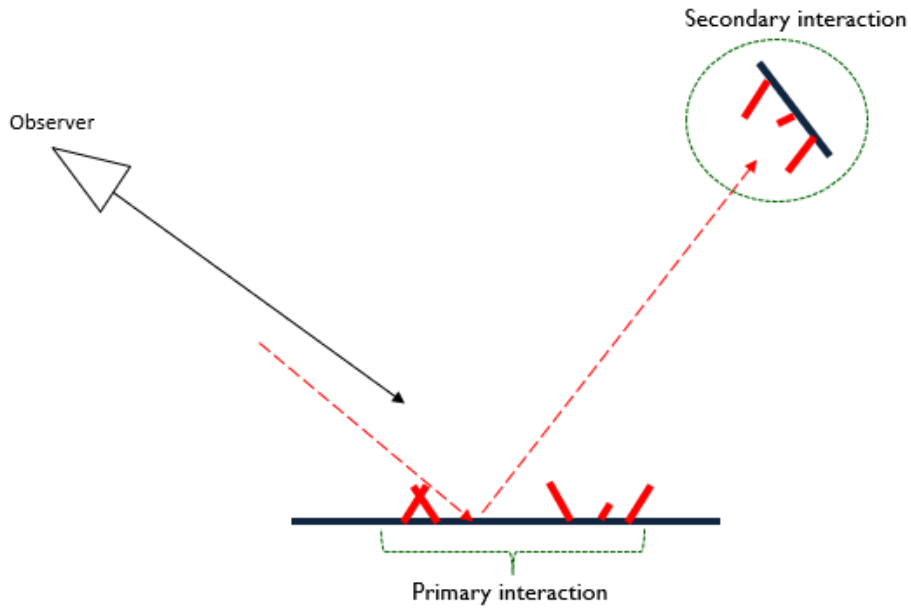
Published: 2023-10-23T14:08:14.04-04:00

Use Final Gathering

> **Note:** Diffuse transmission is not taken into account with **Final gathering**.

This option allows you to exploit the secondary rays and not the primary impacts.

This algorithm produces better results but is much slower to compute.



- **Final gathering max neighbor** allows you to pilot the number of neighbors after the secondary rays. The neighbors are used to compute the luminance for each split ray.
- **Splitting Number** allows you to set the number of split rays.

> **Note:** The value usually used is15.
>
> If there is an ambient source, the splitting number is not taken into account and is replaced by the ambient sampling value.

Fast Transmission Gathering

Fast Transmission Gathering accelerates the simulation by neglecting the light refraction that occurs when the light is being transmitted though a transparent surface.

This option is useful when transparent objects of a scene are flat enough to neglect the refraction effect on the direction of a ray (windows, windshield etc).

> **Note:** Fast Transmission Gathering does not apply to 3D Texture, Polarization Plate and Speos Component Import.

With **Fast transmission gathering** activated:

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- The result is correct only for flat glass (parallel faces).
- The convergence of the results is faster.
- The effect of the refraction on the direction is not taken into account.
- Dispersion is allowed.

Related information

Understanding Photon Mapping for a Deterministic Simulation on page 418
The Deterministic simulation is fast and targeted but is best suited to analyze simple optical paths. When contributions and optical interactions are multiple, you should use Monte Carlo algorithm or generate a photon map.

Deterministic Calculation Properties without Photon Map on page 420
This page describes the parameters to set when wanting to create a deterministic inverse simulation without generating a photon map.

10.6. Understanding Advanced Simulation Settings

The following section describes the advanced parameters to set when creating a simulation.

10.6.1. Meshing Properties

This page describes the different parameters to set when creating a Meshing and helps you understand how meshing properties impact performance and result quality.

> **Note:** For same values of meshing, meshing results can be different between the CAD platforms in which Speos is integrated.

> **Note:** In case of a thin body, make sure to apply a fixed meshing sag mode and a meshing sag value smaller than the thickness of the body. Otherwise you may generate incorrect results.

Creating a meshing on an object, a face or a surface allows you to mobilize and concentrate computing power on one or certain areas of a geometry to obtain a better level of detail in your results. In a CAD software, meshing helps you to subdivide your model into simpler blocks. By breaking an object down into smaller and simpler pieces such as triangular shapes, you can concentrate more computing power on them, and therefore improve the quality of your results. During a simulation, it will no longer be one single object that interprets the incoming rays but a multitude of small objects.

Meshing Migration Warning

For files created in version 2021 R1 or before: if **Sag** / **Step** type was **Proportional**, the file is migrated to used **Proportional to Face size** in version 2022 R2.

For files created in versions 2021 R2 or 2022 R1: if **Sag** / **Step** type was **Proportional to Body size**, the file is migrated with the same settings in version 2022 R2.

For file created before version 2022 R2: if **Sag** / **Step** type was **Fixed**, the file is migrated with no change.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Warning: if you created a file in version 2021 R1, then migrated to 2021 R2 and changed the values for **Sag / Step** type (when it became **Proportional to Body size**), these values may not be good in 2022 R2 when the document is migrated back to **Proportional to Face size**. You cannot know that the values were changed over the versions.

Meshing Mode: Fixed, Proportional to Face size Proportional to Body size

The meshing values can be set as Proportional to Body size, Proportional to Face size or Fixed:

- **Proportional to Face size** means that the tolerance adapts and adjusts to the size of each face of the object. The sag and maximum step size will, therefore, depend on the size of each face.
- **Proportional to Body size** means that the tolerance adapts and adjusts to the size of the object. The sag and maximum step size will, therefore, depend on the size of the body.
- **Fixed** means that the tolerance will remain unchanged no matter the size or shape of the object. The mesh of triangles will be forced on the object. The sag and maximum step size is, therefore, equal to the tolerance you entered in the settings.

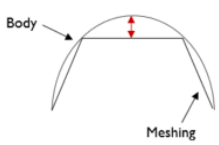
Note: From 2022R2, the new default value is **Proportional to Face size**. Selecting between Proportional to Face size and Proportional to Body size may slightly affect the result according to the elements meshed.

Note: When setting the meshing to **Proportional to Face size**, the results may return more faces than **Proportional to Body size**. These additional faces should be really small and they should not influence the ray propagation.

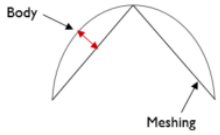
Note: When running a simulation for the first time, Speos caches meshing information if the Meshing mode is **Fixed** or **Proportional to Body size**. This way, when you run a subsequent simulation and you have not modified the Meshing mode, the initialization time may be a bit faster than the first simulation run.

Sag Tolerance

The sag tolerance defines the maximum distance between the geometry and the meshing.



Small Sag Value



Large Sag Value

By setting the sag tolerance, the distance between the meshing and the surface changes. A small sag tolerance creates triangles that are smaller in size and generated closer to the surface. This will increase the number of triangles and potentially computation time. A large sag tolerance will generate looser triangles that are placed farther from the surface. A looser meshing can be used on objects that do not require a great level of detail.

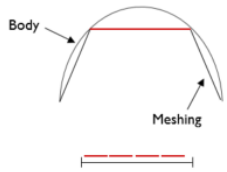
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Note: If the **Meshing sag value** is too large compared to the body size, Speos recalculate with a **Meshing sag value 128** to better correspond to the body size.

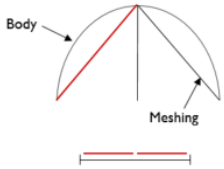
Maximum step size

Note: For a Heavyweight body, the **Meshing step value** precision decreases when applying a value below 0.01mm.

The maximum step size defines the maximum length of a segment.



Small maximum step size



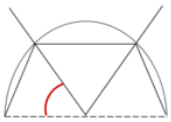
Large maximum step size

A small maximum step size generates triangles with smaller edge lengths. This usually increases the accuracy of the results.

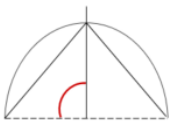
A greater maximum step size generates triangles with bigger edge lengths.

Angle Tolerance

The angle tolerance defines the maximum angle tolerated between the normal of the tangent formed at each end of the segments.



Small angle tolerance



Large angle tolerance

Related information

Adjusting Interactive Simulation Settings on page 392
This page describes the simulation settings that you can adjust to customize your interactive simulation.

Adjusting Direct Simulation Settings on page 401
This page describes the simulation settings that you can adjust to customize your direct simulation.

Adjusting Inverse Simulation Settings on page 409
This page describes the simulation settings that you can adjust to customize your inverse simulation.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

10.6.2. Tangent Bodies Management

The following page describes you how the propagation is managed when a ray interacts with an interface between two bodies having tangent faces.

Tangent Bodies Management

The Tangent Bodies Management applies when an interaction ray-face occurs. The propagation engine searches if ray intersects another face (meshing triangle) nearby the interaction along the incoming ray direction. The neighborhood range of an interaction is limited by the Geometrical Distance Tolerance parameter.

Tangent Bodies Management only applies to tangent volume bodies. A surface tangent to a volume body or another surface will generate a 2D Tangency.

> **Note:** When a 3D Texture support is tangent to another body, rays intersecting the 3D Texture are set in error. The html report indicates the percentage of rays in error with the **3D Texture support body tangent to another body** entry.

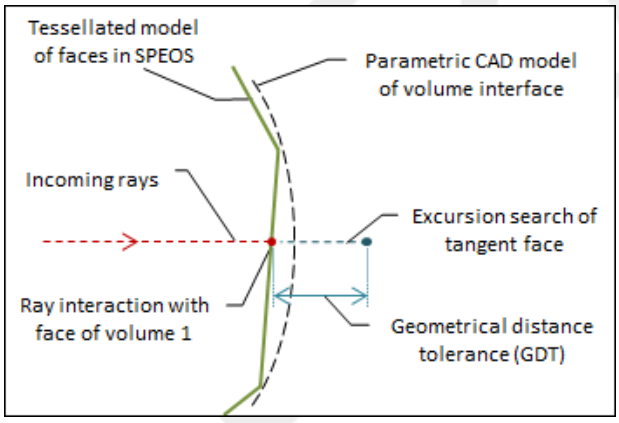
You can find two cases:

- **Case 1: No interaction is found.**

  The Surface Optical Properties is applied on the face. The next Volume Optical Properties is set according to the new optical properties.

  For example, if a ray coming from the ambient material enters in a material, then the new Volume Optical Properties of this material is applied.

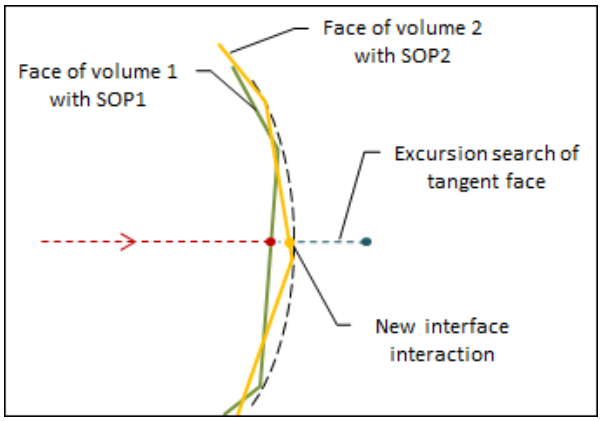
  If the ray coming from a body does not intersect any face within the Geometrical Distance Tolerance, no tangent surface is detected, so the ray propagates in the ambient material (AIR by default).



- **Case 2: A new interaction is found.**
  - Principle 1: If the Surface Optical Properties of face 1 and face 2 are the same, then the tangent faces are considered as one interface. Example: two plastic components overmoulded.
  - Principle 2: If the Surface Optical Properties of face 1 and face 2 are different, then an infinitesimal ambient material is considered as laying between faces. Example: junction between two tangent faces with two different surfacings. This configuration can lead to a **wrong impact status for this interaction**. Please **create a real geometric gap** to better model this configuration

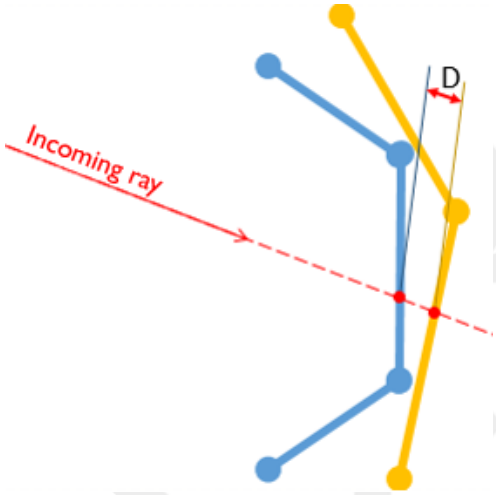
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Note: You must correctly consider the design intent by setting the same or different SOP between the two interfaces.



Geometrical Distance Tolerance (GDT) Management

Geometrical distance tolerance defines the maximum distance to consider two faces as tangent. When the distance between two faces is smaller than the maximum distance defined, the faces are considered as tangent.



D : distance between the two projected impacts of the incoming ray on two faces (projection without any shift).

If **D** is **inferior** to the **Geometrical distance tolerance**, then the two faces **are tangent** (% error added to the Speos report). If **D** is **superior**, then the **faces are not tangent**.

Note: It is not possible to manage geometries that are smaller than a nanometer (1e-6 mm). However, we do not recommend you to set a smaller value than 1e-3 mm as the value is small enough in most cases.

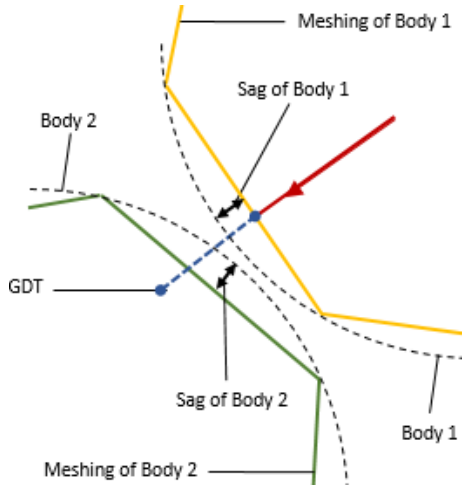
To correctly set the Geometrical Distance Tolerance (GDT) option, the following criteria must be fulfilled:

- Two faces are tangent if the distance between the intersection points of the two faces is smaller than the Geometrical Distance Tolerance.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

- The Geometrical Distance Tolerance must be much larger than the meshing tolerance: **GDT >> sag(Body1) + sag(Body2)**(see figure below)



- Speos tangent faces management can handle only two tangent faces. The Geometrical Distance Tolerance must be smaller than the smallest thickness of the bodies.
- Make sure that sources or sensor distance to geometries are greater than the Geometrical Distance Tolerance.

When two faces are very close (closer than **Geometrical Distance Tolerance**) with a big angle between them, the faces are not considered as tangent and it may cause propagation error.

Related information

Adjusting Interactive Simulation Settings on page 392
This page describes the simulation settings that you can adjust to customize your interactive simulation.

Adjusting Direct Simulation Settings on page 401
This page describes the simulation settings that you can adjust to customize your direct simulation.

Adjusting Inverse Simulation Settings on page 409
This page describes the simulation settings that you can adjust to customize your inverse simulation.

10.6.3. Smart Engine

This page describes the Smart Engine ray tracing technique.

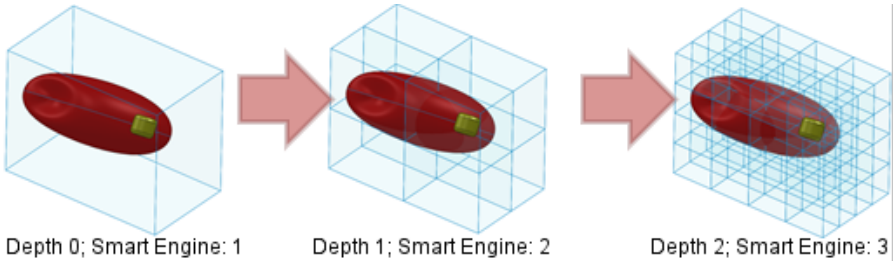
Smart Engine is a pre-calculation method meant to improve the definition of the rays' impact area in the scene.

The scene (the simulation environment comprising the geometries) is subdivided into blocks to help the rays find/locate the elements they need to interact with.

The Smart Engine value defines a balance between the speed and the memory. The higher the value, the more subdivided the scene becomes.

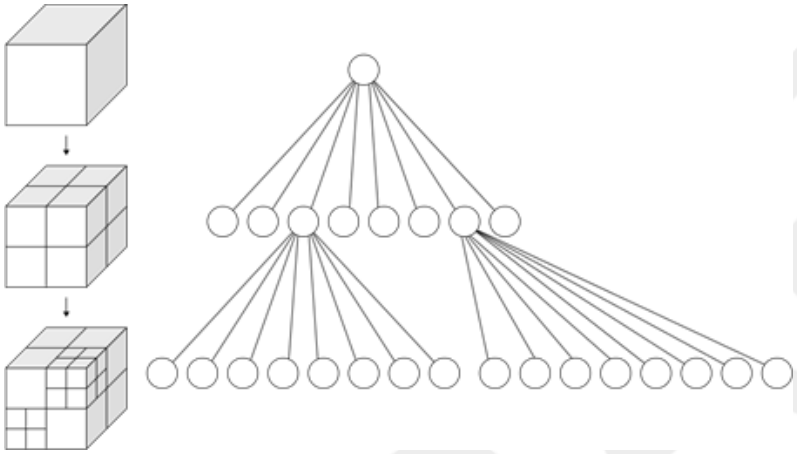
DRAFT

Smart engine value is the depth allowed of the octree structure to partition the geometry. This geometry sorting in bounding box speeds the computation of ray intersection with the geometry.



Geometry sub-division according to the smart engine value

An octree structure is a partitioned meshed three dimensional geometry by recursively subdividing it into eight octants. An octant without geometries is not subdivided.



Note: It is not recommended to change the default smart engine value for a classical use of Speos.

However in some cases when memory use is critical due to huge geometries (complete cockpit, cabin, car or building), this value must not exceed 9 and can be reduced in order to save memory.

Also in other cases when a simulation contains small detailed geometries inserted in a big scene (detail of headlamp bulb placed in a simulation with a 50m long road geometry) this value can be increased to reach better performances.

It becomes interesting to use the smart engine parameter when sending a large number of rays. As an example, it is not the case for a Light Modeling interactive simulation with around 100 rays, and it is the case for a Digital Vision and Surveillance interactive simulation with around 300k rays.

Related information

Adjusting Interactive Simulation Settings on page 392
This page describes the simulation settings that you can adjust to customize your interactive simulation.

Adjusting Direct Simulation Settings on page 401
This page describes the simulation settings that you can adjust to customize your direct simulation.

Adjusting Inverse Simulation Settings on page 409
This page describes the simulation settings that you can adjust to customize your inverse simulation.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

10.6.4. Dispersion

Dispersion refers to the chromatic dispersion of visible light based on the Snell-Descartes law.

> **Note:** The **Dispersion** option is only available for Inverse and Direct simulations. In Interactive simulation, the **Dispersion** option is always activated (and so hidden in the interface).

Allowing the Dispersion option activates the dispersion calculation. In optical systems in which the dispersion phenomena can be neglected, the colorimetric noise is reduced by deactivating the Dispersion option.

Dispersion influences the Monte Carlo algorithm in terms of ray number generated by pass in Inverse simulation. It increase the number of rays in Inverse simulation compared to when the Dispersion option is not activated.

> **Important:** When a dispersive material is present in the optical system, we highly recommend you to activate the dispersion.

Refraction Calculation

From version 2023 R2, in order for both the CPU and the GPU simulations to converge towards the same result, Speos uses a new algorithm to calculate the refraction of a ray.

When Dispersion Activated

Each ray carries only one monochromatic wavelength. The Snell-Descartes law is applied without any approximation. That means the ray will be refracted according to the refractive index corresponding to its own wavelength when hitting a dispersive material.

When Dispersion Deactivated

Each ray carries all wavelengths. When a ray hits a dispersive material, the ray is refracted using the refraction direction of the median wavelength of the propagated spectrum, while keeping the whole spectrum.

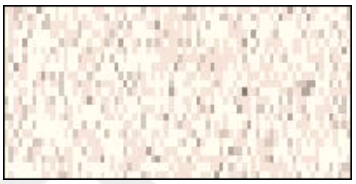
> **Note:** In case of a Direct Simulation, activating or deactivating **Dispersion** for a Ray File Source has no effect as each ray of the source is composed of only one wavelength.

Both algorithms generate colorimetric and photometric noises. Comparing them, you can observe that:

- When activating Dispersion, noise has a more colorimetric nature.
- When deactivating Dispersion, noise has a more photometric nature.

In terms of rendering, noise obtained without dispersion is visually more appealing than the one with dispersion.

DRAFT

	With Dispersion (Simulation Time = 290s)	Without Dispersion (Simulation Time = 10s)
Simulation Result		
Noise		

CPU vs GPU Calculation

In Direct Simulation, in case of a Photometric or Colorimetric sensor, the spectral range used is different between a GPU Simulation and a CPU Simulation for performance reasons.

> **Note:** In case of a Radiometric sensor, no difference is made. Speos takes the entire spectral range in both CPU and GPU Simulations

With a CPU Simulation, the spectral range of the sources is used.

- Advantages
  - You can access the Power report generated that provides radiometric information of the result as it takes the spectral range of the sources.
  - There is no spectral range sensor dependency as it takes the spectral of the sources. Indeed if you add other sensor, the sensors will not influence each other as in a GPU Simulation (see GPU drawbacks below).
- Drawbacks
  - Not all rays are integrated in the sensor as the spectral range of the sources may be larger than the spectral range of the sensor (which is the opposite in GPU as all rays generated should be integrated in the sensor). Rays propagated which do not contribute to the result occupy the CPU anyway.
  - You can have differences between a CPU inverse and CPU direct simulations as it does not generate rays from the same origin (The Direct simulation generates rays from the source with a spectral range of the sources, whereas the Inverse Simulation generates rays from the sensor with a spectral range of the sensor).

With a GPU Simulation, the spectral range of the sensors is used.

- Advantages
  - There is no difference between GPU inverse and GPU direct simulations in case of a one-sensor simulation because in both cases, Speos takes the spectral range of the sensor. If there are more than one sensor with different spectral ranges, you may have difference.
- Drawbacks
  - There is a sensor dependency as it takes the spectral range of sensors.

DRAFT

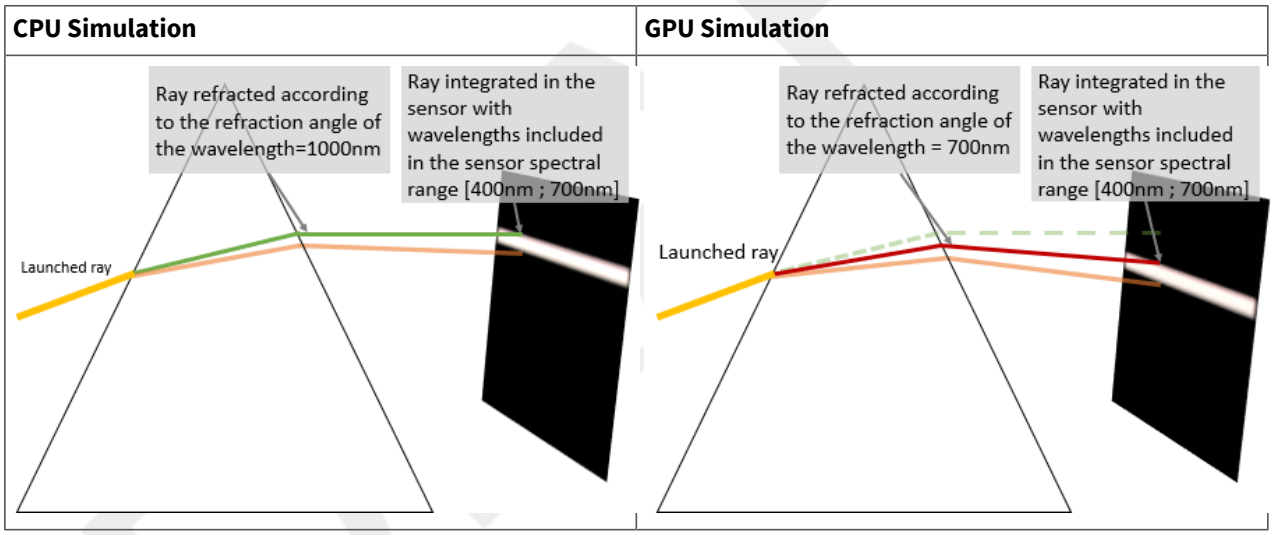
Example: let's take a sensor of [400 ; 700] spectral range. In simulation, the spectral range used will be the sensor one in [400 ; 700]. If you add another sensor in Infrared for example [800 ; 1000], then both sensors will influence each other and the spectral range used in simulation will be [400 ; 1000].

Example

Let's take a simulation with a source containing a *.spectrum file ranged from 400nm to 1000nm. The colorimetric sensor wavelength integration is defined between 400nm and 700nm.

- The CPU Direct Simulation will launch rays from all the spectral range of the source ([400nm ; 1000nm]). When passing through the dispersive material, each ray is refracted with the refraction angle of the median wavelength of the propagated spectrum [400 ; 1000nm]. If the ray hits the sensor, the sensor integrates the ray only with the wavelengths included in the spectral range of the sensor [400 ; 700].
- The GPU Direct Simulation will launch rays only from the spectral range of the sensor ([400nm ; 700nm]). When passing through the dispersive material each ray is refracted with the refraction angle of the median wavelength of the propagated spectrum [400 ; 700nm]. As the spectral range of the sensor is used for both the wavelength selection of the refraction angle and the integration on the sensor, all rays are integrated in the sensor, generating a low-noise result.

With this algorithm, a CPU simulation should spread refraction more than a GPU simulation due to the different spectral range.



Note: Note that the following example is used with a sensor which size is large enough to integrate the rays with all refraction angles of the wavelengths spectral range of the source. If the sensor size were much smaller the spread and noise would be different as some rays would not hit the sensor due to their refraction angle.

10.6.5. Weight

The page provides advanced information on the **Weight** parameter.

Note: It is highly recommend to set this parameter to true excepted in interactive simulation.

Deactivating this option is useful to understand certain phenomena as absorption.

Weight and Minimum energy percentage

The **Weight** represents the ray energy. In real life, a ray looses some energy (power) when it interacts with an object.

The **Minimum energy percentage** parameter defines the minimum energy ratio to continue to propagate a ray with weight. This parameter helps the solver to better converge according the simulated lighting system.

If you do not activate weight, rays' energy stay constant and probability laws dictate if they continue or stop propagating.

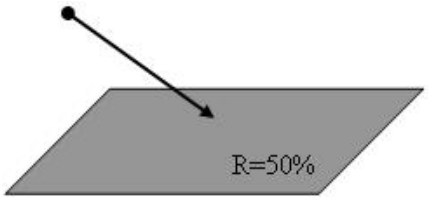
If you do activate weight, the ray's energy evolves with interactions until they reach the sensors.

Specific scenarios

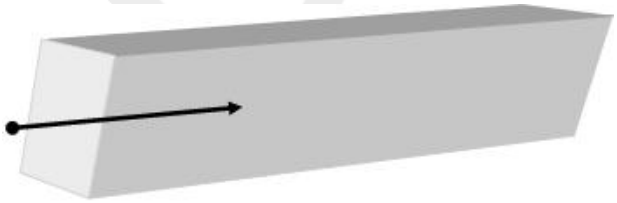
According to the configuration, activating the weight parameter may have some impacts on the simulation calculation.

The following configurations illustrate the behavior of the rays depending if the weight has been activated or not.

- Ray/Face interaction Consider rays reaching an optical surface having a 50% reflectivity.
  - If you do not activate weight, rays have 50% probability to be reflected.
  - If you do activate weight, all the rays are reflected with 50% of their initial energy.

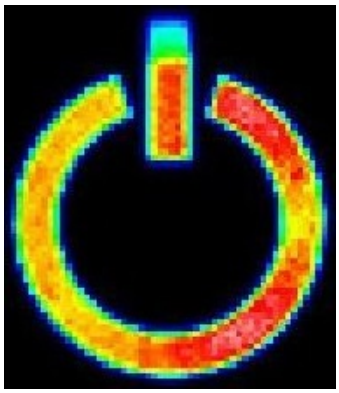


- Ray/Volume interaction Consider rays propagating inside an absorbing material.
  - If you do not activate weight, rays have a probability to be absorbed or transmitted according to their path through the material.
  - If you activate weight, rays' energy decreases exponentially according to the material absorption value and the path of rays through it.

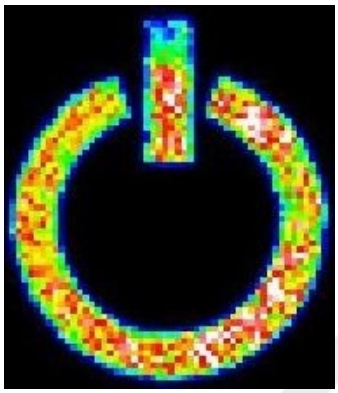


Tip: Practically, using weight in simulation improves results' precision as more rays with contributing energy reach the sensors. So, to get the same amount of rays on sensors without the Weight parameter, you need to set more rays in simulations, which tends to increase simulation time.

DRAFT



Direct simulation result when weight is activated

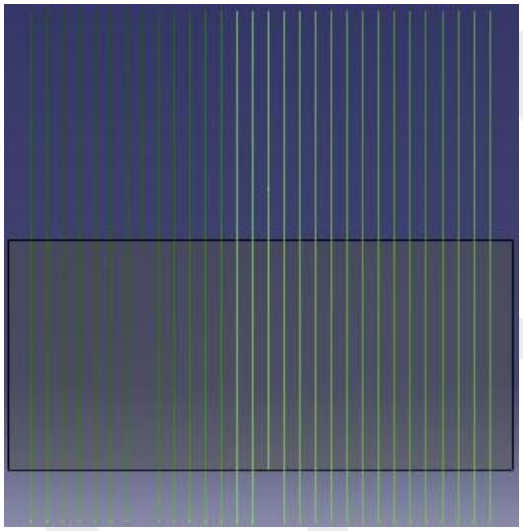


Direct simulation result when weight is deactivated

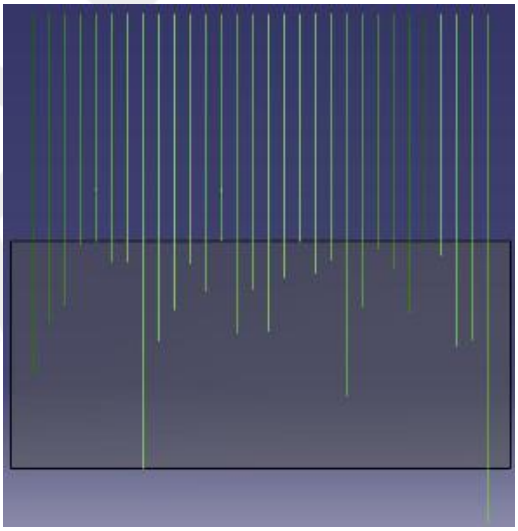
Weight Deactivation

Deactivating the weight is useful in two specific cases.

1. When you analyze phenomena such as absorption. Considering a material with absorption, here is the observation of the absorbed rays using an interactive simulation.



Interactive simulation result when weight is activated



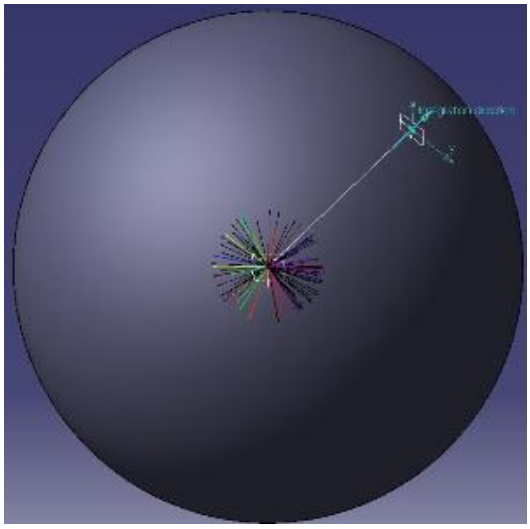
Interactive simulation result when weight is deactivated

2. If you want a simulation performance improvement in a closed system.

Let us consider an integrating sphere with a light source and sensor inside of it.

The surface inside the sphere has a high reflectivity value. The system is set so the sensor is protected from direct illumination from the light source.

DRAFT



In this context, activating the Weight would highly extend simulation time.

When weight is activated, simulation time corresponds to 1747.

When weight is not activated, simulation time corresponds to 440.

This difference is due to the fact that low energy rays are still propagating after several bounds in the system for simulations using weight whereas the probability the rays still propagate decreases each bound they make for simulations not using weight.

Related information

Adjusting Direct Simulation Settings on page 401
This page describes the simulation settings that you can adjust to customize your direct simulation.

Adjusting Inverse Simulation Settings on page 409
This page describes the simulation settings that you can adjust to customize your inverse simulation.

10.7. LiDAR

LIDAR is a remote sensing technology using pulsed laser light to collect data and measure the distance to a target. LIDAR sensors are used to develop autonomous driving vehicles.

10.7.1. Understanding LiDAR Simulation

This page gives a global presentation on LiDAR principles and simulation.

LiDAR Principle

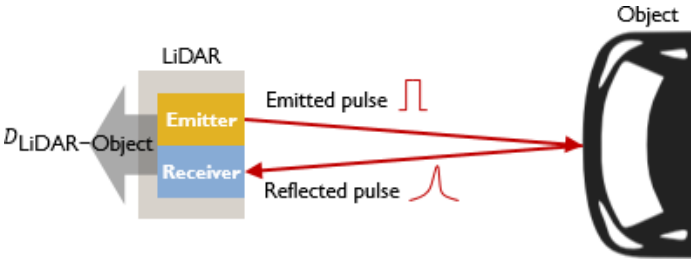
A LiDAR is a device measuring the distance to a target by sending pulsed laser light. It works on the principle of a radar but uses light instead of radio waves.

A LiDAR is composed of a light source (an emitter) and a sensor (a receiver).

The emitter illuminates a target by sending pulsed laser light and evaluates the distance to that target based on the time the reflected pulse took to hit the receiver.

DRAFT

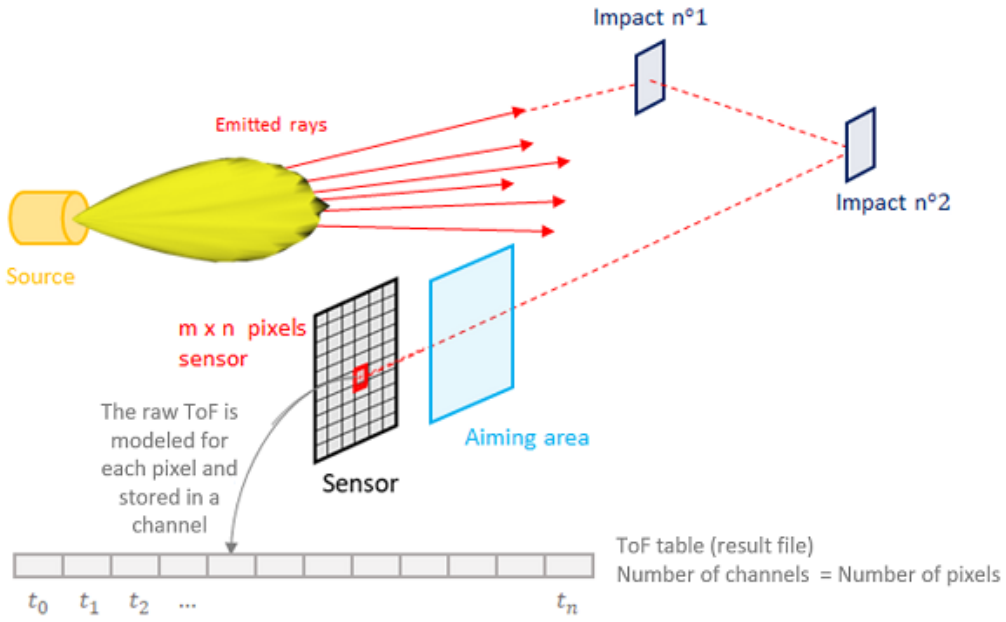
LiDAR Simulation in Speos



The ray sent by the LiDAR source (emitter channel) interacts with a geometry or vanishes in the environment.

The interaction is managed by the optical properties of the target geometry. A part of the ray energy is reflected towards the LiDAR sensor (receiver channel). This energy contributes to the signal of a pixel. After all contributions are integrated, Speos models the Raw time of flight for each pixel. The Raw time of flight expresses the temporal power on a pixel. A specific power is integrated in each pixel for a given distance.

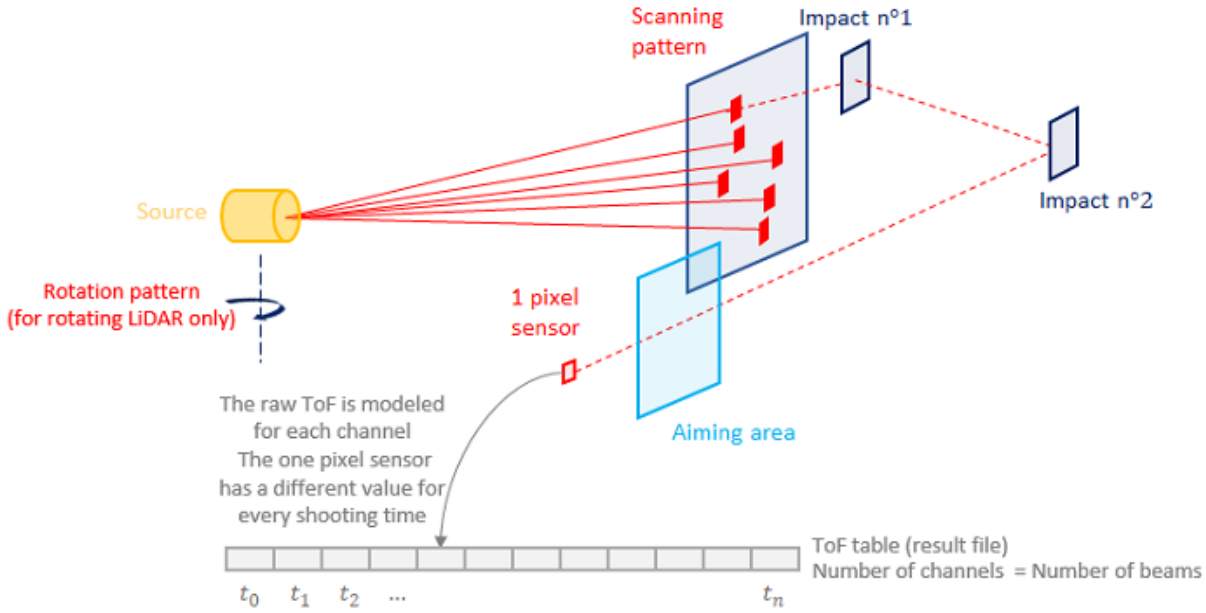
In the case of static LiDAR simulation, the time of flight is modeled for each pixel of the sensor with one pixel corresponding to one channel in the result file.



In the case of a scanning or rotating simulation, the record of data is slightly different. Indeed, the number of channels is no longer equal to the number of pixels but to the number of beams. As the beams are sent at different times with a very short interval (of micro seconds- µs), a one-pixel sensor can be used and will have a different time of flight value for each time.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00



At the end of the optical simulation, Speos applies a signal post-processing to interpret the Raw time of flight signal in a distance. The signal then identifies the time when the maximum power is received by a pixel and computes the distance based on this time and the light celerity.

LiDAR Simulation Results

Speos LiDAR Simulation can provide three types of result: (the results vary based on the type of LiDAR)

- Fields of view : visualization of the source, sensor and LiDAR field of view projection in the 3D view.
- Raw time of flight signal: power arriving on the sensor, saved in a binary format (*.OPTTimeOfFlight).
- Map of depth: LiDAR-object distance measured for each pixel, saved in a xmp map.

Related concepts

Understanding LiDAR Simulation Results on page 449
This section gathers the different types of results that can be obtained from a LiDAR simulation.

Related information

Creating a LiDAR Simulation on page 446
Creating a LiDAR simulation allows you to generate output data and files that enable to analyze a LiDAR system and configuration. The LiDAR simulation supports several sensors at a time.

10.7.2. Understanding LiDAR Simulation with Timeline

The following page presents you how a timeline coupled with a trajectory file works in a LiDAR simulation.

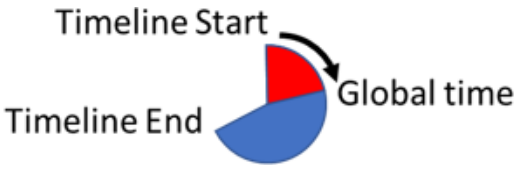
DRAFT

Warning: From the version 2022 R2, the Timeline **Start** and **End** parameters have been improved in order to enter time values below 1ms. This change will impact your custom scripts, and will need to be modified consequently, as previous time format is no longer supported.

Time Consideration

LiDAR simulation considers all the timestamps of the complete acquisition sequence described in the scanning and rotation sequence files.

Starting from **Timeline Start**, time will increase by considering the timestamps.



Global time = Timeline Start + Timestamp min

```
ANSYS - LiDAR scanning sequence file v1.0
LiDAR with 64 beams
Dirac
64
3.5     0.0000002       0.578361878     -0.022349783
3.5     0.0000001       0.917972642     -12.02531402
4.7     0.0000002       -1.55626933     -0.362407375
4.7     0.0000001       -2.284297681    -12.53631345
5.9     0.0000002       1.35466933      2.02081238
5.9     0.0000001       2.082697681     -8.949727577
7.2     0.0000002       -0.779961878    1.679889013
7.2     0.0000001       -1.119572642    -9.463678954
10.4    0.0000002       -3.678524303    -0.702363193
10.4    0.0000001       -5.47901546     -13.04696774
```

Global time = Timeline Start + Timestamp

```
ANSYS - LiDAR scanning sequence file v1.0
LiDAR with 64 beams
Dirac
64
3.5     0.0000002       0.578361878     -0.022349783
3.5     0.0000001       0.917972642     -12.02531402
4.7     0.0000002       -1.55626933     -0.362407375
4.7     0.0000001       -2.284297681    -12.53631345
5.9     0.0000002       1.35466933      2.02081238
5.9     0.0000001       2.082697681     -8.949727577
7.2     0.0000002       -0.779961878    1.679889013
7.2     0.0000001       -1.119572642    -9.463678954
10.4    0.0000002       -3.678524303    -0.702363193
10.4    0.0000001       -5.47901546     -13.04696774
```

Global time = Timeline Start + Timestamp max

```
ANSYS - LiDAR scanning sequence file v1.0
LiDAR with 64 beams
Dirac
64
3.5     0.0000002       0.578361878     -0.022349783
3.5     0.0000001       0.917972642     -12.02531402
4.7     0.0000002       -1.55626933     -0.362407375
4.7     0.0000001       -2.284297681    -12.53631345
5.9     0.0000002       1.35466933      2.02081238
5.9     0.0000001       2.082697681     -8.949727577
7.2     0.0000002       -0.779961878    1.679889013
7.2     0.0000001       -1.119572642    -9.463678954
10.4    0.0000002       -3.678524303    -0.702363193
10.4    0.0000001       -5.47901546     -13.04696774
```

Once the acquisition sequence described in the LiDAR sensor is complete, a new sequence begins from the Timestamp min.

The loop lasts until Timeline End is reached.

DRAFT

Figure 55. Scanning type LiDAR Sensor Workflow

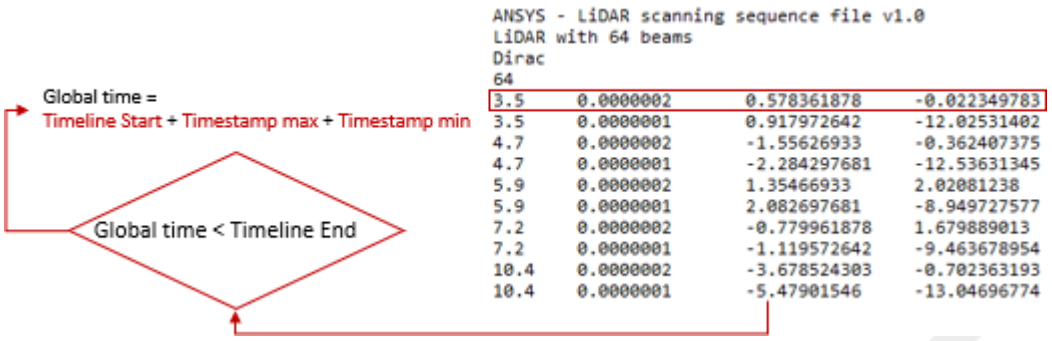
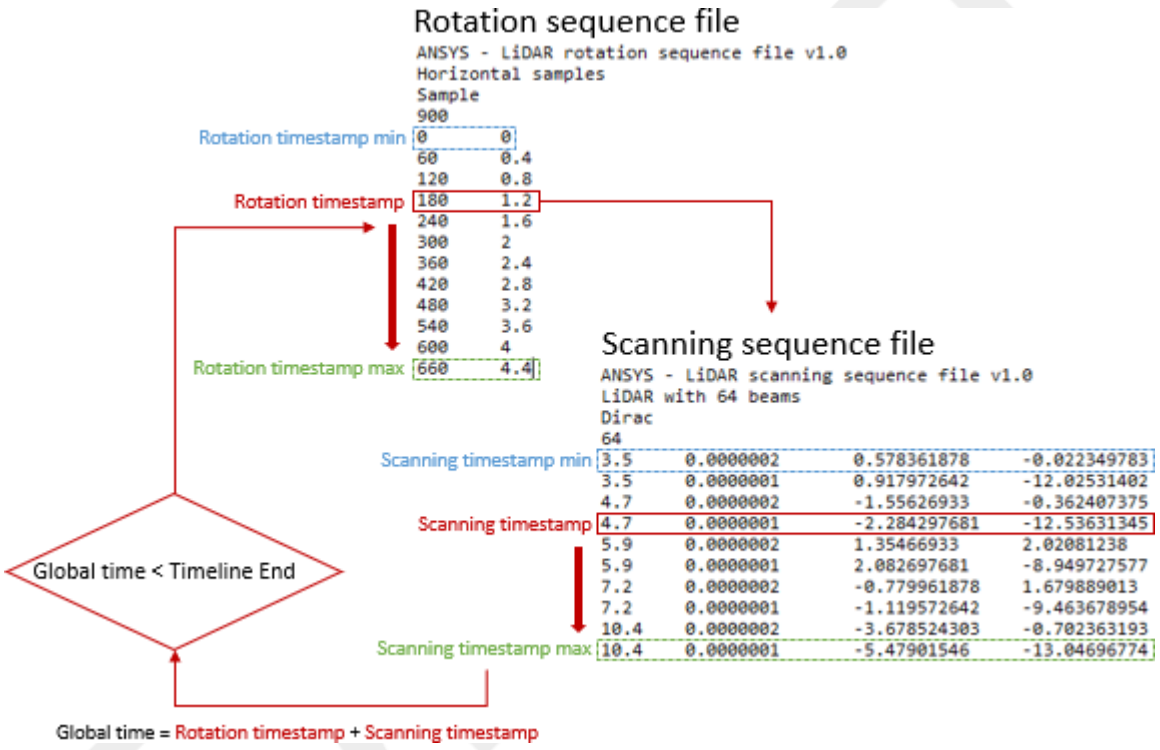


Figure 56. Rotating type LiDAR Sensor Workflow



Note: For more information about Rotation and Scanning Sequence File, refer to Firing Sequence Files on page 229

Trajectory Consideration

At each timestamp, the global time is used to position and orientate features using a Trajectory file.

A linear interpolation is done between two trajectory samples.

Note: The number of samples needs to reflect the shape of the trajectory. For instance, few samples are necessary for a linear trajectory. On the other hand, the number of samples needs to be high enough for a curvy trajectory.



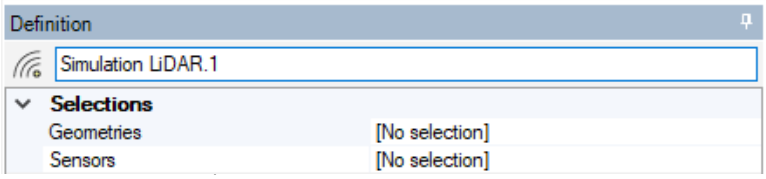
10.7.3. Creating a LiDAR Simulation

Creating a LiDAR simulation allows you to generate output data and files that enable to analyze a LiDAR system and configuration. The LiDAR simulation supports several sensors at a time.

To create a LiDAR Simulation:

A sensor and source must already be created.

1. From the **Light Simulation** tab, click **System** >



2. In the 3D view click  to select the geometries for which you want to calculate the distance to the Lidar.

   Your selection appears in the **Geometries** list.

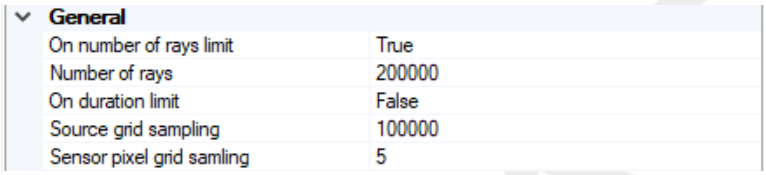
> **Tip:** You can select elements from the 3D view or directly from the tree.
>
> To deselect an element, click back on it.
>
> To rapidly select all the faces of a same element, click one face of this element and press CTRL+A.

3. In the 3D view click  and from the tree select the previously created LiDAR sensor(s).

   Your selection appears in the **Sensors** list.

> **Note:** When selecting several sensors in one LiDAR simulation, the sensors do not illuminate each other and their contributions are not merged. The simulation will produce individual results for each sensor selected.

4. Define the criteria to reach for the simulation to end:



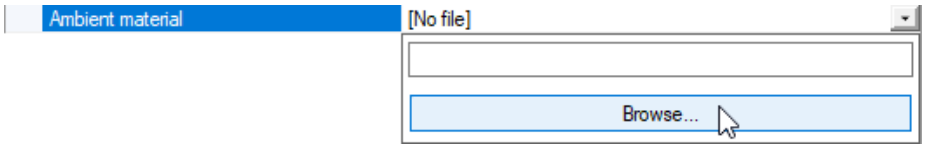
   - To stop the simulation after a certain number of rays were sent, set **On number of rays limit** to **True** and define the number of rays. The number can be larger than 2 Giga rays.
   - If you are working with static LiDARs and want to stop the simulation after a certain duration, set **On duration limit** to **True** and define a duration.

> **Note:** If you activate both criteria, the first condition reached ends the simulation.
>
> If you select none of the criteria, the simulation ends when you stop the process.

5. In **Source grid sampling**, define the number of samples used to calculate the field of views.
6. In **Sensor pixel grid sampling**, define the number of samples used along each pixel side to calculate the field of views.
7. In **Ambient material**, browse a .material file if you want to define the environment in which the light will propagate (water, fog, smoke etc.).

   The ambient material allows you to specify the media that surrounds the optical system. Including an ambient material in a simulation brings realism to the optical result.

   The .material file is taken into account for simulation.



> **Note:** If you do not have the .material file corresponding to the media you want to use for simulation, use the **User Material Editor**, then load it in the simulation.

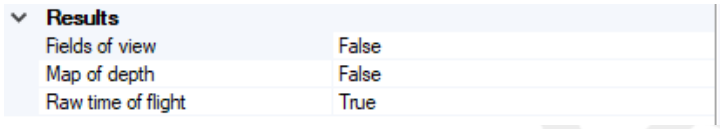
8. If you want to consider time during the simulation, set **Timeline** to **True**.

> **Note:** This allows dynamic objects, such as Speos Light Box Import and Scanning or Rotating LiDAR Sensors, to move along their defined trajectories and consider different frames corresponding to the positions and orientations of these objects in simulation.

> **Note:** For more information on Timeline, refer to Understand LiDAR Simulation with Timeline.

9. From the **Results** section, filter the results you want to generate by setting them to **True** or **False**:

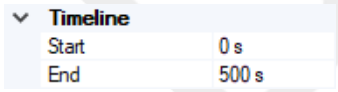
> **Note:** The **Fields of view** and **Map of depth** result types are only available for static LiDARs.



- If you want a visualization of the source, sensor and LiDAR fields of view to be displayed in the 3D view after simulation, activate **Fields of view**.
- If you want a map of depth to be generated after simulation, activate the corresponding option.
- If you want to generate a **Raw time of flight** (*.OPTTimeOfFlight) result file, activate the corresponding option.

> **Note:** If you need more information about grid sampling or results, see Understanding LIDAR Simulation Results.

10. If **Timeline** is set to **True**, from the **Timeline** section, define the **Start** and **End** times.



> **Warning:** From the version 2022 R2, the Timeline **Start** and **End** parameters have been improved in order to enter time values below 1ms. This change will impact your custom scripts, and will need to be modified consequently, as previous time format is no longer supported.

11. In the 3D view, click **Compute**



to launch the simulation.

The simulation is created and results appear both in the tree and in the 3D view.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

If Timeline is activated, the Raw time of flight simulation result will contain all the timestamps of the simulation.

When running a Rotating LiDAR Simulation with **Speos HPC Compute**, each thread sends one ray per pulse.

Related concepts

Understanding LiDAR Simulation Results on page 449
This section gathers the different types of results that can be obtained from a LiDAR simulation.

Related information

Understanding LiDAR Simulation on page 441
This page gives a global presentation on LiDAR principles and simulation.

10.7.4. Understanding LiDAR Simulation Results

This section gathers the different types of results that can be obtained from a LiDAR simulation.

10.7.4.1. Fields of View

This page describes the Fields of View result, obtained from a LiDAR simulation and helps you understand how the grid sampling can impact result visualization.

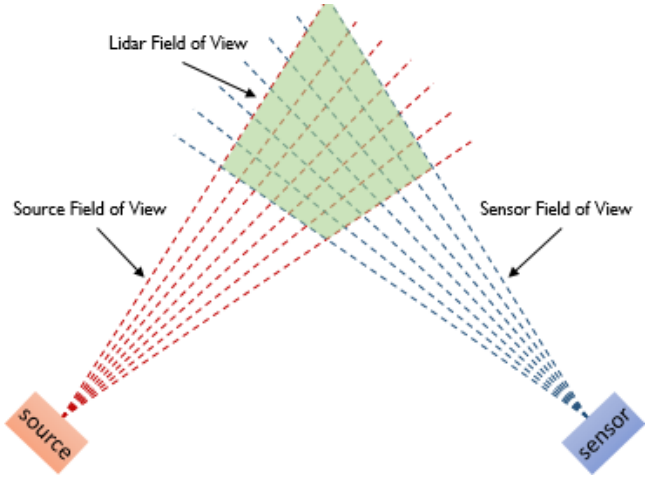
Field of View

The **Fields of view** allow to project in the 3D view a visualization grid of the source, sensor and LIDAR fields of view (the LiDAR field of view being the overlap of the source and the sensor fields of view).

> **Note:** Projected grids are generated for solid-state LiDAR simulations only (LiDAR simulations using a Static LiDAR sensor).
>
> You can edit the visualization of the LiDAR Projected Grids.
>
> You can export the projected grids as geometry to convert them into construction lines.

- The Source Field of View represents the illuminated area.
- The Sensor Field of View represents the area observed by the sensor.
- The LiDAR Field of View is the overlap of the Source and Sensor Fields of View.

DRAFT



Grid Sampling

The quality of the projected grid depends on the source and sensor sampling. With an approximate sampling, you may miss some thin geometries or get an inaccurate grid on geometries' edges.

Sensor Grid Sampling

Below are examples of the level of accuracy obtained depending on the sampling precision.

Figure 57. Poor sampling level - The pixel corner 2 does not intersect any geometry. The last edge is not drawn.

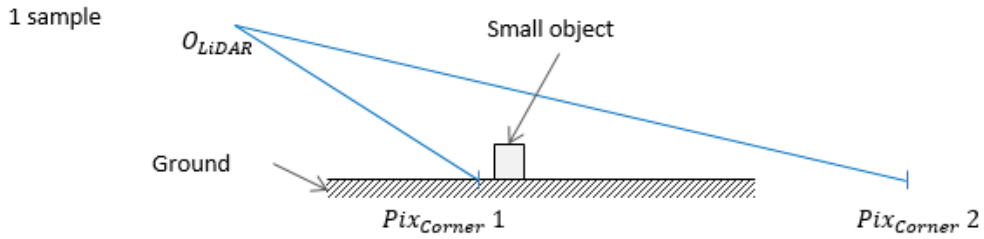


Figure 58. Intermediate sampling level - The sub-pixel's sample intersects the ground. An edge is drawn, but passing through the object.

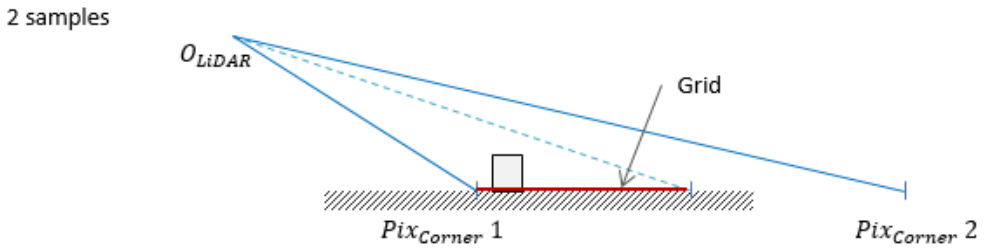
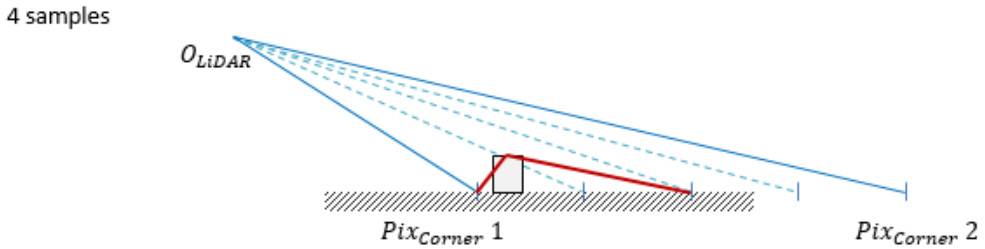


Figure 59. High sampling level - The sub-pixel's sample intersects the ground and the object. The displayed pixel edge takes into account the ground and the object intersection.

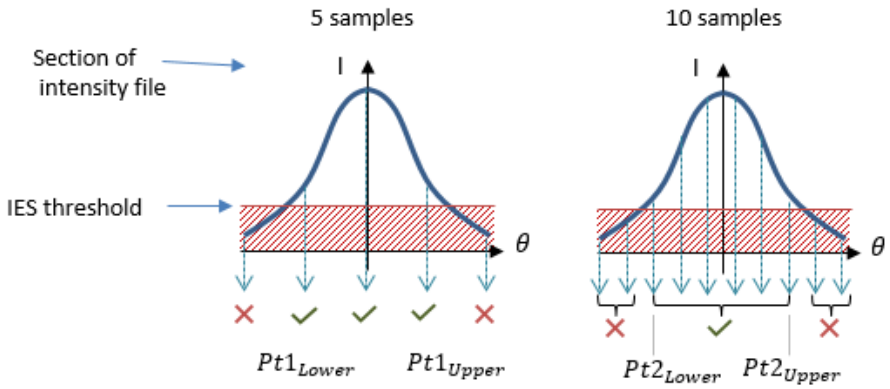


Source Grid Sampling

The shape of the source grid can be impacted by its sampling.

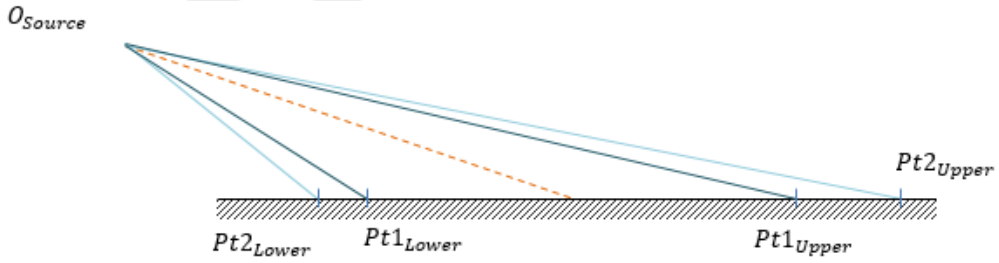
To project the source, samples are taken at different positions with a regular step. A high number of samples improves the accuracy and reduces the risk of error.

> **Note:** Only samples having an intensity higher than the defined threshold are taken into account.



> **Note:** Angles of lower and upper bound of the Field of View can change with the sampling.

Depending on the sampling and shape of the intensity, this difference can be noticeable on the projected grid results.



Related information

Creating a LiDAR Simulation on page 446
Creating a LiDAR simulation allows you to generate output data and files that enable to analyze a LiDAR system and configuration. The LiDAR simulation supports several sensors at a time.

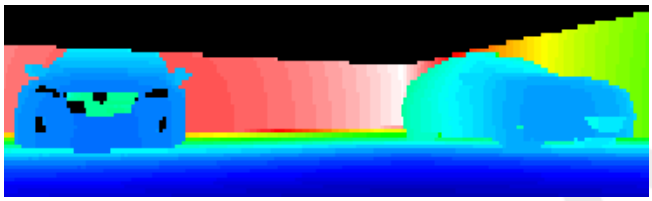
10.7.4.2. Map of Depth

This page describes the map of depth result type that can be obtained from a LiDAR simulation.

> **Note:** Depth maps are generated for solid-state LiDAR simulations only.

The **Map of depth** is an extended map (*.XMP) that saves, for each pixel, the distance from the LiDAR to the detected object. This distance is extracted from the raw signal by catching the maximum peak position detected in every pixel. The closeness to objects is expressed with a color scale.

In the illustration below, the smaller the distance from the LIDAR, the closer to the blue.



Related information

Creating a LiDAR Simulation on page 446
Creating a LiDAR simulation allows you to generate output data and files that enable to analyze a LiDAR system and configuration. The LiDAR simulation supports several sensors at a time.

10.7.4.3. Raw Time of Flight

This page provides more information on the Raw Time of Flight (*.OPTTimeOfFlight) file and describes how to operate/analyze its content.

Description

The *.OPTTimeofFlight is a result file of a LiDAR simulation used to store the raw time of flight of each pixel of the LiDAR sensor.

You can use this file to read and export specific data or post-treat it to output results, like a 3D representation of the scene in the form of a point cloud (impacts collected during simulation).

This file is a compressed **binary** file that can store large amounts of data but that can only be accessed through APIs.

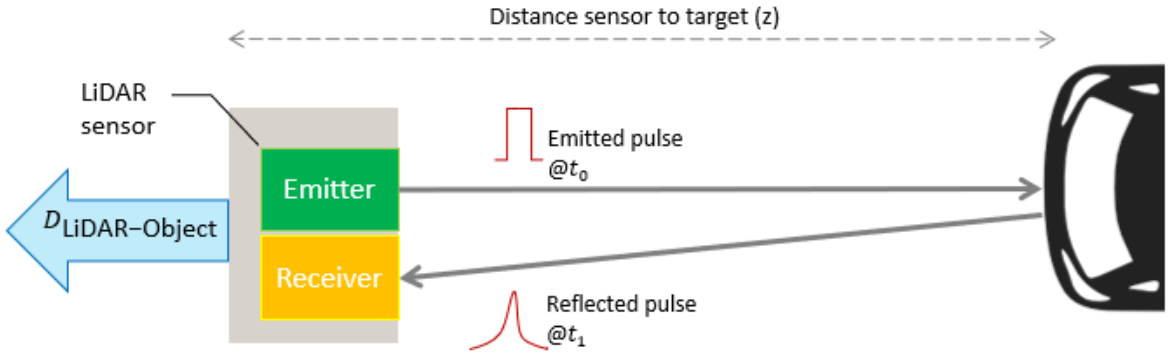
Principle

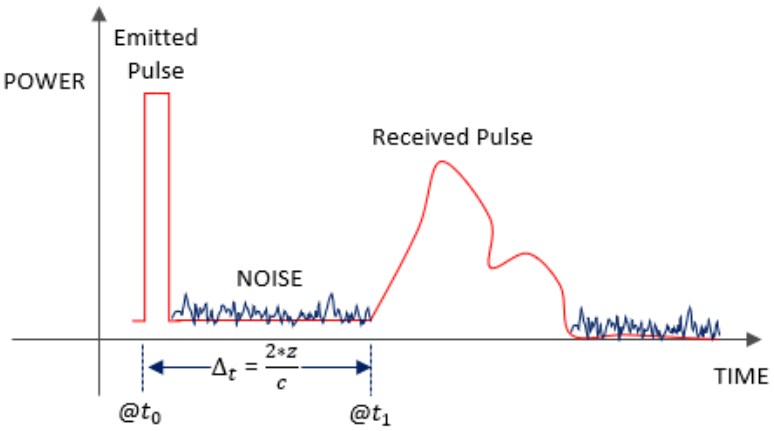
The **Raw Time of Flight** essentially represents the temporal power of a pixel expressed in Watts. It describes the time interval between the emission of the light pulse and its detection after being reflected by an object in the scene. Then, through data conversion, the LiDAR-to-object distance is derived from this time of flight.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Important: Speos LiDAR Simulation has been designed so the emitter (source) and receiver (sensor) of the LiDAR system are located at the same place (meaning equal or less than the **Spatial accuracy** parameter of the LiDAR Sensor). The distance traveled by a ray from the source to the sensor corresponds to twice the distance between the sensor and the target.

Figure 60. Raw Time of Flight Principle

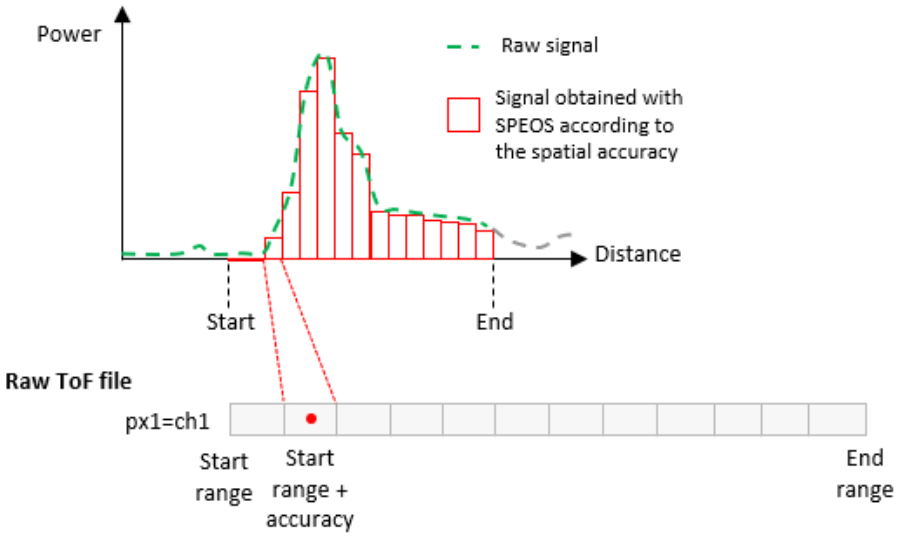




$$\Delta_t = t_1 - t_0 = \frac{2 * z}{c}$$

C: speed of light

In Speos, the raw time of flight data correspond to the optical **power** integrated in the pixel for a given **distance**. This distance depends on the **Spatial accuracy** of the sensor, that is its discrimination step between two measurement points (in mm).



Data Storage

> **CAUTION:** The simulation can take time according to how you configured the LiDAR sensor, due to the memory needed to generate a large amount of data. To give an idea of the amount of data to be generated you can make this operation:
>
> **Number of rotations** * **Number of scans** * **Number of time of flight samples** * **8 bytes**
>
> With **Number of time of flight samples** = (End range - Start range) / Spatial accuracy

The raw ToF result file includes different types of information:

- Generic information (often related to input parameters): sensor pixels' number and size, sensor min and max detection range, spatial accuracy, number of channels, source and sensor origins and orientation etc.
- Channels (corresponding to beams) with the different time of flight data comprising the power, line of sight directions and detection range.

Data is organized by list of channels, one channel corresponding to a fire of the emitter:

- Static LiDAR sensor generates one channel.
- Scanning LiDAR sensor generates one channel per scan configuration.
- Rotating LiDAR sensor generates one channel per rotation configuration, per scan configuration.

Then, each channel contains a list of sub-channels, each sub channel corresponding to one pixel and contains a raw time of flight:

- the unique channel of the Static LiDAR sensor contains a number of sub-channels that corresponds to its resolution.
- each channel of a Scanning / Rotating LiDAR sensor **with no resolution** contains one sub-channel.
- each channel of a Scanning / Rotating LiDAR sensor **with a resolution** contains a number of sub-channels that corresponds to its resolution.

> **Note:** The number of pixels (corresponding to the resolution) is set according to the Horizontal and Vertical pixels parameters of the Sensor Imager.

DRAFT

To extract or post process the raw data stored in the compressed binary file, you must parse the result file using scripts and dedicated APIs.

Related tasks

Accessing the OPTTimeOfFlight File on page 460
This procedure describes how to read an *.OPTTimeofFlight result file through scripts.

Related reference

List Of Methods on page 455
This page describes the methods that should be used to access the data stored in the *.OPTTimeOfFlight file generated as an output of a LiDAR simulation.

10.7.4.3.1. List Of Methods

This page describes the methods that should be used to access the data stored in the *.OPTTimeOfFlight file generated as an output of a LiDAR simulation.

As the *.OPTTimeOfFlight is a binary file, you must use specific methods to access its content and retrieve its data. Make sure to use the **3.9 version** of IronPython or Python language to write your scripts.

> **Tip:** Use script examples as a starting point. The provided scripts are directly compatible with Speos and contain all the methods described below.

Basic Functions

Name	Description	Syntax
OpenFile	Opens the OPTTimeOfFlight file and loads its content	object.OpenFile(BSTR bstrFileName) as Boolean • Object: Raw Time Of Flight File Editor • bstrFileName: path and filename Should end by .OPTTimeOfFlight
GetNumberOfChannels	Returns the number of channels (i.e. the sensor resolution)	object.GetNumberOfChannels() as Short Object: Raw Time Of Flight File Editor

Axis system

Name	Description	Syntax
GetSourceOrigin	Returns the (x, y, z) coordinate corresponding to the origin of the source	object.GetSourceOrigin() as Variant Object: Raw Time Of Flight File Editor
GetSourceXDirection	Returns the (x, y, z) vector corresponding to the X direction of the source	object.GetSourceXDirection() as Variant Object: Raw Time Of Flight File Editor

DRAFT

Name	Description	Syntax
GetSourceYDirection	Returns the (x, y, z) vector corresponding to the Y direction of the source	object.GetSourceYDirection() as Variant Object: Raw Time Of Flight File Editor
GetSensorOrigin	Returns the (x, y, z) coordinate corresponding to the origin of the sensor	object.GetSensorOrigin() as Variant Object: Raw Time Of Flight File Editor
GetSensorXDirection	Returns the (x, y, z) vector corresponding to the X direction of the sensor	object.GetSensorXDirection() as Variant Object: Raw Time Of Flight File Editor
GetSensorYDirection	Returns the (x, y, z) vector corresponding to the Y direction of the sensor	object.GetSensorYDirection() as Variant Object: Raw Time Of Flight File Editor
GetSystemOrigin	Returns the (x, y, z) coordinate corresponding to the origin of the system	object.GetSystemOrigin() as Variant Object: Raw Time Of Flight File Editor
GetSystemXDirection	Returns the (x, y, z) vector corresponding to the X direction of the system	object.GetSystemXDirection() as Variant Object: Raw Time Of Flight File Editor
GetSystemYDirection	Returns the (x, y, z) vector corresponding to the Y direction of the system	object.GetSystemYDirection() as Variant Object: Raw Time Of Flight File Editor

Operating range

Name	Description	Syntax
GetSensorMinRange	Returns the minimum range of the sensor	object.GetSensorMinRange() as Double Object: Raw Time Of Flight File Editor
GetSensorMaxRange	Returns the maximum range of the sensor	object.GetSensorMaxRange() as Double Object: Raw Time Of Flight File Editor
GetSensorAccuracy	Returns the distance step between two energy measurements	object.GetSensorAccuracy() as Double Object: Raw Time Of Flight File Editor

Sensor

Name	Description	Syntax
GetSensorFocal	Returns the sensor's focal length (in mm)	object.GetSensorFocal() as Double Object: Raw Time Of Flight File Editor
GetSensorPixelHorizontalSize	Returns the horizontal size of pixel	object.GetSensorPixelHorizontalSize() as Double Object: Raw Time Of Flight File Editor

DRAFT

Name	Description	Syntax
GetSensorPixelVerticalSize	Returns the vertical size of pixel	object.GetSensorPixelVerticalSize() as Double Object: Raw Time Of Flight File Editor
GetSensorHorizontalPixels	Returns the number of horizontal pixels	object.GetSensorHorizontalPixels() as Double Object: Raw Time Of Flight File Editor
GetSensorVerticalPixels	Returns the number of vertical pixels	object.GetSensorVerticalPixels() as Double Object: Raw Time Of Flight File Editor
GetSensorPupilDiameter	Returns the pupil diameter of the sensor	object.GetSensorPupilDiameter() as Double Object: Raw Time Of Flight File Editor
GetDistortionObjectAngles	Returns the object angle values from the distortion file	object.GetDistortionObjectAngles() as Double object: Raw Time Of Flight File Editor object
GetDistortionImageAngles	Returns the image angle from the distortion file	object.GetDistortionImageAngles() as Double object: Raw Time Of Flight File Editor object

Raw Time Of Flight Data

Name	Description	Syntax
GetChannelPulseEnergy	Returns the value of the pulse energy in Joules (J)	object.GetChannelPulseEnergy(int iChannel) as Double • object: Raw Time Of Flight File Editor object • iChannel: index of the channel
GetChannelAzimuthAngle	Returns the global azimuth angle of a channel in degrees (°)	object.GetChannelAzimuthAngle(int iChannel) as Double • object: Raw Time Of Flight File Editor object • iChannel: index of the channel
GetChannelElevationAngle	Returns the global elevation angle of a channel in degrees (°)	object.GetChannelElevationAngle(int iChannel) as Double • object: Raw Time Of Flight File Editor object • iChannel: index of the channel
GetChannelLineOfSightDirection	Returns the (x, y, z) vector corresponding to the line of sight direction of a channel	object.GetChannelLineOfSightDirection(int iChannel) as Variant • object: Raw Time Of Flight File Editor object • iChannel: index of the channel
GetChannelTimeStamp	Returns the time stamp of a channel	object.GetChannelTimeStamp(int iChannel) as Double • object: Raw Time Of Flight File Editor object • iChannel: index of the channel

DRAFT

Name	Description	Syntax
GetChannelTimeOfFlight	Returns a table corresponding to the received energy during the time of flight for a channel in Watt (W)	object.GetChannelTimeOfFlight(int iChannel, iPixel) as Variant • object: Raw Time Of Flight File Editor object • iChannel: index of the channel • iPixel: index of the pixel
GetChannelSensorOrigin	Returns the (x, y, z) coordinates corresponding to the origin of the sensor at a channel timestamp	object.GetChannelSensorOrigin(int iChannel) as Variant • object: Raw Time Of Flight File Editor object • iChannel: index of the channel
GetChannelSensorXDirection	Returns the (x, y, z) vector corresponding to the X direction of the sensor at a channel timestamp	object.GetChannelSensorXDirection (int iChannel) as Variant • object: Raw Time Of Flight File Editor object • iChannel: index of the channel
GetChannelSensorYDirection	Returns the (x, y, z) vector corresponding to the Y direction of the sensor at a channel timestamp	object.GetChannelSensorYDirection (int iChannel) as Variant • object: Raw Time Of Flight File Editor object • iChannel: index of the channel
GetChannelSourceOrigin	Returns the (x, y, z) coordinates corresponding to the origin of the source at a channel timestamp	object.GetChannelSourceOrigin (int iChannel) as Variant • object: Raw Time Of Flight File Editor object • iChannel: index of the channel
GetChannelSourceXDirection	Returns the (x, y, z) vector corresponding to the X direction of the source at a channel timestamp	object.GetChannelSourceXDirection (int iChannel) as Variant • object: Raw Time Of Flight File Editor object • iChannel: index of the channel
GetChannelSourceYDirection	Returns the (x, y, z) vector corresponding to the Y direction of the source at a channel timestamp	object.GetChannelSourceYDirection (int iChannel) as Variant • object: Raw Time Of Flight File Editor object • iChannel: index of the channel
GetChannelSystemOrigin	Returns the (x, y, z) coordinates corresponding to the origin of the system at a channel timestamp	object.GetChannelSystemOrigin (int iChannel) as Variant • object: Raw Time Of Flight File Editor object • iChannel: index of the channel
GetChannelSystemXDirection	Returns the (x, y, z) vector corresponding to the X direction of the system at a channel timestamp	object.GetChannelSystemXDirection (int iChannel) as Variant • object: Raw Time Of Flight File Editor object • iChannel: index of the channel

DRAFT

Name	Description	Syntax
GetChannelSystemYDirection	Returns the (x, y, z) vector corresponding to the Y direction of the system at a channel timestamp	object.GetChannelSystemYDirection (int iChannel) as Variant • object: Raw Time Of Flight File Editor object • iChannel: index of the channel

Related concepts

Raw Time of Flight on page 452
This page provides more information on the Raw Time of Flight (*.OPTTimeOfFlight) file and describes how to operate/analyze its content.

Related tasks

Accessing the OPTTimeOfFlight File on page 460
This procedure describes how to read an *.OPTTimeofFlight result file through scripts.

10.7.4.3.2. Script Examples

Different script (*.scscript and *.py) files are provided for each type of LiDAR sensor to help you read and extract data from the Raw ToF result file.

The following script examples files are directly compatible with Speos and provide different types of results as they extract specific data out of the Raw time of flight result file.

> **Note:** Scripts are provided in IronPython language and, when possible, in Python language. With Python scripts, make sure to use Python **3.9 version**.

- **Static_LiDAR**

  This script allows you to parse the .OPTTimeOfFlight result in order to export the coordinates of the impacts and their associated power and distance into a *txt file.

- **Scanning_Rotating**

  This script allows you to parse the .OPTTimeOfFlight result in order to extract the following parameters:

  - Get (scanning, rotating) or calculate (static) the line of sight vector.
  - Find the peak at 0.1% of the maximum value or obtain the list of peaks above the minimum signal level (peformed with the **isSinglePeak** parameter).
  - Export the x, y, z coordinates and the energy value for a given distance in a text file.

- **Draw_point_cloud**

  This script includes the same methods as the **Scanning_Rotating** script but uses the point cloud export to project it directly in the 3D view. 3D points with a distance-related color scale (from closest in red, to far in blue) are projected in the 3D view to easily visualize the impacts of the LiDAR sensor.

Related reference

List Of Methods on page 455

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

This page describes the methods that should be used to access the data stored in the *.OPTTimeOfFlight file generated as an output of a LiDAR simulation.

10.7.4.3.3. Accessing the OPTTimeOfFlight File

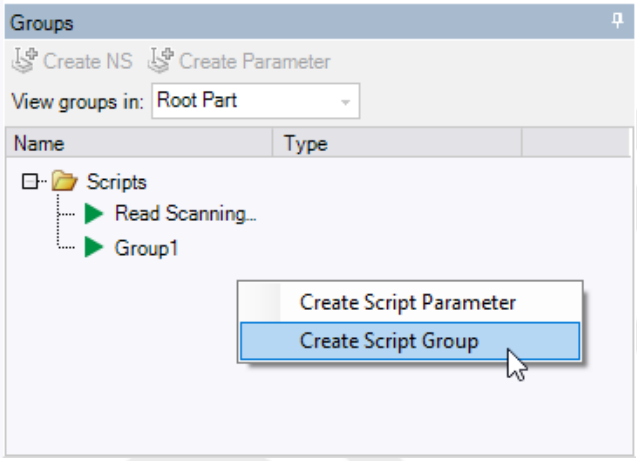
This procedure describes how to read an *.OPTTimeofFlight result file through scripts.

The result file must be generated and stored in folder with full access rights.

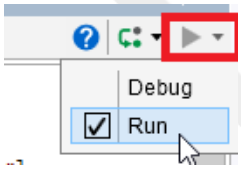
1. Prepare the script file that will be used to read the result:
   a) Download a script example.
   b) Adjust the result file paths to point to your OPTTimeOfFlight file(s).
   c) Enrich the script with your own functions (for example, peak detection functions etc.).

> **Note:** You can also write your script in IronPython or Python language from scratch using dedicated APIs.

2. In Speos, right-click in the **Groups** panel and click **Create Script Group**.



3. Right-click the newly created script group and click **Edit Script** to open the script in the command interpreter.
4. From the script editor, click  to browse and load a .scscript or .py script file.
5. To improve performance, make sure the script editor is not in **Debug** but in **Run** mode.



6. Click **Run**.

Depending on your script configuration, different outputs are generated in the **SPEOS output files** folder and/or visualizations are generated in the 3D view.

Related concepts

Raw Time of Flight on page 452

This page provides more information on the Raw Time of Flight (*.OPTTimeOfFlight) file and describes how to operate/analyze its content.

10.7.4.4. LiDAR Projected Grid Parameters

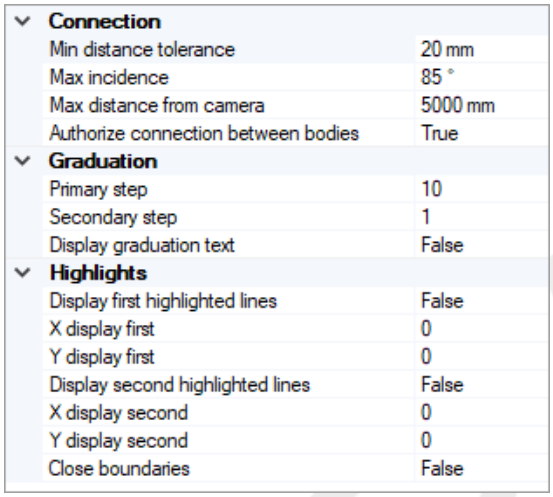
Three projected grid are generated as field of view result from a Static LiDAR simulation.

The first projected grid represents the overlap of the source field of view and sensor field of view corresponding to the LidAR field of view.

The second projected grid represents the source field of view.

The last projected grid represents the sensor field of view.

Once a grid is generated, you can edit its parameters. Each time you generate a grid, the grid uses the parameters of the previous generated grid as default parameters.



Connection

With grid connection parameters, you can connect two adjacent samples of the grid that do not belong to the same body.

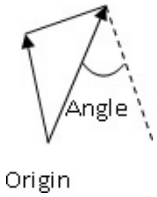
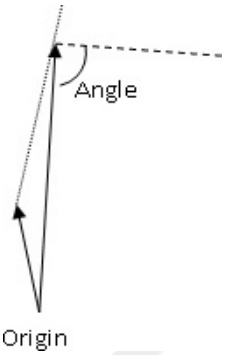
To connect two adjacent samples, they need to fulfill one of the two parameters **Min distance tolerance** (mm) or **Max incidence** (deg):

- The parameter Min distance tolerance (mm) has priority over the parameter Max incidence (deg).
- If the two adjacent samples do not fulfill the parameter Min. distance tolerance (mm), then Speos checks if they fulfill the parameter Max incidence (deg).
- The two adjacent samples can fulfill both parameters.

Parameters

- **Min distance tolerance** (mm): The distance tolerance for which two adjacent samples to be connected by a line. Example: for a Min. distance tolerance of 5mm, all adjacent samples, for which the distance is less than 5mm, are connected by a line.
- **Max incidence**: Maximum angle under which two projected samples should be connected by a line. Example: for a Max. incidence of 85°, if the angle to the normal (normal of the plane of the two pixels) of the farther sample from the origin is less than 85°, then the two samples are connected by a line.

DRAFT

	
Angle 45°: Connection	Angle 88°: No connection

- **Max distance from camera** (mm): Maximum distance between a sample and the LiDAR source/sensor. With maximum distance from camera, you can limit the visualization at a specific distance of the LiDAR source/sensor.
- **Authorize connection between bodies**: allows you to decide to display the connection between bodies that fulfill one of the parameters (Min. distance tolerance or Max. incidence).

Graduation

With the grid graduations, you can modify the two levels of graduation, **Primary step** (yellow default color) and **Secondary step** (green default color).

To lighten the visualization, we recommend you to increase the graduation step parameters when the grid resolution becomes high.

> **Note:** Setting the graduation steps to zero prevents the display of the grids.

Highlights

These parameters allow to define four lines to highlight on the grid.

10.8. Geometric Rotating LiDAR Simulation

This section introduces Rotating LiDAR's principles and describes how to create a Geometric Rotating LiDAR simulation.

10.8.1. Understanding Rotating LiDAR Simulation

This page describes core LiDAR principles and introduces the Rotating LiDAR feature.

LiDAR Principle

A rotating LiDAR is a distance measuring device sending short and rapid pulses of light in specific angular directions while rotating to capture information about its surrounding environment. It works on the principle of a radar but uses light instead of radio waves.

The most obvious advantage of a rotating LiDAR lies in its capacity to cover a 360° field of view. In contrast, Solid State LiDARs rarely exceed a 120° field of view.

Speos Geometric Rotating LiDAR Simulation allows you to reproduce the behavior of a rotating LiDAR.

DRAFT

Geometric Rotating LiDAR Simulation in Speos

The rays are sent in specific angular directions and either interact with a geometry or vanish in the environment.

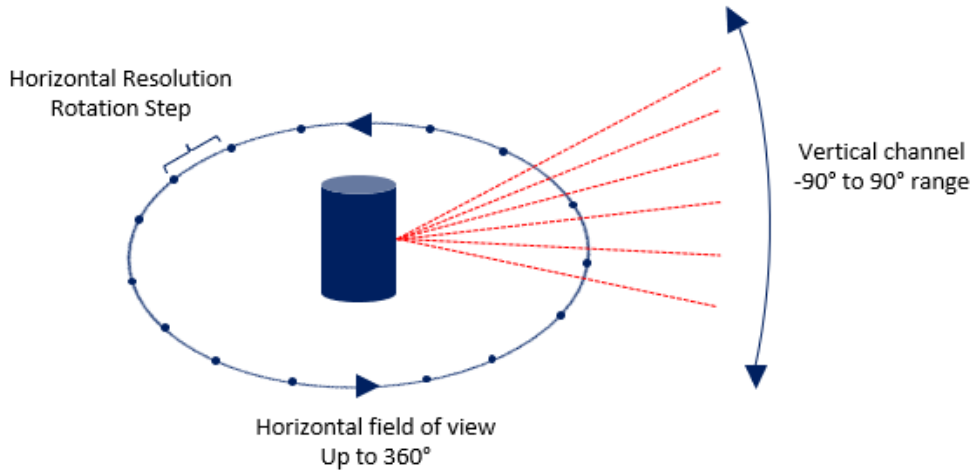
> **Note:** The sensor does not consider the geometries' optical properties.

At the end of the simulation, interactions and impacts are displayed in the 3D view to represent the LiDAR field of view.

To perform a custom field of view study, you can control every aspect of the LiDAR scanning pattern:

- Horizontal field of view: allows you to define the azimuthal range (up to 360°) and sampling of LiDAR's aiming area.
- Vertical field of view:allows you to define the elevation for each channel of the LiDAR.

Figure 61. Geometric Rotating LiDAR scanning pattern



Rotating LiDAR Configuration

Speos LiDAR covers the standard Rotating LiDAR configuration, which means:

- The Speos LiDAR can analyze a 360° field of view,
- The distance is measured with a single light pulse.

LiDAR Simulations Results

Geometric Rotating LiDAR Simulation provides two types of results:

- **Fields of View**: Each ray and its corresponding impact with the geometries are displayed in the 3D view. The set of rays allows you to visualize the sensor's viewing range and reach.
- **Point Cloud**: This .txt file contains all points (impacts) collected during simulation. Each impact represents an individual point with its own set of coordinates x, y, z (millimeters), the power (Watt) measured at the impact, and the distance (millimeter) between the impact and the receiver. This file can then be used as a future input in other simulation software.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Related information

Creating a Geometric Rotating LiDAR Simulation on page 464
Creating a Geometric Rotating LiDAR simulation allows you to perform field of view studies. A field of view study allows you to quickly identify what can or must be optimized (for example, the number, position and direction of sensors) in a LiDAR system.

10.8.2. Creating a Geometric Rotating LiDAR Simulation

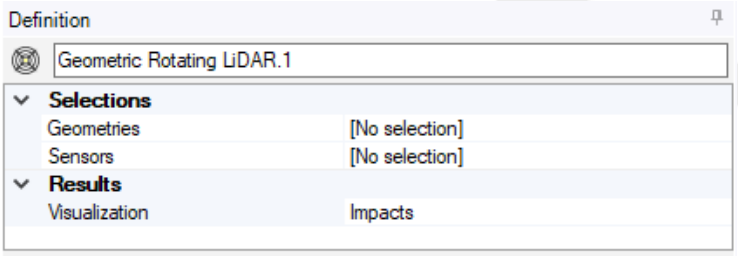
Creating a Geometric Rotating LiDAR simulation allows you to perform field of view studies. A field of view study allows you to quickly identify what can or must be optimized (for example, the number, position and direction of sensors) in a LiDAR system.

> **Note:** The Geometric Rotating LiDAR simulation supports several sensors at a time.

To create a Geometric Rotating LiDAR Simulation:

A sensor must already be created.

1. From the **Light Simulation** tab, click **System** > **Geometric Rotating LiDAR**  .



2. In the 3D view or from the tree, click  to select the geometries for which you want to calculate the distance to the rotating LiDAR.

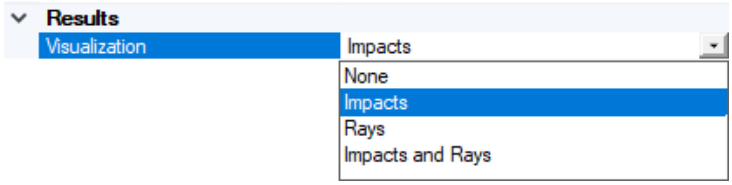
   *Your selection appears in the **Geometries** list.*

   > **Note:** In some cases, using geometries that have no optical properties applied for simulation can lengthen the simulation's initialization time.

3. In the 3D view click  and from the tree select the previously created LiDAR sensor(s).

   *Your selection appears in the **Sensors** list.*

4. From the **Visualization** drop-down list, select which type of results you want to display in the 3D view:



Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- **None**
- **Impacts**
- **Rays**
- **Impacts and Rays**

5. In the 3D view, click **Compute**  to launch the simulation.

The simulation is created and results appear both in the tree and in the 3D view.

Depending on the visualization type selected, impacts and /or rays are displayed in the scene, allowing you to visualize the LiDAR's field of view and viewing range. The distance from the LiDAR to the detected object is represented with a color scale.

Related tasks

Creating a Geometric Rotating LiDAR Sensor on page 298
This page shows how to create a Geometric Rotating LiDAR sensor that will be used for LiDAR simulation.

10.9. Light Expert

The Light Expert is a tool that allows you to specify what ray path to display in the 3D view.

10.9.1. Understanding the Light Expert

The following page presents you the Light Expert and the context of use.

Light Expert is a tool that allows you to interact dynamically with the rays and visualize in real-time the impacts of certain modifications on the rays' behavior.

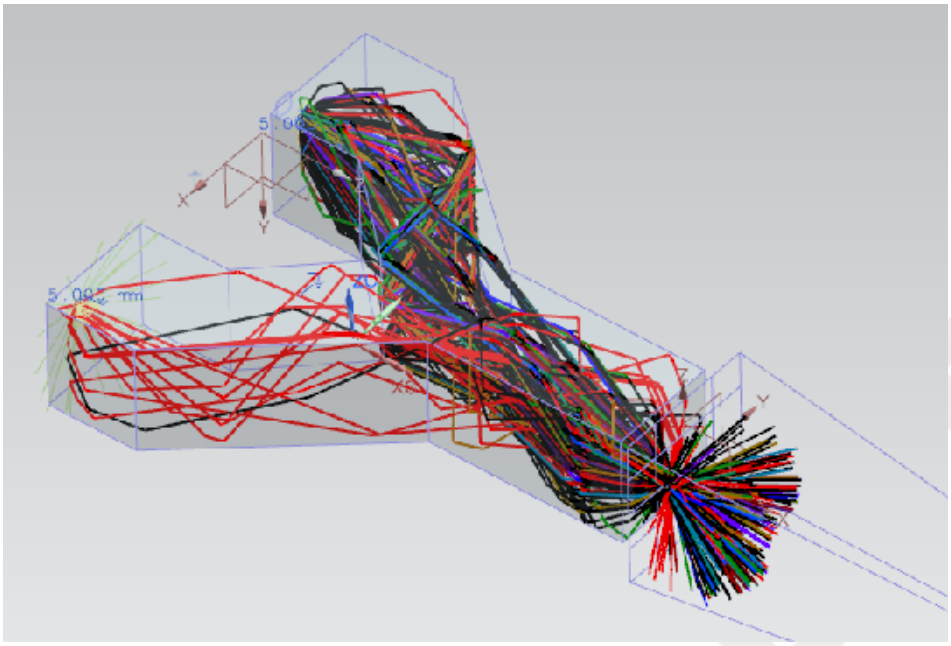
This tool is useful to analyze an optical system, identify light paths, source's contribution and see how light adjusts its trajectory with certain constraints.

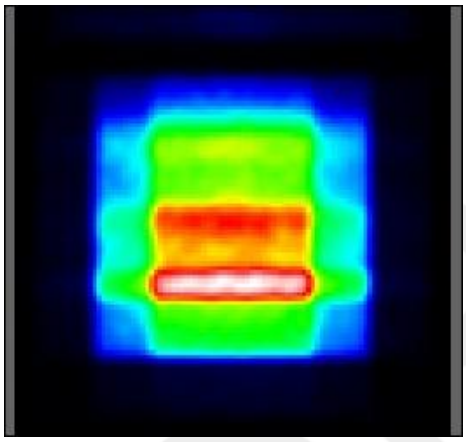
On the technical part, with the Light Expert, you can dynamically filter the rays displayed in the 3D view thanks to two parameters (Required faces / Rejected faces). These two parameters allow to filter the results and visualize light behavior and adjustment when defining path constraints. The rays displayed in the 3D view automatically adjust as you modify these parameters.

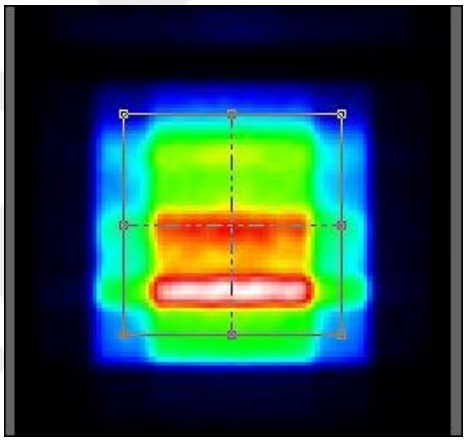
In the case of a XMP map (for direct or inverse simulations), you can create an area of measure on the map to get the ray tracing corresponding to that specific area (the light path from the sources to the sensors).

DRAFT

Figure 62. Example of rays displayed for an interactive simulation







Area of measure applied on a XMP map to display the rays passing through that area

Types of Light Expert Analysis

According to your optical system and your configuration you can perform a single-sensor Light Expert analysis or a multi-sensors Light Expert analysis.

Single-Sensor Light Expert Analysis

The Single-Sensor Light Expert Analysis allows you to analyze each sensor individually.

Depending on the selected sensors, two types of files may be generated:

- ***.lpf files**: they are standard light expert files generated when LXP has been activated for interactive, direct or inverse simulations.
- ***.lp3 files**: they are generated when 3D sensors are comprised in the simulation.

Multi-Sensors Light Expert Analysis

The Multi-Sensors Light Expert Analysis allows you to perform a Light Expert analysis for several sensors at the same time.

- A Multi-Sensors Light Expert Analysis can only be performed with a Direct Simulation for Irradiance and Intensity sensors included in a Light Expert Group.
- A Direct Simulation can only contain one Light Expert group.
- At the end of the simulation a *.lpf file is generated and one xmp is generated for each sensor contained in the group.

> **Note:** For more information about *.lpf result management, refer to the Light Path Finder Result section.

10.9.2. Understanding the Light Expert Parameters

The following section describes the Light Expert parameters to set in simulation.

LPF Max Path Parameter

The LPF max path corresponds to the maximum number of rays the Light Path Finder file (*.lpf or *.lp3) can contain.

According to the simulation, the rays are integrated differently in the file:

- In Direct Simulation, rays are selected randomly.
- In Inverse Simulation, rays are basically selected according to the order in which the pixels of the map are computed (1st pixel with all wavelength sample, 2nd pixel with all wavelength samples, etc.).

  Example: For a 1000*1000 pixel map with 10 wavelength, 10e6 rays are propagated per pass in the system. When setting LPF max path to 10e6, only 10% of the map is covered.

We recommend you to set a LPF max path parameter equivalent to the number of rays propagated in the system. Otherwise, when LPF max path is much more smaller than the number of rays, information will not be integrated in the analysis.

> **Warning:** The *.lpf file size increases with the number of rays interaction. The more interactions in the system, the bigger the *.lpf file size.
>
> To avoid such situation, you can create small sensor (not a lot of pixel) to cover the area on which you want to perform a Light Expert analysis with a *.lpf file.

10.9.3. Performing a Single-Sensor Light Expert Analysis

The following procedure helps you perform a light expert analysis on one sensor.

To perform a Single-Sensor Light Expert Analysis:

1. Create a Direct, Inverse, or Interactive simulation.
2. Set **Light Expert** to **True**.
3. In case of a Direct or Inverse simulation:

a) Define the **LPF max path**.

We recommend you to set a LPF max path parameter equivalent to the number of rays propagated in the system. Otherwise, when LPF max path is much more smaller than the number of rays, information will not be integrated in the analysis.

> **Warning:** The *.lpf file size increases with the number of rays interaction. The more interactions in the system, the bigger the *.lpf file size. To avoid such situation, you can create small sensor (not a lot of pixel) to cover the area on which you want to perform the Light Expert analysis with the *.lpf file.

b) Add one or more sensors to the **Sensors** list of the simulation.

c) In the **Sensors** list, check **LXP** to define which sensor(s) to analyze.

4. Run the simulation.
5. To visualize and analyze the result, double-click the *.lpf or *.lp3.

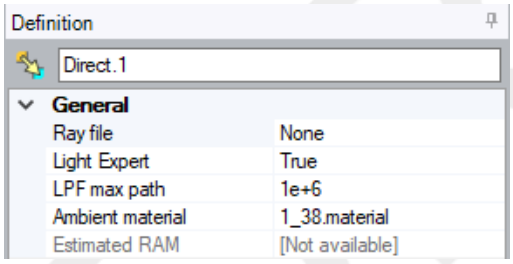
   Refer to Light Path Finder Results according to the analysis to perform.

10.9.4. Performing a Multi-Sensors Light Expert Analysis

The following procedure helps you perform a light expert analysis on several sensors at the same time.

To perform a Multi-Sensors Light Expert Analysis:

1. Create a Light Expert Group.
2. Create a Direct simulation.
3. In the **General** section, set **Light Expert** to **True**.



4. Add the **Light Expert Group** to the **Sensors** list of the simulation.

   As Light Expert is set to **True**, the **LXP** check box is of the Light Expert Group is automatically activated.

> **Note:** Only one Light Expert Group can be added to the simulation.

5. Define the **LPF max path**.

   We recommend you to set a LPF max path parameter equivalent to the number of rays propagated in the system. Otherwise, when LPF max path is much more smaller than the number of rays, information will not be integrated in the analysis.

DRAFT

> **Warning:** The *.lpf file size increases with the number of rays interaction. The more interactions in the system, the bigger the *.lpf file size. To avoid such situation, you can create small sensor (not a lot of pixel) to cover the area on which you want to perform the Light Expert analysis with the *.lpf file.

6. Run the simulation.
7. To visualize and analyze the result, double-click the *.lpf file.
   The Light Expert opens as well as each XMP map generated for each sensor of the Light Expert Group.

10.10. Propagation Errors

The following chapter helps you understand how to visualize and manage potential propagation errors in your optical system.

10.10.1. Propagation Errors Overview

The following page introduces you with the propagation erros and how to analyze and solve them.

Context

During the ray propagation of a Speos simulation, errors may occur and have different origins.

The simulation tracks the number of encountered errors, according to different causes.

- The error report part of the simulation report indicates the corresponding percentage of errors. However, most of the time, it is difficult to identify the root causes only through the report and it requires a lot of Speos experience to decrease the error rate.
- A specific *.lpf file can be generated storing the rays in error to help you analyze and decrease the percentage of errors.

Error Report

The propagation error is expressed as a fraction of the rays emitted (**Total number of errors** in the HTML report) or a fraction of the energy emitted (**Error power** in the HTML report).

We usually recommend a Total number of errors rate below 3%, and below 1% in case of a sensitive project.

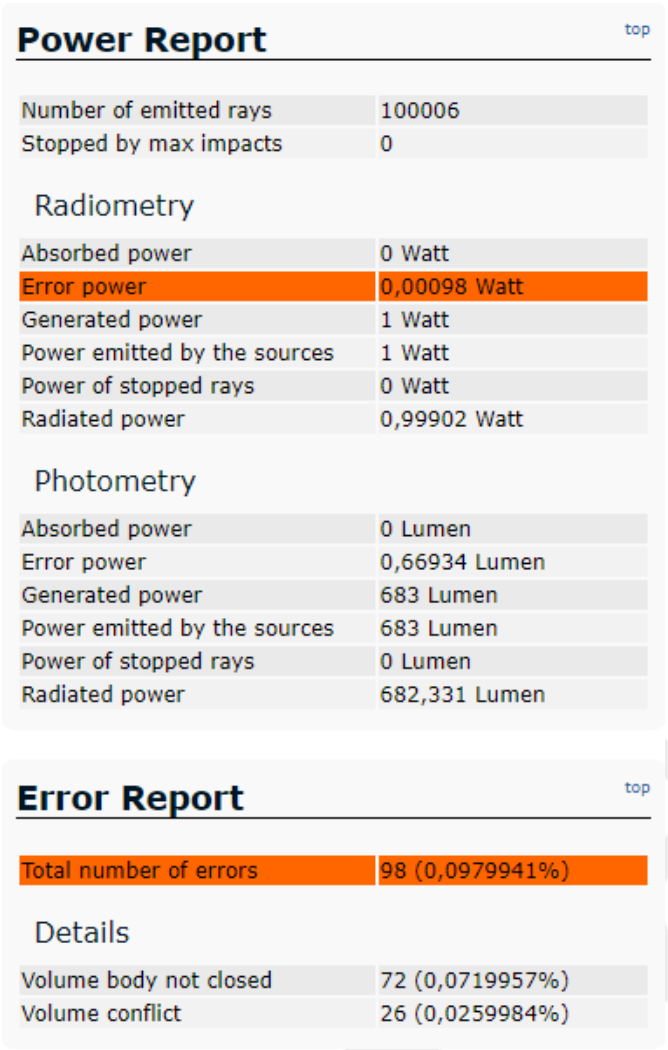
> **Note:** The fraction of energy emitted is usually more valuable to assess the criticality of error amount: you may have plenty of rays in error in a part insignificant to your project because almost no energy is propagated there.

In the following report example, 100006 rays are emitted. 98 rays are in error representing an energy of 0,00098 Watt.

> **Warning:** The more propagation errors you have, the more visible the difference between the CPU result and the GPU result for a same simulation. For more information, refer to GPU/CPU Differences.

DRAFT

Figure 63. Report Example



Power Report top

Number of emitted rays	100006
Stopped by max impacts	0

Radiometry

Absorbed power	0 Watt
Error power	0,00098 Watt
Generated power	1 Watt
Power emitted by the sources	1 Watt
Power of stopped rays	0 Watt
Radiated power	0,99902 Watt

Photometry

Absorbed power	0 Lumen
Error power	0,66934 Lumen
Generated power	683 Lumen
Power emitted by the sources	683 Lumen
Power of stopped rays	0 Lumen
Radiated power	682,331 Lumen

Error Report top

Total number of errors	98 (0,0979941%)

Details

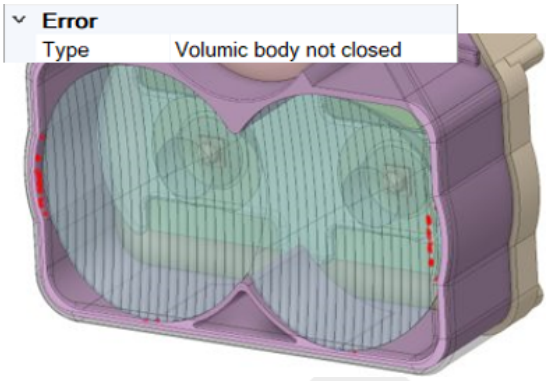
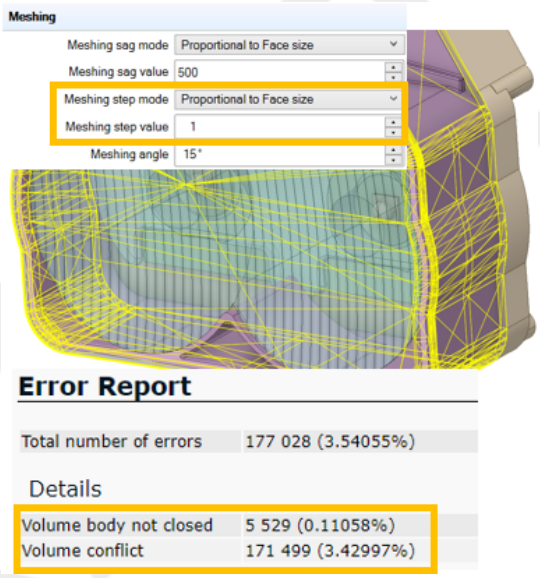
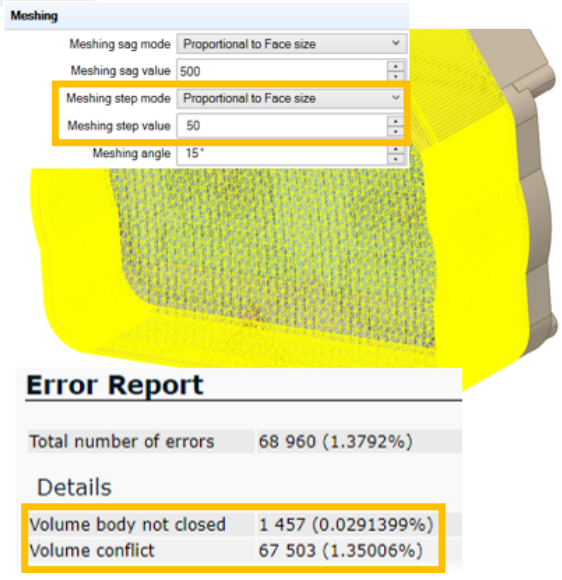
Volume body not closed	72 (0,0719957%)
Volume conflict	26 (0,0259984%)

Propagation Error File

To help you identify the root causes and decrease the percentage of errors, a *.lpf file dedicated to store only the rays in error can be generated out of the simulation and analyzed. Through this file you can display the rays in error in the 3D view and the type of error so that you can correct your optical system.

DRAFT

Example: decreasing the error rate

The body being the cause of the errors is identified thanks to the rays in error.	
Meshing settings and error report before potential modifications. The error rate is high.	
Meshing settings modifications and error report after modifications. The error rate has decreased.	

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

10.10.2. Understanding Propagation Errors

This page describes the different types of propagation errors that can be encountered during a simulation.

Volume body not closed error

Volume body not closed error occurs when a ray inside a solid body cannot hit a face of this given solid (to go out of this solid), only rays located really close to a corner can be involved. This error can occur either if the solid faces are not really closed (imported geometry) or if a ray cannot detect a face due to the geometrical optical precision. This error can occur especially when VOP on Surface parameter is used. This error can occur if using a geometry thinner than the minimal distance tolerance.

This error can be reduced by improving the precision of the CAD geometry or reducing the geometrical optical precision.

You can also calibrate the propagation parameters and the meshing parameters to reduce the error percentage.

Volume conflict error

Volume conflict error occurs when a photon inside a body hits another solid body. This error can occur especially when VOP on Surface parameter is used.

> **Note:** This propagation error should be corrected by changing the geometry modeling.

Propagation Errors on Corners of 3D geometries

In case of reflection of rays near an angle of a 3D geometry (geometry with corners an edges such as cube, rectangle, prisms, etc.), the next intersection with the geometry could be missed due to a too high geometrical distance tolerance.

Try reducing the Geometrical Distance Tolerance to be as close as possible to the angle.

2D tangency error

2D tangency error occurs when a solid geometry and surface geometry, or a surface geometry and another surface geometry are tangent in a same simulation.

> **Note:** Speos can manage two tangent solid geometries in a same simulation.

> **Note:** This propagation error can be avoided by separating geometric elements by at least geometrical optical precision.

Non optical material error

Non optic material error occurs when a ray enters in a solid body having non optic as volume optical properties.

Note: This propagation error should be corrected by changing the modeling.

Non optical material at emission error

Non optical material at emission error occurs when a ray is directly emitted inside a solid body having non optic as volume optical properties.

10.10.3. Performing a Propagation Errors Analysis

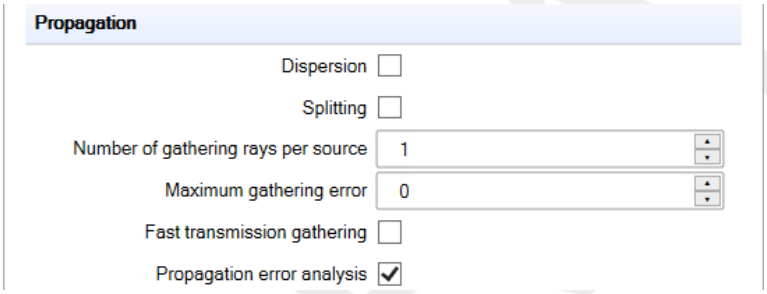
The following procedure helps you perform a propagation errors analysis in order to decrease the errors rate.

Note: The **Show rays in error** tool is in BETA mode for the current release.

To perform a propagation errors analysis:

You must have created a Direct or Inverse simulation.

1. In the **Simulation** tree, right-click the simulation and click **Options** to open the advanced settings.
2. In the Direct/Inverse Simulation tab, check **Propagation error analysis**.

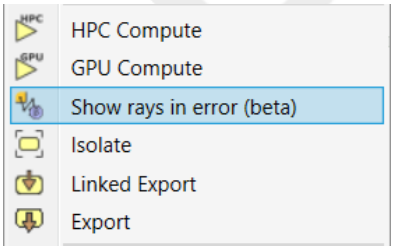


3. Run the simulation.

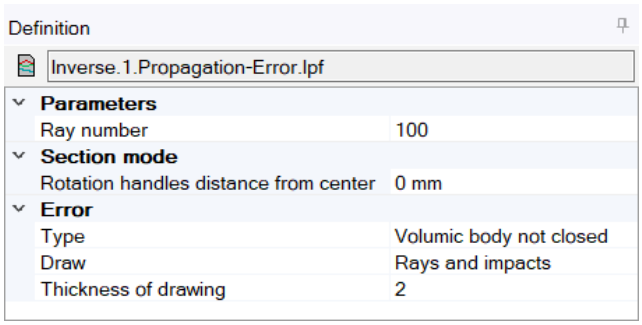
   A *.lpf is generated in the Speos output files folder following this filename format SimulationName.Propagation-Error.lpf.

   The *.lpf file does not appear as a child result of the simulation in the tree.

4. Once the simulation is finished, right-click the simulation in the tree and click **Show rays in error (beta)**  .



5. In the *.lpf definition panel, define what type of error to display and how you want to display them.



- With **Ray number**, control the number of rays to be displayed in the 3D view.
- In the **Error** section, define the error **Type**.

  For more information on the different error types, refer to Understanding Propagation Errors.

- Define what visual information to display between **Last impacts**, **Rays**, **Impacts**, **Rays and Impacts**.
- Define the **Thickness of drawing** that suits best for you.

6. If you want to better visualize where do rays in error pass:

   a) Click **Section mode**  .
   The Section mode appears in the 3D view and you can move it thanks to the 3D Manipulator to modify its position and orientation therefore cutting the geometry and see where do rays pass.

   b) If the 3D Manipulator is not enough discernible, in the **Section mode** section, with the **Rotation handle distance from center** parameter you can modify the distance of the rotation manipulators to the center of the 3D Manipulator.

7. Modify your optical system according to the error analysis.

DRAFT

11: Results

This section describes how to manage result files and reports generated from simulations.

11.1. Reading the HTML Report

This page lists and describes all the information contained in an HTML report (systematically generated when you perform a simulation).

Performance Analysis

- Time analysis: Date and time of the initialization/termination and duration of the simulation.
- Computer Details: list of the computer resources used to run the simulation.

> **Note:** The computer details are useful to assess your needs with Speos HPC.

Power Report

- Number of emitted rays
- **Power emitted by the sources** corresponds to the user defined power of the sources that could be radiometric or photometric. When the user defined power is defined in radiometric unit, the photometric power is computed using the source's spectrum Watt to lumen ratio, and reciprocally.
- **Generated power** corresponds to the power of the rays randomly emitted by the sources the using Monte Carlo algorithm, during the direct simulation.

  Radiometric generated power is always equal to the radiometric power emitted by the sources.

  Photometric generated power is computed from the individual wavelengths of the Monte Carlo emitted rays.

  This value converges towards the photometric emitted power when the number of rays increases.
- **Radiated power** corresponds to the power of the rays that leave the system and that will not interact with it anymore.
- **Absorbed power** corresponds to the power of the rays that are absorbed during the propagation by one element of the system.
- **Error power** corresponds to the power of the rays that are lost to propagation error.

  Refer to the Propagation Errors page for more information the different types of error.
- **Power of stopped rays** corresponds to the power of the rays that are stopped during the propagation because they have reached the **Maximum number of surface interactions**.

> **Note:** **Error power** and **Power of stopped rays** need to be carefully checked after a simulation as they need to be as low as possible to avoid biased result.

DRAFT

Important: In case of Scanning or Rotating LiDAR simulation, Speos compute the sources total power from scanning and rotating pulse parameters contained in the Scanning and Rotating Sequences files:

- For Scanning LiDAR, Speos accumulates the pulses energy.
- For Rotating LiDAR, Speos takes the value computed for scanning and multiplies it by number of rotations.

Error Report

Total number of errors (only for Direct, Inverse simulations)

Results

Preview of the results and associated details.

- **Pixels per body** represents the coverage of a body by the pixels of the sensor.

Note: Pixels per body is available only for inverse simulations with radiance sensors or camera sensors.

Geometry Report

- Number of geometries included in the simulation (number of faces/bodies)
- Optical Properties applied to each geometry.
- Ray tracing technique used for the simulation (Smart Engine, Embree).

Simulation Parameters

Simulation Parameters represents all simulation settings (Meshing, FTG, Weight, etc).

- Comments: user defined comments (only for Interactive, Direct, Inverse simulations)
- CAD Parameters: user defined parameters (for Interactive, Direct, Inverse simulations)
- Design Screenshots: user defined screenshots (for Interactive, Direct, Inverse simulations)
- Statistical Analysis (for Interactive and Inverse simulations)
- Status Details: statistical information about interaction status (for Interactive and Inverse simulations)
- **Luminaire Wattage**: only if luminaire in simulation (for Direct and Inverse simulations)
- **Impacts Report**: detail of impact for all rays (for Interactive simulation)

Related information

Interactive Simulation on page 391
The Interactive Simulation allows you to visualize the behavior of light rays in an optical system.

Inverse Simulation on page 405
The Inverse Simulation allows you to propagate a large number of rays from a camera or a sensor to the sources through an optical system.

Direct Simulation on page 398
The Direct Simulation allows you to propagate a large number of rays from sources to sensors through an optical system.

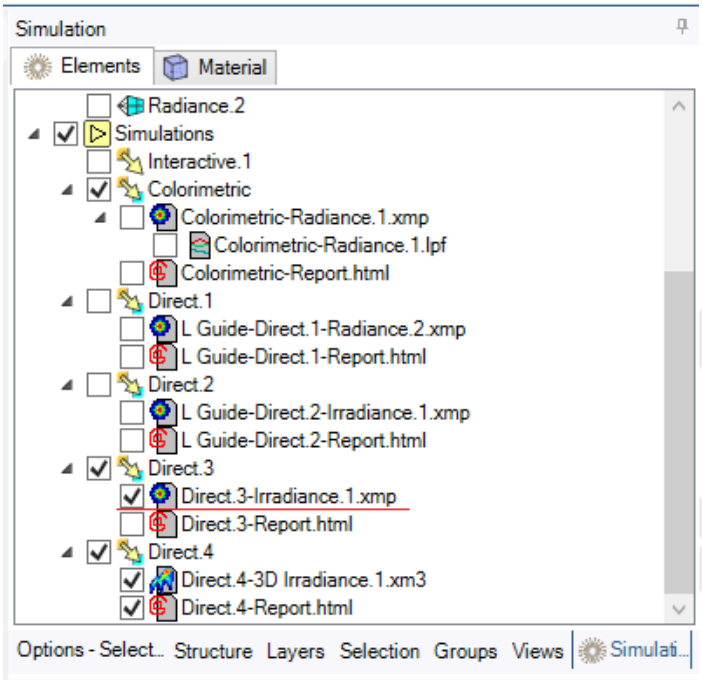
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

11.2. Visualizing Results

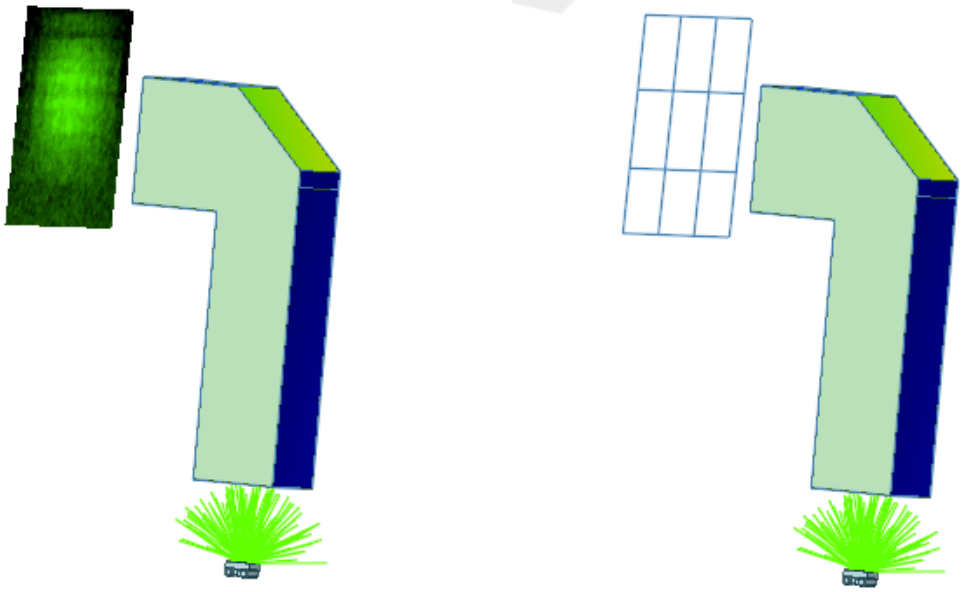
This page describes how to visualize various types of results (.xmp, .xm3, .lpf, .ies or .ldt files etc.).

Simulations results are stored in the SPEOS output files directory of the project and are available from the Speos tree.

To open a simulation result, double-click the result from Speos tree to automatically open it with the associated viewer/editor.



To display or hide a result in the 3D view, check or clear the result's check box.



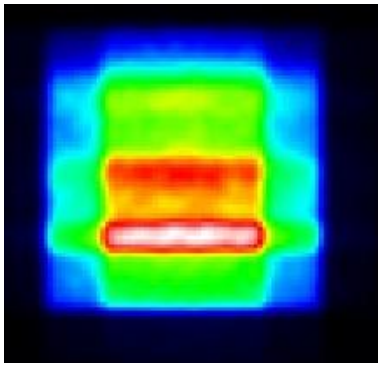
XMP Map (*.xmp)

Extended maps provide photometric or colorimetric information and measures.

XMP results are opened with the **Virtual Photometric Lab**.

> **Note:** XMP results appear blurry if the resolution of the XMP preview displayed in the 3D view is 128x128 or under.
>
> XMP results do not appear in the 3D view when an intensity sensor with a Conoscopic orientation is used for simulation.

Figure 64. Example of XMP Result



Spectral Irradiance Map (*.Irradiance.xmp)

Spectral irradiance maps are a type of extended map that provide the radiometric power or flux received by the pixels of the sensor in W/m^2.

They are generated when using a Camera Sensor with an *.OPTDistortion file V1 or V2 in an Inverse Simulation.

> **Note:** The XMP map generated using either the *.OPTDistortion file V1 or V2 is displayed upside-down in the 3D view and Virtual Photometric Lab to represent the signal measured on the imager of the camera.
>
> The PNG file resulting from the simulation of the Camera Sensor using either the *.OPTDistortion file V1 or V2 is right side up as it corresponds to the post-processed image.

Spectral Exposure Map (*.Exposure.xmp)

Spectral Exposure maps are a type of extended map that provide information on the acquisition of Camera Sensors and express the data for each pixel in $Joules/m^2/nm$.

They are generated for each Camera Sensor during a dynamic Inverse Simulation, that means:

- If the Timeline parameter is actived in the Inverse Simulation
- And at least one Camera Sensor is moving

DRAFT

Note: The XMP map generated using either the *.OPTDistortion file V1 or V2 is displayed upside-down in the 3D view and Virtual Photometric Lab to represent the signal measured on the imager of the camera.

The PNG file resulting from the simulation of the Camera Sensor using either the *.OPTDistortion file V1 or V2 is right side up as it corresponds to the post-processed image.

XM3 Map (*.xm3)

Note: When you open Virtual 3D Photometric Lab from Speos, Virtual 3D Photometric Lab inherits the default navigation commands from SpaceClaim.

The navigation commands are:

- spin: middle mouse button
- spin + center definition: middle mouse button when clicking on the mesh
- pan: SHIFT + middle mouse button
- zoom: CTRL + middle mouse button

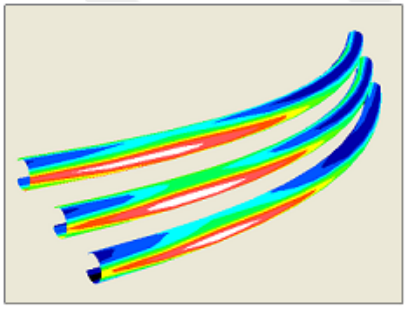
XM3 maps allow to analyze light contributions on a geometry.

XM3 results are opened with the **Virtual 3D Photometric Lab**.

Note: In some cases, the result preview displayed in the 3D view can differ from the viewer's result. It is often the case when working with poorly meshed surfaces.

Note: In case of a large XM3 file, we recommend you to use a powerful GPU for better performance of Virtual 3D Photometric Lab.

Figure 65. Example of XM3 Result



Ray File (*.ray)

Ray files are generated when the ray file generation is activated during a Direct Simulation.

Ray file results store all rays passing through the sensor during propagation. They can be reused to describe a light source emission.

The results are opened with the **Ray File Editor**.

Intensity File (*.ies, *.ldt)

Intensity files come in various formats and are created when using an Intensity Sensor in an Interactive or Direct Simulation.

Depending on the file format, the results might open with:

- **Eulumdat Viewer** for a *.ldt file.
- **IESNA Viewer** for an *.ies file.

Speos360 File (*.speos360)

Speos360 files are generated when an Observer Sensor is used during simulation.

The results are opened with the **Virtual Reality Lab**.



HDRI File (*.hdr)

HDR images are generated at the end of an Inverse Simulation.

The results are opened with the **Virtual Reality Lab**.

Light Path Finder Files (*.lpf, *.lp3)

Light Path Finder files are generated when the Light Expert option is activated during a simulation.

For more information, see Light Path Finder Results.

XMP Map of Sensors Included in Light Expert Group

When a Light Expert Group has the **Layer** type set to **Data separated by Sequence**, the sequence order displayed in Virtual Photometric Lab for each sensor's XMP map changes compared to whether each sensor is included individually in a simulation.

In Virtual Photometric Lab:

- When a sensor is included individually in the simulation, the sequences in the Layer list are sorted by descending order of energy.

DRAFT

- In case of a sensor group:
  - the first sensor (according to its position related to the source) sequences in the Layer list are sorted by descending order of Energy(%) values.
  - all following sensors will display the exact same sequences as the first sensor.

The goal of it is in case of a Light Expert analysis of a sensors's group, when selecting a layer in one of the XMP maps, it applies the same layer to all XMP maps so that it displays the ray path of the layer. This explains why sequences of 2nd and beyond sensors may not be sorted by descending order of energy.

11.3. Light Path Finder Results

This section describes how to manage Light Path Finder results (LPF or LP3 files) for interactive, direct and inverse simulations. *.lpf and *.lp3 files are generated when activating the Light Expert in simulation.

11.3.1. Understanding the Light Path Finder Parameters

The following section describes the Light Path Finder parameters used to visualize the results.

Ray Length Parameter According to Sensors

> **Note:** The Ray Length parameter only set a preview of the length. If you want to export the rays as geometry, the parameter has no impact on the length exported when you use export as geometry. The length of the rays exported depends on the optical system and the simulation parameters.

In Direct Simulation, the **Ray Length** parameter has different meanings according to the sensor used.

- Irradiance Sensor: it corresponds to the length of the rays after the last impact, located on the sensor.

  Irradiance Sensors have a physical location in the system.

- Intensity Sensor: it corresponds to the length of the rays after the last impact on the geometries.

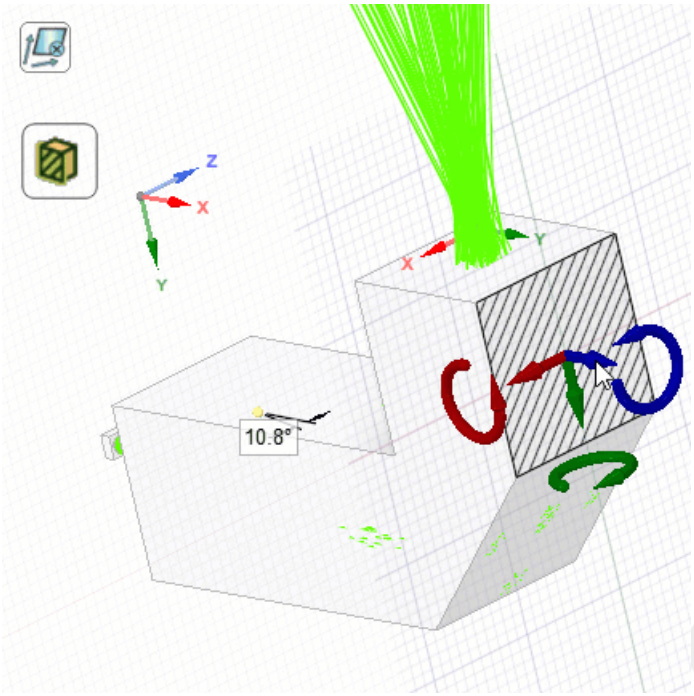
  Intensity Sensors have no physical location in the system, radius is only used to size the sensor in the 3D view and to display the results.

- Radiance Sensor: the **Ray Length** parameter is not used as the propagation stops when rays are integrated by the sensor. What you observe is the path to the observer point.

Section Mode

The Section mode  is a useful tool that allows you to go further into a Light Expert analysis. The tool permits you to modify the position and orientation of a section plan with a 3D Manipulator while performing a Light Expert analysis that requires ray tracing. By positioning the section mode, you can cut a geometry and therefore better see the rays passing through, according to the parameters and area of measure set.

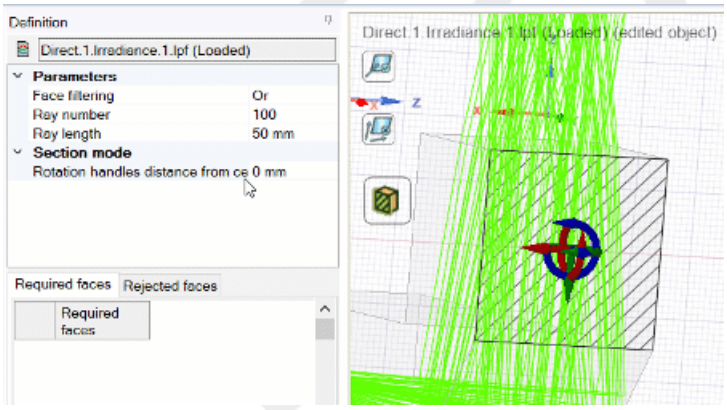
> **Note:** The Section mode only affects the geometries. Rays are not cropped.



Note: The Section mode tool is also useful in case of a Propagation Errors Analysis.

Rotation Handle Distance from Center

The **Rotation handle distance from Center** parameter allows you to modify the distance of the rotation manipulators to the center of the Section mode 3D Manipulator for a better visibility and use.



11.3.2. Visualizing an Interactive Simulation LPF Result

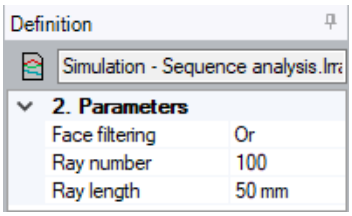
This page shows how to use a LPF result generated from an interactive simulation. LPF files appear only if light expert is activated during the simulation.

To visualize an Interactive Simulation LPF result:

1. To ensure the proper visualization of the *.lpf result, re-compute the simulation when:

- the Draw Rays option was activated for simulation. As this option causes LXP rays and rays of the simulation to overlap, therefore making them indistinguishable from one another, it should be deactivated for simulation.
- the file was generated prior to 2020 R2.

2. From Speos tree, double-click the *.lpf file.

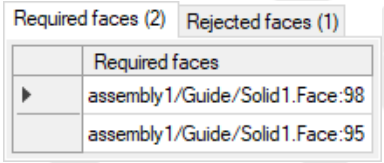


The Light Expert panel is displayed. The Interactive Simulation ray tracing is automatically hidden so that only the ray tracing of the *.lpf result is displayed in the 3D view.

> **Note:** If other Interactive Simulations are displayed in the 3D view, hide them all to display the *.lpf ray tracing wanted only.

> **Note:** When you close the Light Expert panel, the Interactive simulation ray tracing is automatically displayed regardless the hide/show status previously defined.

3. From the **Faces filtering** drop-down list:
   - Select **And** to display rays that have at least one intersection with each selected faces.
   - Select **Or** to display rays that have at least one intersection with one of the selected faces.
4. Define the **Number of rays** to be displayed in the 3D view.
5. In **Ray length**, set a value to define the ray length preview.
6. If you want to filter the rays displayed in the 3D view:



   - Click  and select the **Required faces** (the faces that are taken into account to filter rays).

     The rays impacting the required faces and reaching the sensor are displayed in the 3D view.

   - Click  and select the **Rejected faces** (the faces that you want to exclude from the light trajectory).

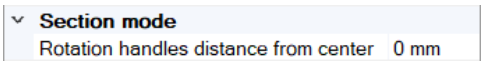
     The rays reaching the sensor, excepted those impacting the rejected faces are displayed in the 3D view.

7. If you want to better visualize rays paths:

   a) Click **Section mode**  .
   The **Section mode** appears in the 3D view and you can move it thanks to the 3D Manipulator to modify its position and orientation therefore cutting the geometry and see the rays paths.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

b) If the 3D Manipulator is not enough discernible, in the **Section mode** section, with the **Rotation handles distance from center** parameter you can modify the distance of the rotation manipulators to the center of the 3D Manipulator.



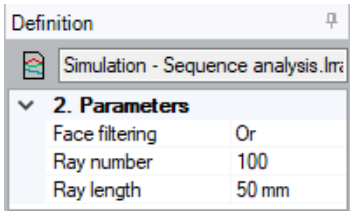
11.3.3. Visualizing an Inverse or Direct Simulation LPF Result

This page shows how to use an LPF result generated from an inverse or a direct simulation. LPF files appear only if light expert is activated during the simulation.

> **Note:** If you need more information, see Light Expert.

To visualize a LPF Result for an Inverse or a Direct Simulation:

1. From Speos tree, double-click the *.lpf file.

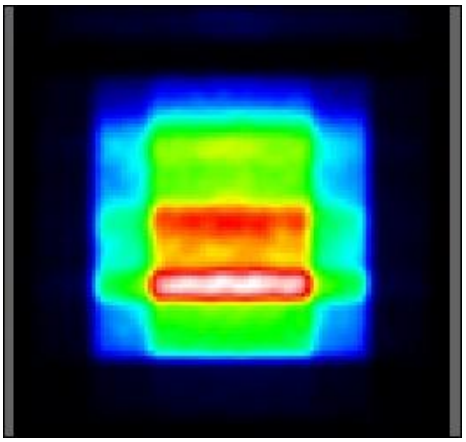
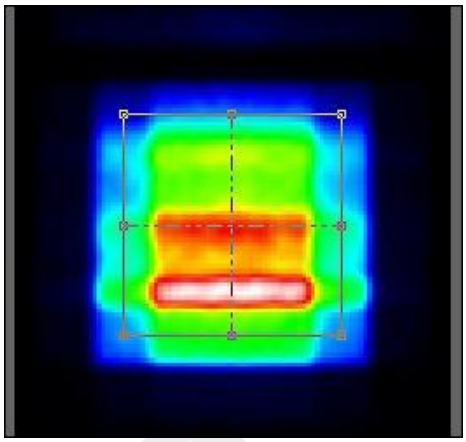


Virtual Photometric Lab loads the XMP result with the first measure area available in the Measures tool on the first opening. The selected measure area (in the Measure tool) upon save is displayed on the next openings. Light Expert panel is displayed.

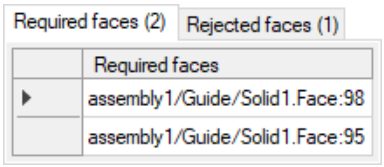
> **Note:** If necessary, you can hide the XMP result of the 3D view for a better display of the *.lpf ray tracing.

2. From the **Faces filtering** drop-down list:
   - Select **And** to display rays that have at least one intersection with each selected faces.
   - Select **Or** to display rays that have at least one intersection with one of the selected faces.
3. Define the **Ray number** to be displayed in the 3D view.
4. In **Ray length**, set a value to define the ray length preview.
5. In **Virtual Photometric Lab**, click **Measure**.

DRAFT

	
Extended map	Extended map with surface tool

6. If necessary, modify the measure area, shape or position on the map to update the ray tracing preview in the 3D view.
   The ray tracing of the 3D view is adjusted in real-time to illustrate the light path from the sources to the sensor.
7. If you want to filter the rays displayed in the 3D view:



   - Click  and select the **Required faces** (the faces that are taken into account to filter rays).

     The rays impacting the required faces and reaching the sensor are displayed in the 3D view.

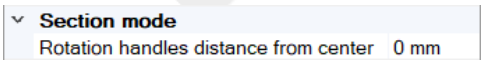
   - Click  and select the **Rejected faces** (the faces that you want to exclude from the light trajectory).

     The rays reaching the sensor, excepted those impacting the rejected faces are displayed in the 3D view.

8. If you want to better visualize rays paths:

   a) Click **Section mode**  .
   The **Section mode** appears in the 3D view and you can move it thanks to the 3D Manipulator to modify its position and orientation therefore cutting the geometry and see the rays paths.

   b) If the 3D Manipulator is not enough discernible, in the **Section mode** section, with the **Rotation handles from center distance** parameter you can modify the distance of the rotation manipulators to the center of the 3D Manipulator.



11.3.4. Visualizing a LP3 Result

This page shows how to use a LP3 result generated from a direct simulation. LP3 files appear only when using light expert in a simulation that contains 3D sensors.

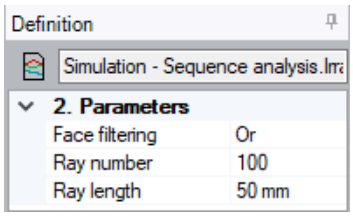
DRAFT

Note: If you need more information on the Light Expert tool, see Light Expert.

To visualize a LP3 Result:

1. From Speos tree, double-click the .lp3 file.

   Virtual 3D Photometric Lab loads the XM3 result. The Light Expert panel is displayed.

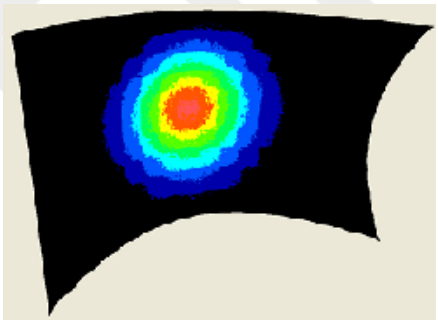
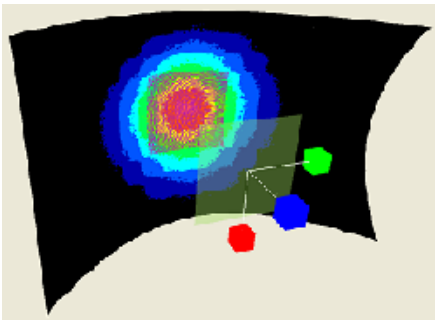


   Note: When you open Virtual 3D Photometric Lab from Speos, Virtual 3D Photometric Lab inherits the default navigation commands from SpaceClaim.

   The navigation commands are:

   - spin: middle mouse button
   - spin + center definition: middle mouse button when clicking on the mesh
   - pan: SHIFT + middle mouse button
   - zoom: CTRL + middle mouse button

2. From the **Faces filtering** drop-down list:
   - Select **And** to display rays that have at least one intersection with each selected faces.
   - Select **Or** to display rays that have at least one intersection with one of the selected faces.
3. Define the **Ray number** to be displayed in the 3D view.
4. In **Ray length**, set a value to define the ray length preview.
5. In **Virtual 3D Photometric Lab**, click **Tools** > **Measure**

	
Extended map	Extended map with measure tool

6. Modify the measure area, shape or position on the map to update the ray tracing preview in the 3D view.
   The ray tracing of the 3D view is adjusted in real-time to illustrate the light path from the sources to the sensor.
7. If you want to filter the rays displayed in the 3D view:

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

- Click and select the **Required faces** (the faces that are taken into account to filter rays).

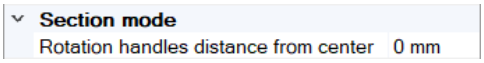
  The rays impacting the required faces and reaching the sensor are displayed in the 3D view.

- Click and select the **Rejected faces** (the faces that you want to exclude from the light trajectory).

  The rays reaching the sensor, excepted those impacting the rejected faces are displayed in the 3D view.

8. If you want to better visualize rays paths:

   a) Click **Section mode** .
   The **Section mode** appears in the 3D view and you can move it thanks to the 3D Manipulator to modify its position and orientation therefore cutting the geometry and see the rays paths.
   b) If the 3D Manipulator is not enough discernible, in the **Section mode** section, with the **Rotation handles distance from center** parameter you can modify the distance of the rotation manipulators to the center of the 3D Manipulator.



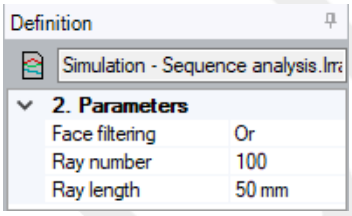
11.3.5. Visualizing a LPF Result for Multi-Sensors Analysis

The Light Expert allows you to specify what ray path to display in 3D view according to required or rejected faces you select and the measures defined in each xmp map.

> **Note:** If you need more information, see Light Expert.

To visualize a LPF Result for a multi sensor analysis:

1. From Speos tree, double-click the *.lpf file.



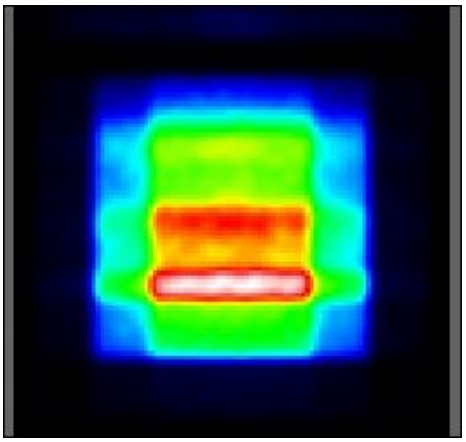
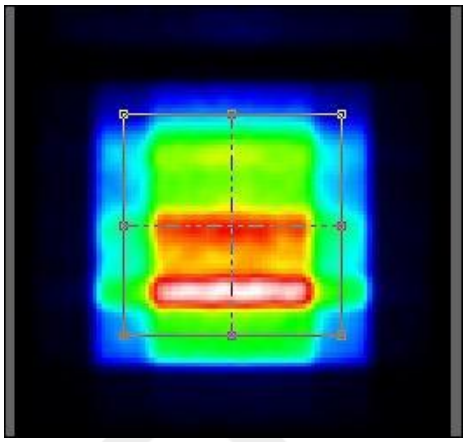
   Virtual Photometric Lab loads each XMP result of each sensor contained in the Light Expert Group with the first measure area available in the Measures tool on the first opening. The selected measure area (in the Measure tool) upon save is displayed on the next openings. Light Expert panel is displayed.

   > **Note:** If necessary, you can hide the XMP results of the 3D view for a better display of the *.lpf ray tracing.

2. From the **Faces filtering** drop-down list:
   - Select **And** to display rays that have at least one intersection with each selected faces.
   - Select **Or** to display rays that have at least one intersection with one of the selected faces.
3. Define the **Ray number** to be displayed in the 3D view.
4. In **Ray length**, set a value to define the ray length preview.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

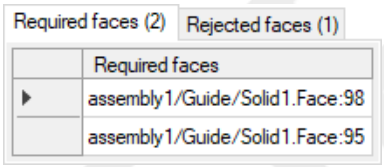
5. In **Virtual Photometric Lab**, click **Measure**.

	
Extended map	Extended map with surface tool

6. If necessary, modify the measure area, shape or position on the maps to update the ray tracing preview in the 3D view.
   The ray tracing of the 3D view is adjusted in real-time to illustrate the light path from the sources to the sensors.
7. Select the rays passing:
   - **Inside all measurement areas**: only the rays passing through all of the measurement areas you defined are displayed in the 3D view.
   - **Outside at least one measurement area**: rays passing outside of at least one measurement area are displayed in the 3D view.

     That means a ray is displayed if he passes outside of one measurement area but passes through all other measurement area. However you don't know outside which measurement areas the ray passes.

8. If you want to filter the rays displayed in the 3D view:



   - Click and select the **Required faces** (the faces that are taken into account to filter rays).

     The rays impacting the required faces and reaching the sensor are displayed in the 3D view.

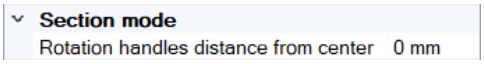
   - Click and select the **Rejected faces** (the faces that you want to exclude from the light trajectory).

     The rays reaching the sensor, excepted those impacting the rejected faces are displayed in the 3D view.

9. If you want to better visualize rays paths:

   a) Click **Section mode** .
   The **Section mode** appears in the 3D view and you can move it thanks to the 3D Manipulator to modify its position and orientation therefore cutting the geometry and see the rays paths.
   b) If the 3D Manipulator is not enough discernible, in the **Section mode** section, with the **Rotation handles distance from center** parameter you can modify the distance of the rotation manipulators to the center of the 3D Manipulator.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



10. In the **Photometric** section, if you want to display only the photometric contribution of the rays displayed in the 3D view, check **Filtered Rays Only**, then click **Update Results** to recalculate the photometric values of all XMP maps from the Light Expert Group.

> **Note:** Selecting a layer in one of the XMP maps applies the layer to all XMP maps. That implies that sequences are not sorted in descending order of energy from the second sensor of the group. For more information on how the sequences are ordered, refer to the section XMP Map of Sensors Included in Sensors' Group.

The optical system (figure 1) displays the rays passing through the area defined in the sensor 1 result (figure 2) and the area defined in the sensor 2 result (figure 3) and the area defined in the sensor 3 result (figure 4).

11.3.6. Light Path Finder Advanced Analysis

The Light Path Finder Advanced Analysis allows you to read a *.lpf file to access specific data contained in it.

11.3.6.1. Understanding the Light Path Finder Advanced Analysis

Thanks to a provided API, you can read a *.lpf file to access the list of traces contained in it, and use the data for post-processing in a third party application.

The *.lpf file contain a number of traces that can be considered as launched rays. Each trace gives access to a number of parameters.

A trace contains:

- a list of impacts, each impact giving access to its coordinates
- a list of IDs of the bodies, each body ID corresponding to one impact of the trace
- a list of IDs of the face, each face ID corresponding to one impact of the trace
- a list of wavelengths, each wavelength corresponding to one impact of the trace
- a list of interaction statuses, each item highlighting the interaction type of the impact
- the last direction of the trace
- the sensor contributions (corresponding to trace contribution on sensors)

Each of the above lists has the same size.

To extract the raw data stored in the *.lpf file, create your own script based on the dedicated APIs.

Speos provides you with a basic example.py python script to help you create your script.

11.3.6.2. List Of Methods

This page describes the methods that should be used to access the data stored in the *.lpf file generated as an output of a simulation with Light Expert activated.

As the *.lpf is a compressed binary file, you must use specific methods to access its content and retrieve its data. Make sure to use the **3.9 version** of IronPython or Python language to write your scripts.

DRAFT

Name	Description	Syntax
CLpfFileReader	Allows you to access the content of a *.lpf file.	IllumineSpeos_pywrap.CLpfFileReader()
Path	Converts the *.lpf file so that it can then be read by CLpfFileReader.	IllumineCore_pywrap.Path(string_LPFfile) • String_LPFfile: path and filename Should end by .lpf
InitLpfFileName	Opens and initializes the LPF file.	Object.InitLpfFileName(illumineCorePath_LPFFile) • Object: LPF File Reader • illumineCorePath_LPFFIle: converted .lpf file by the Path method
GetNbOfXMPs	Returns the number of XMPs associated to the LPF file.	Object.GetNbOfXMPs() • Object: LPF File Reader
GetNbOfTraces	Returns the number of traces stored in the LPF file.	Object.GetNbOfTraces() • Object: LPF File Reader
HasSensorContributions	Allows you to know if the LPF file has the information on the sensor(s) contribution of the traces.	Object.HasSensorContributions() • Object: LPF File Reader
GetSensorNames	Returns the names of the sensors which contributed in the LPF result.	Object.GetSensorNames() • Object: LPF File Reader
Vector_COptRayPath	Creates a table of the traces stored in the LPF file.	IllumineSpeos_pywrap.Vector_COptRayPath()
Resize	Sizes the table according to the number of traces.	Object.Resize(string_numberoftraces) • Object: table to store the traces • string_numberoftraces: name of the variable corresponding to the number of traces retrieved
GetRayPathBundle	Returns all traces and stores them in the table.	Object.GetRayPathBundle(string_TableVariableName.ToSpan()) • Object: LPF File Reader • String_TableVariableName: name of the variable corresponding to the table to store the traces • ToSpan(): method that permits to correctly order and optimize data in the table
vImpacts	Returns all impacts for one trace stored in the LPF.	Object.vImpacts • Object: trace
Size	Returns the number of impacts for one trace.	Object.Size() • Object: table of impacts

DRAFT

Name	Description	Syntax
Get(impactIndex)	Returns the index of the impact in the table of impacts.	Object.Get(impactIndex) • Object: table of impacts • impactIndex: index of the impact in the table
Get(impact_coordinates)	Returns the coordinates (x, y, z) of the impact.	Object.Get(0) Object.Get(1) Object.Get(2) • Object: impact • 0: X coordinate • 1: Y coordinate • 2: Z coordinate
vUniqueFaceIds	Returns the ID of the faces impacted by the trace.	Object.vUniqueFaceIds.Get(impactIndex) • Object: trace • impactIndex: index of the impact in the table
vBodyContextIds	Returns the ID of the bodies impacted by the trace.	Object.vBodyContextIds.Get(impactIndex) • Object: trace • impactIndex: index of the impact in the table
vWavelengths	Returns the wavelength at each impact of the trace.	Object.vWavelengths.Get(impactIndex) • Object: trace • impactIndex: index of the impact in the table
vInteractionStatuses	Returns the interaction type at each impact of the trace with the elements encountered.	Object.vInteractionStatuses.Get(impactIndex) • Object: trace • impactIndex: index of the impact in the table • Possible enumerate value for string_InteractionStatus: º Lambertian reflected º Gaussian reflected º Specular reflected º Absorbed º Specular transmitted º Gaussian transmitted º Lambertian transmitted º Volumic diffused º Emitted º Other type of interaction

DRAFT

Name	Description	Syntax
vSensorContributions	Returns the table of the sensors in which the trace has contributed: • the SensorId • the coordinates on the sensor	Object.vSensorContributions • Object: trace
LastDirection	Returns the last direction of the trace after the last impact.	Object.LastDirection • Object: trace
Get(last_direction)	Returns the last direction coordinates (x, y; z) of the trace after the last impact.	Object.Get(0) Object.Get(1) Object.Get(2) • Object: last direction of the trace • 0: X coordinate • 1: Y coordinate • 2: Z coordinate

11.4. Export as Geometry

This page describes the **Export Rays as Geometry** option that allows you to convert light rays into construction lines.

> **Note:** You can only export rays as geometry from an interactive simulation or from a light expert analysis file when this file is active (when the rays are displayed in the 3D view).
>
> The rays exported correspond to the rays propagated during the simulation. Therefore the length of the rays exported depends on the optical system and the simulation parameters.

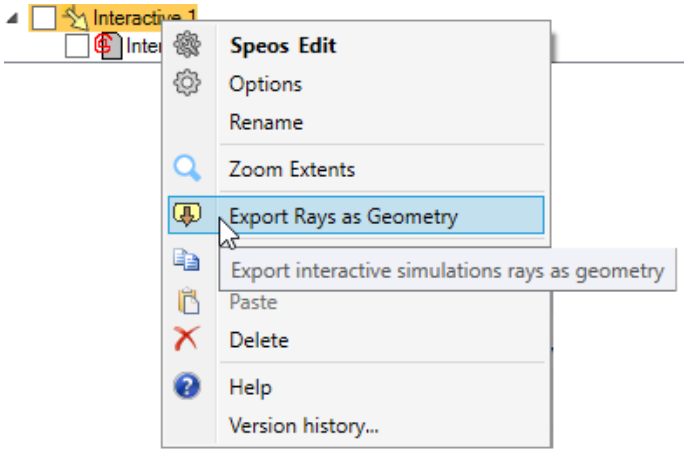
Exporting rays as geometries is useful to verify, assess and modify an optical system to optimize it.

For example, it can help you place the elements of your system ideally, or use a specific construction line as an axis, a base to orient a light guide.

To export ray as geometry, right-click an interactive simulation or *.lpf active file and click **Export Rays as Geometry**.

The rays geometry is exported in the active component.

DRAFT



When the rays are exported as geometries, they appear in the 3D view. They are stored as construction lines in the **Curves** folder of the **Structure** panel.

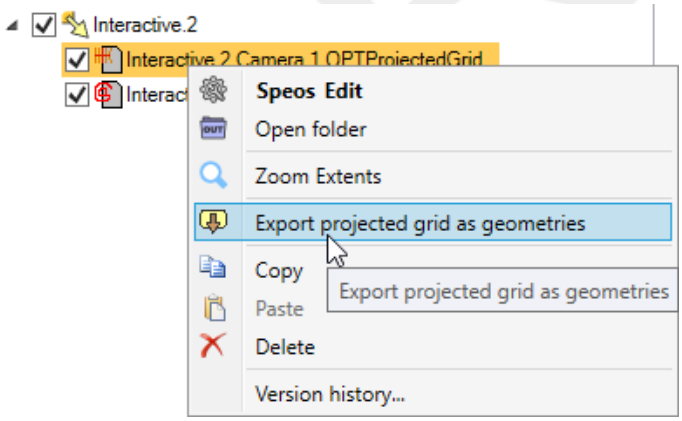
11.5. Export Projected Grid as Geometry

This page describes the **Export Projected Grid as Geometries** option that allows you to convert projected grid files into construction lines.

> **Note:** This option is only available for simulations generating projected grids, that is interactive simulations containing a camera sensor, and Static LiDAR simulations generating the Field of View result as *.OPTProjectedGrid file.

To export projected grid as geometries, right-click the **.OPTProjectedGrid** result file and click **Export projected grid as geometries**.

The projected grid geometry is exported in the active component.



When the grid is exported as geometries, construction lines appear in the 3D view. One line is created per element of the grid. The lines are stored in the **Structure** panel, in a **Curves** sub-folder placed under the "projected grid" geometry.

Related reference

Camera Projected Grid Parameters on page 396

The projected grid represents the sensor pixels on the simulation geometry. This projection is done following the camera's distortion.

11.6. Isolating a Simulation Result

This page shows how to isolate a simulation result in Speos tree to access result files.

> **Note:** An isolated simulation cannot be updated or run anymore.
>
> Isolating a simulation result also works with exported simulations.
>
> Renaming an isolated simulation does not rename the isolated folder.

Isolating a Simulation Result is useful when wanting to save a result in a specific state.

If the result is isolated, further parameter changes and simulations run will not overwrite the result.

To isolate a simulation result, from the Simulation panel, right-click a simulations and click **Isolate**.

Once isolated, the simulation is available from Speos tree and in the isolated results folder.

DRAFT

12: Speos Labs Documentation Links

The following page presents you the different tools available to create or edit Optical Properties, analyze simulation results and perform virtual numerical certifications to ensure you meet the industry standards.

Labs

- **Virtual Photometric Lab**

  Analyze spectral luminance, irradiance, intensity results. Measure photometric values, perform colorimetric analysis, load measurement templates, control the contribution of light source, analyze different result layers based on the sensor type.

- **Virtual Human Vision Lab**

  Display the visual appearance of modeled scenes restoring the contrasts perceived by the human eye.

- **Virtual Reality Lab**

  Read natively environment maps (Speos360), OpenEXR/HDR, XMP spectral picture, analyze multiple view points based on the type of sensor used (observer immersive sensor), deploy results on multiple screens, create complex screen scenarios such as a CAVE.

- **Virtual 3D Photometric Lab**

  Analyze photometric or radiometric information such as reflection, transmission, absorption or irradiance directly on your 3D model.

- **3D Energy Density Lab**

  Visualize the absorption per volume unit.

- **Virtual Lighting Animation Tool**

  Generate videos by defining timelines for each layer of a loaded XMP result and by animating power variations of the different sources.

- **Photometric Calc**

  Make operations on *.xmp maps or *.ldt files to create new ones based on the operation type (addition, subtraction, merge, union, etc.).

Optical Properties Editors

- **Simple Scattering Surface**

  Simulate the surface's behavior with the classical model (Specular, Lambertian, Gaussian) for both transmitted and reflected rays without wavelength information.

- **Advanced Scattering Surface**

  Simulate the surface's behavior for both transmitted and reflected rays for Specular, Lambertian, and Gaussian models, and depending on incidence angle, polarization, and wavelength information.

- **User Material Editor**

  Build complex volume materials, taking into account the diffusion, index and absorption variation, gradient index and birefringent properties, and store them in a library.

DRAFT

- **Spectrum Editor**

  Draw the spectrum curve using linear interpolation between defined particular points
- **Polarizer Surface**

  Model a perfect polarizer.
- **Coated Surface**

  Define the quality of a surface to which a coating is applied.
- **Retro Reflecting Surface**

  Model a surface for which rays may be reflected towards the direction they come from.
- **DOE and Thin Lens Surface**

  Simulate a diffractive surface. Model a thin lens that corresponds to a theoretical lens.
- **Grating Surface**

  Model the diffraction of light.
- **Fluorescent Surface**

  Take into account a spectrum for absorption and use another spectrum for emission.
- **Rough Mirror Surface**

  Define the surface quality of a colored diffuse surface.

DRAFT

13: Optical Part Design

Optical Part Design provides geometrical modeling capabilities dedicated to optical and lighting systems.

> **Important:** All features contained in this section require **Speos Optical Part Design** add-on and Premium or Enterprise license.

13.1. Optical Part Design Overview

Optical Part Design allows you to create optical components and lighting systems such as optical lenses, total internal reflection lenses, light guides etc.



Head Lighting



Rear Signal Lighting (reflex reflectors)



Dashboard - interior lighting (light guide)

Speos Optical Part Design provides modeling capabilities dedicated to optical and lighting systems design mainly for automotive industry.

A wide variety of optical parts and components such as lenses, surfaces or reflectors are available in Speos to cover different needs and configurations.

Once modeled, the optical components can be integrated in an optical system and be tested out through simulation.

Optical Components

- The Parabolic Surface is a collimating surface, that is a surface that allows a perfectly specular, mirror-like reflection of rays.
- The TIR (Total Internal Reflection) lens is a collimating optical component, highly efficient to capture and redirect light emitted from a light source.
- The Light Guide is a thin pipe made of transparent material. Light Guides are highly efficient in transmitting light and have various possible applications (interior lighting, accent lighting etc.)
- The Optical Lens allows you to create pillow, prismatic, pyramidal or reflex reflectors lenses. Optical lenses are key components of lighting systems are often used to design rear signal lenses.
- The Optical Surface allows you to generate rectangular, circular or faceted reflectors.
- The Projection Lens allows you to create optical lenses used for automotive projector modules.
- The Poly Ellipsoidal Reflector is a reflector that is mainly used in automotive projector modules to produce spread/driving beams.
- The Freeform Lens allows you to create a collimating lens from any freeform surface.
- The Micro Optical Stripes helps you create a feasible light guide by defining a Tool Bit Shape used for processing the Light Guide mold.

DRAFT

In Speos

You can manage all the characteristics of the optical component: its source, focal, shape, size and distribution.

> **Warning:** Opening an Optical Part Design project in two different Speos versions may present different results.

13.2. Migration Warnings and Differences between Versions

The following page presents you with warnings on Optical Part Design features migrations from a version to another and on the differences you can find.

> **Important:** The Optical Part Design (OPD) algorithm is continuously evolving along the software releases. That means, from one release to another for a same OPD feature, the computation result might differ. Therefore, We highly recommend you to keep the previous version of Speos and a copy of your OPD project to be able to maintain the construction algorithm and behavior if the result does not fit your expectations.

Computing an Optical Part Design Feature after Facing an Error

When you face an error during the compute of a feature, this prevents the construction of the feature. Then if you get back to the previous state (before the error) and recompute the feature, a new geometry will be built in the Structure tree.

The links between the Optical Part Design feature and Speos features (material, simulation, etc.) are then not related to the new geometry built:

- In version 2022 R2, links are broken.
- In version 2023 R1, links are kept to the previous isolated geometry result.

In version 2023 R2, the issue is corrected: recomputing the feature updates the original geometry and links are kept.

Projection Lens Migration from 2022 R1 to 2022 R2

When migrating a Projection Lens from Speos 2022 R1 to Speos 2022 R2, aspheric coefficients are shifted.

Aspheric coefficients go from 2 to 31 instead of 2 to 30, and so do the values. The issue goes also for the Aspheric coefficient of a Zernike face.

In 2022 R1

- i = 1 value = 0
- i = 2 value = 0.02
- i = 3 value = 0.001
- ...
- i = 30 value = 0,001

In 2022 R2 after migration

- i = 2 value = 0 (this corresponds to the first index 1)
- i = 3 value = 0.02
- i = 4 value = 0.001
- ...

DRAFT

- i = 31 value = 0,001 (this corresponds to the last index 30)

Projection Lens Migration from 2022 R2 to 2023 R1

When migrating a Projection Lens from Speos 2022 R2 to Speos 2023 R1, if the Projection Lens from 2022 R2 is already a migration from a previous version, then aspheric coefficients are shifted.

- In 2022 R2, Aspheric coefficients go from 2 to 31 instead of 2 to 30, and so do the values.
- In 2023 R1, Aspheric coefficients go from 2 to 30.

The issue goes also for the Aspheric coefficient of a Zernike face.

In 2022 R2

- i = 2 value = 0 (this corresponds to the first index 1)
- i = 3 value = 0.02
- i = 4 value = 0.001
- ...
- i = 31 value = 0,001 (this corresponds to the last index 30)

In 2023 R1 after migration

- i = 2 value = 0.02
- i = 3 value = 0.001
- ...
- i = 30 value = 0,001

> **Note:** The index 1 is no longer present in the interface as it is always 0. Only edible coefficient are present.

13.3. Parabolic Surface

The Parabolic Surface allows you to create a collimating (perfectly specular) surface.

13.3.1. Parabolic Surface Overview

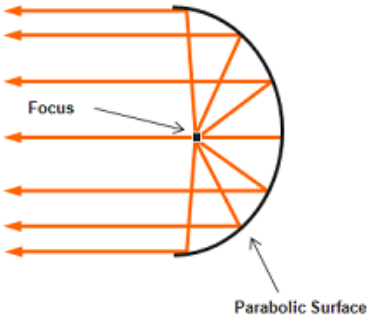
This page provides an overview of the parabolic surface and its different applications and uses.

The Parabolic Surface allows you to create a collimating surface, that is a surface that allows a perfectly specular, mirror-like reflection of rays.



In Speos, a focus point (representing the light source) sends rays that are reflected on the specular surface and collimated in the surface's optical axis direction.

DRAFT



Section view of the collimation carried out by a parabolic surface



3D view of a parabolic surface

Related tasks

Creating a Parabolic Surface on page 501
This page shows how to create a Parabolic Surface that can, later on, be used as a support to create optical lenses.

Related reference

Understanding Parabolic Surface Parameters on page 500
This page describes parameters to set when creating a Parabolic Surface.

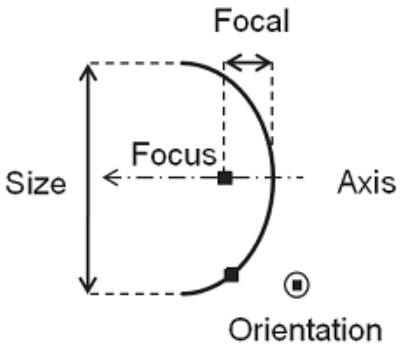
13.3.2. Understanding Parabolic Surface Parameters

This page describes parameters to set when creating a Parabolic Surface.

The Parabolic Surface is basically driven by three parameters:

- An axis system comprising an origin, optical and orientation axis.
- A focal
- A size

Figure 66. Top view of a Parabolic Surface and associated settings



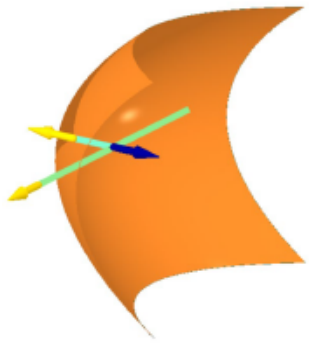
Axis System

- The **origin point** represents the focus point (the point giving the position of the source).

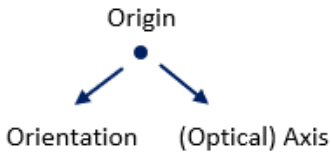
- The **Axis** refers to the optical axis of the surface. This axis is used to define the direction of the surface revolution axis.

> **Note:** The focus does not necessarily belongs to this Axis. In this case, the revolution axis of the surface is defined by the direction of **Axis** and by the Focus.

- The **Orientation** fixes the orientation of the surface around the optical **Axis**. The orientation is usually but not necessarily defined on a plane normal to the Axis. When the orientation is defined on a plane that is not normal to the Axis, orientation is projected onto the defined plane,



3D view of Parabolic Surface



Axis System

Focal

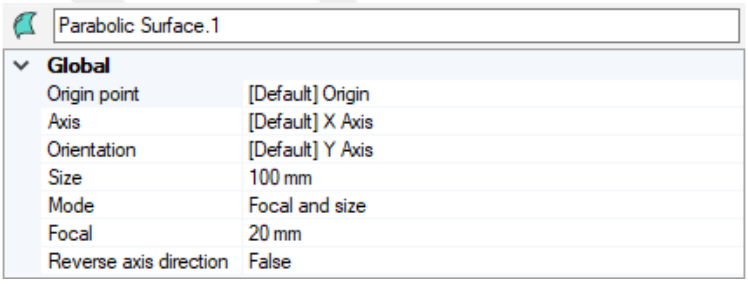
The focal length represents the distance between the top and the focus of the parabolic surface.

13.3.3. Creating a Parabolic Surface

This page shows how to create a Parabolic Surface that can, later on, be used as a support to create optical lenses.

To create a Parabolic Surface:

1. From the **Design** tab, click **Parabolic Surface** .



2. Set the axis system of the surface:
   - to select the surface's **Origin point** (the source point).

- 
  and select a line giving the direction of the optical axis.
- 
  and select a line fixing the orientation of the surface around **Axis**.
- or click
  
  and select a coordinate system to autofill the Axis System.

> **Note: Orientation** is not necessarily defined on a plane normal to **Axis**. In this case, **Orientation** is projected onto such plane.

3. In **Size**, adjust the length of the surface's edges.
4. Adjust the focal length of the surface, that is the distance between the top and the focus.
5. Press the **E** key to leave the feature edition mode.

The Parabolic Surface is created and appears both in the tree and in the 3D view.

Related reference

Understanding Parabolic Surface Parameters on page 500
This page describes parameters to set when creating a Parabolic Surface.

Related information

Parabolic Surface Overview on page 499
This page provides an overview of the parabolic surface and its different applications and uses.

13.4. Optical Surface

The Optical Surface feature allows you to generate rectangular, circular or faceted reflectors.

13.4.1. Optical Surface Overview

The Optical Surface feature allows you to generate faceted reflectors.

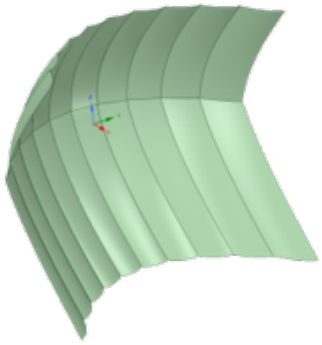
Optical surfaces are used to create reflectors that depend on specific support and for which elements are shaped to light specific areas.



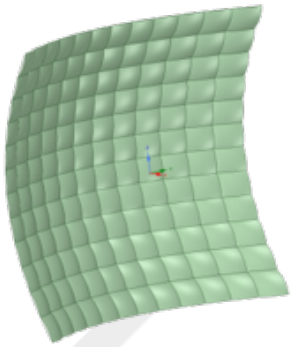


DRAFT

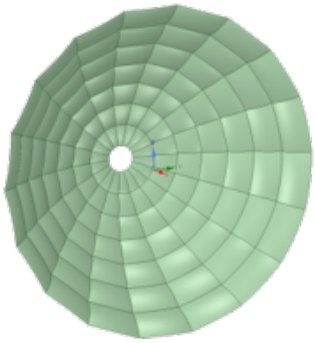
In Speos, optical facets distribution, shape and size can be fully customized by the user.



3D view of a segmented reflector



3D view of an optical surface with rectangular grid



3D view of an optical surface with circular grid

Related tasks

Creating an Optical Surface on page 503
This page shows how to create an optical surface from a parabolic or freeform support.

Related information

Optical Surface Parameters on page 511
This section provides more information about the parameters to set when creating an optical surface.

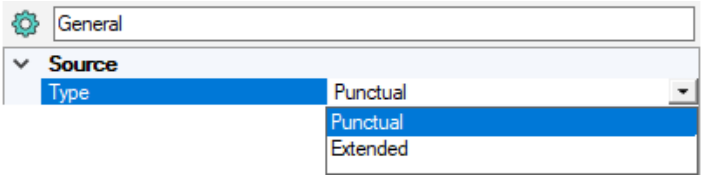
13.4.2. Creating an Optical Surface

This page shows how to create an optical surface from a parabolic or freeform support.

To create an Optical Surface:

1. From the **Design** tab, click **Optical Surface** and select a grid type/shape (Rectangular, Circular, Stripes, Freestyle Rectangular or Freestyle Circular).
2. In Source, from the **Type** drop-down list, select:

DRAFT



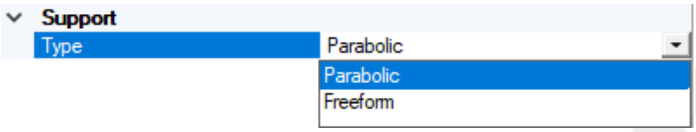
- **Punctual** to position the source by selecting a point in the 3D view.
- **Extended** to select an emitting surface like a filament, cylinder, LED, chip etc.

3. If you select **Extended**, you can define the **Flux** of the source.

   The Flux will be used to run the **Photometry** tool simulation that generates a photometric preview displayed in the feature viewer, a HTML report and a XMP file of the selected element(s) of the feature.

   For more information on the Photometry tool, refer to Understanding Display Properties.

4. In Support, from the **Type** drop-down list select which support to use to build the optical surface:



> **Important:** When computing the Optical Lens, the support is meshed. The bigger the support, the longer the computation of the feature. We recommend you to define a support size as close as possible to the feature size.

- Select **Parabolic** to create the parabolic support by defining its origin, axis and orientation or click  and select a coordinate system to autofill the Axis System.

  > **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

- Select **Freeform** to build the optical surface on an existing surface. Then, in the 3D view click  to select the freeform surface.

  > **Note:** A face or a multifaces body can be selected as surface.

5. In **Target**, if you want to verify that your current design passes regulation standards, select a **XML Template** corresponding to the regulation standard.

   The XML template will be used by the **Photometry** tool simulation to generates the photometric preview displayed in the feature viewer, the HTML report and the XMP file of the selected element(s) of the feature.

   > **Note:** You can find existing XML templates of regulation standards in the Ansys Optical Library.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

For more information on the Photometry tool, refer to Understanding Display Properties.

Target	
Type	Intensity
XML Template	[No file]
Axis	[Default] (Style)
Reverse direction	False
Orientation	[Default] (Style)
Reverse direction	False
Intensity result viewing direction	From source looking at sensor

6. Define the result viewing direction and the position of the observer point:
   a) If you want to define specific axes for the sensor, in the 3D view click  to select a projection axis and  to select an orientation axis or click  and select a coordinate system to autofill the Axis System.
   b) From the **Intensity result viewing direction** drop-down list:
   - Select **From source looking at sensor** to position the observer point from where light is emitted.
   - Select **From sensor looking at source** to position the observer in the opposite of light direction.
7. In the Style tab, set the axis system of the grid define the distribution, pattern and size of the elements to be created on the support.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

8. In Manufacturing, activate the sewing and drafting of the elements if you want mechanical constraints to be taken into account:

Manufacturing	
Sewing	True
Drafting	Drafting
Draft length	0.3 mm
Draft all elements	True

   - Activate the **Sewing** if you want any gaps between elements to be automatically filled.
   - Activate the drafting of the elements if you want mechanical constraints coming from unmolding to be taken into account.

     If activated, define a **Draft length** in mm. The draft length defines the sewing surface created between two adjacent faces.

9. From the **Design** tab, click **Compute**  to generate the surface.

The Optical Surface is created and built in the 3D view. You can edit the surface at any time.

Related tasks

Managing Groups and Elements on page 506
This page shows how to create and manage groups. When designing an optical surface, you can create different groups of elements to apply specific parameters to these groups.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Related information

Optical Surface Parameters on page 511
This section provides more information about the parameters to set when creating an optical surface.

13.4.3. Managing Groups and Elements

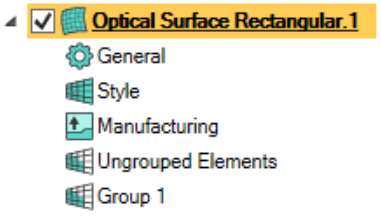
This page shows how to create and manage groups. When designing an optical surface, you can create different groups of elements to apply specific parameters to these groups.

> **Note:** When no group is created, all the elements (facets) are stored in **Ungrouped Elements** section.

To create and manage groups:

An Optical surface must already be defined.

1. From the **Design** panel, reach the sub-feature level.



2. From the **Groups** panel, click **Add** + to add as many groups as needed.

Groups (3)

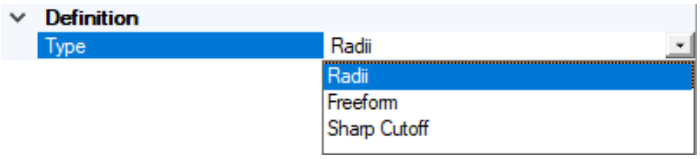
	Name	Type	Orientation	Focal	X center	Y center	X radius	Y radius	+
	Ungrouped Elements	Radii	Convex	20 mm	0 °	0 °	15 mm	20 mm	
▶	Group 1	Radii	Convex	20 mm	0 °	0 °	15 mm	20 mm	
	Group 2	Radii	Convex	20 mm	0 °	0 °	15 mm	20 mm	

> **Tip:** For a quicker way, right-click the feature and select **Add new group**.

The groups appear in the design panel.

3. Select each group to define its parameters. From the **Type** drop-down list, select the type of lens you want to apply to the current group:

> **Note:** Available beam types depend on the optical surface selected.
>
> For more information on lens types, see Beams .



- Select Radii to create spherical facets that ideally diffuse light.
- Select Freeform to create facets that are shaped by specifying the area where the light should be sent in.
- Select Sharp Cutoff if you want to create an optical surface beam that directs light underneath a defined cut-off line.
- Select Flutesif you want to create facets by adjusting the flutes curvature.

4. In the 3D view, click  and select the faces to include in the group:
   - In case of a Rectangular or Freestyle Rectangular part, you can directly **select a row**

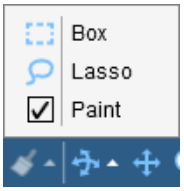
     or **select a column**

     
   - In case of a Circular or a Freestyle Circular part, you can directly **select a radius**

     or **select a circle**

     
   - For a free selection of faces, you can choose a SpaceClaim selection mode in the bottom right corner of the session and use **Box**, **Lasso** or **Paint**.

   - You can add Named Selections composed of faces to Optical Part Design groups.

   Note: You can only add a same type of element in a group. Example: a group composed of faces only, or a group composed of Named Selections only.

5. Once, the group is defined, click  to validate the selection.
6. Adjust the Support settings according to the selected beam type.

The Group is created and the properties are correctly applied to it. You can now access the feature viewer to obtain pieces of information regarding facets.

Related tasks

Creating an Optical Surface on page 503
This page shows how to create an optical surface from a parabolic or freeform support.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Related information

Beams on page 521
This section gathers all beam types available when designing an optical surface.

Support on page 535
This page describes the parameters to set when working with a **Parabolic** or **Freeform** support.

13.4.4. Managing Elements from an Excel File

The following procedure shows how to create and use and Excel file that allows you to drive the parameters of each facet of the Optical Surface individually.

To create and use an Excel file:

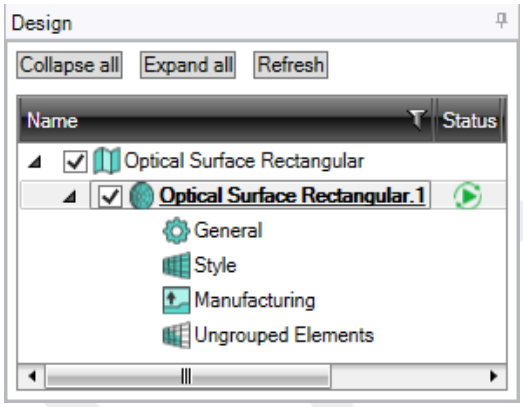
An Optical Surface Rectangular or Circular must already be defined.

No group must have been created. Only the Ungrouped Elements must be present.

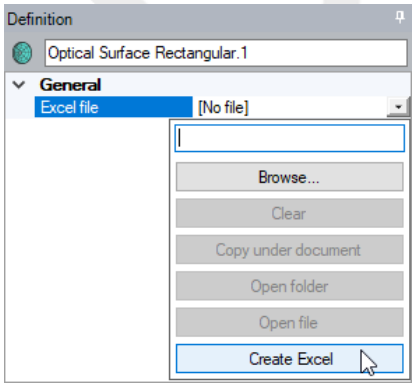
Make sure the Optical Surface is compatible with the Excel file definition.

1. From the **Design** panel, open the Optical Surface.
2. If you want, modify the parameters related to the facets in the **Style** and **Ungrouped Elements** definitions.

   The **Style** and **Ungrouped Elements** parameters corresponds to those who will be created and applied when you create the Excel file.
3. Open the main node of the Optical Surface.



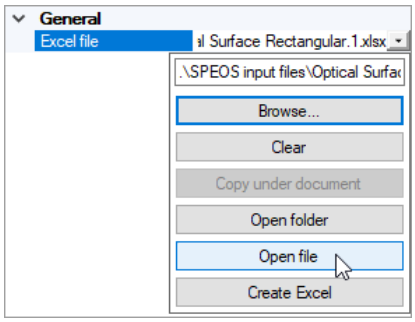
4. In the main node definition, from the **Excel file** drop-down list, click **Create Excel**.



An Excel file is created based on the input parameters set in the Style and Ungrouped Elements definition.

The Excel file opens automatically upon creation.

5. If you had already created the Excel File (automatic opening of the file already done), from the **Excel file** drop-down list, click **Open file**.



> **Important:** For Speos to know that you modified the Excel file, you must open the file from here. Otherwise, if you open the file outside of the interface, Speos cannot detect if you modified it.

6. Modify the Excel file as you want.

   In the Excel file, each sheet correspond to a parameter and each cell in a sheet corresponds to one specific facet of the Optical Surface.

> **Tip:** To find the match between a cell in the Excel file and its facet in the 3D view, you can hover over the feature. This will give you the cell coordinates.
>
> 
>

7. Save the Excel file.

8. In Speos, click **Compute** to generate the Optical Surface and take into account the changes.

The Optical Surface is generated and built in the 3D view.

13.4.5. Understanding the Excel File

The following page helps you understand the use of an Excel file in the definition of an Optical Surface.

Description

The Excel file is an input file that allows you to drive every parameter that you can drive in a group for every facet of the Optical Surface individually. Basically the Excel file represents every facet of the feature as if one face = one group. It saves you from creating lots of groups in the Speos interface and allows you to quickly modify each parameter for each facet so that you can create a smooth evolution of the parameters value along the feature.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Note: The excel file replaces the management of the facets by group. Thus you cannot create groups when the Excel definition is activated.

Excel File Compatibility with Optical Surface Parameters

The Excel file is compatible with the following types of Optical Surface. All others types not mentioned in the table are not compatible.

Optical Surface	Beam	Parameters displayed in Excel
Rectangular	Radii	• X radius • Y radius • Focal (used in case of a Parabolic support) • X Center (used in case of a Parabolic support) • Y Center (used in case of a Parabolic support) • Shift (used in case of a Freeform support)
Rectangular	Freeform	• X start • X end • Y start • Y end • X spread • Y spread • Focal (used in case of a Parabolic support) • X Center (used in case of a Parabolic support) • Y Center (used in case of a Parabolic support) • Shift (used in case of a Freeform support)
Circular	Radii	• Radial radius • Start radius • End radius • Focal (used in case of a Parabolic support) • X Center (used in case of a Parabolic support) • Y Center (used in case of a Parabolic support) • Shift (used in case of a Freeform support) **Important:** The behavior of radii pillows in circular is different between a definition by group and a definition using an Excel file. When using the Excel File definition, the radii pillows are built according to the values written in the Excel. They are not built according to a ratio of the position between the Start radius and End radius values.

DRAFT

Excel File Format

Each sheet corresponds to a parameter driving the facets.

Each cell corresponds to a facet in the 3D view.

Each cell can be defined use a formula or a macro. Speos will only interpret the result of the formula or the macro.

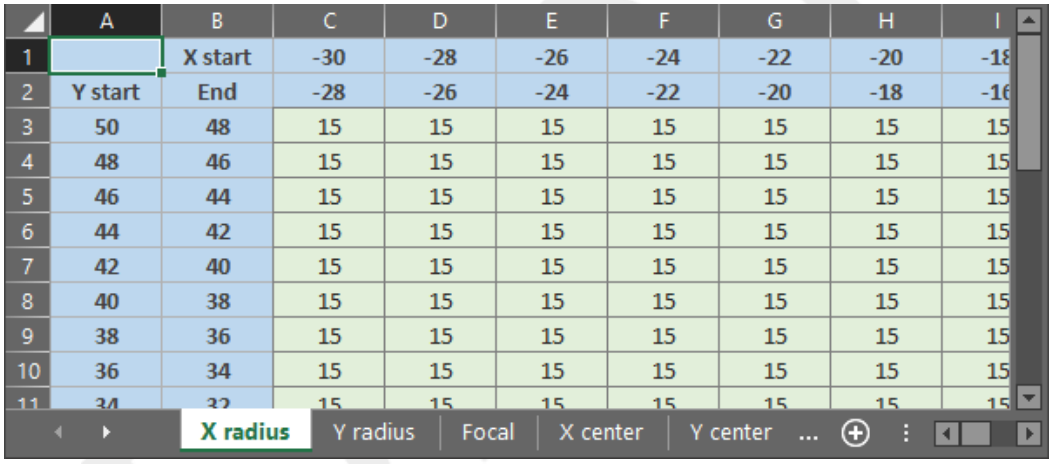
If a cell is empty in the Excel file, Speos will use the value from the Speos definition.

If a sheet does not exist for a parameter or a grid value, an message is raised suggesting you to create a new valid excel file. If you use the Excel file with the missing sheet, then the parameter values from the Ungrouped Elements will be used to replace the missing sheet.

Warning: If you create a new Excel file from the definition in Speos, you will not have the modifications you applied directly in the previous used Excel file. You will have to re-type them manually.

Note: We recommend you to define an Excel file of a maximum size of 100x100. A bigger file would affect the performance.

Figure 67. Example of a the sheet for an Optical Surface



13.4.6. Optical Surface Parameters

This section provides more information about the parameters to set when creating an optical surface.

13.4.6.1. Source Types

This page describes the different types of sources available when designing an optical surface.

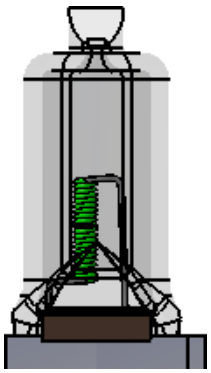
The source corresponds to the source that is used to illuminate the optical surface. In the interface, 2 types of sources are available.

- A **Punctual** source is a simplified source that emits light from a single point and position.
- An **Extended** source is used to emit light from a surface. To use this option, a source geometry (like a light bulb, LED, surface) must already exist or be created.

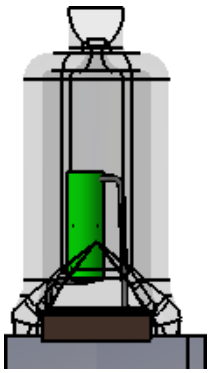
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

With an extended source, the source images are available in the feature viewer.

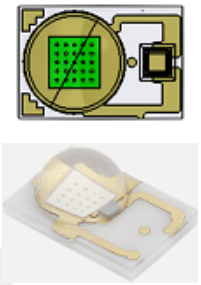
Example of extended sources:



Filament - H1 lamp



Cylinder - H1 lamp



Chip - Luxeon Rebel ES

Related tasks

Creating an Optical Surface on page 503
This page shows how to create an optical surface from a parabolic or freeform support.

13.4.6.2. Support

The support drives the construction of the surface's elements. Elements can be built on a Parabolic or Freeform surface.

Parabolic

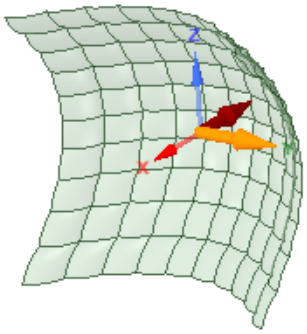
With the Parabolic type, the four corners of each element belong to the parabolic surface.

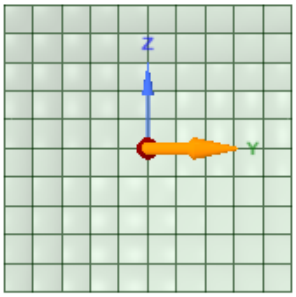
To generate the support, an origin, axis and orientation need to be selected.

- The **Origin** point determines the absolute position of the surface. By default, the source position is used.
- The **Axis** determines the optical axis of the surface.
- The **Orientation** is defined by selecting a line fixing the orientation of the surface around the **Axis**.

> **Note: Orientation** might not be defined on a plane normal to the **Axis**. In this case, orientation is automatically projected onto such plane.

DRAFT



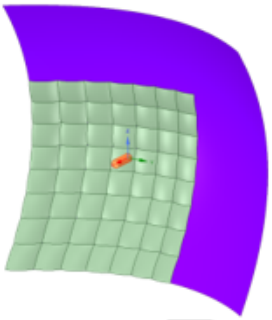


Axis depicted in red, Orientation in orange.

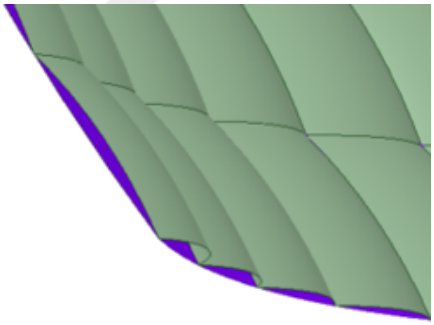
Freeform

With the **Freeform** type, elements are built on a user-defined freeform surface.

A surface must be selected as support.



Optical surface created on a freeform support (depicted in purple).



The 4 edges of the elements belong the freeform support.

Related tasks

Creating an Optical Surface on page 503
This page shows how to create an optical surface from a parabolic or freeform support.

13.4.6.3. Target

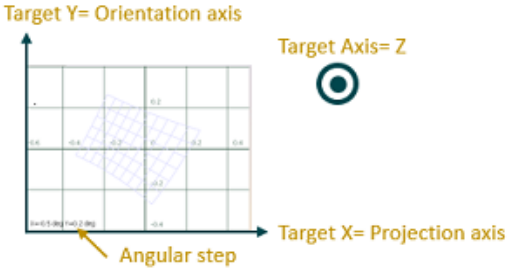
The target allows you to define how the results are shown in the feature viewer.

Intensity Target

The target is more or less acting like a sensor. You can choose from this section how the results are shown and what pieces of information are going to be displayed in the feature viewer.

Beam patterns and source images are displayed on an angular grid in the feature viewer. It is often used when the area to light is defined angularly.

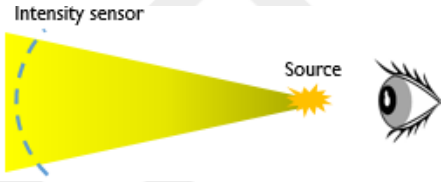
DRAFT



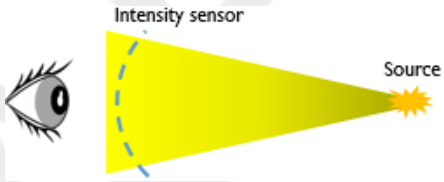
Intensity result viewing direction

This parameter adjusts the viewing direction of the observer.

From source looking at sensor(Intensity/Illuminance target): The viewing direction of the observer is the same as the light direction emitted.



From sensor (Intensity/Illuminance target) **looking at source:** The viewing direction of the observer is in the opposite of the light direction.



13.4.6.4. Style

This section describes all grid types available to design an optical surface.

> **Note:** In some cases, when modifying the grid of an Optical Surface, group definitions must be updated.

13.4.6.4.1. Rectangular

The grid determines the size and distribution of the elements over the support.

Axis System

An axis system is required to define the elements' orientation and projection on the support.

This axis system is, by default, inherited from the support definition.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

DRAFT

Rectangular Grid

Elements are distributed according to a rectangular pattern. The following parameters can be set to customize this distribution:

- **X start**: Start value for the feature size along **X Grid** (mm).
- **X end**: End value for the feature size along **X Grid** (mm).
- **Y start**: Start value for the feature size along **Y Grid** (mm).
- **Y end**: End value for the feature size along **Y Grid** (mm).
- **X Angle**: Angle made by X axis and the horizontal edge of the elements (deg).
- **Y Angle**: Angle made by Y axis and the vertical edge of the elements (deg).
- **X count**: Number of elements along X axis.
- **Y count**: Number of elements along Y axis.
- **X size**: Size of the elements along X axis (mm).
- **Y size**: Size of the elements along Y axis (mm).

13.4.6.4.2. Circular

The grid determines the size and distribution of the elements over the support.

Axis System

An axis system is required to define the elements' orientation and projection on the support.

This axis system is, by default, inherited from the support definition.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

Circular Grid

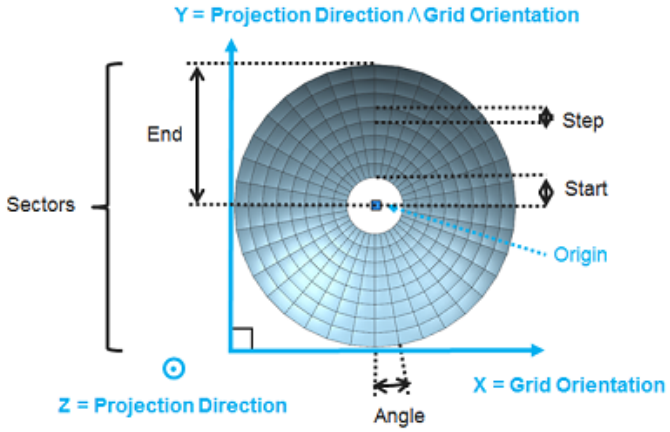
> **Note:** Circular grid distribution is not compatible with **Flute** or **Freeform** beam type.

Elements are distributed according to a circular pattern. The following parameters can be set to customize this distribution:

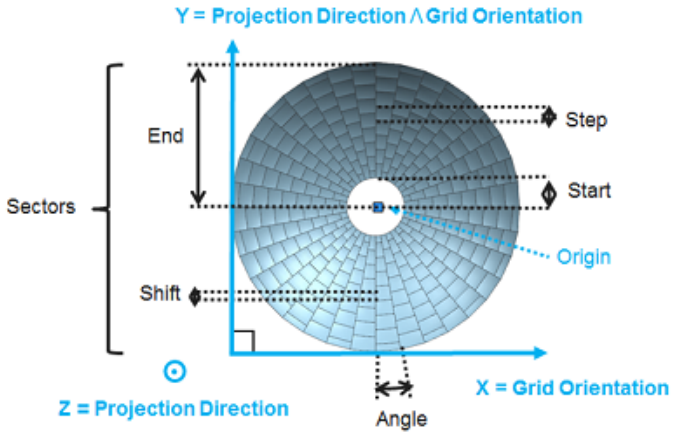
Shift

- **None**

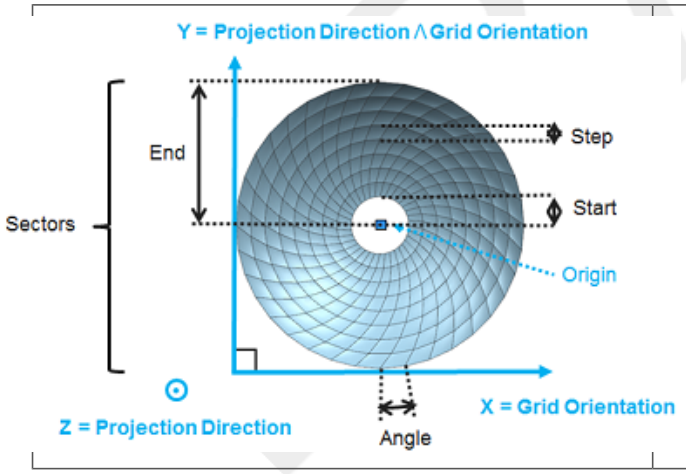
DRAFT

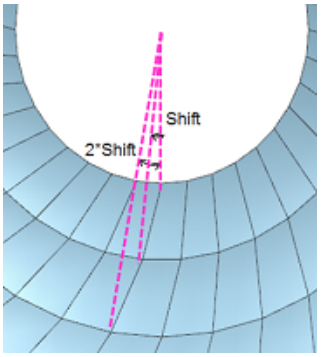


- **Radial**



- **Circular**





Grid Definition

- **Start**: Interior radius of the feature (mm).
- **End**: Exterior radius of the feature (mm).
- **Step**: Radial length of the elements (mm).
- **Sectors**: Number of angular subdivisions of the feature.
- **Angle**: Angle made by the sectors (deg).

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- **Shift**: length (radial shift type) or angle (circular shift type) driving the elements distribution (in mm or deg).

13.4.6.4.3. Stripes

The grid determines the size and distribution of the elements over the support.

> **Tip: Stripes** grid is only compatible with the **Flute** beam type.

Style curves define the stripes on the optical surface.

The shape of the stripes, their number and repartition over the support depend on the style curves selected for definition.

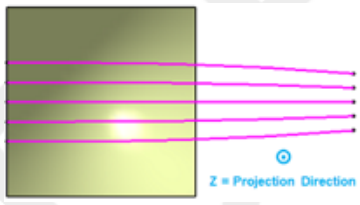
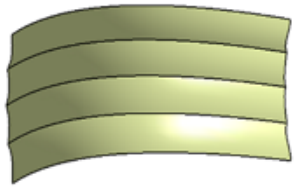
Axis System

The grid axis system is used to define the way that the style curves are projected onto the support.

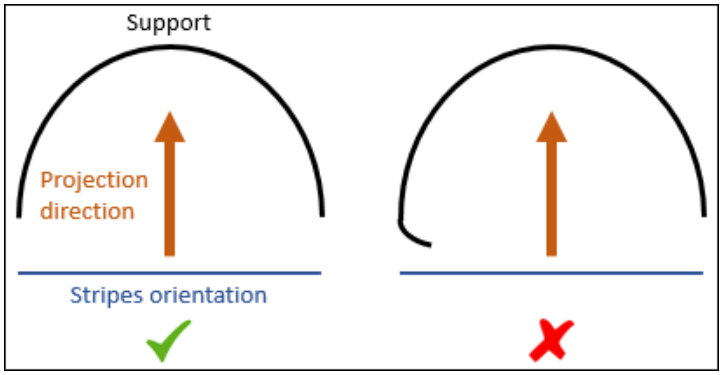
The axis system origin is optional. It is, by default, inherited from the support.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

> **Note:** This axis system has an impact on some parameters of the flutes beams:
>
> - Start angle/End angle.
> - Flutes' concavity/convexity.

	
Support and Style Curves viewed from the plane normal to Projection Direction.	Result of the feature when stripes are generated on the support according to style curves definition.

To ensure a correct construction when using a curved support, make sure the support does not close in on itself (see figure below). Otherwise, the stripes will not be projected correctly.



Definition

Styles curves are selected to demarcate the stripes.

These curves can be lines, splines or sketches. The stripes are oriented along X Grid, which is given by the tangent to the set of curves.

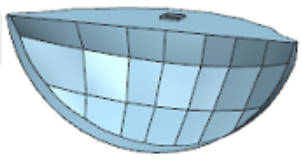
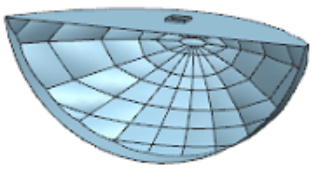
> **Note:** The stripes curves are not necessarily continuous in tangency.
>
> One incomplete style curve is enough to provoke a null result for the whole feature. To avoid construction issues, always use support surfaces larger than style curves.

13.4.6.4.4. Freestyle

The grid allows you to determine the lens' distribution onto the support.

> **Note:** **Freestyle** grid is not compatible with the **Flute** beam type, and the **Freestyle circular** grid is not compatible with the **Sharp Cutoff** beam type.

Two sets of curves are used to delimit the elements of the lens. These curves can give either a rectangular-based or a circular-based pattern.

	
Reflector with rectangular-based freestyle grid	Reflector with circular-based freestyle grid

Axis System

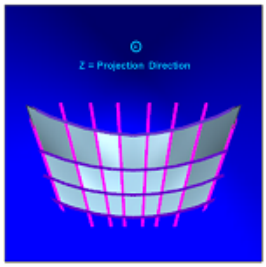
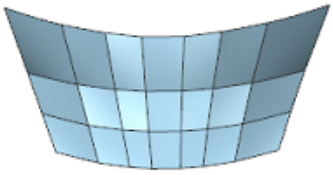
The grid axis system is used to define the way that the sets of curves are projected onto the support.

The axis system origin is optional. It is, by default, inherited from the source.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Note: If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

	
Support and the two sets of curves viewed from the plane normal to Projection Direction	Result of the feature generated with the support and the two sets of curves of the picture on the left

Definition

Two sets of curves (**X curves** and **Y curves**) are used to demarcate the elements.

These curves can be lines, splines or sketches.

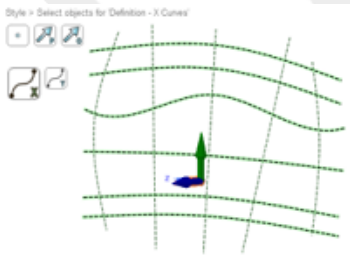
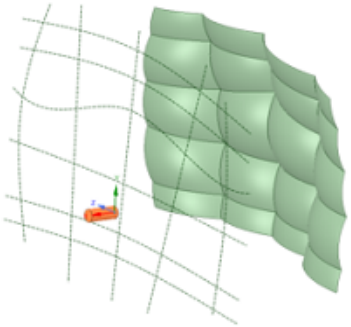
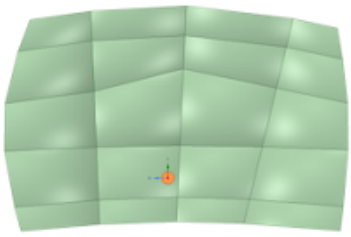
Note: The curves are not necessarily continuous in tangency.

Each set of curves has to follow some rules for the feature to be built properly :

- Each curve of a set cannot intersect any other curve from this set.
- Each curve of a set cannot intersect several times any other curve.
- All the curves of a set has to intersect all the curves of the other set.
- All the curves of a set has to be closed if one curve of this set is closed.

X curves are dedicated to vertical curves and are defined along Y grid axis.

Y curves are dedicated to horizontal curves and are defined along X grid axis.

		
Y curves are the highlighted curves and drive X grid. X curves drive Y grid	Curves projected onto support	Final result

13.4.6.5. Manufacturing

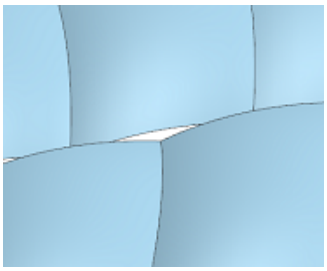
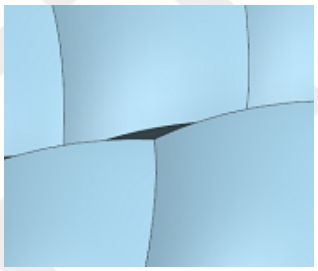
The **Manufacturing** section allows you to consider and anticipate mechanical constraints coming from the surface's manufacture.

> **Note:** The sewing and drafting options are not available for the **Stripes** grid.

Sewing

Sometimes a gap may appear between the elements.

Activating the **Sewing** option automatically fills any gaps that might appear during construction.

	
Optical surface without sewing - Gaps remain between the elements	Optical surface with sewing - Gaps between the elements are filled

Drafting

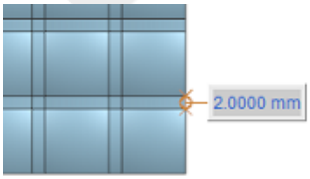
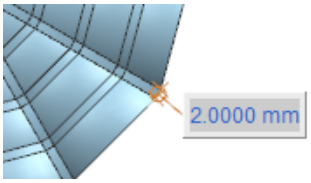
Lenses generated without drafting induce a manufacturing issue when they are removed from their mold.

Activating the **Drafting** allows you to take into account mechanical constraints and ensure an accurate manufacture.

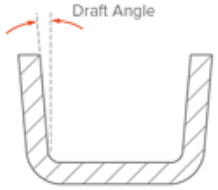
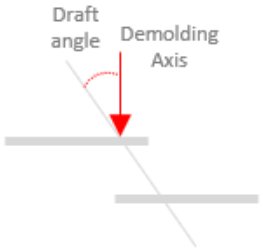
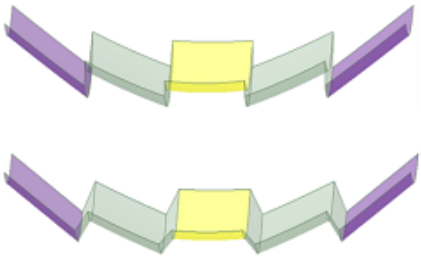
> **Note:** Drafting reduces the facets' size. This operation is automatically taken into account by the algorithm generating the elements. As a consequence, the photometry is kept during the drafting operation no matter how high the drafting value is.

Drafting can be done by length or by angle:

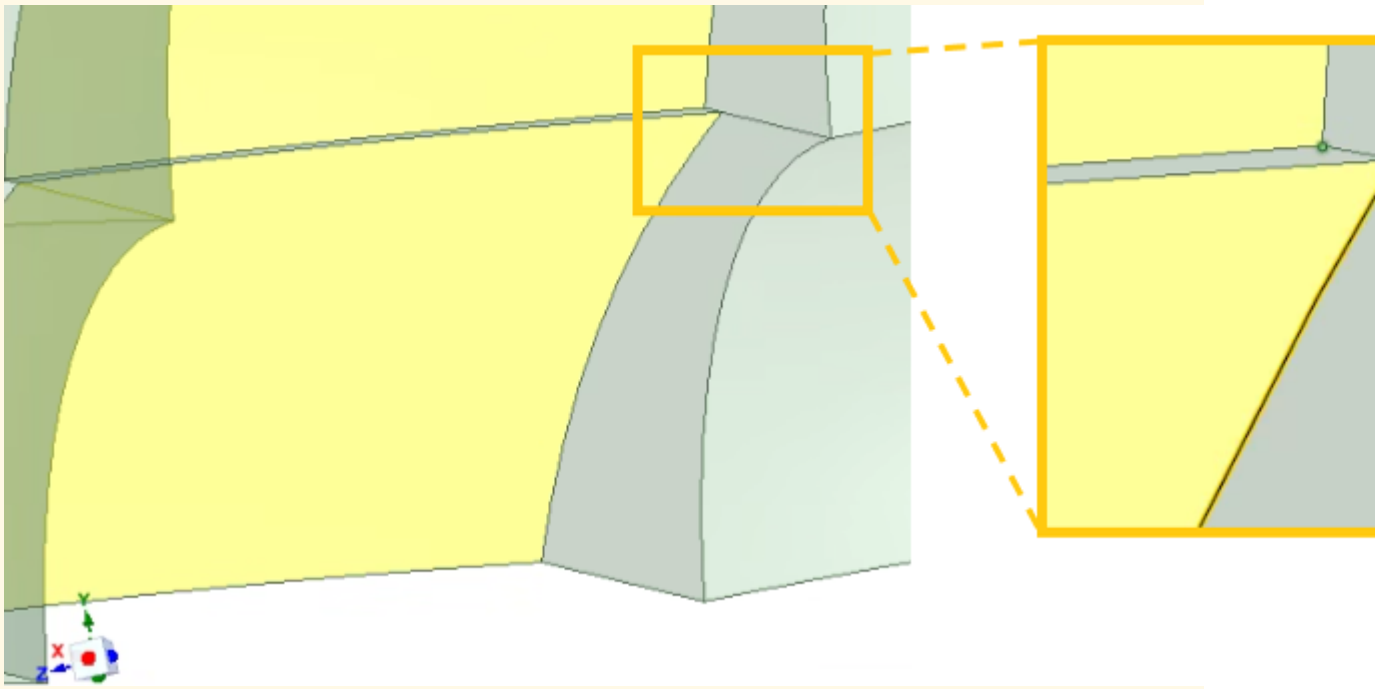
- When selecting **Draft length**, the value to define determines the size of the sewing surface created between two adjacent elements.

	
Rectangular Optical lens with a drafting of 2mm	Circular Optical lens with a drafting of 2mm

- When selecting **Draft angle**, the value to define determines the angle to create between the demolding axis and the facet.

		
Draft angle	Demolding axis	Impact of drafting angle on an optical surface

Note: The **Draft angle**Draft Angle is not supported when pillows/faces have edges in common connected to vertices that are at different levels, as seen in the following example:



In this case, we recommend you to use the **Draft length**.

Related tasks

Creating an Optical Surface on page 503
This page shows how to create an optical surface from a parabolic or freeform support.

13.4.6.6. Beams

This section gathers all beam types available when designing an optical surface.

13.4.6.6.1. Radii

The Radii beam type allows you to create spherical facets that ideally diffuse light.

With the radii beam type, a definition of certain parameters is required:

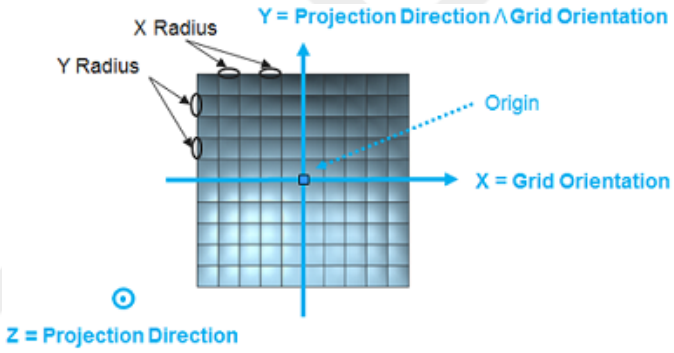
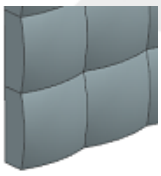
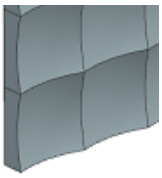
- You need to define the **Radius** of the facet.
- You need to select an **Orientation**: concave or convex.

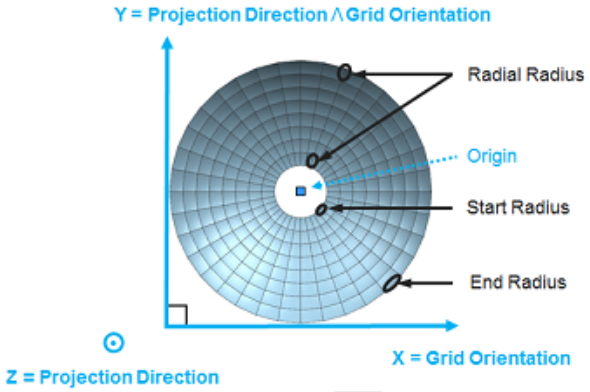
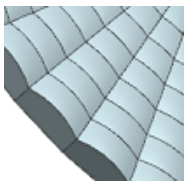
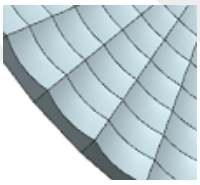
A concave surface is curved inwards.

A convex surface is curved outwards.

> **Note:** For each element, the convexity/concavity is defined with regards to the source.

Radii is available for Rectangular, Circular, Freestyle grid types.

<table>
<tr><td rowspan="4">Rectangular</td><td colspan="2">X radius: Radius of the facet along X grid axis.
Y radius: Radius of the facet along Y grid axis.</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Convex Lens</td><td>Concave Lens</td></tr>
</table>

<table>
<tr><td rowspan="4">Circular</td><td>Start radius: Smallest value of the radius of curvature of the elements along the transverse axis.
End radius: Highest value of the radius of curvature of the elements along the transverse axis.
Radial radius: Radius of the elements along the radial axis.</td></tr>
<tr><td> 
</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Convex Lens</td><td>Concave Lens</td></tr>
<tr><td>Freestyle</td><td colspan="2">Freestyle Rectangular: Parameters are the same as the ones used for the rectangular type.
Freestyle Circular: Parameters are the same as the ones used for circular type.</td></tr>
</table>

13.4.6.6.2. Freeform

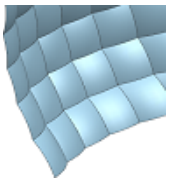
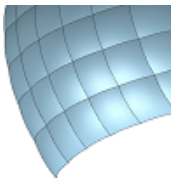
With the Freeform beam type, elements are shaped by specifying the rectangular area where the light should be sent in.

- **Orientation**: concave or convex.

  A concave surface is curved inwards.

  A convex surface is curved outwards.

> **Note:** If **X start** > **X end** or **Y start** > **Y end**, a concave freeform element becomes convex and a convex freeform element becomes concave respectively on X or Y axis.

	
Convex parameter influence	Concave parameter influence

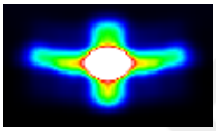
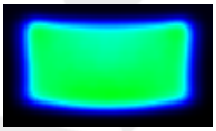
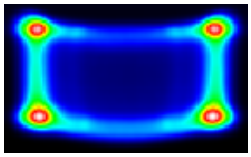
- **X start**: Left angular boundary in the target.
- **X end**: Right angular boundary in the target.
- **Y start**: Bottom angular boundary in the target.
- **Y end**: Top angular boundary in the target.
- **Custom spread**: Boolean to drive the beam spread inside the rectangle of specification. This option better respects beam specification but tends to cause gaps between facets.

> **Note:** We recommend you to set Custom spread to True. Otherwise you may face incorrect pillow construction.

> **Important:** **Custom spread** will be removed (hidden) from the interface and always activated in a coming release.

- **X spread** and **Y spread**: Ratio driving the beam spread along X and Y Target axes.

> **Note:** These parameters are only available when **Custom spread** is activated.

0% < Ratio < 100%	100%	100% < Ratio < 200%
Light distribution inside the specification rectangle tends to concentrate toward the center.	Light distribution inside the specification rectangle tends to be uniform.	Light distribution inside the specification rectangle tends to concentrate toward the edges.
		

13.4.6.6.3. Sharp Cutoff

The following section focuses on Sharp Cutoff beam type. A Sharp Cutoff beam directs light in a certain way so that it remains under a defined line.

13.4.6.6.3.1. Sharp Cutoff Overview

This page presents the Sharp Cutoff beam type and the two construction modes that are available to build the beam.

The Sharp Cutoff is a type of optical surface beam shaped to direct light underneath a specified cut-off line on the target.

The Sharp Cutoff is a target-oriented task definition: you define the light behavior of the extended source on the target thanks to the parameters and the facet will shape accordingly.

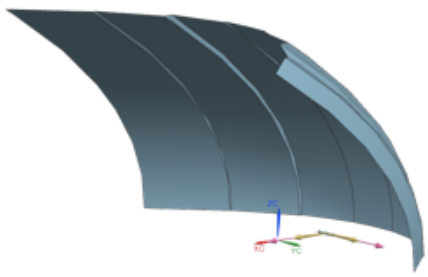
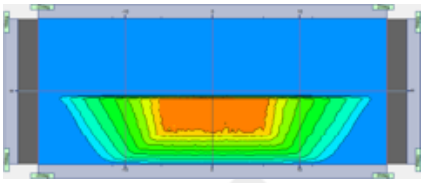
Two modes are available to create a Sharp Cutoff beam: the **Standard Mode** and the **Advanced Mode**.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Standard Mode

The Standard mode allows you to easily design a sharp cutoff beam.

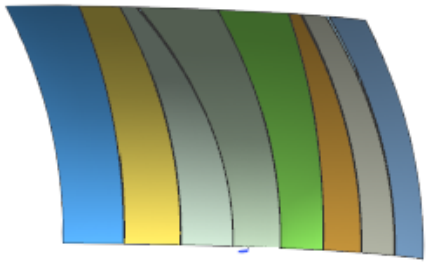
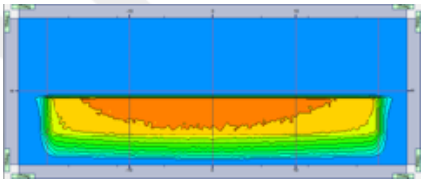
This mode provides a standard set of parameters and the definition of the beam is partly automated.

	
Simplified Sharp Cutoff Reflector	Target Result

Advanced Mode

The Advanced mode is a flexible mode that allows you to fully control the light direction according to the reflecting area of the facet.

This mode is useful to design low beams, fog lamps or cornering lights.

	
Advanced Sharp Cutoff Reflector	Target Result

Related information

Sharp Cutoff in Standard Mode on page 525
This section focuses on Sharp Cutoff definition and parameters in Standard mode.

Sharp Cutoff Advanced on page 529
This section focuses on Sharp Cutoff definition and parameters in advanced mode.

13.4.6.6.3.2. Sharp Cutoff in Standard Mode

This section focuses on Sharp Cutoff definition and parameters in Standard mode.

DRAFT

13.4.6.6.3.2.1. Creating a Sharp Cutoff Beam in Standard Mode

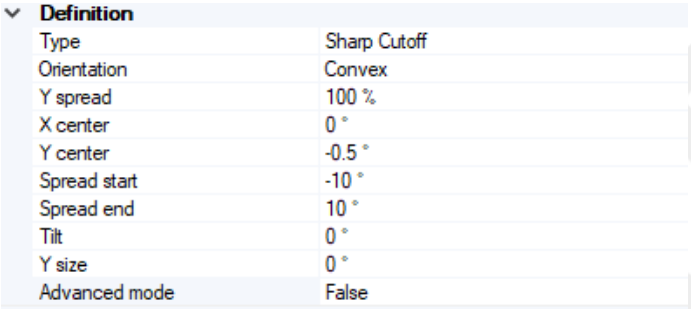
This procedure shows how to create a Sharp Cutoff beam in Standard mode.

> **Note:** You can repeat the following task for each created group. For more details about the beam parameters, see Standard Sharp Cutoff Beam Parameters

To create a Sharp Cutoff beam:

Make sure to use an extended source.

1. From the **Groups** panel, click **Add**  to add as many groups as needed.
2. Select a Group and from the **Type** drop-down list, select **Sharp Cutoff**.



Definition	
Type	Sharp Cutoff
Orientation	Convex
Y spread	100 %
X center	0 °
Y center	-0.5 °
Spread start	-10 °
Spread end	10 °
Tilt	0 °
Y size	0 °
Advanced mode	False

3. Make sure the **Advanced mode** is set to **False**.
4. In **Orientation**, select **Concave** or **Convex** to determine the behavior of the rays on the target.
5. Define the cutoff line points coordinates:
   - In **X center**, define the middle point's position of the specification line along **X target** axis.
   - In **Y center**, define the position of the specification line along **Y target** axis.
6. Define the cut-off line's amplitude:
   - In **Spread start**, define the length between the X center of the specification line and its left extremity.
   - In **Spread end,** define the length between the middle point of the specification line and its right extremity.
7. In **Tilt**, define the cut-off line's orientation by specifying the degree angle between **X Target** axis and the specification line.
8. Define the light distribution under the cut-off line:
   - In **Y size**, define the beam spread along **Y Target** axis.
   - In **Y spread**, specify the ratio (in %) driving the beam spread along **Y Target** axis.

> **Note:** When the ratio equals 100%, the light distribution inside the specification line tends to be uniform. The less the ratio is, the more concentrated the beam light will be under the specification line.

DRAFT

Related tasks

Managing Groups and Elements on page 506
This page shows how to create and manage groups. When designing an optical surface, you can create different groups of elements to apply specific parameters to these groups.

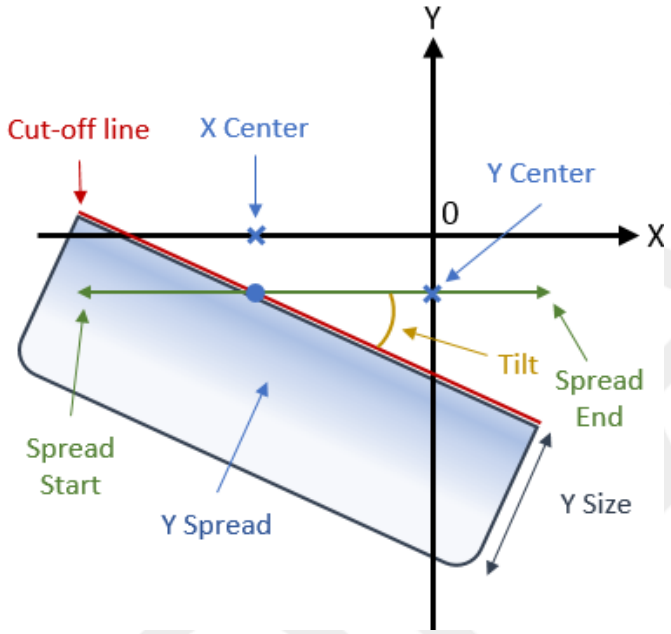
Related reference

Standard Sharp Cutoff Beam Parameters on page 527
This page describes the parameters used to create a Sharp Cutoff beam in standard mode.

13.4.6.6.3.2.2. Standard Sharp Cutoff Beam Parameters

This page describes the parameters used to create a Sharp Cutoff beam in standard mode.

Figure 68. Target schema with Sharp Cutoff parameters



Cut-off line points' coordinates

To place the cut-off line on the target, two coordinates must be created. These coordinates (X and Y centers) are used to determine the top borderline of the beam on the target.

- **X center**: X coordinate of the point used to place the cut-off line on the target.
- **Y center**: Y coordinate of the point used to place the cut-off line on the target.

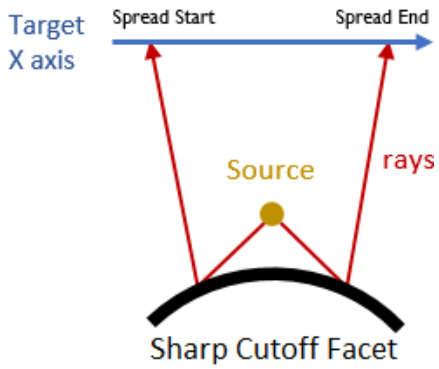
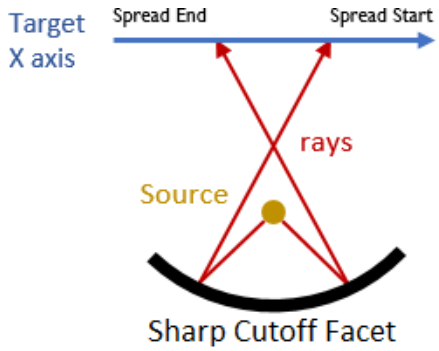
Cut-off line amplitude

The beam's amplitude is defined along X axis and determines the emission pattern of the beam (that is, if the beam has a narrow or broad emission).

- **Spread start**: Length between the middle point of the specification line and its left extremity.
- **Spread end**: Length between the middle point of the specification line and its right extremity.
  - When **Spread start < Spread end**, the rays never cross each other. The Sharp Cutoff element is convex in X direction.

- When **Spread start > Spread end**, the rays cross each other. The Sharp Cutoff element is concave in X direction.

> **Note:** **Convex** and **Concave** parameters have no impact on the rays direction on the Target X axis.

	
Spread Start < Spread End	Spread Start > Spread End

Cut-off line orientation

The orientation of the cut-off line is controlled by the tilt angle.

Tilt: Angle between **X target** axis and the specification line.

Light distribution under the cut-off line

- **Y size**: Beam spread along **Y target**.
- **Y spread**: Ratio driving the beam spread along **Y target** axis.

> **Note:** When the ratio equals 100%, the light distribution inside the specification line tends to be uniform. The less the ratio is, the less spread/the more concentrated the beam light will be under the specification line.

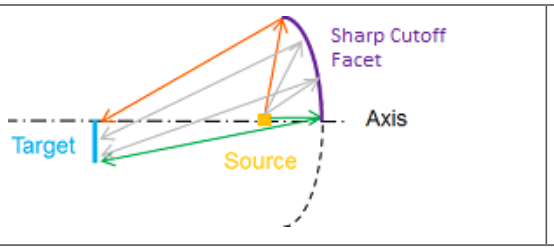
Elements shape

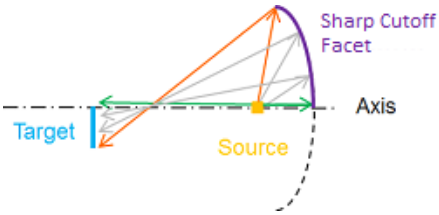
The convexity/concavity of the elements determine the behavior of the rays on the target.

> **Note:** The facets of the reflector will not necessarily appear concave or convex.

- **Convex**: a ray reflected on the top of the element will come to the top of the target when **Y size** is non-null. Rays never cross each other.
- **Concave**: a ray reflected on the top of the element will come to the bottom of the target when **Y size** is non-null. Rays cross each other.

DRAFT





Beam ray tracing of a sharp cutoff element and a non-null Y Size - Convex on the left, Concave on the right

Performance

A progress bar displays the progress of the feature generation algorithm. The displayed information are:

- Element Numbers: Current and total elements of the feature.
- Status: Progress on the feature generation algorithm (percent).
- Estimated time remaining: Remaining time for the feature generation algorithm (seconds).

13.4.6.6.3.3. Sharp Cutoff Advanced

This section focuses on Sharp Cutoff definition and parameters in advanced mode.

13.4.6.6.3.3.1. Creating a Sharp Cutoff Beam in Advanced Mode

This procedure shows how to create a Sharp Cutoff beam in advanced mode. The advanced mode allows you to control the light distribution on the target per point on facet.

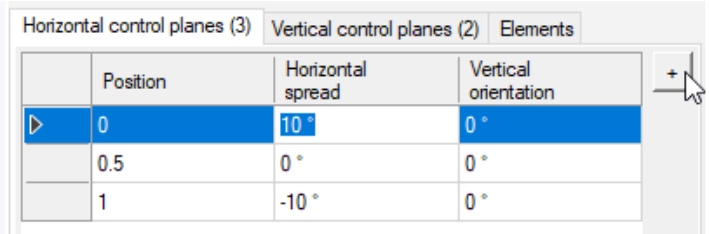
To create a Sharp Cutoff beam in Advanced Mode:

Make sure to use an extended source.

1. From the **Groups** panel, click **Add** + to add as many groups as needed.
2. Select a Group and from the **Type** drop-down list, select **Sharp Cutoff**.
3. Set **Advanced mode** to **True**.

Definition	
Type	Sharp Cutoff
Orientation	Convex
Advanced mode	True
Horizontal control planes	2 items
Vertical control planes	2 items

4. Define the beam parameters:
   a) Define the horizontal and vertical control planes position on X and Y directions:
      1. Click in a line of the table and click
         +
         to create a new control plane.



2. In **Horizontal control planes** and **Vertical control planes**, define the X and Y position of the current plane.

b) For each plane created, define the reflected rays direction:

1. In **Horizontal Spread**, define the direction of the reflected rays along the horizontal axis of the target (X).
2. In **Vertical Spread**, define the direction of the reflected rays along the vertical axis of the target (Y).

c) In **Vertical orientation**, type a value to adjust the horizontal spread size of the beam on the target.

d) In **Tilt**, enter an angular value (in degrees) define the cut-off line's orientation.

Related tasks

Managing Groups and Elements on page 506
This page shows how to create and manage groups. When designing an optical surface, you can create different groups of elements to apply specific parameters to these groups.

Related reference

Advanced Sharp Cutoff Beam Parameters on page 530
This page describes the parameters used to create a Sharp Cutoff beam in advanced mode.

13.4.6.6.3.3.2. Advanced Sharp Cutoff Beam Parameters

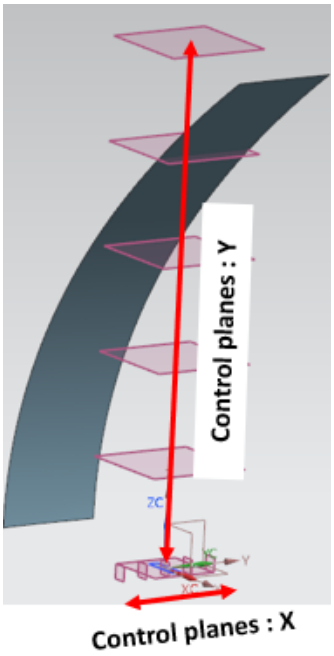
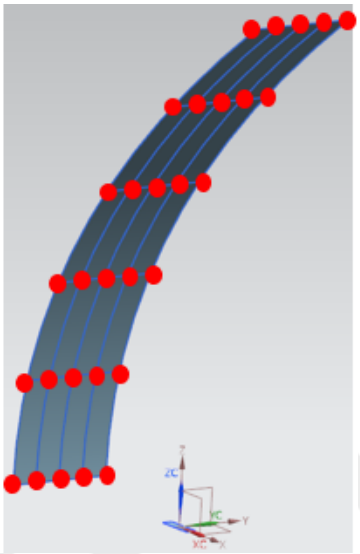
This page describes the parameters used to create a Sharp Cutoff beam in advanced mode.

In Advanced Mode, you control the light distribution on the target per point on facet. That means you can apply a different set of parameters to different points of the facet. The points are defined by the intersections of the control planes with the surface.

Control Planes

The control planes determine the intersection points made with the facet. These intersections generate points that allow to apply different parameters on different areas of a same facet.

DRAFT

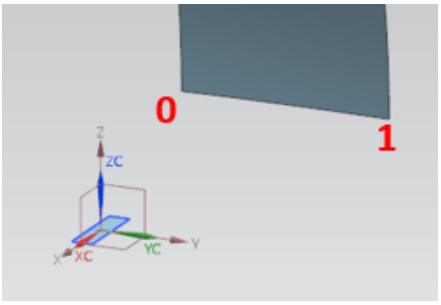
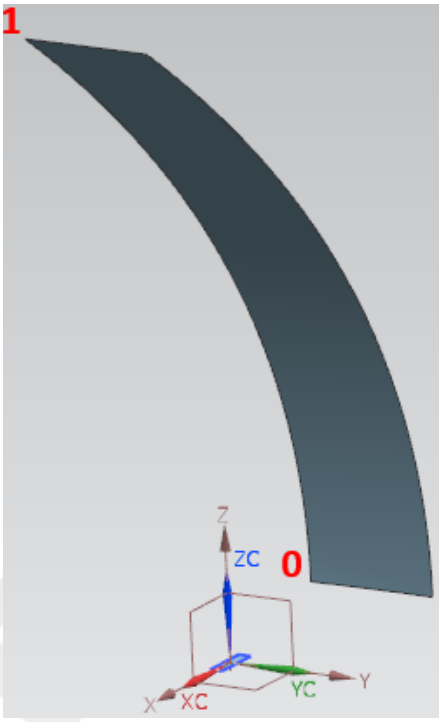
	
Control planes	Points created on the facet by the intersection curves of the control planes and the facet.

- **X** corresponds to the direction of the grid orientation axis. The **vertical control planes** are placed along X.
- **Y** corresponds to the direction normal to the plane made with the grid's projection and orientation. The **horizontal control planes** are placed along Y.

Position

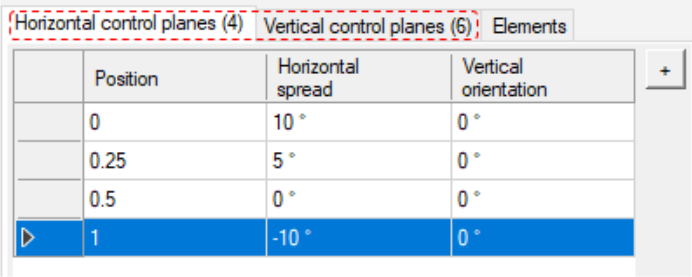
The control planes need to be defined and positioned independently on X (horizontal axis) and Y (vertical axis).

DRAFT

	
X axis	Y axis

To position the control planes, you need to define a ratio (X and Y value) between 0 and 1.

0 corresponds to the negative direction of the axis, and 1 to the positive direction of the axis.



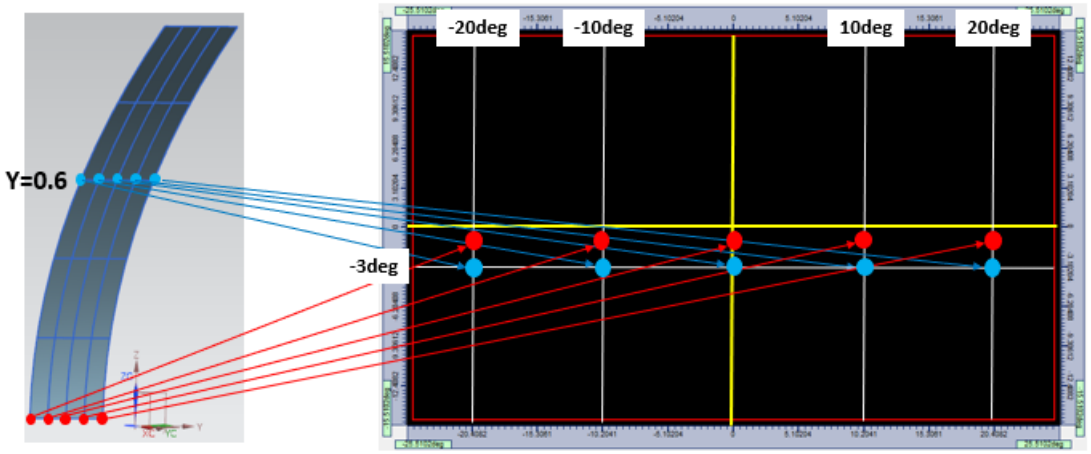
Reflected Rays Direction

- **Horizontal spread** drives the direction of the reflected ray along the horizontal axis. The value to set is expressed angularly (in degrees), 0° corresponding to the same direction as the target axis.
- **Vertical spread** drives the direction of the reflected rays along the vertical axis. The value is expressed angularly (in degrees), 0° corresponding to the same direction as the target axis.

Horizontal and Vertical Spread values between control planes are linearly interpolated.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

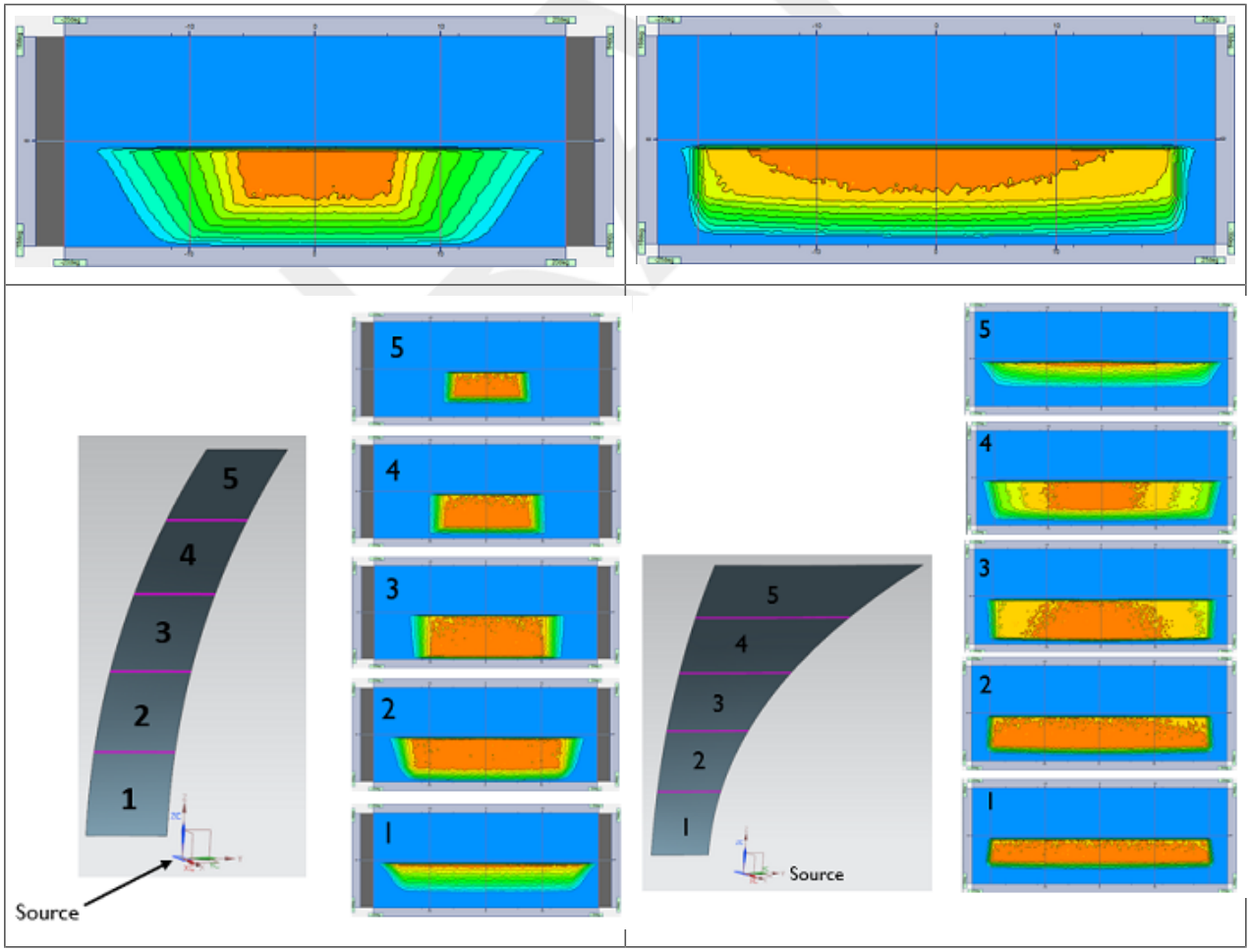
Figure 69. Impact of the Vertical / Horizontal Spread on the Target



Control planes orientation

The **Vertical orientation** adjusts the horizontal spread size of the beam on the target.

This parameters allows you to adjust the extent of the beam's emission and therefore to generate different emission patterns (wider or narrower emission).



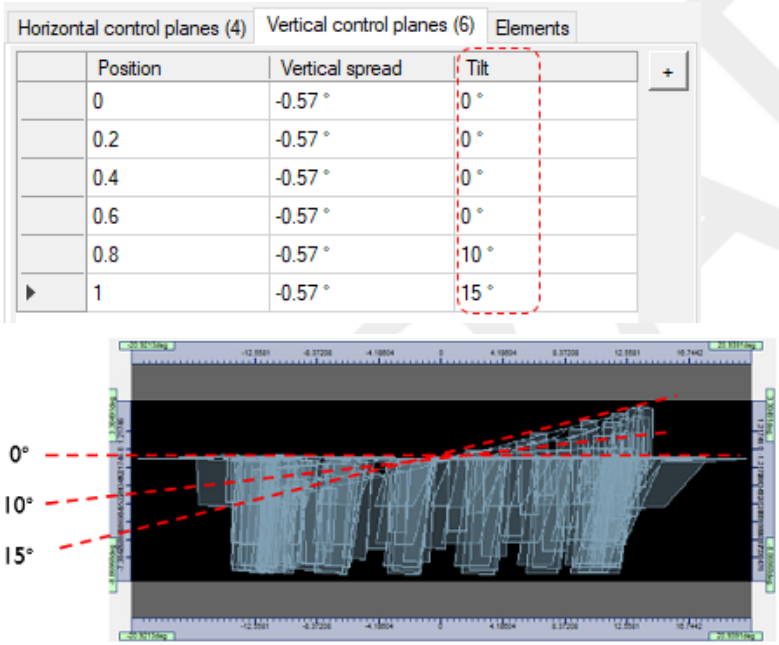
Standard Sharp Cutoff (Simplified behavior)	Sharp Cutoff in Advanced Mode
Vertical Orientation to 0° The farther the facets are from the source, the less respected the Horizontal Spread.	Vertical Orientation = Horizontal Spread. The Horizontal Spread is respected on the target and the shape of the facet is modified with respect to optical principles. The result geometry takes space of adjacent facets and you need to consider the balance beam pattern and the size/shape of each facet.

Cutoff line orientation

The **Tilt** allows you to define the orientation of the cut-off line for each area of the facet defined by the control planes.

A different value can be applied to each control plane to create a progressive tilt.

Figure 70. Tilt Impact on the Target



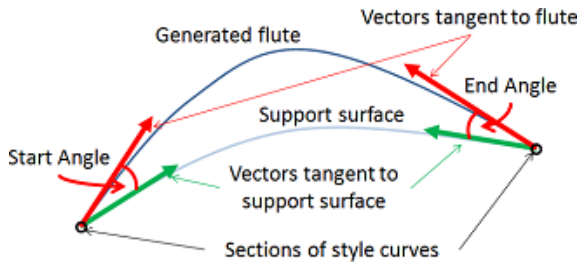
13.4.6.6.4. Flutes

Flutes are shaped by the geometrical angle between tangents of the support.

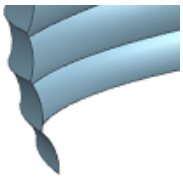
Flutes are shaped by specifying the geometrical angles between tangents of the support and of the optical surface.

By adjusting the flutes curvature, you can drive the angular spread of the beam.

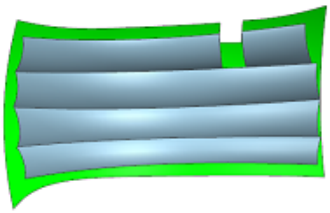
- **Start angle**: Angle between the vector tangent to the support and the vector tangent to the Flute.



- **End angle**: Angle between the vector tangent to the support and the vector tangent to the Flute.
- **Orientation**: the flutes can be defined as concave or convex.

	
Convex Flutes	Concave Flutes

Flutes can also be made of several parts when the support has holes in it or has an irregular shape.



13.4.6.7. Support

This page describes the parameters to set when working with a **Parabolic** or **Freeform** support.

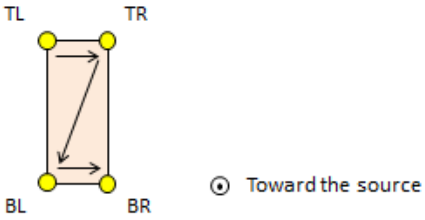
Parabolic Surface

Support	
Focal	20 mm
X center	1 °
Y center	10 °
Centered on X spread	False
Centered on Y spread	False

- **Focal**: The focal length of the support is defined differently depending on the beam type selected.
  - For **Freeform** and **Radii** beam types: the focal length corresponds to the distance between the source barycenter and the support apex.
  - For **Sharp Cutoff** beam: the focal length is the distance between the barycenter of the source and the corner of the element closest to this source. If this distance is not unique, the **Focal** matches the distance related to the first corner met when going through them in the following order:

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Figure 71. Corners are swept in this order: Top Left (TL), Top Right (TR), Bottom Left (BL), Bottom Right (BR)



- **X center**: tilt of the parabolic support axis around X support axis.
- **Y center**: tilt of the parabolic support axis around Y support axis.
- **Centered on X/Y spread**: Links the value of X/Y center to the middle of the horizontal/vertical specification of the beam when set to **True**.

> **Note:** This parameter is only available with the **Freeform** beam type.

This parameter horizontally and/or vertically orientates the parabolic support to make the beam pattern match the most with the specification rectangle.

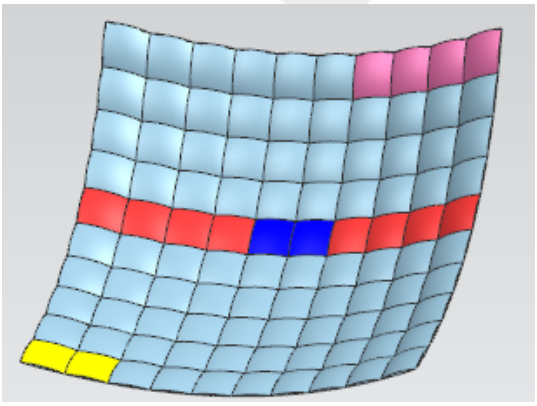
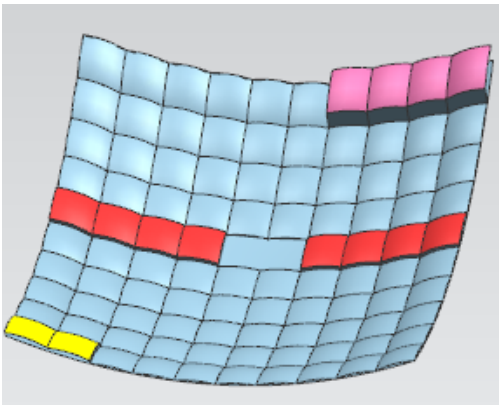
The formulas used when the Boolean is true are the following:

- **X center** = (**X start** + **X end**) / 2
- **Y center** = (**Y start** + **Y end**) / 2

Freeform Surface

When working with a freeform surface as support, the only parameter to adjust is the Shift of the facets from the surface.

- **Shift** corresponds to a translation of the selected group along the Grid Axis. If Shift > 0, the translation is done in the Grid Axis direction. If Shift < 0, the translation is done in the opposite of the Grid Axis direction.

	
Shift = 0 for all groups	Pink group Shift = 5mm Red group Shift = 1mm Blue group Shift = -3mm Yellow group Shift =2mm

- **Only one point on support** is a boolean imposing that all the elements have only one point belonging to the support.

Note: This parameter is visible at the feature level only.

When this parameter is set to **True**, elements are tilt in order for their beam patterns to be centered on the target origin.

Related tasks

Managing Groups and Elements on page 506
This page shows how to create and manage groups. When designing an optical surface, you can create different groups of elements to apply specific parameters to these groups.

13.4.7. Display Properties

The Display Properties allow to customize the information displayed in the feature viewer.

13.4.7.1. Understanding Display Properties

The viewer and display properties assist the design process by helping to understand the feature behavior.

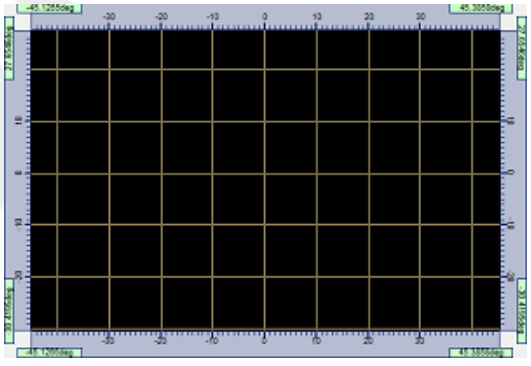
Note: X and Y target axes used to display values in the viewer are defined in the target.

The viewer gives information about the feature (surface) behavior and characteristics.

In the viewer, different types of informations are displayed according to your configuration of the optical surface.

Several types of elements can be displayed.

Grid



Grid with X Step = 10deg and Y Step = 10deg

A grid is displayed in the viewer and gives information about the size of the beam.

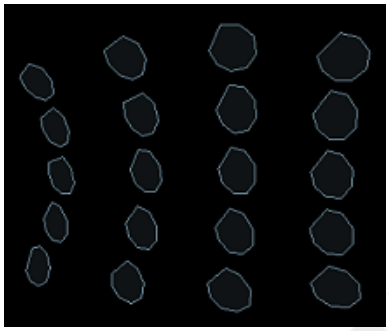
The grid is defined angularly and the step is expressed in degrees.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Source Images

Source Images is only available with the Extended source type.

The source images give information about the surface behavior inside the target. Their color depend on the group's color of the selected element.



Source Images with U Samples = 5 and V Samples = 4

U Samples indicate the number of source images along the first parametric surface direction.

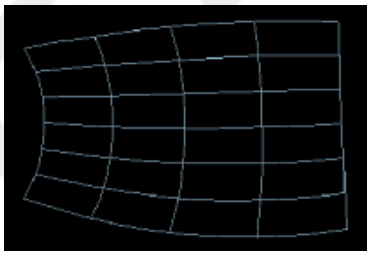
V Samples indicate the number of source images along the second parametric surface direction.

The external face of the elements is discretized according to **U Samples** and **V Samples** giving particular points. The image of the extended source is then calculated for each of these particular points using the Snell's law.

The source images are approximated because of the meshing and the extended source convex hull considered for the calculation.

Beam Pattern

A beam pattern is depicted as a grid (a network of lines) and gives information about the beam's shape.



Beam pattern with U Samples = 7 and V Samples = 5

U Samples indicate the number of isoparametrics along the first parametric surface direction.

V Samples indicate the number of isoparametrics along the second parametric surface direction.

The external face of the element is discretized according to **U Samples** and **V Samples** giving a particular network of lines on this element. The reflection of this network of lines is carried out using the Snell's law from the source point.

The calculation of the beam is calculated considering a punctual source. If an extended source is used, the barycenter of the extended source is considered as the punctual source.

DRAFT

Photometry

> **Note:** The Photometry is in BETA mode for the current release.

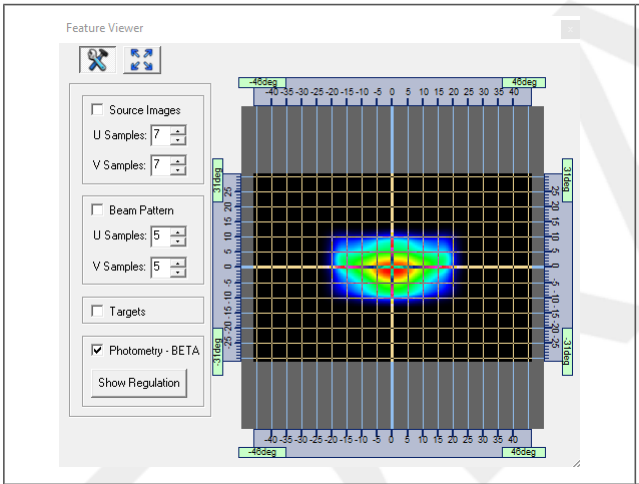
Description

The Photometry tool is an integrated simulation that allows you to have a quick preview and result of the selected element(s) of the feature.

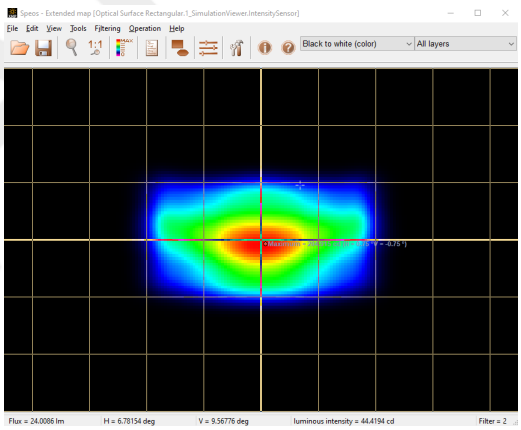
When activating the **Photometry**, the simulation is automatically launched using the GPU Compute functionality and generates a photometric preview displayed in the feature viewer, a HTML report and a XMP file in a folder named as **FeatureName_InteractivePhotometry** located in the Speos Output Files folder.

> **Note:** Each time you modify the selection of the element(s), the simulation is launched again, which overwrites the previous HTML report and XMP file.

Therefore, the Photometry tool allows you to directly iterate during the design process of the Optical Part Design feature in order to reach the regulation standards required, without having to run a Speos simulation that can take times to generate the output results.



Photometry of a group of elements from the feature viewer



XMP result of the group of elements selected

Click **Show Regulations** to open the HTML report generated.

Simulation Parameters

The Photometry tool simulation needs some information in order to run correctly. Therefore, the simulation considers the following parameters:

- **Source Parameters**
  - The simulation considers the **Extended** source type of the feature definition with a lambertian emission

> **Note:** The **Punctual** source type is not compatible.

- The simulation considers the **Flux** of the feature definition (100 lm by default).
- Optical Properties of the extended source: the simulation considers by default a **Surface Optical Property** as a **Mirror** set to **0%** and a **Volume Optical Property** set to **None**.

- **Geometry Parameters**

  The simulation considers the Optical Part Design feature geometry with the Optical Properties you applied manually.

  If you have not set any Optical Properties, the simulation considers by default a **Surface Optical Property** as a **Mirror** set to **85%**.

- **Sensor Parameters**

  The simulation considers the **Intensity** Target type of the feature definition.

  The size of the sensor used corresponds to the defined **Beam Pattern** parameters of the feature viewer.

- **Regulation Standards**

  The simulation considers the **XML template** selected in the Target definition.

  The XML template corresponds to the standard that you want the feature to pass.

> **Note:** You can find existing XML templates of regulation standards in the Ansys Optical Library.

Beam Type Compatibility

The following table describes the compatibility between the beam types and the photometry:

Beam type	Radii	Freeform	Sharp Cutoff	Flute
Compatibility	✓	✓	✓	✗

Related tasks

Adjusting Display Properties on page 540
The Display Properties allows you to customize the viewer and the information displayed in it.

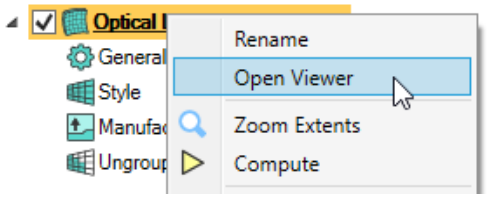
13.4.7.2. Adjusting Display Properties

The Display Properties allows you to customize the viewer and the information displayed in it.

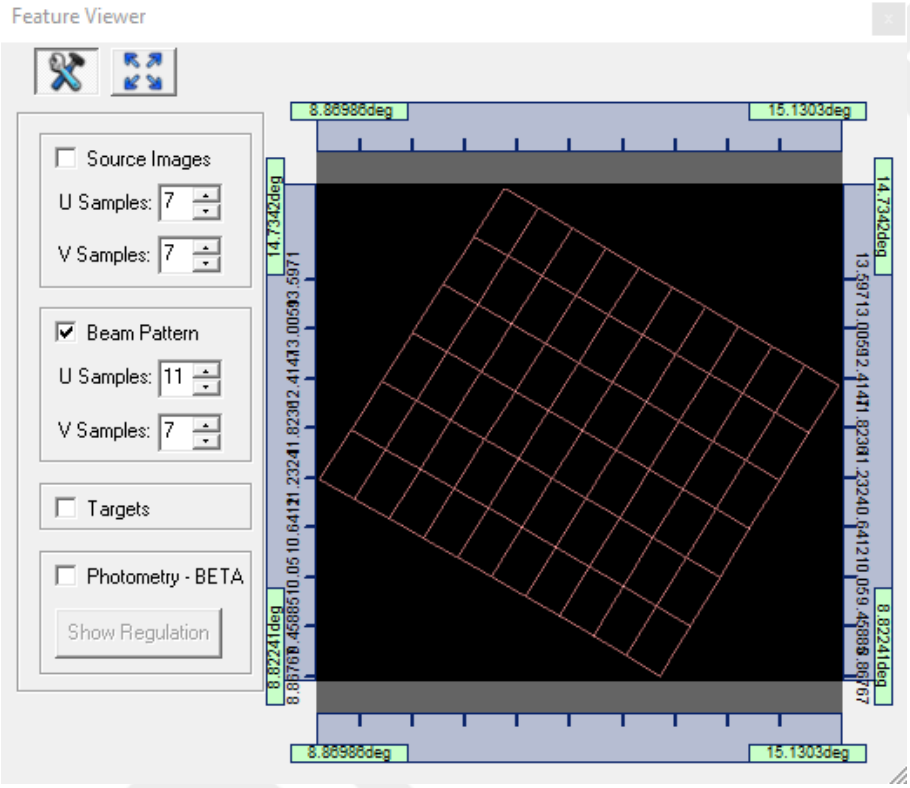
To adjust Display Properties:

1. From the **Design** panel, right-click the optical lens feature and click **Open viewer**.

DRAFT



2. In the 3D view, click a facet.
   Informations are displayed in the viewer.
3. From the **Feature Viewer**, click to adjust the display properties.



4. In **Source Images** and **Beam Pattern**, adjust the number of samples for U and V axes.
5. Check **Photometry - BETA** to run an integrated simulation that generates a photometric preview displayed in the feature viewer, a HTML report and a XMP file of the selected element(s) of the feature.

   This will help you verify if your current design passes the regulation standards that you selected in the XML template parameter of the feature definition.

   Fore more information, refer to Understanding Display Properties.
6. Click **Show Regulation** if you want to open the HTML report of the simulation.
7. To access grid parameters, right-click in the viewer and click **Grid Parameters**.
8. Click **User** and define the step you want for the gridline for U and V axes.

The display properties are set and the modifications are taken into account in the viewer.

Related concepts

Understanding Display Properties on page 537
The viewer and display properties assist the design process by helping to understand the feature behavior.

13.4.8. Interactive Preview

The interactive preview allows you to dynamically visualize different construction elements in the 3D view to help you modify and design the Optical Part Design feature as needed.

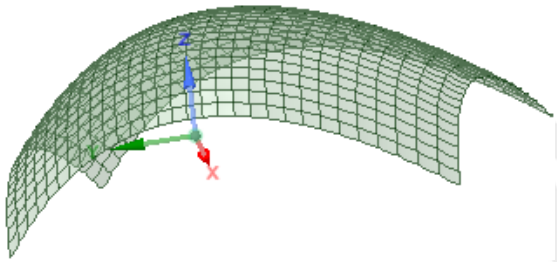
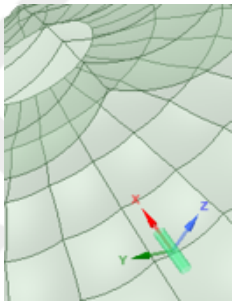
13.4.8.1. Understanding the Interactive Preview

The Interactive Preview allows you to visualize the grid of the Optical Surface, the support used, its source, its projected grid on the support.

Source

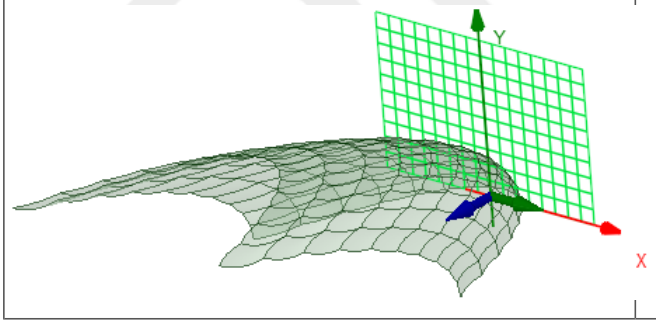
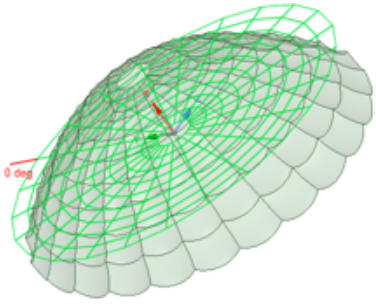
All source types can be displayed:

- Point Source: point highlighted in the 3D view
- Extended Source: emitting surface highlighted in the 3D view

Example of point source interactive preview	Example of extended source interactive preview
	

Grid

Only rectangular and circular grids can be previewed in the 3D view.

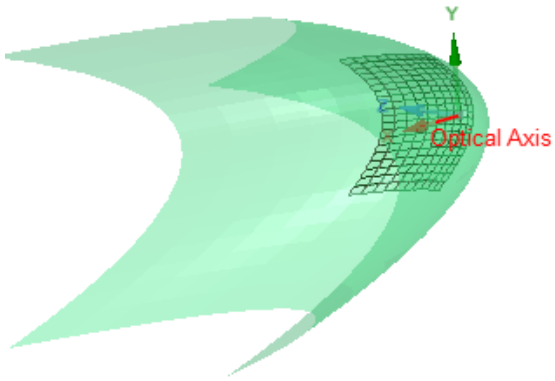
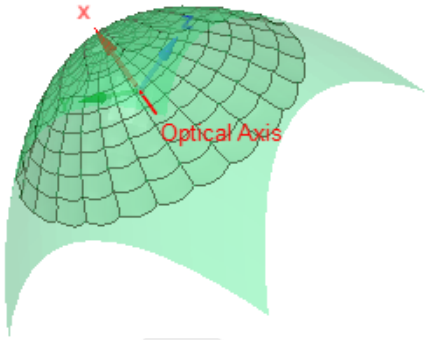
Example of rectangular grid interactive preview	Example of circular grid interactive preview
	

Support

Only Parabolic support can be previewed in the 3D view.

Created groups are not considered in the preview.

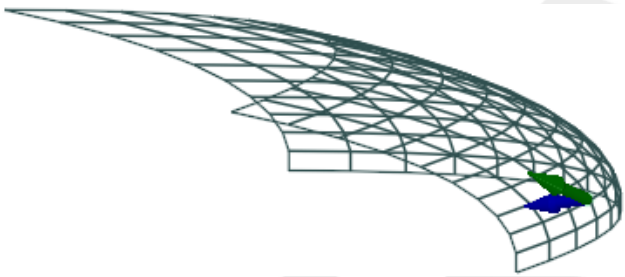
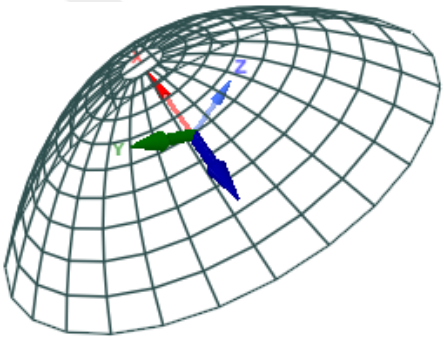
DRAFT

Example of parabolic support interactive preview for a rectangular grid	Example of parabolic support interactive preview for a circular grid
	

Projected Grid

All projected grid types can be previewed in the 3D view.

Created groups are not considered in the preview.

Example of rectangular projected grid interactive preview	Example of circular projected grid interactive preview
	

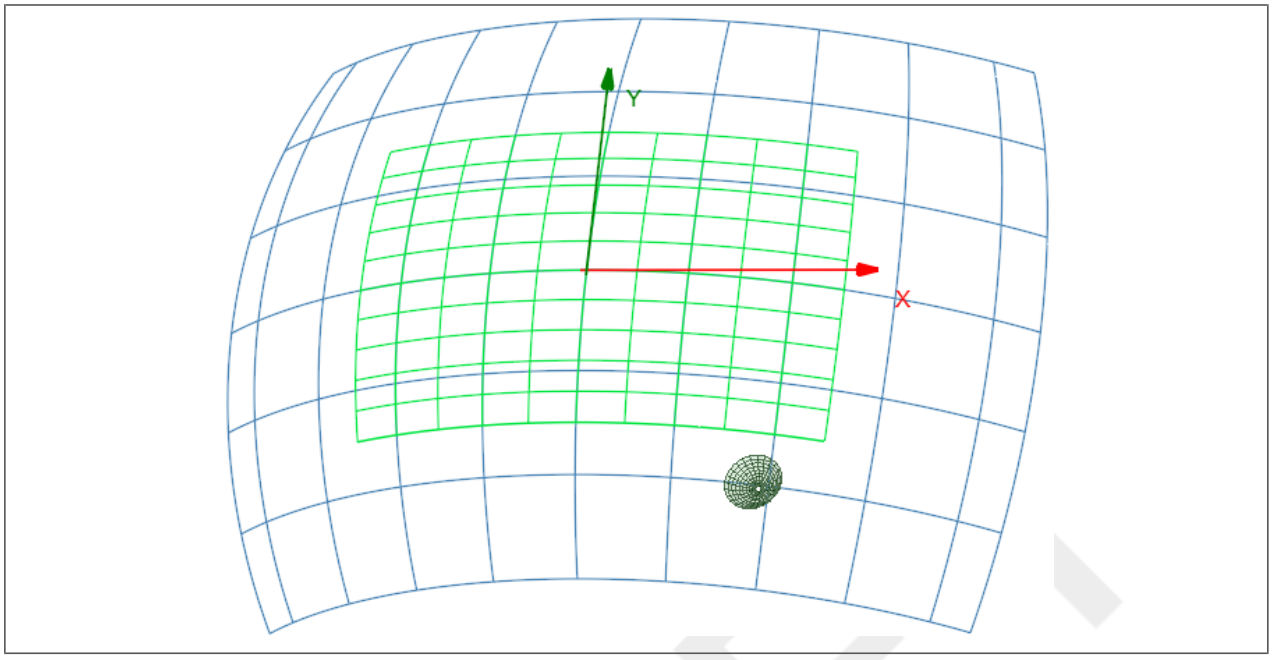
Target

The preview of the target is useful to correctly define your target and avoid useless steps of try and error.

Thin X and Y axes represent the axis system of the target.

Example of Optical Surface target preview

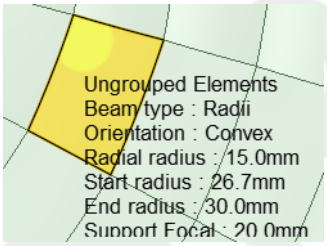
DRAFT

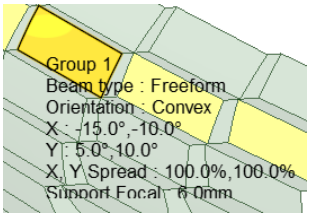


Facet Information

The Facet Information corresponds to the beam parameters of the facet you are hovering over. Facet Information is useful when your Optical Part Design feature is using multiple groups or an Excel file.

The Facet Information is always displayed when hovering over a facet.





13.4.8.2. Displaying the Interactive Preview Parameters

The following procedure helps you display the preview of parameters of an Optical Surface or an to give a dynamic overview upon feature modification.

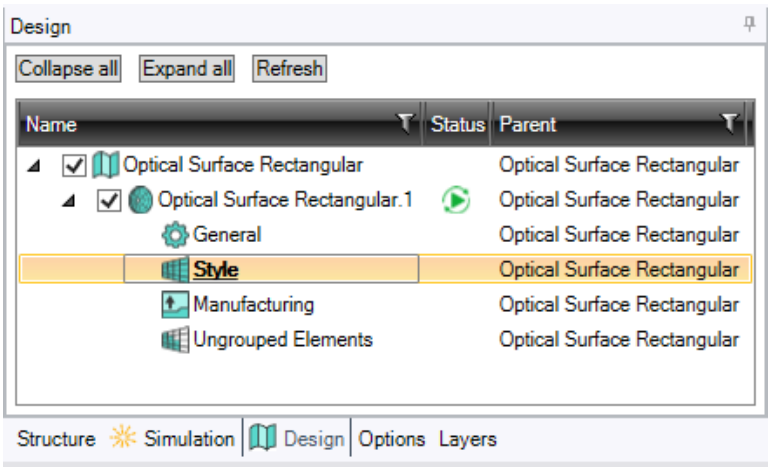
> **Important:** The Interactive Preview can be time-consuming, especially for the Projected grid preview in case of a huge number of grid intersections.

To display a parameter's interactive preview:

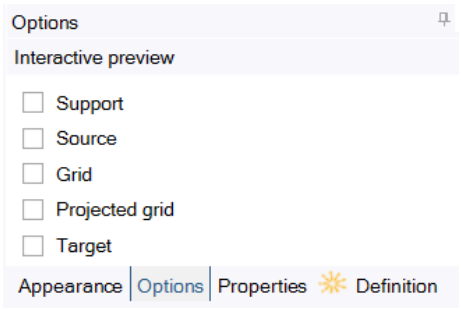
An Optical Surface must already be created.

1. In the **Design** panel, open one of the sub-definition group of the feature.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



2. Open the **Options** panel.



3. In the **Options** panel, check the parameters' interactive previews you want to display in the 3D view.
   - Source
   - Support (only parabolic support can be previewed)
   - Grid (only rectangular and circular grids can be previewed)
   - Projected Grid
   - Target

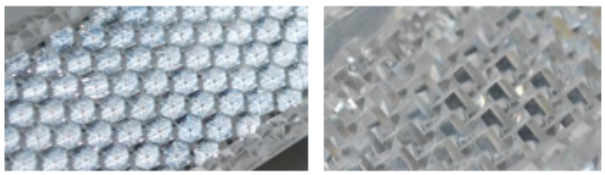
The parameters' previews are displayed in the 3D view and change dynamically upon feature modifications.

13.5. Optical Lens

The Optical Lens allows you to create a pillow, prismatic and pyramidal lenses.

13.5.1. Optical Lens Overview

The Optical Lens feature allows you to generate optical elements such as pillow, prismatic, pyramidal or reflex reflectors.

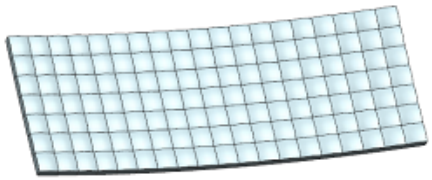
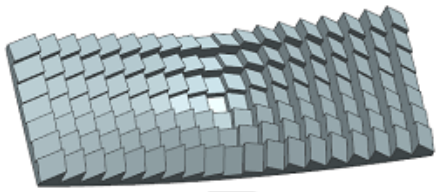
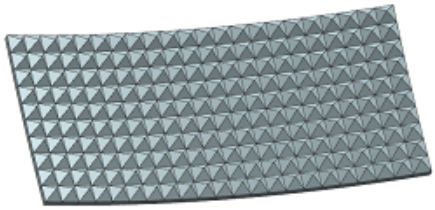
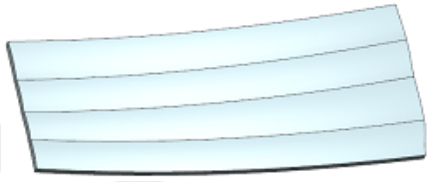
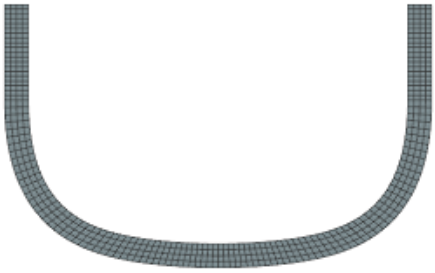
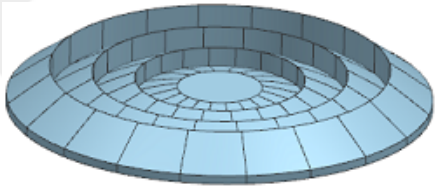


DRAFT

Optical lenses are key components of lighting systems. They allow to create elements to transmit, reflect or diffuse light.

They are often used to design automotive signal lighting.

Optical lenses distribution, shape and size can be fully customized by the user.

	
3D view of an optical lens used as pillow lens	3D view of an optical lens used as prismatic lens
	
3D view of an optical lens used as pyramid lens	3D view of an optical lens used as flute lens
	
3D view of an optical lens used as freestyle pillow lens	3D view of an optical lens used as freestyle pillow lens
	
3D view of an optical lens used as fresnel lens	

Related information

Creating an Optical Lens on page 546
This page shows how to create optical lenses.

13.5.2. Creating an Optical Lens

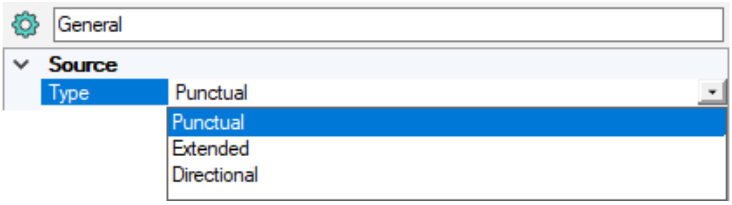
This page shows how to create optical lenses.

To create an Optical Lens:

1. From the **Design** tab, click **Optical Lens**

   and select a lens type/shape (Rectangular, Circular, Honeycomb, Stripes, Freestyle).
2. In **General**, from the Source **Type** drop-down list, select:

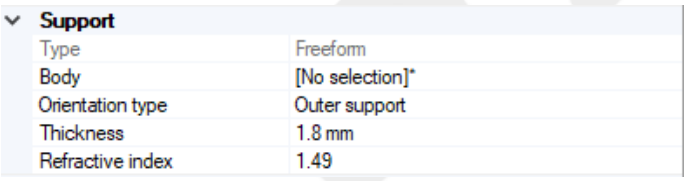
   

   - **Punctual** to position the source by selecting a point in the 3D view.
   - **Extended** to select an emitting surface like a filament, cylinder, LED, chip etc.
   - **Directional** to define the direction of the rays by clicking a line in the 3D view.

     In case of **Directional** source, the rays are collimated. A yellow arrow indicates the light direction. If needed, use **Reverse direction** to adjust the direction of the rays.
3. If you select **Extended**, you can define the **Flux** of the source.

   The Flux will be used to run the **Photometry** tool simulation that generates a photometric preview displayed in the feature viewer, a HTML report and a XMP file of the selected element(s) of the feature.

   For more information on the Photometry tool, refer to Understanding Display Properties.
4. Define the **Refractive index** of the medium from which the light comes.
5. To define the support:

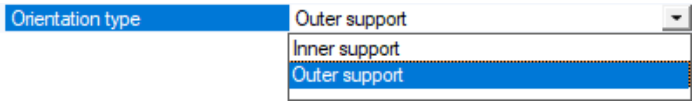
   

   Important: When computing the Optical Lens, the support is meshed. The bigger the support, the longer the computation of the feature. We recommend you to define a support size as close as possible to the feature size.

   a. Click  to select a surface in the 3D view.

   Note: A face or a multifaces body can be selected as surface.

   b. Define the **Orientation type** of the support, that is on which side of the surface the elements will be created:

   - Select **Outer support** to consider the outer surface as the support and create the elements on the inside.

DRAFT

- Select **Inner support** to consider the inner face as the support and create the elements on the outside.

> **Note:** The principle of inner and outer surface is determined by the source position. The source is always facing the inner surface of the support.

c. If needed, adjust the **Thickness** of the support. This value is used to define an intermediary surface carrying the elements.

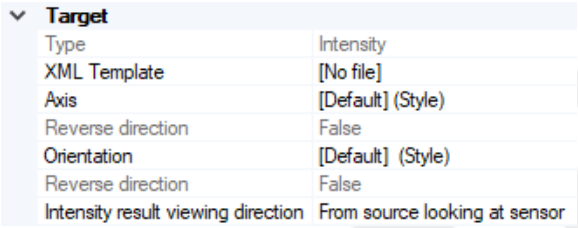
d. Define the **Refractive index** of the lenses.

6. In **Target**, if you want to verify that your current design passes regulation standards, select a **XML Template** corresponding to the regulation standard.

   The XML template will be used by the Photometry tool simulation to generates the photometric preview displayed in the feature viewer, the HTML report and the XMP file of the selected element(s) of the feature.

> **Note:** You can find existing XML templates of regulation standards in the Ansys Optical Library.

   For more information on the Photometry tool, refer to Understanding Display Properties.



7. From the **Type** drop-down list, select the type of metrics used for reviewing the beam pattern:
   - If the area to light is in a plane placed at a known distance from the source, select **Illuminance**.

     Light is focalized in one point/plane. The viewer displays illuminance values.
   - If the area to light is defined angularly, select **Intensity**.

     Light is directed in a given direction. The viewer displays intensity values.
   - If you want to define specific axes for the sensor, in the 3D view click
     
     to select a projection axis and
     
     to select an orientation axis or click
     
     and select a coordinate system to autofill the Axis System.
   - In case of an Illuminance target, define the integration direction of the sensor with **Reverse integration direction** if necessary. The Integration direction (non visible in 3D view) should be in the opposite direction of the target axis (red arrow). The target axis should face the Optical Lens.
   - In **Intensity result viewing direction**:
     - Select **From source looking at sensor** to position the observer point from where light is emitted.
     - Select **From sensor looking at source** to position the observer in the opposite of light direction.

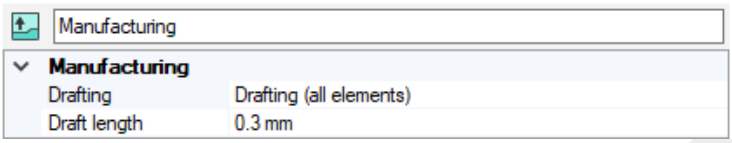
DRAFT

8. In the Style tab, set the axis system of the grid define the distribution, pattern and size of the elements to be created on the support.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

9. In Manufacturing, activate the drafting of the elements if you want mechanical constraints coming from unmolding to be taken into account.

   If activated, define a **Draft length** in mm. The draft length defines the sewing surface created between two adjacent faces.



10. From the **Design** tab, click **Compute**



to generate the lenses.

Optical Lenses are created are built in the 3D view. You can edit the lens at any time. The generation time of the Optical Lens depends on the complexity of the settings.

Related tasks

Managing Groups and Elements on page 549
This page shows how to create and manage groups. When designing an optical lens, you can create different groups of elements to apply specific parameters to these groups.

Related information

Style on page 558
This section describes all grid types available to design an optical lens.

Optical Lens Parameters on page 555
This section provides more information about the parameters to set when creating an optical lens.

13.5.3. Managing Groups and Elements

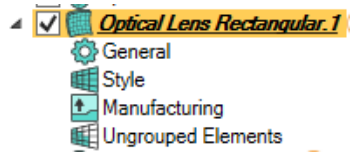
This page shows how to create and manage groups. When designing an optical lens, you can create different groups of elements to apply specific parameters to these groups.

> **Note:** When no group is created, all the elements (facets) are stored in **Ungrouped Elements** section.

To create and manage groups:

An Optical lens must already be defined.

1. Reach the feature level, and from the **Groups** panel, click **Add**  to add as many groups as needed.



Groups (3)

	Name	Type	Shift	Orientation	X radius	Y radius
▶	Ungrouped Elements	Radii	0 mm	Convex	15 mm	20 mm
	Group 1	Radii	0 mm	Convex	15 mm	20 mm
	Group 2	Radii	0 mm	Convex	15 mm	20 mm

> **Tip:** For a quicker way, right-click the feature and select **Add new group**.

The groups appear in the design panel.

2. Select each group to define its parameters. From the **Type** drop-down list, select the type of lens you want to apply to the current group:

> **Note:** Available beam types depend on the optical lens selected.
>
> For more information on lens types, see Beams .



- Select Radii to create a spherical lens that spreads light.
- Select Prism to create lenses that redirect light.
- Select Pyramid to create pyramid lenses shaped by their heights.
- Select Freeform to create lenses that are shaped by specifying targeted points or directions.
- Select Reflex reflector to create a lens that reflects light back to the source.
- Select Step Pillow to create facets that are shaped by specifying the area where the light should be sent in.

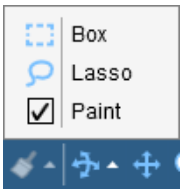
3. Click  and select the faces to include in the group:
   - In case of a Rectangular, Freestyle Rectangular, or HoneyComb part, you can directly **select a row**
     
     or **select a column**
     
   - In case of a Circular or a Freestyle Circular part, you can directly **select a radius**
     

or **select a circle**



.

- For a free selection of faces, you can choose a SpaceClaim selection mode in the bottom right corner of the session and use **Box**, **Lasso** or **Paint**.



- You can add Named Selections composed of faces to Optical Part Design groups.

> **Note:** You can only add a same type of element in a group. Example: a group composed of faces only, or a group composed of Named Selections only.

4. Once, the group is defined, click  to validate the selection.
5. If you want to apply a shift from the support, type a value in **Shift** to define thickness of the elements. The thickness set here adds up to the thickness set in the **General** section.

The Group is created and the properties are correctly applied to it. You can now access the feature viewer to obtain pieces of information regarding facets.

Related information

Creating an Optical Lens on page 546
This page shows how to create optical lenses.

Beams on page 568
This section gathers all beam types available when designing an optical lens.

13.5.4. Managing Elements from an Excel File

The following procedure shows how to create and use and Excel file that allows you to drive the parameters of each facet of the Optical Lens individually.

To create and use an Excel file:

An Optical Lens Rectangular or Circular must already be defined.

No group must have been created. Only the Ungrouped Elements must be present (and Central Closing group in case of an Optical Lens Circular).

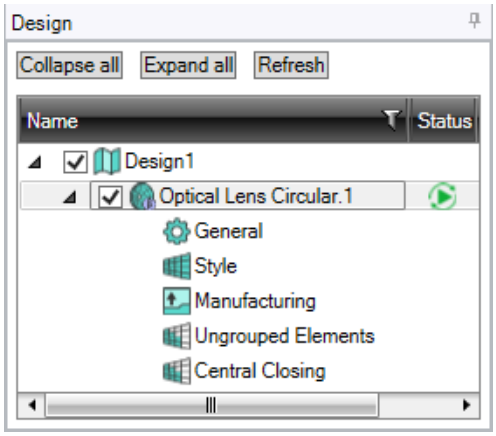
Make sure the Optical Lens is compatible with the Excel file definition.

1. From the **Design** panel, open the Optical Lens.
2. If you want, modify the parameters related to the facets in the **Style** and **Ungrouped Elements** definitions.

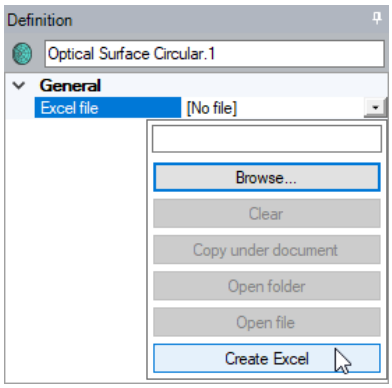
   The **Style** and **Ungrouped Elements** parameters corresponds to those who will be created and applied when you create the Excel file.
3. Open the main node of the Optical Lens.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00



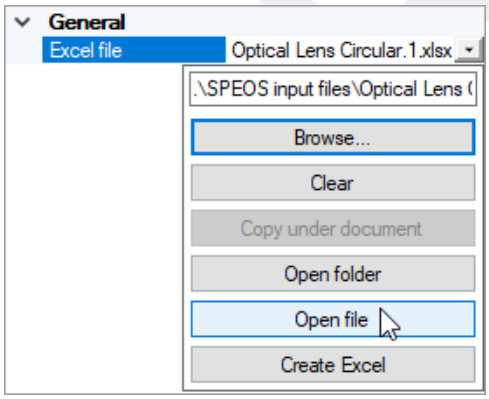
4. In the main node definition, from the **Excel file** drop-down list, click **Create Excel**.



An Excel file is created based on the input parameters set in the Style and Ungrouped Elements definition.

The Excel file opens automatically upon creation.

5. If you had already created the Excel File (automatic opening of the file already done), from the **Excel file** drop-down list, click **Open file**.



Important: For Speos to know that you modified the Excel file, you must open the file from here. Otherwise, if you open the file outside of the interface, Speos cannot know if you modified it.

6. Modify the Excel file as you want.

In the Excel file, each sheet correspond to a parameter and each cell in a sheet corresponds to one specific facet of the Optical Lens.

DRAFT

Tip: To find the match between a cell in the Excel file and its facet in the 3D view, you can hover over the feature. This will give you the cell coordinates.



7. Save the Excel file.
8. In Speos, click **Compute** to generate the Optical Lens and take into account the changes.

The Optical Lens is generated and built in the 3D view.

13.5.5. Understanding the Excel File

The following page helps you understand the use of an Excel file in the definition of an Optical Lens.

Description

The Excel file is an input file that allows you to drive every parameter that you can drive in a group for every facet of the Optical Lens individually. Basically the Excel file represents every facet of the feature as if one face = one group. It saves you from creating lots of groups in the Speos interface and allows you to quickly modify each parameter for each facet so that you can create a smooth evolution of the parameters value along the feature.

Note: The excel file replaces the management of the facets by group. Thus you cannot create groups when the Excel definition is activated.

Excel File Compatibility with Optical Lens Parameters

The Excel file is compatible with the following types of Optical Lens. All others types not mentioned in the table are not compatible.

Optical Lens	Beam	Parameters displayed in Excel
Rectangular	Radii	• X radius • Y radius • Shift
Rectangular	Prism	• X angle • Y angle • X radius • Y radius • Shift
Rectangular	Pyramid	• Height • Shift

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Optical Lens	Beam	Parameters displayed in Excel
Rectangular	Freeform	• X angle • Y angle • Shift
Circular	Radii	• Radial radius • Start radius • End radius • Shift **Important:** The behavior of radii pillows in circular is different between a definition by group and a definition using an Excel file. When using the Excel File definition, the radii pillows are built according to the values written in the Excel. They are not built according to a ratio of the position between the Start radius and End radius values.
Circular	Prism	• X angle • Y angle • X radius • Y radius • Shift
Circular	Pyramid	• Height • Shift
Circular	Freeform	• X angle • Y angle • Shift

Excel File Format

Each sheet corresponds to a parameter driving the facets.

Each cell corresponds to a facet in the 3D view.

Each cell can be defined use a formula or a macro. Speos will only interpret the result of the formula or the macro.

If a cell is empty in the Excel file, Speos will use the value from the Speos definition.

If a sheet does not exist for a parameter or a grid value, an error is raised suggesting you to create a new valid excel file. If you use the Excel file with the missing sheet, then the parameter values from the Ungrouped Elements will be used to replace the missing sheet.

Warning: If you create a new Excel file from the definition in Speos, you will not have the modifications you applied directly in the previous used Excel file. You will have to re-type them manually.

DRAFT

Note: We recommend you to define an Excel file of a maximum size of 100x100. A bigger file would affect the performance.

Figure 72. Example of a the sheet for an Optical Lens



	A	B	C	D	E	F	G	H	I	J	K
1		Start angle	0	5	10	15	20	25	30	35	40
2	Start radiu	End	5	10	15	20	25	30	35	40	45
3	200	195	20	20	20	20	20	20	20	20	20
4	195	190	20	20	20	20	20	20	20	20	20
5	190	185	20	20	20	20	20	20	20	20	20
6	185	180	20	20	20	20	20	20	20	20	20
7	180	175	20	20	20	20	20	20	20	20	20
8	175	170	20	20	20	20	20	20	20	20	20
9	170	165	20	20	20	20	20	20	20	20	20
10	165	160	20	20	20	20	20	20	20	20	20
11	160	155	20	20	20	20	20	20	20	20	20
12	155	150	20	20	20	20	20	20	20	20	20
13	150	145	20	20	20	20	20	20	20	20	20

Radial radius | Start radius | End radius | Shift

13.5.6. Optical Lens Parameters

This section provides more information about the parameters to set when creating an optical lens.

13.5.6.1. Source Types

This page describes the different types of sources available when designing an optical lens.

The source corresponds to the source that is used to illuminate the optical lens. In the interface, 3 types of sources are available.

- A **Punctual** source is a simplified source that emits light from a single point.
- An **Extended** source is used to emit light from a surface. To use this option, a source geometry (light bulb, LED, surface) must already exist or be created.

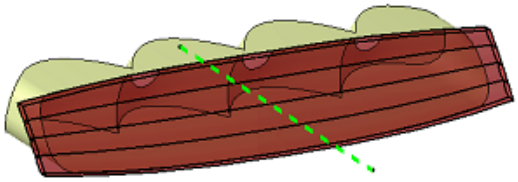
  With an extended source, the source images are available in the feature viewer.
- A **Directional** source determines the direction of the rays by selecting a line in the interface. The light source is considered as perfectly collimated before interacting with the lens.

  When selecting a directional source, a refractive index is required. This refractive index refers to the medium from which rays are coming. In most cases, this medium corresponds to the air, but if rays are coming through a different medium, you should apply the material's refractive index here.

  For example, if you want to create a lens on top of a plexiglass (PMMA) block, you need to apply the refractive index of the PMMA instead of the refractive index of the air.

  With a directional source, source images are not available and cannot be displayed in the feature viewer.

DRAFT



The line indicates the light direction and is considered as collimated by the reflectors.

Related information

Creating an Optical Lens on page 546
This page shows how to create optical lenses.

13.5.6.2. Support

The support is a construction geometry (usually a surface) used to carry the optical lenses.

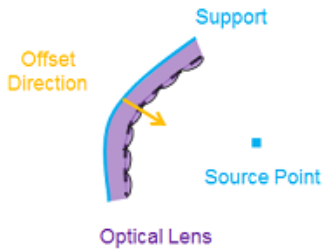
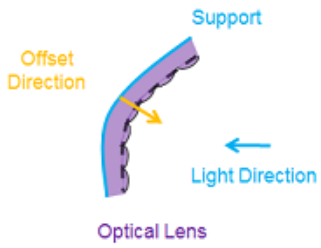
Support

The lenses are created on a support geometry. The lenses can be created on any freeform surface.

Outer/Inner Surface

The outer/inner surface parameter defines on which side of the support the elements will be created.

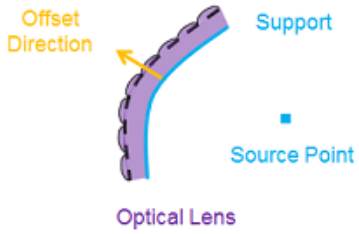
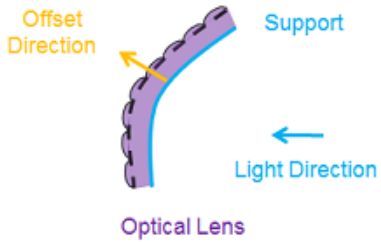
With **OuterSurface**, the outer face of the surface is considered as the support and the elements are created on the inside. The elements are facing the source.

	
Outer surface with a punctual source	Outer surface with a directional source

With **Inner Surface**, the inner face of the surface is considered as the support and the elements are created on the outside.

> **Note:** With this mode, you cannot use the feature viewer.

DRAFT

	
Inner surface with a punctual source	Inner surface with a directional source

The meaning of outer/inner is related to how the source has been defined, the inner surface being always on the side of the source unlike outer surface.

If a punctual source is moved to the other side of the support or if a directional source is reversed, then the roles of outer and inner surface are switched.

Thickness

The **Thickness** allows you to adjust the offset of the elementary elements. The thickness corresponds to the distance between the support and the "intermediary surface" of the support.

Type a value for the length used to define an intermediary surface carrying the elementary elements.

- **Radii**: The four corners of each optical face of the pillows belong to the intermediary surface.
- **Prism**: At least one corner of the optical face of the prisms belong to the intermediary surface.
- **Pyramid**: The four corners of the pyramid bases belong to the intermediary surface.
- **Flute**: The four corners of the optical face of the flutes belong to the intermediary surface.

Refractive Index

The **Refractive Index** of a material is a pure number that describes how light propagates through that medium.

It is defined as n=c/v (where *c* is the speed of light in vacuum and *v* is the phase velocity of light in the medium.)

Most transparent materials have refractive indices between 1 and 2. The default refractive index used here (1.49) is the index of plexiglass. This material is commonly used to design lenses.

Related information

Creating an Optical Lens on page 546
This page shows how to create optical lenses.

13.5.6.3. Target

The target defines how the results are going to be interpreted in the feature viewer.

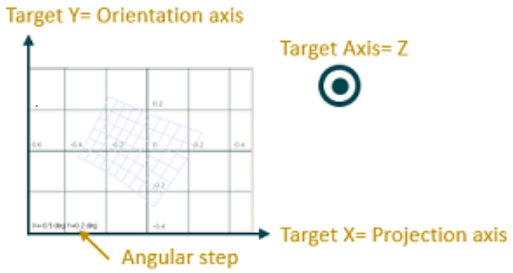
The target is more or less acting like a sensor. You can choose from this section how the results are going to be interpreted and what pieces of information are going to be displayed in the feature viewer.

Target Type

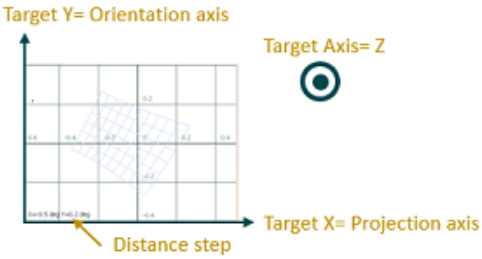
The target type defines what kind of values/metrics will be available in the feature viewer (**Optics** tab). Two target types are available:

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- **Intensity**: with this type, intensity values will be displayed on an angular grid in the feature viewer. It is often used when the area to light is defined angularly.

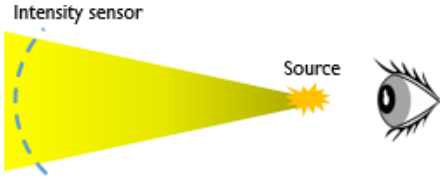
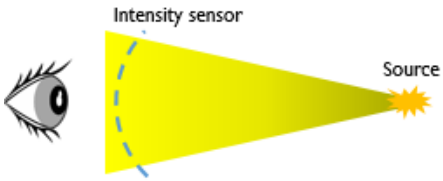


- **Illuminance**: with this type, illuminance values will be displayed in the feature viewer. Illuminance target is often used when the area to light is located at a known distance from the source.



Intensity result viewing direction

This parameter allows you to adjust the viewing direction of the observer.

From source looking at sensor (Intensity/Illuminance target): The viewing direction of the observer is the same as the light direction emitted.	
From sensor (Intensity/Illuminance target) **looking at source**: The viewing direction of the observer is in the opposite of the light direction.	

Related information

Creating an Optical Lens on page 546
This page shows how to create optical lenses.

13.5.6.4. Style

This section describes all grid types available to design an optical lens.

DRAFT

Note: In some cases, when modifying the grid of an Optical Lens, group definitions must be updated.

13.5.6.4.1. Rectangular

The grid determines the size and distribution of the lenses over the support.

By default, elements are distributed onto the support according to a rectangular pattern. This distribution can be customized as well as the shape and size of the lens itself.

Axis System

An axis system is required to define the element's orientation and projection on the support.

This axis system is, by default, inherited from the source definition.

Note: If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

Rectangular Grid

Elements are distributed according to a rectangular pattern. The following parameters can be set to customize this distribution:

- **X start**: Start value for the feature size along **X Grid** (mm).
- **X end**: End value for the feature size along **X Grid** (mm).
- **Y start**: Start value for the feature size along **Y Grid** (mm).
- **Y end**: End value for the feature size along **Y Grid** (mm).
- **X Angle**: Angle made by X axis and the horizontal edge of the elements (deg).
- **Y Angle**: Angle made by Y axis and the vertical edge of the elements (deg).
- **X count**: Number of elements along X axis.
- **Y count**: Number of elements along Y axis.
- **X size**: Size of the elements along X axis (mm).
- **Y size**: Size of the elements along Y axis (mm).

Related information

Creating an Optical Lens on page 546
This page shows how to create optical lenses.

DRAFT

13.5.6.4.2. Circular

The grid determines the size and distribution of the lenses over the support.

By default, elements are distributed onto the support according to a rectangular pattern. This distribution can be customized as well as the shape and size of the lens itself.

Axis System

An axis system is required to define the element's orientation and projection on the support.

This axis system is, by default, inherited from the source definition.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

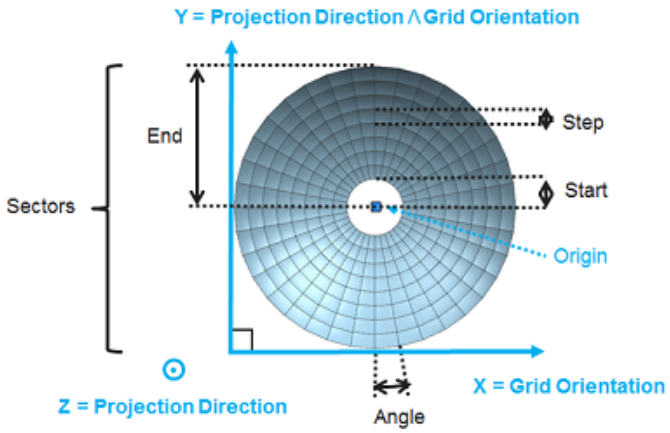
Circular Grid

> **Note:** Circular grid distribution is not compatible with **Flute** beam type.

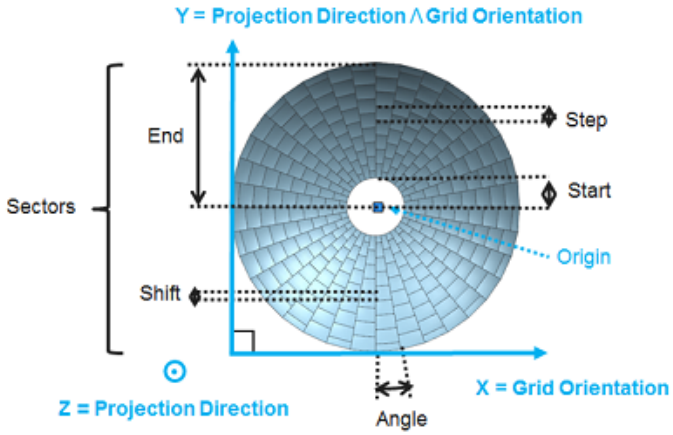
Elements are distributed according to a circular pattern. The following parameters can be set to customize this distribution:

Shift

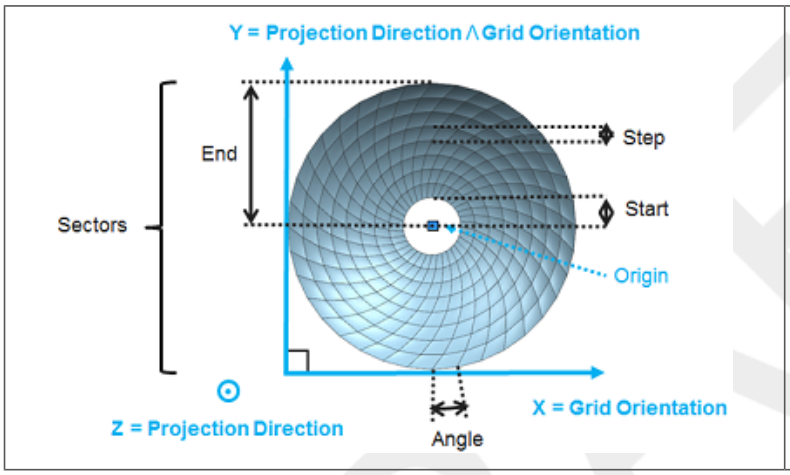
- **None**

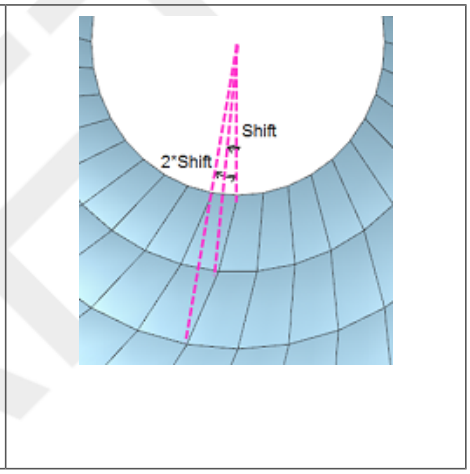


- **Radial**



- **Circular**





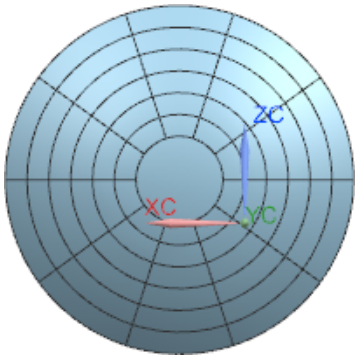
Grid Definition

- **Start**: Interior radius of the feature (mm).
- **End**: Exterior radius of the feature (mm).
- **Step**: Radial length of the elements (mm).
- **Sectors**: Number of angular subdivisions of the feature.
- **Angle**: Angle made by the sectors (deg).

Circular Edges

The **Circular edges** option allows you to define **Freeform** beam elements with circular edges. This option allows you to create a Fresnel lens.

DRAFT



Related information

Creating an Optical Lens on page 546
This page shows how to create optical lenses.

Beams on page 568
This section gathers all beam types available when designing an optical lens.

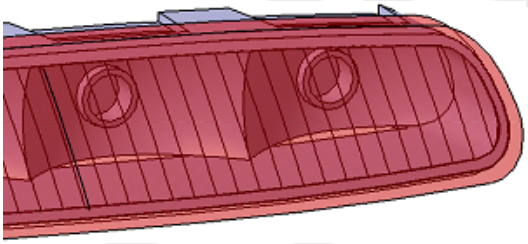
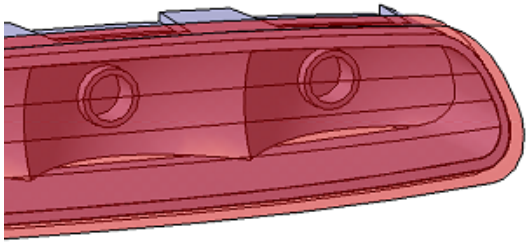
13.5.6.4.3. Stripes

The grid allows you to determine the lens' distribution onto the support.

> **Note:** **Stripes** grid is only compatible with the Flute beam type.

Style curves define the stripes on the optical lens.

The shape of the stripes, their number and their repartition over the support depend on the style curves selected for definition.

	
Lens with vertical stripes	Lens with horizontal stripes

Axis System

The grid axis system is used to define the way that the style curves are projected onto the support.

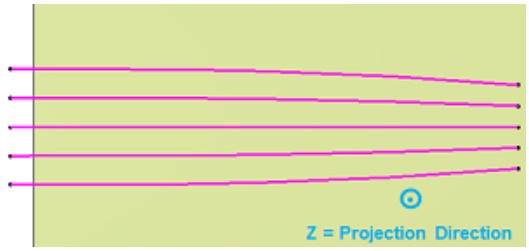
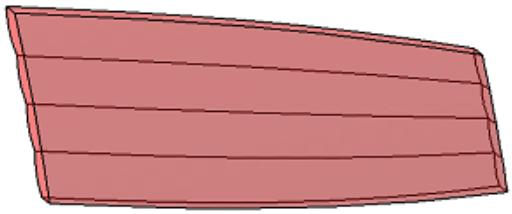
The axis system origin is optional. It is, by default, inherited from the source.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

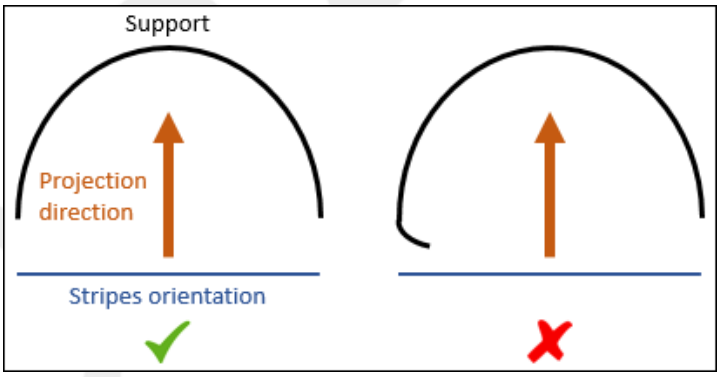
Note: If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

Note: This axis system has an impact on some parameters of the flutes beams:

- Start angle/End angle.
- Flutes' concavity/convexity.

	
Support and style curves viewed from the plane normal to Projection direction (Z axis)	Result of the feature when stripes are generated on the support according to style curves definition.

When using a curved support, use a support that does not close in on itself (see figure below). Otherwise, the stripes will not be projected correctly.

	
Support ensuring good construction	Incorrect curved support

Definition

Styles curves are selected to demarcate the stripes.

These curves can be lines, splines or sketches. The stripes curves are not necessarily continuous in tangency.

Note:

- An incomplete style curve is enough to provoke a null result for the whole feature.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- To avoid construction issues, always use support surfaces larger than style curves.
- Stripes do not work with large curve length variation. To limit the size of the last stripes, we recommend you to cut the support.

Related information

Creating an Optical Lens on page 546
This page shows how to create optical lenses.

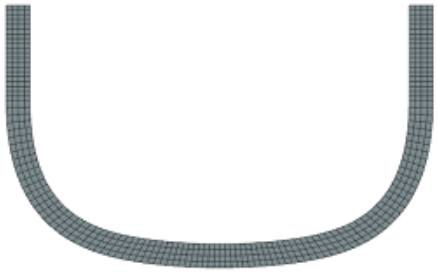
Beams on page 568
This section gathers all beam types available when designing an optical lens.

13.5.6.4.4. Freestyle

The grid allows you to determine the lens' distribution onto the support.

Note: **Freestyle** grid is not compatible with the **Flute** beam type.

Two sets of curves are used to delimit the elements of the lens. These curves can give either a rectangular-based or a circular-based pattern.

	
Lens with rectangular-based freestyle grid	Lens with circular-based freestyle grid

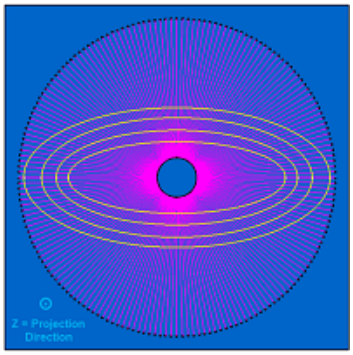
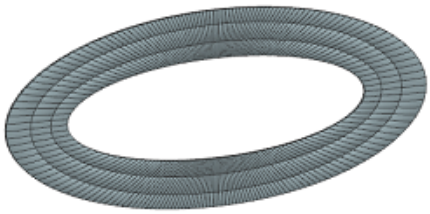
Axis System

The grid axis system is used to define the way that the sets of curves are projected onto the support.

The axis system origin is optional. It is, by default, inherited from the source.

Note: If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

DRAFT

	
Support and the two sets of curves viewed from the plane normal to Projection Direction	Result of the feature generated with the support and the two sets of curves of the picture on the left

Definition

Two sets of curves (**X curves** and **Y curves**) are used to define the freestyle grid.

These curves can be lines, splines or sketches.

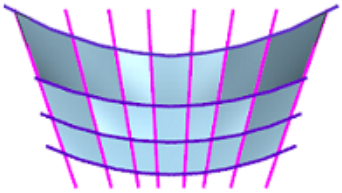
> **Note:** The curves are not necessarily continuous in tangency.

Each set of curves has to follow some rules for the feature to be built properly :

- Each curve of a set cannot intersect any other curve from this set.
- Each curve of a set cannot intersect several times any other curve.
- All the curves of a set has to intersect all the curves of the other set.
- All the curves of a set has to be closed if one curve of this set is closed.

X curves are dedicated to vertical curves and are defined along Y grid axis.

Y curves are dedicated to horizontal curves and are defined along X grid axis.



X curves in magenta define the curves along Y grid and Y curves in purple define the curves along X Grid

Related information

Creating an Optical Lens on page 546
This page shows how to create optical lenses.

Beams on page 568
This section gathers all beam types available when designing an optical lens.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

13.5.6.4.5. Honeycomb

The grid determines the size and distribution of the lenses over the support.

By default, elements are distributed onto the support according to a rectangular pattern. This distribution can be customized as well as the shape and size of the lens itself.

Axis System

An axis system is required to define the element's orientation and projection on the support.

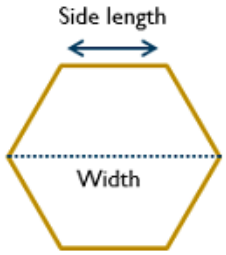
This axis system is, by default, inherited from the source definition.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

Definition

From the definition section you can:

- Define how the elements are going to be distributed on the support by setting start and end values for X and Y axes.
- Apply a rotation to the lenses (the X axis is taken as a reference to apply the rotation to the elements).
- Adjust the dimensions of the lenses by adjusting either their width or side length.



From the specification tree, under the feature node, more parameters are available to customize the pattern distribution:

- **X Start**: Start value for the feature size along **X Grid** (mm).
- **X End**: End value for the feature size along **X Grid** (mm).
- **Y Start**: Start value for the feature size along **Y Grid** (mm).
- **Y End**: End value for the feature size along **Y Grid** (mm).
- **X Count**: Number of elements along **X Grid**.
- **Y Count**: Number of elements along **Y Grid**.
- **Rotation**: Rotation of elements along **X** axis (deg).
- **Side Length**: Size of the hexagons' side (mm).
- **Width**: Hexagons' width (mm).

Related information

Creating an Optical Lens on page 546
This page shows how to create optical lenses.

Beams on page 568
This section gathers all beam types available when designing an optical lens.

13.5.6.5. Manufacturing

The **Manufacturing** or **Drafting** option allows you to consider and anticipate mechanical constraints coming from the lenses' manufacture.

> **Note:** This option is not available for the **Stripes** grid type and is compatible with a circular grid only if the **Shift** type is **Radial**.

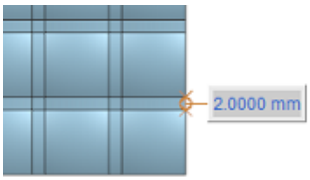
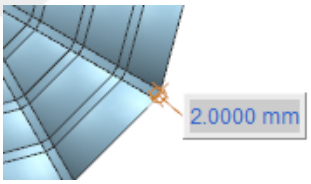
Lenses generated without drafting induce a manufacturing issue when they are removed from their mold.

Activating the **Drafting** allows you to take into account mechanical constraints and ensure an accurate manufacture.

> **Note:** Drafting reduces the facets' size. This operation is automatically taken into account by the algorithm generating the elements. As a consequence, the photometry is kept during the drafting operation no matter how high the drafting value is.

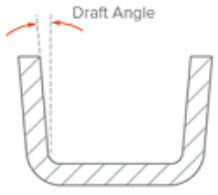
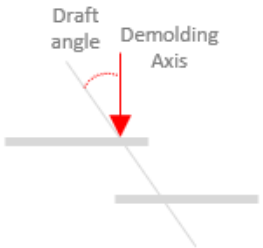
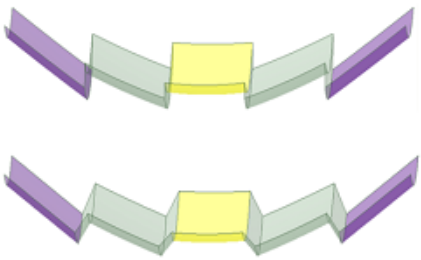
Drafting can be done by length or by angle:

- When selecting **Draft length**, the value to define determines the size of the sewing surface created between two adjacent elements.

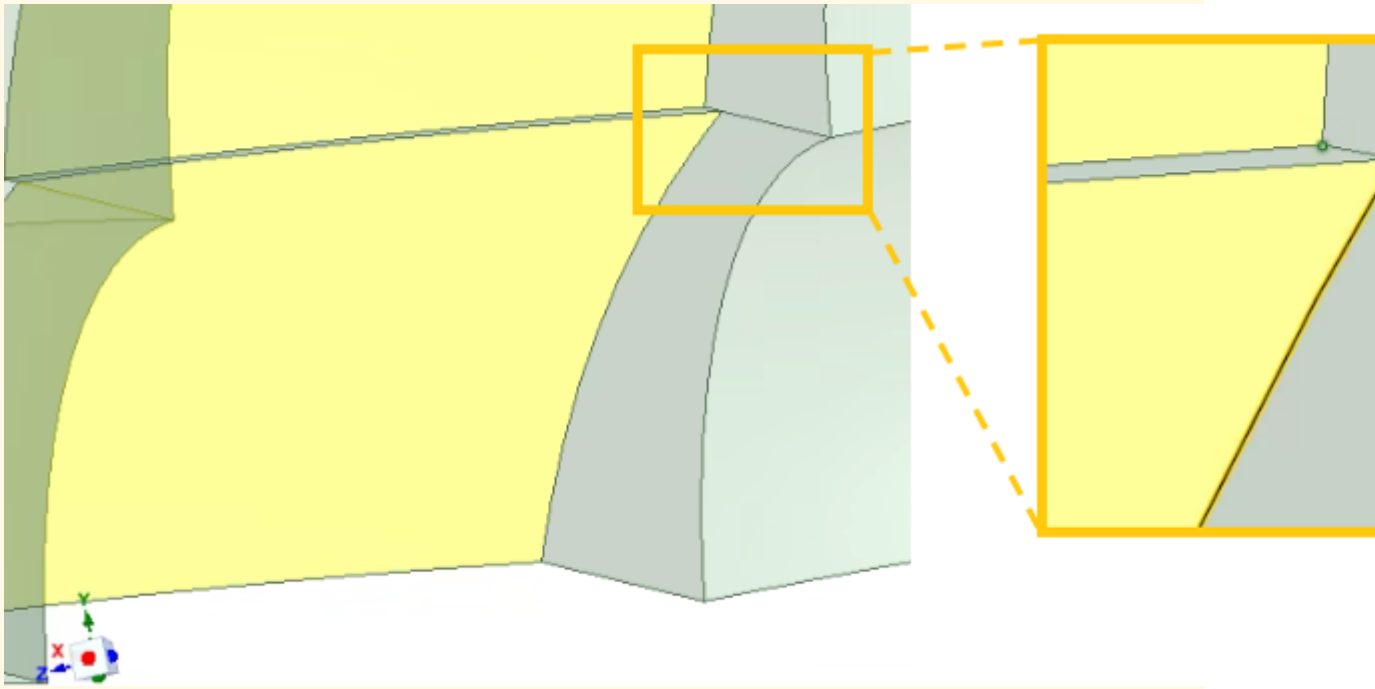
	
Rectangular Optical lens with a drafting of 2mm	Circular Optical lens with a drafting of 2mm

- When selecting **Draft angle**, the value to define determines the angle to create between the demolding axis and the facet.

> **Note:** Draft Angle is not supported if you try to apply a draft angle on 3 groups of one or more facets that share a same vertex.

		
Draft angle	Demolding axis	Impact of drafting angle on an optical surface

Note: The **Draft angle**Draft Angle is not supported when pillows/faces have edges in common connected to vertices that are at different levels, as seen in the following example:



In this case, we recommend you to use the **Draft length**.

Related information

Creating an Optical Lens on page 546
This page shows how to create optical lenses.

13.5.6.6. Beams

This section gathers all beam types available when designing an optical lens.

13.5.6.6.1. Radii

The Radii lens is a spherical lens used to ideally diffuse light.

Pillow or **Radii** lens is a spherical shaped lens that transmits and spreads light. Light goes through the lens and illuminates objects placed in front of it.

With the pillow lens, a definition of certain parameters is required:

- You need to define the **Radius** of the lens.
- You need to select an **Orientation**: concave or convex.

A concave lens has a surface that is curved inwards.

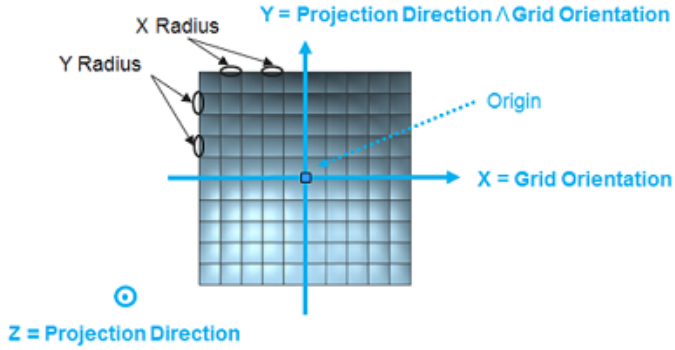
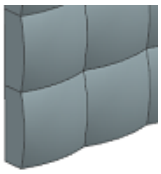
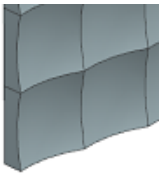
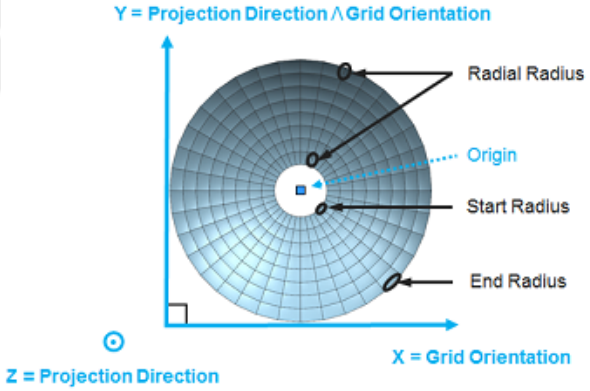
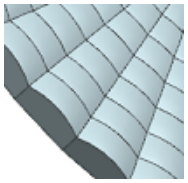
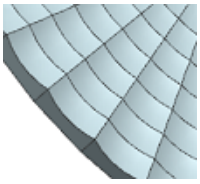
A convex lens has a surface that is curved outwards.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

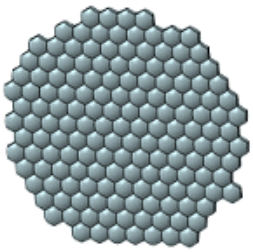
Published: 2023-10-23T14:08:14.04-04:00

Note: For each element, the convexity/concavity is defined with regards to the source.

Radii is available for Rectangular, Circular, Freestyle and Honeycomb lens types.

<table>
<tr><td rowspan="4">Rectangular</td><td colspan="2">X radius: Radius of the lens along X grid axis.
Y radius: Radius of the lens along Y grid axis.</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Convex Lens</td><td>Concave Lens</td></tr>
<tr><td rowspan="4">Circular</td><td colspan="2">Start radius: Smallest value of the radius of curvature of the elements along the transverse axis.
End radius: Highest value of the radius of curvature of the elements along the transverse axis.
Radial radius: Radius of the elements along the radial axis.</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Convex Lens</td><td>Concave Lens</td></tr>
</table>

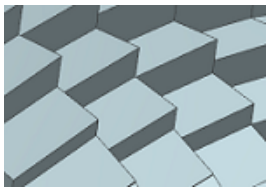
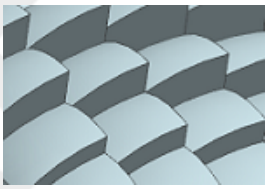
DRAFT

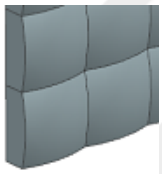
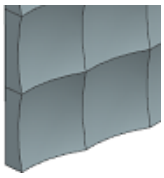
Freestyle	**Freestyle Rectangular**: Parameters are the same as the ones used for the rectangular type. **Freestyle Circular**: Parameters are the same as the ones used for circular type.	
Honeycomb	Radius of the lens	

13.5.6.6.2. Prism

The Prism or prismatic lens spreads and redirects light in a specific direction.

Prismatic lenses are shaped by specifying target points or directions. Light passes through the prisms and is redirected to illuminate the defined area/direction.

	
Flat prisms - No spreading applied	Curved prisms - Spreading applied

	
Influence of the Convex/Concave parameter - Convex elements on the left/Concave elements on the right.	

The parameters to set depend on the Target type selected in the **General** tab.

Illuminance	• X position: Coordinate of the point targeted by the prism along X target axis. • Y position: Coordinate of the point targeted by the prism along Y target axis. • X radius: Radius of curvature applied on the prism along X grid axis. • Y radius: Radius of curvature applied on the prism along Y grid axis.
Intensity	• X Angle: Angle defining the direction targeted by the prism along X Target axis. • Y Angle: Angle defining the direction targeted by the prism along Y Target axis. • X Radius: Radius of curvature applied on the prism along X Grid axis. • Y Radius: Radius of curvature applied on the prism along Y Grid axis.

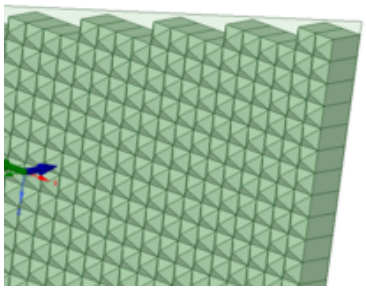
DRAFT

Note: Setting Radius to 0 gives a flat shape (meaning no spread).

Note: Radius must be higher than 2*grid step value.

13.5.6.6.3. Pyramid

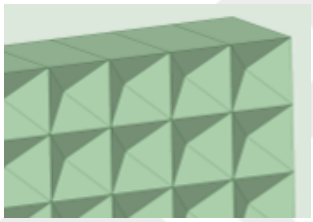
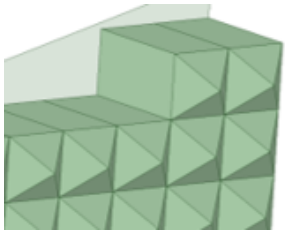
This lens type allows you to generate pyramid-shaped lenses.



The pyramids are shaped based on their heights.

The **Height** refers to the distance between the base of the pyramid and its top, along the direction that is normal to the support.

Pyramids can be concave or convex.

	
Concave	Convex

13.5.6.6.4. Freeform

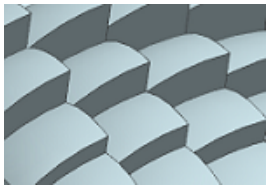
The Freeform lens is shaped according to target points or directions.

Note: This beam type is not available if you defined the Inner support option.

The freeform lens is generated to illuminate a desired area. You define this area of interest and the lenses generated accordingly.

The elements of this lens basically look like prisms having a pillow on their tops.

DRAFT



Contrary to the prism beam type whose curvature is defined with a constant radius, here the curvature is the result of an optimization minimizing the beam pattern size With this beam type all the rays goes in the targeted points or directions while with the basic prism beam type the matching is made sure only for one prism point.

Intensity	• X angle: Angle defining the direction targeted by the prism along X Target axis. • Y angle: Angle defining the direction targeted by the prism along Y Target axis.
Illuminance	• X position: Coordinate of the point targeted by the prism along X Target axis. • Y position: Coordinate of the point targeted by the prism along Y Target axis.

13.5.6.6.5. Flute

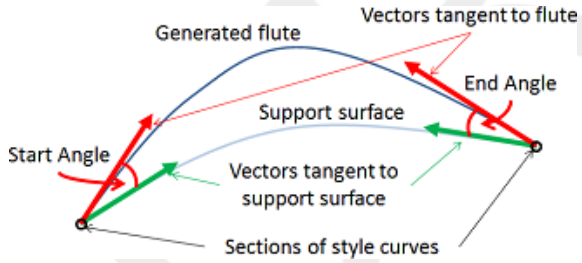
Flute lenses are shaped by the geometrical angle between tangents of the support.

Flute lenses are shaped by specifying the geometrical angles between tangents of the support and of the optical surface.

By adjusting the flutes curvature, you can drive the angular spread of the beam.

You can drive the flute horizontal spread by using vertical flutes and reciprocally.

- **Start angle**: Angle between the vector tangent to the support and the vector tangent to the Flute.
- **End angle**: Angle between the vector tangent to the support and the vector tangent to the Flute.
- **Orientation**: the lenses can be defined as concave or convex.



Note:

- Flutes can be made of several parts when the support has holes in it or an irregular shape.
- In some cases, stripes (usually the most external stripes of a group) can present a curvature inversion.

13.5.6.6.6. Reflex reflector

The Reflex reflector is a lens meant to reflect light.

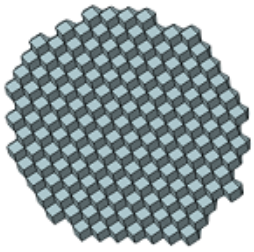
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Note: This beam type is only compatible with the Honeycomb lens.

Retroreflectors are composed of small corner reflectors (truncated cubes) that redirect light back to its source.

Reflex reflector lenses are designed to reflect light in the same direction as incident light.

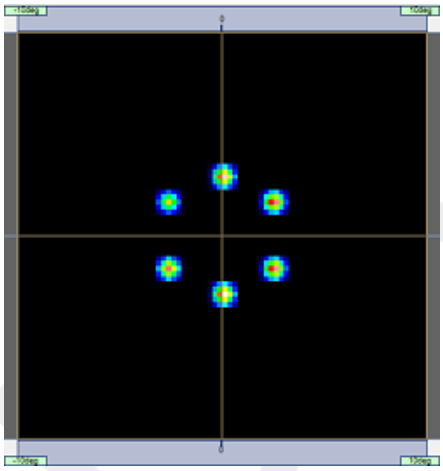
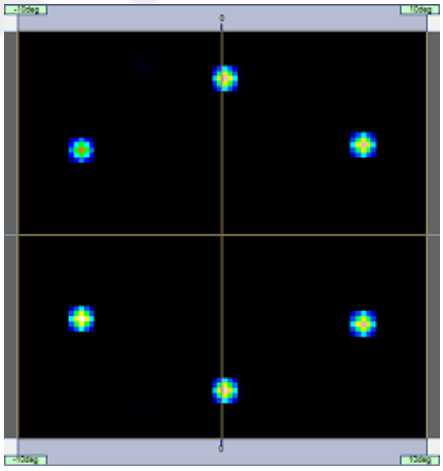
Figure 73. Honeycomb lens built with reflex reflector beam type



With this beam type, no further definition is required. But you can:

- adjust the offset of the group from the intermediary surface (thanks to the **Shift** parameter).
- adjust the angular spacing between the input and reflected ray (thanks to the Angle parameter).

  The Angle parameter allows you to control the cubes' corners flattening. This adjustment impacts the spacing/spread of the reflected points. The 6 reflected points correspond to the 6 retro-reflections on the corners of the cube.

	
3 degrees angular spread	8 degrees angular spread

Note: According to ECE R3 regulation on retro-reflecting devices, available on the Ansys Library, the measurements are made around the direction of the source within 20 arc minutes (0.33˚) and 1˚30 arc minutes (1.5˚).

13.5.6.6.7. Step Pillow

With the Step Pillow beam type, elements are shaped by specifying the area where the light should be sent in.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

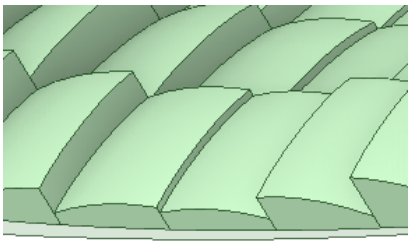
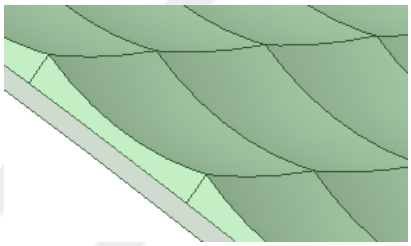
The **Step Pillow** beam type is only available when the **Orientation type** is set to **Outer support**.

Orientation

Concave: A concave surface is curved inwards.

Convex: A convex surface is curved outwards.

> **Note:** If **X start** > **X end** or **Y start** > **Y end**, a concave element becomes convex and a convex element becomes concave respectively on X or Y axis.

	
Convex parameter influence	Concave parameter influence

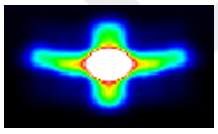
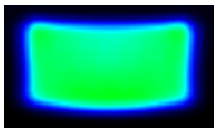
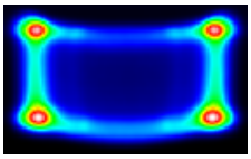
Boundaries

The boundaries define the area on the target where the light should be sent in.

- **X start**: Left boundary on the target.
- **X end**: Right boundary on the target.
- **Y start**: Bottom boundary on the target.
- **Y end**: Top boundary on the target.

Spread

X spread and **Y spread**: Ratio driving the beam spread on the target along X and Y Target axes.

0% < Ratio < 100%	100%	100% < Ratio < 200%
Light distribution inside the specification area tends to concentrate toward the center.	Light distribution inside the specification area tends to be uniform.	Light distribution inside the specification area tends to concentrate toward the edges.
		

Shift

Shift define additional thickness applied to each element. The thickness set here adds up to the **Thickness** set in the **General** section.

DRAFT

13.5.7. Display Properties

The Display Properties allow to customize the information displayed in the feature viewer.

13.5.7.1. Understanding Display Properties

The viewer assist the design process by helping to understand the feature behavior.

> **Note:** X and Y target axes used to display values in the viewer are defined in the target.

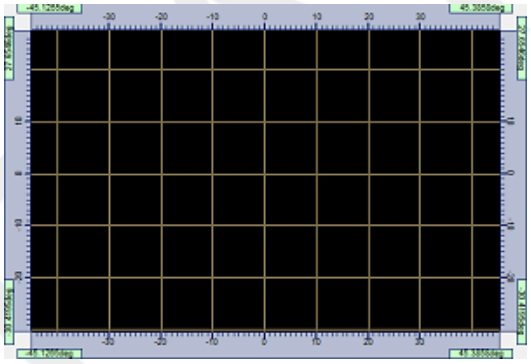
> **Note:** The viewer does not support the Reflex Reflector beam type.

The viewer gives information about the feature (lens) behavior and characteristics.

In the viewer, different types of informations are displayed according to your configuration of the optical lens.

Several types of elements can be displayed.

Grid



Grid with X Step = 10deg and Y Step = 10deg

A grid is displayed in the viewer and gives information about the size of the beam.

According to the definition made in the target, the grid is defined differently:

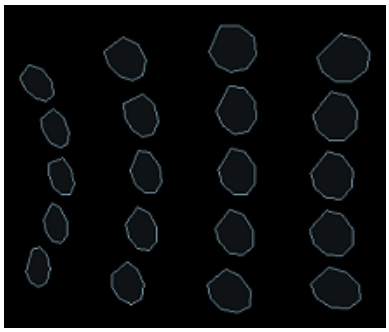
- If you selected the intensity target type, the grid is defined angularly and the step is expressed in degrees.
- If you selected the illuminance target type, the step is expressed in mm.

Source Images

Source Images is only available with the Extended source type.

The source images give information about the lens behavior inside the target. Their color depend on the group's color of the selected element.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



Source Images with U Samples = 5 and V Samples = 4

U Samples indicate the number of source images along the first parametric surface direction.

V Samples indicate the number of source images along the second parametric surface direction.

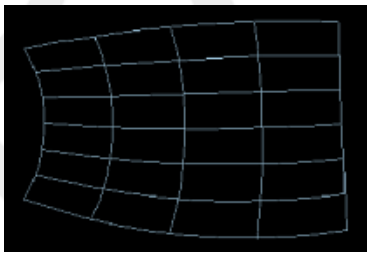
The external face of the elements is discretized according to **U Samples** and **V Samples** giving particular points. The image of the extended source is then calculated for each of these particular points using the Snell's law.

The source images are approximated because of the meshing and the extended source convex hull considered for the calculation.

Beam Pattern

> **Note:** Beam patterns are not displayed when using the inner surface mode with Radii elements. The property must be checked in the feature viewer to be able to display beam patterns.

A beam pattern is depicted as a grid (a network of lines) and gives information about the beam's shape.



Beam pattern with U Samples = 7 and V Samples = 5

U Samples indicate the number of isoparametrics along the first parametric surface direction.

V Samples indicate the number of isoparametrics along the second parametric surface direction.

The external face of the element is discretized according to **U Samples** and **V Samples** giving a particular network of lines on this element. The reflection of this network of lines is carried out using the Snell's law from the source point.

The calculation of the beam is calculated considering a punctual source. If an extended source is used, the barycenter of the extended source is considered as the punctual source.

DRAFT

Photometry

> **Note:** The Photometry is in BETA mode for the current release.

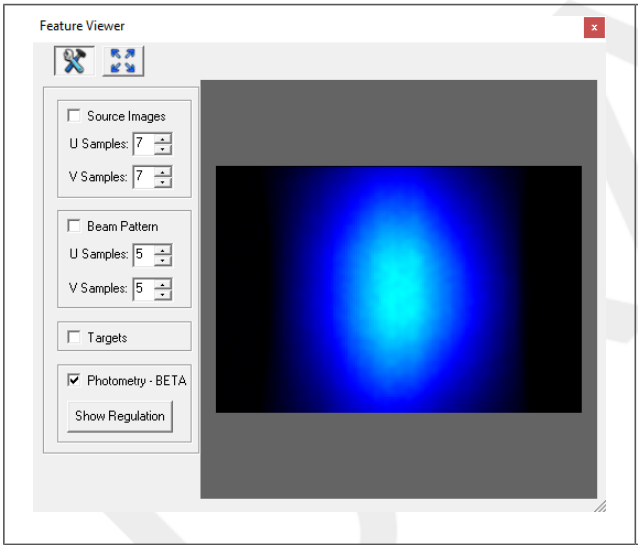
Description

The Photometry tool is an integrated simulation that allows you to have a quick preview and result of the selected element(s) of the feature.

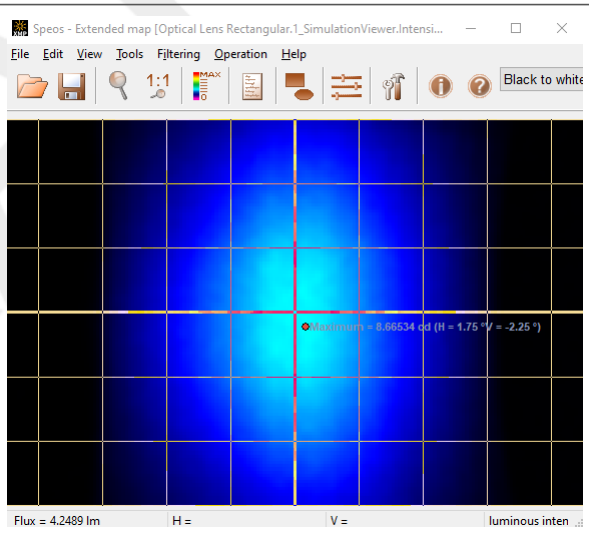
When activating the **Photometry**, the simulation is automatically launched using the GPU Compute functionality and generates a photometric preview displayed in the feature viewer, a HTML report and a XMP file in a folder named as **FeatureName_InteractivePhotometry** located in the Speos Output Files folder.

> **Note:** Each time you modify the selection of the element(s), the simulation is launched again, which overwrites the previous HTML report and XMP file.

Therefore, the Photometry tool allows you to directly iterate during the design process of the Optical Part Design feature in order to reach the regulation standards required, without having to run a Speos simulation that can take times to generate the output results.



Photometry of a group of elements from the feature viewer



XMP result of the group of elements selected

Click **Show Regulations** to open the HTML report generated.

Simulation Parameters

The Photometry tool simulation needs some information in order to run correctly. Therefore, the simulation considers the following parameters:

- **Source Parameters**
  - The simulation considers the **Extended** source type of the feature definition with a lambertian emission

> **Note:** The **Punctual** and **Directional** source types are not compatible.

- The simulation considers the **Flux** of the feature definition (100 lm by default).
- Optical Properties of the extended source: the simulation considers by default a **Surface Optical Property** as a **Mirror** set to **0%** and a **Volume Optical Property** set to **None**.

- **Geometry Parameters**

  The simulation considers the Optical Part Design feature geometry with the Optical Properties you applied manually.

  If you have not set any Optical Properties, the simulation considers by default the **Refractive index** set in the feature definition.

- **Sensor Parameters**

  The simulation considers the **Intensity** or**Illuminance** Target type of the feature definition.

  The size of the sensor used corresponds to the defined **Beam Pattern** parameters of the feature viewer.

- **Regulation Standards**

  The simulation considers the **XML template** selected in the Target definition.

  The XML template corresponds to the standard that you want the feature to pass.

> **Note:** You can find existing XML templates of regulation standards in the Ansys Optical Library.

Related tasks

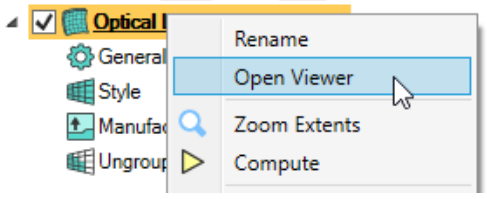
Adjusting Display Properties on page 578
The Display Properties allows you to customize the viewer and the information displayed in it.

13.5.7.2. Adjusting Display Properties

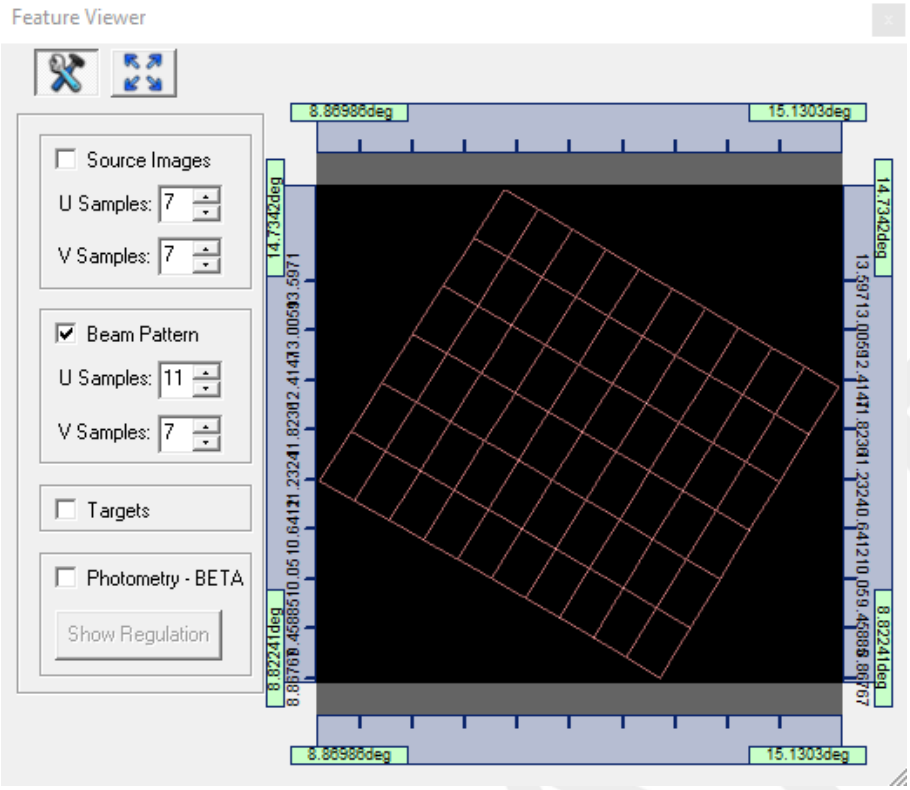
The Display Properties allows you to customize the viewer and the information displayed in it.

To adjust Display Properties:

1. From the **Design** panel, right-click the optical lens feature and click **Open viewer**.



2. In the 3D view, click a facet.
   Informations are displayed in the viewer.
3. From the **Feature Viewer**, click  to adjust the display properties.



4. In **Source Images** and **Beam Pattern**, adjust the number of samples for U and V axes.
5. Check **Photometry - BETA** to run an integrated simulation that generates a photometric preview displayed in the feature viewer, a HTML report and a XMP file of the selected element(s) of the feature.

   This will help you verify if your current design passes the regulation standards that you selected in the XML template parameter of the feature definition.

   Fore more information, refer to Understanding Display Properties.
6. Click **Show Regulation** if you want to open the HTML report of the simulation.
7. To access grid parameters, right-click in the viewer and click **Grid Parameters**.
8. Click **User** and define the step you want for the gridline for U and V axes.

> **Note:** According to the target type selected, the step is expressed in degrees (intensity type) or in mm (illuminance type).

The display properties are set and the modifications are taken into account in the viewer.

Related concepts

Understanding Display Properties on page 575
The viewer assist the design process by helping to understand the feature behavior.

13.5.8. Interactive Preview

The interactive preview allows you to dynamically visualize different construction elements in the 3D view to help you modify and design the Optical Part Design feature as needed.

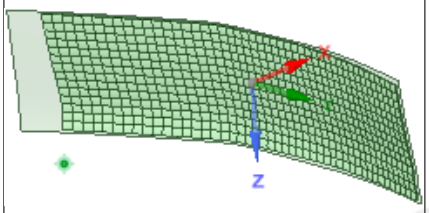
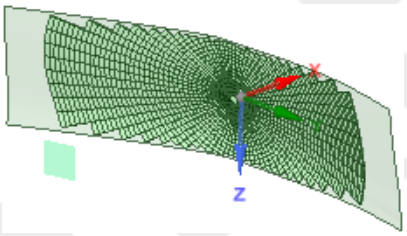
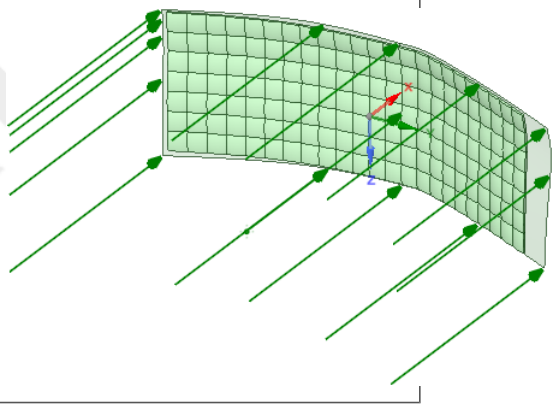
13.5.8.1. Understanding the Interactive Preview

The Interactive Preview allows you to visualize the original grid of the Optical Lens, the support used, its source, its projected grid on the support.

Source

All source types can be displayed:

- Point Source: point highlighted in the 3D view
- Extended Source: emitting surface highlighted in the 3D view
- Directional Source: displays a pattern of axis on the lens that represent the directional source

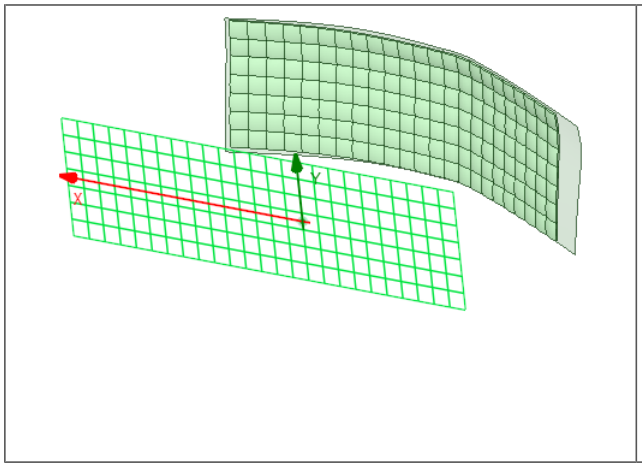
Example of point source interactive preview	Example of extended source interactive preview	Example of directional source interactive preview
		

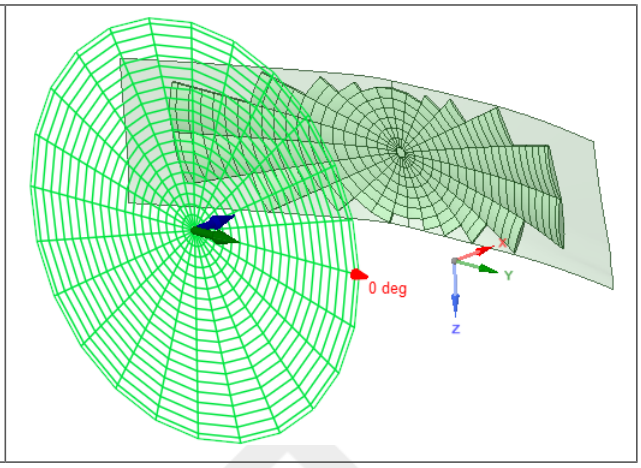
Grid

Only rectangular and circular grids can be previewed in the 3D view.

Example of rectangular grid interactive preview	Example of circular grid interactive preview

DRAFT



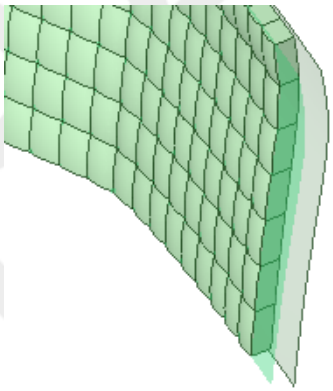


Support

The offseted support is displayed.

Created groups are not considered in the preview.

> **Note:** When the offseted support computation fails, the support surface is highlighted in red color in the 3D view.

Example of support interactive preview


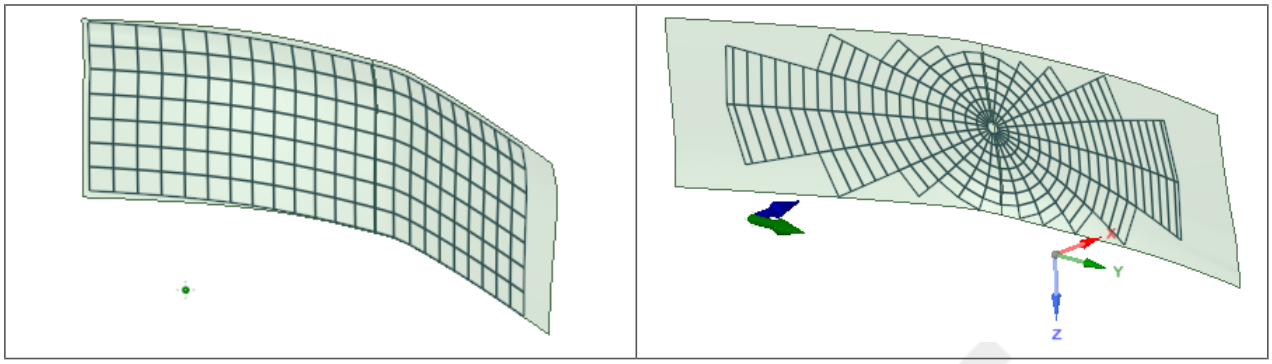
Projected Grid

All projected grid types can be previewed in the 3D view.

Created groups are not considered in the preview.

Example of rectangular projected grid interactive preview	Example of circular projected grid interactive preview

DRAFT

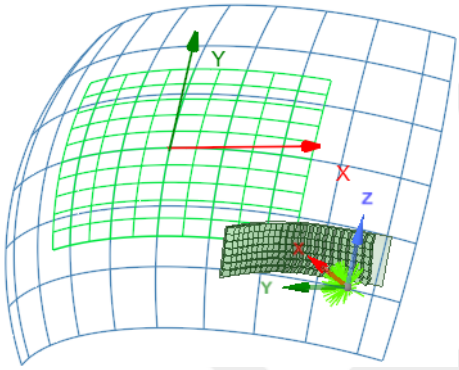
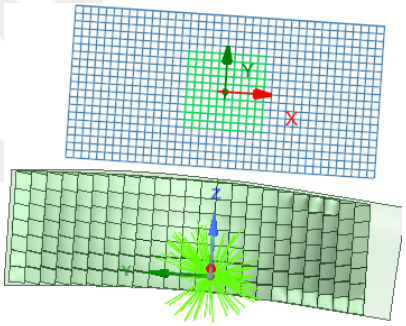


Target

The preview of the target is useful to correctly define your target and avoid useless steps of try and error.

Example: with the Step Pillow beam type, previewing the target will helps you design the sensor correctly.

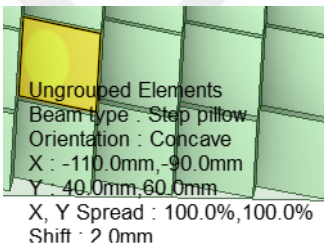
Thin X and Y axes represent the axis system of the target.

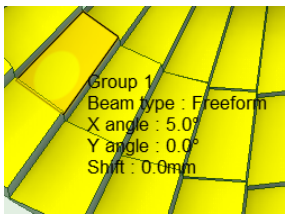
Example of Optical Lens with Intensity Target	Example of Optical Lens with Illuminance Target
	

Facet Information

The Facet Information corresponds to the beam parameters of the facet you are hovering over. Facet Information is useful when your Optical Part Design feature is using multiple groups or an Excel file.

The Facet Information is always displayed when hovering over a facet.





Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

13.5.8.2. Displaying the Interactive Preview Parameters

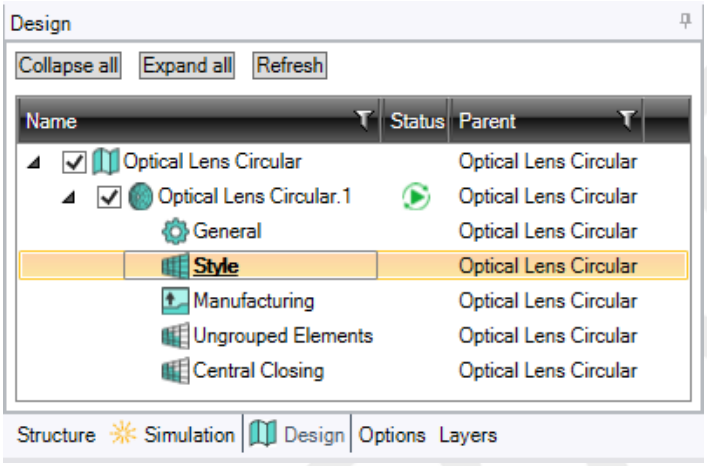
The following procedure helps you display the preview of parameters of an Optical Lens to give a dynamic overview upon feature modification.

> **Important:** The Interactive Preview can be time-consuming, especially for the Projected grid preview in case of a huge number of grid intersections.

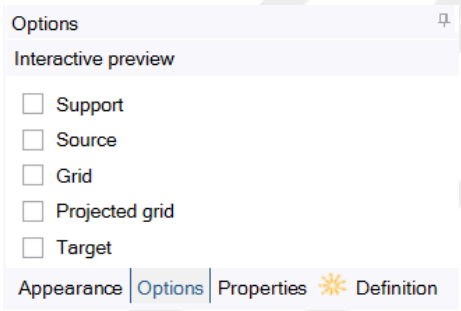
To display a parameter's interactive preview:

An Optical Lens must already be created.

1. In the **Design** panel, open one of the sub-definition group of the feature.



2. Open the **Options** panel.



3. In the **Options** panel, check the parameters' interactive previews you want to display in the 3D view.
   - Source
   - Support
   - Grid (only rectangular and circular grids can be previewed)
   - Projected Grid
   - Target

The parameters' previews are displayed in the 3D view and change dynamically upon feature modifications.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

13.6. Light Guide

Light Guides are pipes guiding/transporting the light.

13.6.1. Light Guide Overview

Light Guides are thin pipes made of transparent material like plastic or glass meant to guide light in a specific direction.

A Light Guide is composed of a body (a pipe) and reflecting elements (prisms). Light travels through the pipe thanks to successive total internal reflections and a part of this light is extracted through reflection on the prisms.

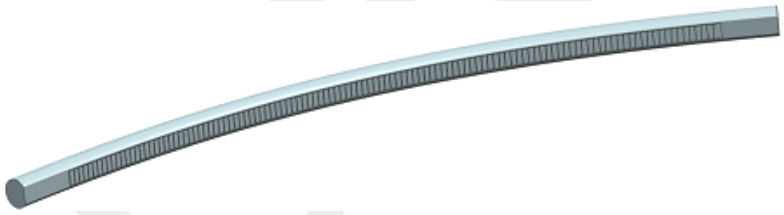
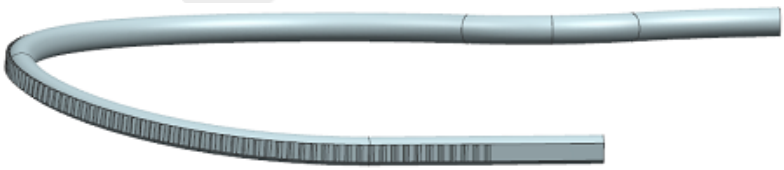
Light Guides are optical components that are highly efficient in transmitting light. For this reason, they have various possible applications.

They are commonly used for interior design, accent lighting, backlighting, exterior lighting or tail lamps.

Figure 74. Example of light guide on a dashboard



In Speos, the light guide can be linear or curved and the prisms' position, size and distribution can be customized.

	
3D view of a linear Light Guide	3D view of a linear Light Guide's prisms
	
3D view of a circular Light Guide	3D view of a circular Light Guide's prisms

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

13.6.2. Creating a Light Guide

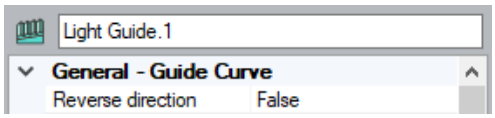
This page shows how to create a Light Guide from any guide curve or shape.

To create a Light Guide:

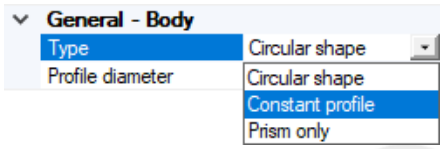
A curve outlining the light guide length should already be sketched.

> **CAUTION:** The **Intersect** tools from the **Design** tab do not work with a created Light Guide. We then recommend you to avoid creating Light Guide tangent to other bodies.

1. From the **Design** tab, click **Light Guide** .



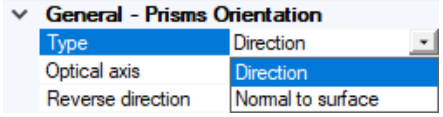
2. Click and select a curve as **Guide curve**.
   The guide curve corresponds to the curve along which the profile is swept to create the Light Guide.
3. From the **Body Type** drop-down list, select the Light Guide's profile:



   - Select **Circular shape** to create a light guide with a circular profile and set its profile diameter.
   - Select **Constant profile** to create a light guide using a profile and click
     to select a surface.
   - Select **Prism only** to create a light guide without body. This option is useful when wanting to use a custom light guide body.

     > **Note:** If the **Add** operation is used with this mode, the height of the prisms is defined to reach the end of the guide curve.

4. If you selected **Prism only** combined with the **Add** or **Hybrid** operation, you can define an **Extra body height**.
   Extra body height corresponds to body height added from the guide curve in the opposite direction of the prisms.
5. Define the prisms' orientation and construction mode:
   a) From the **Type** drop-down list, select how the prism should be oriented in relation to the light guide body:



   - Select **Direction** if you want all prisms to have the same orientation.
   - Select **Normal to surface** if you want prisms to potentially have different orientations.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

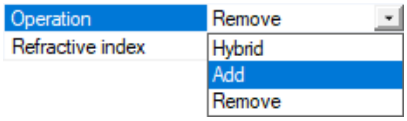
b) In the 3D view, click  and select a line to define the optical axis.

The optical axis is used to define the average orientation in which the light is going to be extracted from the light guide.

> **Note:** The optical axis must not be collinear with any tangent to the guide curve.

c) If you selected the **Normal to surface** type, click  and select a face to define the orientation of the prisms.

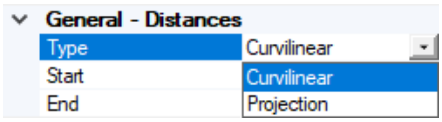
d) From the **Operation** drop-down list, select how you want the prisms to be generated on the light guide:



- Select **Add** if you want the prisms to be added to the body.
- Select**Remove** if you want the prisms to be removed from the body.
- Select **Hybrid** if you want prisms to be both removed/added based on prisms parameters.

e) Define the **Refractive index** of the light guide.

6. In **Distances**, from the **Type** drop-down list:



- Select **Curvilinear** to base the distances on the guide curve.
- Select **Projection** to define curvilinear distances based on the guide curve projected on a projection plane.

  Click  and in the 3D view, select a line to define the **Projection axis**.

  The projection plane is defined as normal to the projection axis.

7. In **Start** and **End** group boxes, define the size of the prism-free zones at the beginning and at the end of the guide curve.

> **Note:** Start value is always respected, however in come cases End value might not be respected because the last prism is not cut by this parameter. Prisms are created to respect the End value as much as possible.

Now, define the prism geometries.

13.6.3. Defining the Light Guide Prisms

This page shows how to define the light guide prisms' and prisms' milling construction mode and parameters.

To define the Light Guide Prisms and their Milling:

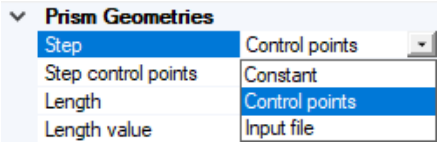
A Light Guide Body should already be defined.

DRAFT

Each parameter of the prism geometries and milling can be set as constant, variable or file based whatever the operation selected (Add, Remove, Hybrid). For more information on the parameters, refer to the Prism Parameters.

Make sure to apply consistent prism parameters values, otherwise you may generate prism with wrong shape. For instance, a milling radius larger than the prism size may generate wrong prism shape after applying the milling.

1. Before setting each parameter's value, define its construction mode:



   - Select **Constant** to exclude parameter variation along the guide. This mode ensures prisms parameters are constant along the guide curve.
   - Select **Control points** if you want to create parameter variation along the guide curve.
   - Select **Input file** to import a *.csv file defining the prisms repartition along the light guide body.

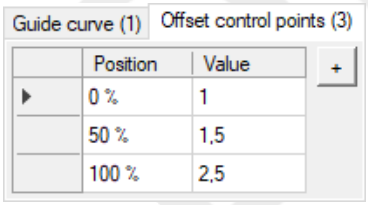
> **Note:** When using *.csv files, make sure the regional settings and number format of the workstation are correctly defined. A dot "." should be defined as **Decimal symbol**.

> **Tip:** From the **Design** panel, right-click the light guide feature and click **Export as CSV file** to export all prisms parameters. The file can then be modified and used as input for light guide definition.

   - Select **Automatic** if you want parameters to be automatically calculated according to other settings.

> **Note:** **Automatic** mode is only available for **Start Angle**, **End Angle** and **Length**.

2. Once the prism's construction mode is defined:
   - In case of a **Constant** mode, set the **Parameter value** that will be constant along the guide curve.
   - In case of a **Control points** mode:

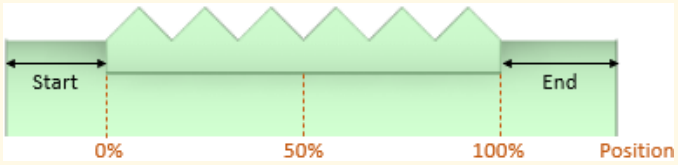


     a. Click

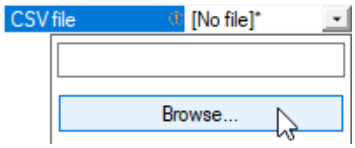
     

     to add as many control points as needed.
     b. Set their **Position** along the curve. The position is expressed in % (a position of 50 places the control point in the middle of the guide).
     c. Define the parameter's value for each control point.

> **Note:** The variation is measured on the curve minus the **Start** and **End**.



- In case of an **Input file** mode, browse and load the **CSV file**.



3. Click **Compute** to build the feature.

The Light Guide is created and appears in the 3D view.

Now you can define the Manufacturing parameters to take into account a correct removal from the mold.

Related reference

Light Guide Body Parameters on page 590
This page provides detailed information on the Light Guide body parameters.

Light Guide Prism Parameters on page 596
This page describes all the parameters to set when defining prisms.

Manufacturing Parameters on page 600
This page describes the parameters to create a light guide that can be manufactured.

13.6.4. Defining the Manufacturing Parameters

This page shows how to define the manufacturing parameters to create a light guide that can be manufactured.

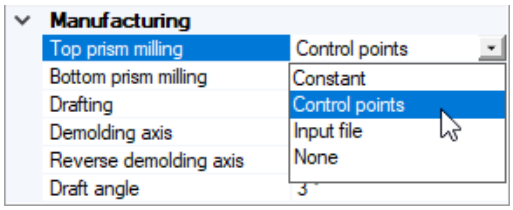
To define the Manufacturing parameters:

A Light Guide Body should already be defined.

The milling parameters can be set as constant, variable or file based whatever the operation selected (Add, Remove, Hybrid). For more information on the parameters, refer to the Manufacturing Parameters.

Make sure to apply consistent parameters values, otherwise you may generate prism with wrong shape. For instance, a milling radius larger than the prism size may generate wrong prism shape after applying the milling.

1. Before setting the **Top prism milling** and/or **Bottom prism milling** parameters value, define its construction mode:



DRAFT

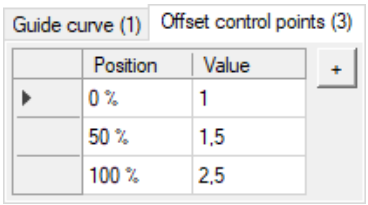
- Select **Constant** to exclude parameter variation along the guide. This mode ensures milling parameters are constant along the guide curve.
- Select **Control points** if you want to create parameter variation along the guide curve.
- Select **Input file** to import a *.csv file defining the prism repartition along the light guide body.

> **Note:** When using *.csv files, make sure the regional settings and number format of the workstation are correctly defined. A dot "." should be defined as **Decimal symbol**.

> **Tip:** From the **Design** panel, right-click the light guide feature and click **Export as CSV file** to export all prisms parameters. The file can then be modified and used as input for light guide definition.

- Select **None** if you do not want to apply a milling.

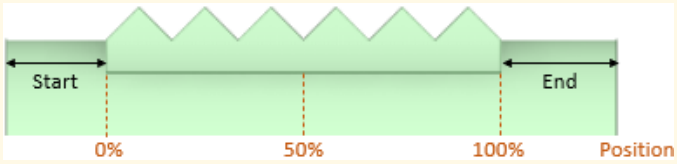
2. Once the milling's construction mode is defined:
   - In case of a **Constant** mode, set the **Parameter value** that will be constant along the guide curve.
   - In case of a **Control points** mode:



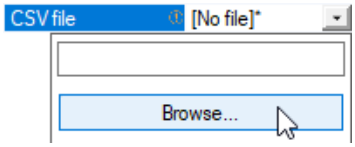
     a. Click

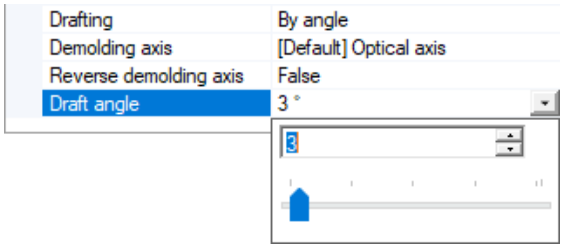
     to add as many control points as needed.
     b. Set their **Position** along the curve. The position is expressed in % (a position of 50 places the control point in the middle of the guide).
     c. Define the parameter's value for each control point.

> **Note:** The variation is measured on the curve minus the **Start** and **End**.
>
> 
>

   - In case of an **Input file** mode, browse and load the **CSV file**.



3. Set **Drafting** to **By angle** if you want to create an angle between the demolding axis and the prism to ensure a safe removal of the guide from the mold.



a) Select a **Demolding axis**.
b) Define a **Draft angle** that takes into account the manufacturing constraints.

> **Important:** The **Drafting** may sometimes be incorrectly applied. Make sure to check the **Drafting** after computing the Light Guide, and increase it if necessary.

4. Click **Compute** to build the feature.

The Light Guide is created and appears in the 3D view.

Related reference

Light Guide Body Parameters on page 590
This page provides detailed information on the Light Guide body parameters.

Light Guide Prism Parameters on page 596
This page describes all the parameters to set when defining prisms.

Manufacturing Parameters on page 600
This page describes the parameters to create a light guide that can be manufactured.

13.6.5. Light Guide Parameters

This section gives more information about the parameters to set when creating a Light Guide.

13.6.5.1. Light Guide Body Parameters

This page provides detailed information on the Light Guide body parameters.

Guide Curve

The **Guide Curve** represents the curve along which the profile is swept to create the Light Guide.

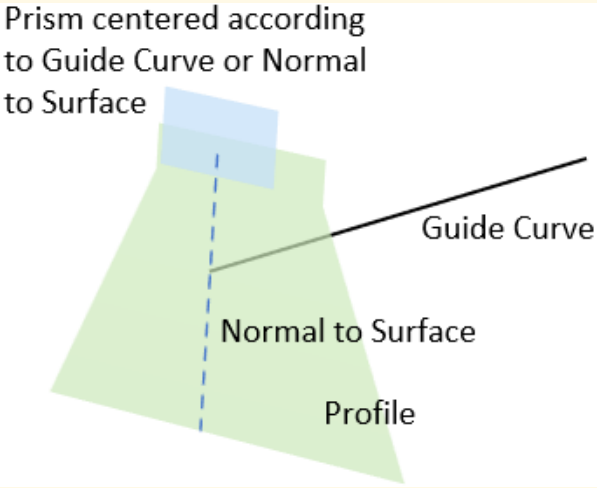
This curve is not necessarily continuous in tangency.

Body Type

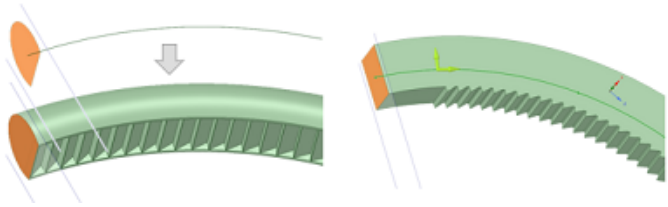
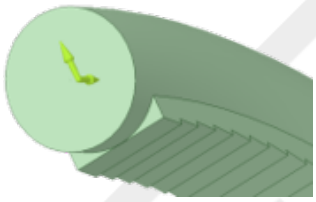
The body type allows you to determine what to base the body construction on (specific shape, curve or diameter).

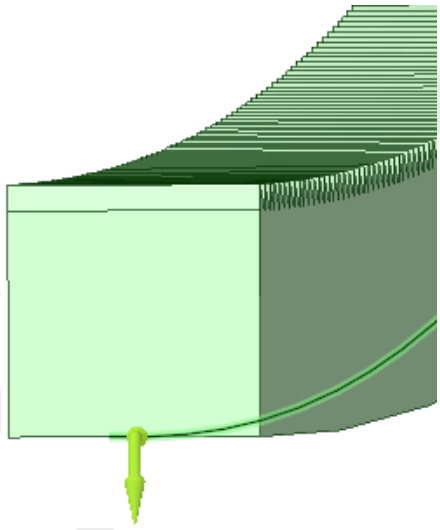
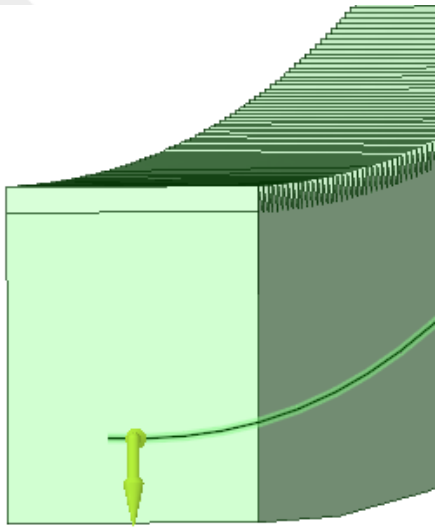
DRAFT

Note: Prisms are built according to the Guide Curve in their middle. The profile you select has no relation to the prisms position and may be shifted.



Prisms only	The prisms are created but not the Light Guide body. This mode is useful when wanting to use a custom light guide body.
	
	Add operation: the height of the prisms is defined to reach the guide curve. **Important:** Up to 2023 R1, a 0.1mm gap is applied automatically between the prisms and the Guide curve. From 2023 R2, the 0.1mm is no longer applied. 
	Remove operation: the height of the prisms is not reliable, it is only used for assembling. **CAUTION:** The visual rendering of a Prism only remove operation corresponds to the negative of the prisms. In other words, it displays the prisms part to be physically removed. 

Constant profile	The body is created using a profile defined by a planar surface and located at the start of the guide curve. Then the profile is swept along the guide curve to create the light guide body.
	
Circular shape	The body is created using a circular shaped profile. The profile diameter allows you to set the diameter of the profile.
	

Extra body height	Extra body height corresponds to body height added from the guide curve in the opposite direction of the prisms.	
	 Extra body height = 0mm	 Extra body height = 1mm

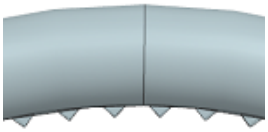
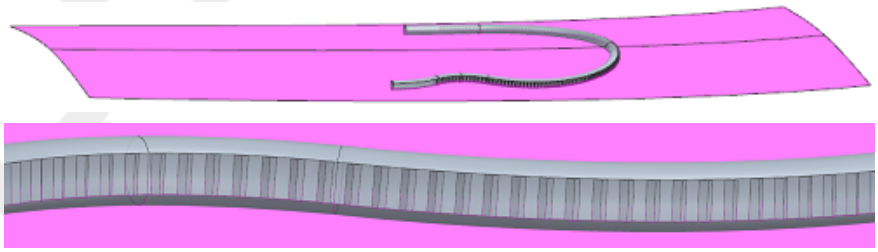
Prism Orientation

The prism orientation type allows you to determine how the prism should be oriented in relation to the light guide body.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Direction	All the prisms have the same orientation all along the guide curve. Light extracted through the prisms has the same direction. This type is particularly suited for linear light guides. **Figure 75. Linear light guide with optical axis in magenta**  **Figure 76. Prisms of the light guide** 

DRAFT

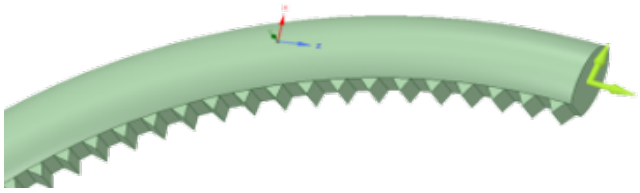
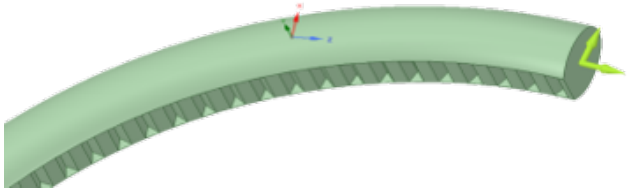
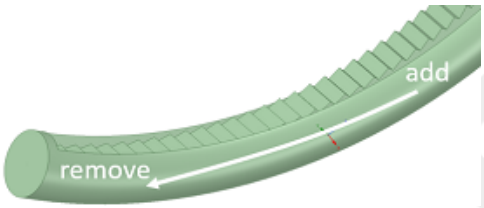
Normal to Surface	All the prisms have potentially a different orientation. Light extracted through the prisms has a different direction all along the guide curve. This type is suited for non-linear light guides. **Figure 77. Circular Light Guide for cup holder**  **Figure 78. Prisms of the light guide**  The position/orientation of the prisms is driven by two elements, a surface and the optical axis. • The optical axis provides the average orientation in which the light is going to be extracted from the light guide. This axis defines the orientation of the first prism and is used as input for other prisms. • The selected surface determines the prisms' orientation. To avoid construction issues, the whole guide curve needs to lay on the normal surface. The normal surface width should equal or exceed the light guide and the built prisms. To create a correct surface, we recommend to create a line at the start of the guide curve defining the optical axis. Then, use the **Pull** command to sweep this line all along the guide curve. **Figure 79. Light Guide surface input in pink - The position/orientation of the prisms is driven by the surface** 

Operation

The Operation corresponds to the prisms generation on the Light Guide.

Add/**Remove**/**Hybrid** operations allow to remove/add prisms to the light guide body.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Add	
Remove	
Hybrid	Hybrid operation is based on the prisms parameters to determine whether a prism is added or removed from the light guide. 

Distances

Mode

Curvilinear	The parameters **Start**, **End**, **Step** and **Length** (for add/remove operations) are curvilinear distances based on the guide curve.  • In **Direction** prism type, the distance are measured directly on the guide curve. • In **Normal to Surface** prism type, the distances are measured on an offset of the guide curve passing by the middle of the top edge of the prism.
Projection	The parameters **Start**, **End**, **Step** and **Length** (for add/remove operations) are defined as curvilinear distances based on the guide curve projected on a projection plane. The **Projection Axis** corresponds to a line that defines the projection plane (plane normal to Projection line). **Note:** With this model, you can obtain style effects as a constant prism length when the Light Guide is seen in a specific direction. 

DRAFT

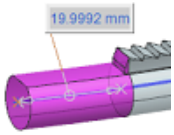
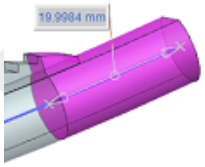
Start & End

Start and **End** define the size of the prism-free zones at the beginning and at the end of the guide curve.

The beginning of the guide curve is the extremity where the source is located.

> **Note: Start** value is always respected, however in come cases **End** value might not be respected because the last prism is not cut by this parameter. Prisms are created to respect the **End** value as much as possible.

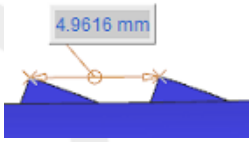
Half of the first prism is included in the **Start** distance. If **Start** is null, the first prism will start by its top at the beginning of the guide curve.

 Start	 End
Prism-free zones	

13.6.5.2. Light Guide Prism Parameters

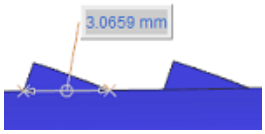
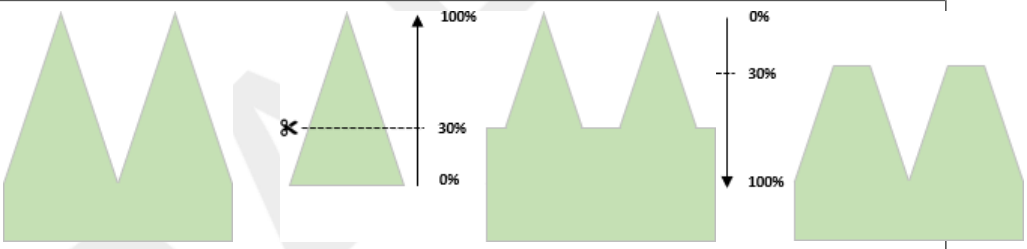
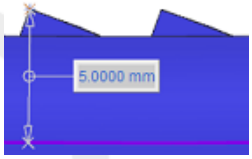
This page describes all the parameters to set when defining prisms.

> **Note:** Prism parameters vary depending on the operation selected.

Step (Add/Remove Operation)	**Step** corresponds to the spacing between the mid points of the top edges of two adjacent prisms.  > **CAUTION:** Make sure to define a Step value different from the Length value. Otherwise it may generate unwanted prism geometries.
Step (Hybrid Operation)	**Step**corresponds to the spacing between two adjacent prisms projected on the guide curve. > **CAUTION:** Make sure to define a Step value different from the Length value. Otherwise it may generate unwanted prism geometries.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

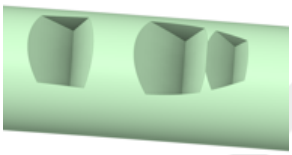
Published: 2023-10-23T14:08:14.04-04:00

Length (Add/Remove Operation)	**Length**corresponds to the length of the prisms.  If the offset is too low regarding the profile, a part of the prisms can be created inside the body. As a consequence, the length of the prisms can appear as not being taken into account. **CAUTION:** Make sure to define a Length value different from the Step value. Otherwise it may generate unwanted prism geometries.
Trimming ratio (Hybrid Operation)	The trimming parameters allows you to cut off the prism either from their peak or from their base: • **Bottom trimming** controls the prisms from their base. • **Peak trimming** controls the prisms from their peak.  Prisms without trimming Prisms trimmed from the bottom Prisms trimmed from the peak
Offset (Add/Remove Operation)	**Offset** corresponds to the distance between the guide curve and the middle point of the top edge of the prism.  Make sure to set appropriate values for the offset parameter: • With **Add**, if the offset is too low, the prisms are inside the body. • With **Remove**, no material is removed from the body if the offset is too high.
Offset (Hybrid Operation)	**Offset** corresponds to the distance between the guide curve and the bottom of prism (start angle side).

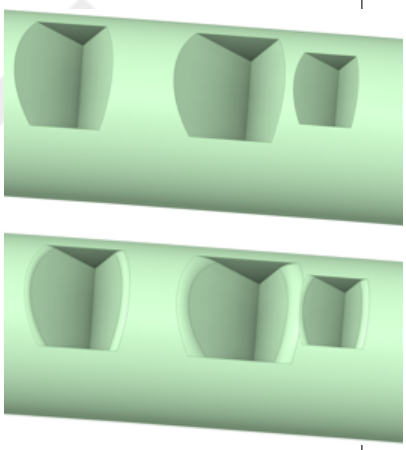
Width	**Width** corresponds to the width of the prisms. Width can be constant over the guide or can vary (Control Points or Input File). Width is always constant in Hybrid Operation. 



When the width varies, the width is progressive in Add Operation.



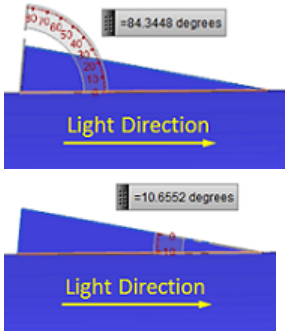
When the width varies, the width is progressive in Remove Operation only if at least two prisms touche themselves (without considering the milling).



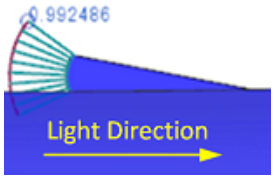
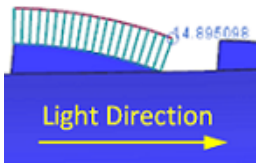
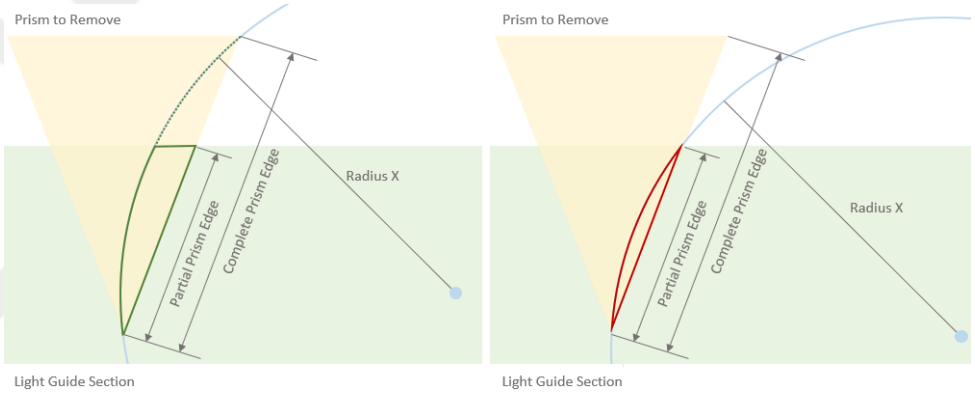
When the width varies, the width is not progressive in Remove Operation if prisms are all separated.

When the milling make two prisms touch each other, the progressive width is not activated. The criterion to determine whether two prisms touch each other is done without the milling.

DRAFT

Start angle End angle	**Start angle** and **End angle** correspond to the angles of the prisms to local neutral fiber (tangent line to guide curve). Start Angle corresponds to the source side angle of the prisms (84,3° in image below). End Angle can be seen as the angle used to change the direction of a reflected ray to let the ray get out of the Light Guide (10,7° in image below). In Add or Hybrid mode, if the start angle is set to automatic, a constant angle of 85° is used to calculate the prisms' start angle. In Remove mode, if the end angle is set to automatic, a constant angle of 85° is used to calculate the prisms' end angle. 

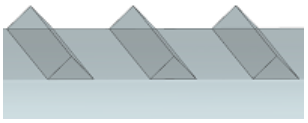
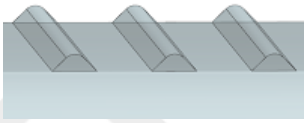
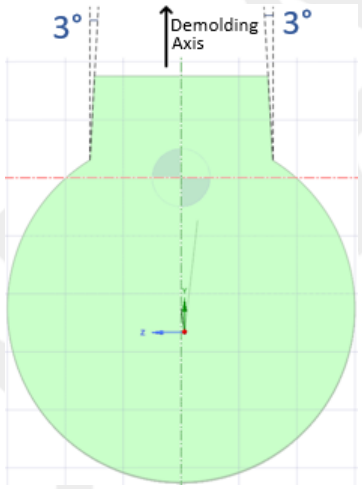
DRAFT

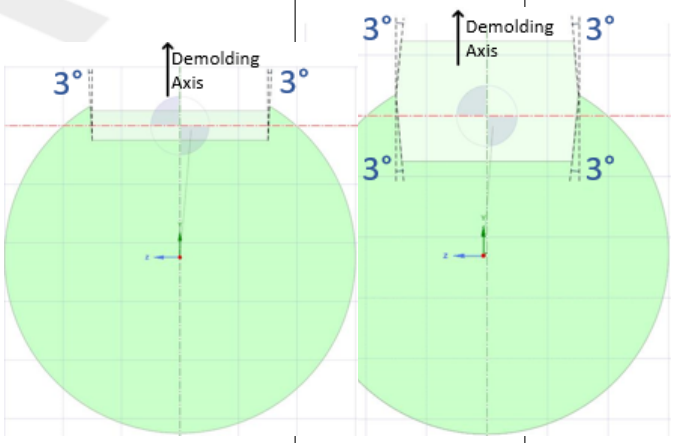
Start Radius End Radius	**Important:** The computation time may be longer when you create curved prisms using the **Start radius** and **End radius** parameters. **Start radius** corresponds to the radius of curvature applied on the source side face of the prisms.  **End radius**corresponds to the radius of curvature applied on the non source side face of the prisms.  Apply a positive value for a convex curvature and a negative value for a concave curvature. **Important:** The Start and End Radius is always applied on the edges of the complete prism. In other words, if you apply a radius on prisms that are partially added or removed according to your setting, the radius is applied on the complete prism edge and not on the partial prism edge. In the following example, you have a prism to remove (yellow). When you apply a radius, the curvature part is generated from the complete prism edge (green), and not from the partial prism edge (red): 

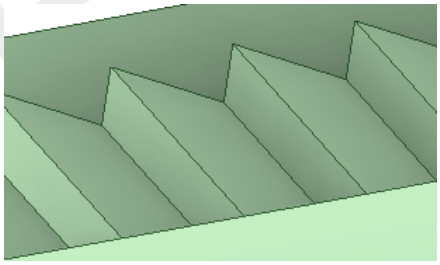
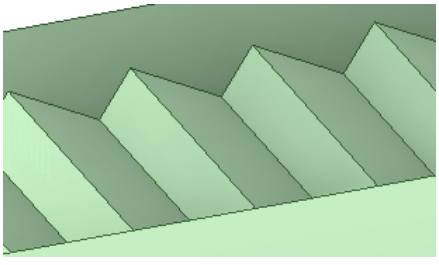
13.6.5.3. Manufacturing Parameters

This page describes the parameters to create a light guide that can be manufactured.

Note: Prism parameters vary depending on the operation selected.

<table>
<tr><td>Top/ Bottom Prism Milling</td><td colspan="2">A filet is applied on the top and/or bottom edge of the prisms when the Milling activated.
Milling is useful to take into account the size of the tool used to shape the surface during the manufacturing.
If the milling is not computed on the prisms, it might be due to an incorrect design. Verify the construction of the light guide.</td></tr>
<tr><td></td><td> 
Light Guide without Milling</td><td> 
Light Guide with top Milling</td></tr>
<tr><td>Drafting</td><td colspan="2">The Drafting allows you to define an angle and a demolding axis to ensure a safe removal of the light guide from the mold while taking into account the manufacturing constraints.
The angle corresponds to the minimum angle the construction algorithm will apply. That means, the algorithm can apply a value higher than the angle defined for a correct demolding on certain prisms if necessary. However the value could slightly impact the photometry. Therefore, we recommend you to check the drafting to avoid unexpected results.</td></tr>
<tr><td></td><td> 

Add Mode</td><td> 

Remove Mode
Hybrid Mode</td></tr>
<tr><td></td><td> 
No Drafting</td><td> 
3° Draft angle</td></tr>
</table>

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

13.7. TIR Lens

The TIR (Total Internal Reflection) Lens allows you to create a lens that collimates rays from a punctual source.

13.7.1. TIR Lens Overview

The Total Internal Reflection (TIR) lens is a cone shaped optical component designed to collimate light.

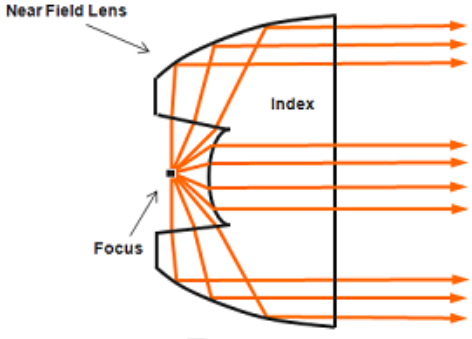
Optical lenses are made to converge or diverge light. The Total Internal Reflection lens is a collimating optical component that is highly efficient to capture and redirect light emitted from a light source.



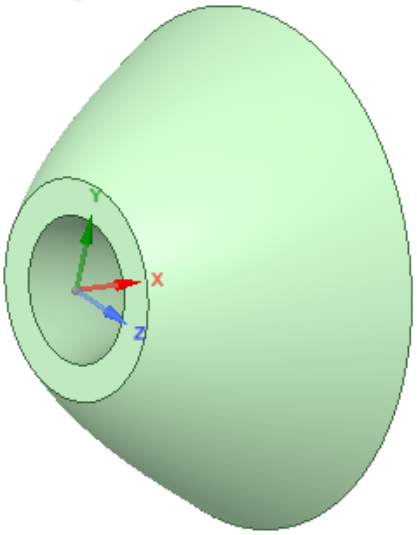


In Speos, the light source is represented by a focus point, placed inside the system.

The output beam is collimated along the optical axis direction after passing through the lens.



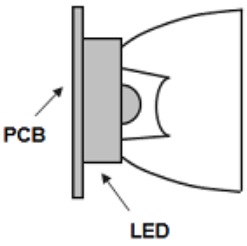
Section view of the collimation carried out by a TIR lens

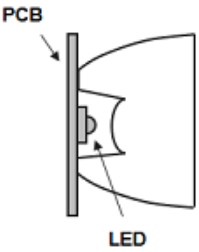


3D view of a TIR lens

The light source defined at the focus point is generally a LED located on a printed circuit board (PCB).

DRAFT





Source at the focus point - Generally a LED located on a printed circuit board (PCB)

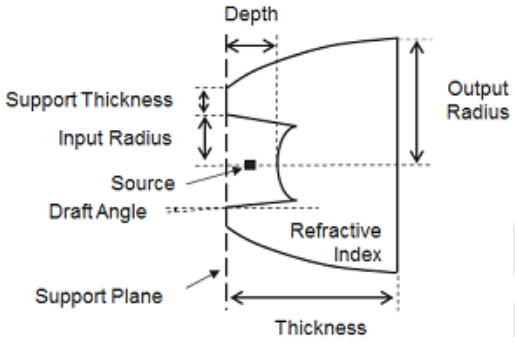
13.7.2. Understanding the Parameters of a TIR Lens

This page describes the parameters to set when creating a TIR Lens.

A TIR lens should be seen as a two-sided lens. The support plane represents the bottom of the lens and the output face represents the top of the lens.

Light travels from the inner to the outer part of the lens.

Figure 80. TIR Lens Settings (Section view)



Axis System

The TIR lens is build around a central axis called revolution or optical axis. This axis is normal to the support plane.

- **Source position**: the source position is set by selecting a point. This point places the light source in the scene and will be used to build the TIR lens around it. The source set here is considered as punctual, which means that it is a simplified source that emits light from a single point and position.

> **Note:** The source of the TIR belongs to this optical axis.

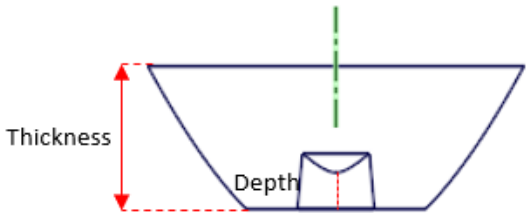
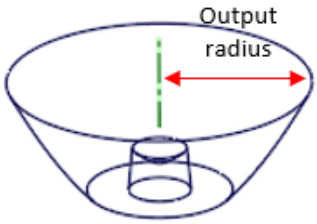
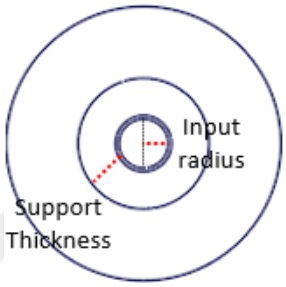
- **Support planeSupport point**: the support defines the TIR lens bottom face.

Dimensions

- **Input radius**: internal radius of the TIR Lens on the support plane.
- **Depth**: distance between the support plane and the first intersection with the lens along the revolution axis.
- **Draft angle**: angle between the internal component of the lens and the revolution axis.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- **Support Thickness**: The TIR Lens is considered fastened on a support (represented by the support plane). The support thickness refers to the thickness of the ring at the bottom of the lens.
- **Thickness**: height of the TIR lens along the revolution axis.
- **Output radius**: radius of the lens' output face.

		
TIR Lens side view	TIR Lens angled view	TIR Lens bottom view

Refractive Index

The **Refractive Index** of a material is a pure number that describes how light propagates through that medium.

Most transparent materials have refractive indices between 1 and 2. Here the refractive index refers to the index of the lenses' material.

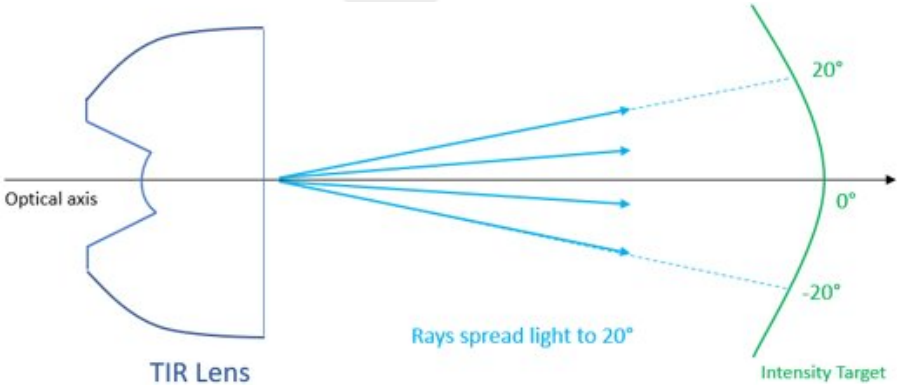
Focal

The focal length represents the distance between the source and the top of the internal collimating surface.

Spread Parameters

Spread and Spread Behavior

The Spread and Spread behavior parameters spread the light at an angle determined on an intensity target.

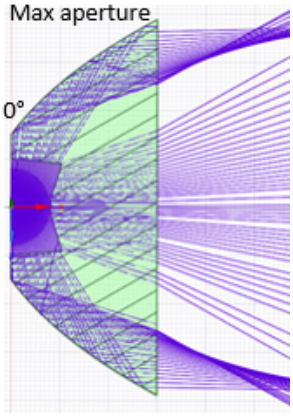
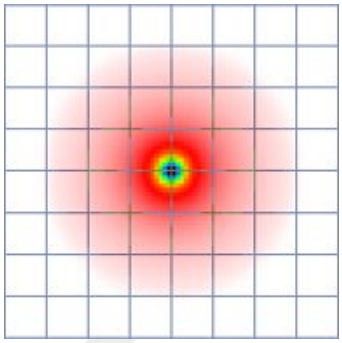
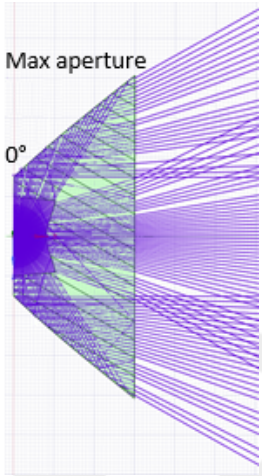
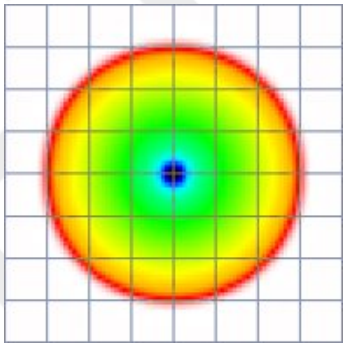


The Spread value controls the maximum angular aperture for the center (dioptric) and the outer (TIR) faces. By defining a Spread value higher than 0°, TIR lens will spread light an intensity target.

The Spread behavior influences the results whether the TIR Lens is Convex or Concave.

- When the TIR Lens is concave: the TIR face spreads from max aperture to 0°, meaning rays are crossing.
- When the TIR Lens is convex: the TIR face spreads from 0° to max aperture, meaning rays are opening.

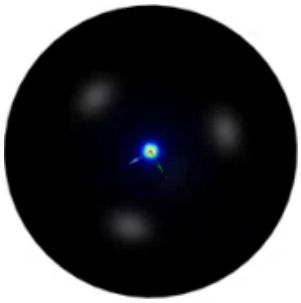
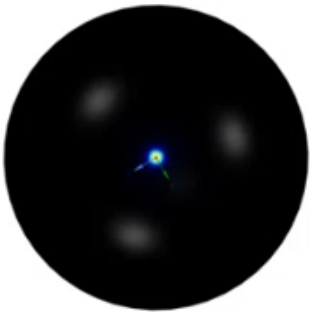
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Concave TIR Lens		
Convex TIR Lens		

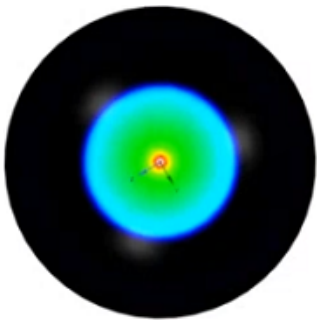
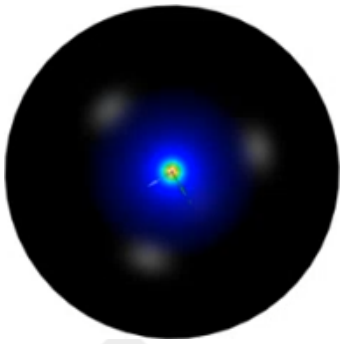
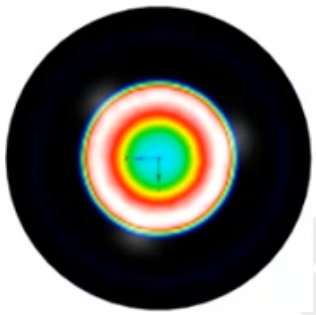
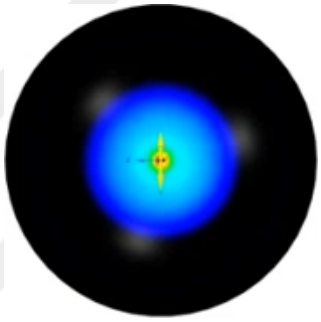
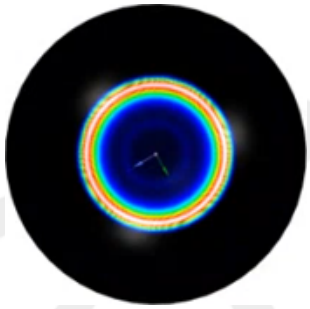
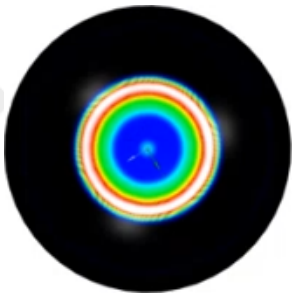
Spread Control

The Spread Control parameter manages how light is spread between 0° and Spread Max.

- When Spread Control is lower than 50, light accumulates on 0° direction.
- When Spread Control is higher than 50, light accumulates on Spread Max direction.

	Mode Convex Spread = 30°	Mode Concave Spread = 30°
Spread Control = 0		

DRAFT

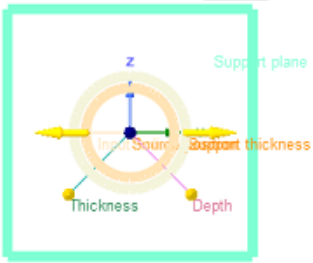
Spread Control = 50		
Spread Control = 60		
Spread Control = 100		

13.7.3. Creating a TIR Lens

This page shows how to create a Total Internal Reflection lens.

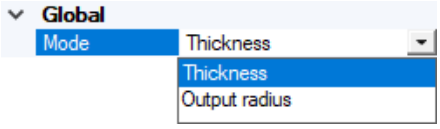
To create a TIR lens:

1. From the **Design** tab, click **TIR lens**  .
   A preview of the TIR lens support plane and associated parameters appear in the 3D view.



2. From the **Mode** drop-down list, select which parameter should drive the lens' dimensions:

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



- Select **Thickness** to define the lens by its height.
- Select **Output radius** to define the lens by the radius of its output face.

3. Click and select a point to define the **Source position**.

> **Note:** The source is considered as punctual.

4. Click and select plane to define the position of the lens input face (the support).
5. In **Input radius**, type a value to define the internal radius of the TIR Lens on the support plane.

Input radius	10 mm
Depth	20 mm
Draft angle	5 °
Support thickness	5 mm
Refractive index	1.49
Thickness	30 mm
Output radius	0 mm
Focal	20 mm
Reverse direction	False
Spread	20 °
Spread behavior	Concave
Spread control	50

6. In **Depth**, type a value to define the distance between the support plane and the first intersection with the lens along the revolution axis.
7. In **Draft Angle**, type a value to define the revolution axis angle.
8. In **Support thickness**, define the thickness of the ring at the bottom of the lens.
9. Define the Refractive index of the lens.
10. If you chose **Thickness**, set a value to define the lens' height, or use the 3D view manipulators.
11. If you chose **Output radius**, type a value to define the radius of the lens' output face.
12. Set the **Focal** of the lens, that is the distance between the source and the top of the internal collimating surface.
13. In **Spread**, type a value in [0° ; 90°[to define the spread angle of the rays. 0° corresponds to collimated rays.
14. In **Spread behavior**, select the spread behavior of the rays:
    - **Concave**: the TIR face spreads from max aperture to 0°, meaning rays are crossing.
    - **Convex**: the TIR face spreads from 0° to max aperture, meaning rays are opening.
15. In **Spread control**, type a value in [0 ; 100] to control the light distribution in the target in the range [0 ; Spread Max].
16. Press **F4** to leave the feature edition mode.

The TIR lens is created and appears both in the tree and in the 3D view.

13.8. Projection Lens

The Projection Lens feature allows you to create plano-concave/convex, aspheric, zernike, toroidal or fresnel lenses that can be used in a projector module.

DRAFT

> **Note:** The Zernike Face type is in BETA mode for the current release.

13.8.1. Projection Lens Overview

Projection lenses are optical lenses that are often used for automotive modules.

> **Note:** The Zernike Face type is in BETA mode for the current release.

Projection lenses are designed to redirect light using the refraction that occurs through the transparent material.

They feature large apertures and short focal lengths and allow to convert a divergent light beam into a collimating one.



In Speos, different types of back and front faces, as well as construction types are available to cover different needs. As a consequence, projection lenses can be toroidal, aspheric, spherical, cylindrical, plano-concave or convex.

13.8.2. Understanding Projection Lens Parameters

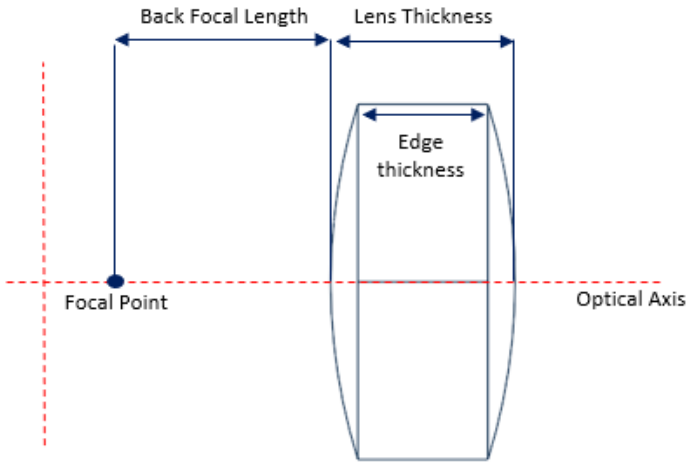
This page describes the parameters to set when creating a Projection Lens.

> **Note:** The Zernike Face type is in BETA mode for the current release.

The Projection lens should be seen as a two-sided lens.

The Back face and Front face of the lens can have different shapes or types.

DRAFT



Construction settings

- **Focal point**: the focal point allows you to position the lens along the optical axis.
- **Optical axis**: line that corresponds to the light direction after passing through the lens.
- **Back Focal Length**: the back focal length corresponds to the distance between the focal point and the lens' back face.
- **Refractive index**: pure number that describes how light propagates through that medium. Most transparent materials have refractive indices between 1 and 2. Here the refractive index refers to the index of the lenses' material.
- **Orientation axis (beta)**: The Orientation axis is used when a Face is defined as Zernike, to consider the polar coordinates for the Zernike coefficients.
- Thickness
  - **Lens Thickness**: size of the central part of the lens.
  - **Edge Thickness**: size of the lens from the back to the front face of the lens.

Face Types

Compatibility Table

The following table provides you with the compatibility between the Front and Back Face types.

	Plan	**Aspherical**	**Automatic**	**Zernike**
Plan	✓	✓	✓	✓
Aspherical	✓	✓	✓	✗
Automatic	✓	✓	✗	✗
Zernike	✓	✗	✗	✗

DRAFT

Plan

Defining a plan face allows you to create a plano-concave/convex lens.

- **Aperture Diameter**: diameter of the Front Face or Back Face of the lens.
- **Aspherics**: corresponds to the 29 aspherical coefficients that you can set to adjust the lens or remove the aberrations.

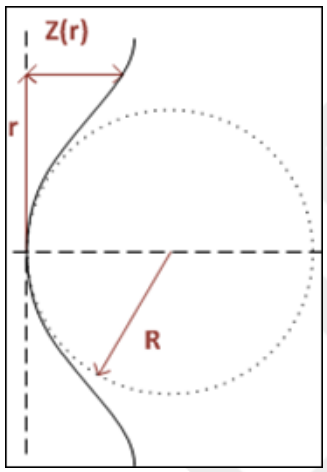
> **Note:** The index 1 is no longer present in the interface as it is always 0. Only edible coefficient are present.

The formula of the aspherical coefficients is $Z_{(r)} = \sum_{i=1}^{\infty} \alpha_i \cdot r^i$ with $\boldsymbol{\alpha}_i$ the ith aspherical value.

Aspheric

- **Aperture Diameter**: diameter of the Front Face or Back Face of the lens.
- **Curvature**: radius of curvature for the spherical part of the Front Face or Back Face of the lens.
- **Conic Constant**: conic constant of the lens.
- **Aspherics**: corresponds to the 29 aspherical coefficients that you can set to adjust the lens or remove the aberrations.

> **Note:** The index 1 is no longer present in the interface as it is always 0. Only edible coefficient are present.



$$Z(r) = \frac{r^2}{R\left(1 + \sqrt{1 - (1 + k)\frac{r^2}{R^2}}\right)} + \sum_{i=2}^{\infty} \alpha_i r^i$$

r: variable corresponding to the position on the lens

k: conic constant

R: radius of curvature

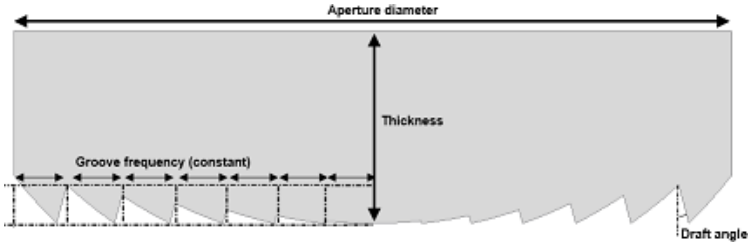
α_i: i^{th} aspherical value

Automatic

- **Aperture Diameter**: diameter of the Front Face or Back Face of the lens.
- **Refractive Index**: refractive index of the lenses' material. This is a pure number that describes how light propagates through that medium. Most transparent materials have refractive indices between 1 and 2. The refractive index impacts how the face of the lens is constructed as rays should be collimated.
- **Fresnel mode**
  - **Constant step**: creates grooves of the same length.

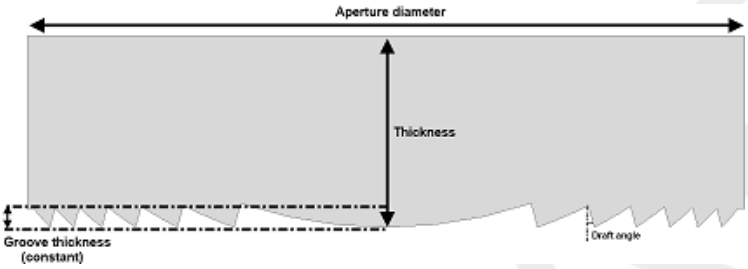
DRAFT

Note: Only full grooves are built. If the last groove cannot be built, a flat face appears instead.



- **Constant height**: creates grooves of the same height.

Note: Only full grooves are built. If the last groove cannot be built, a flat face appears instead.



- **Draft Angle**: angle to respect to be able to remove the lens from the mold when the face is in Fresnel mode. The draft angle should be between 0° and 15°.

Zernike

- **Aperture Diameter**: diameter of the Front Face or Back Face of the lens.
- **Curvature**: radius of curvature for the spherical part of the Front Face or Back Face of the lens.
- **Conic Constant**: conic constant of the lens.
- **Aspherics**: corresponds to the 8 aspherical coefficients that you can set to adjust the lens or remove the aberrations.
- **Zernike coefficients**: correspond to the 28 first Zernike coefficients that you can set to adjust the lens or remove the aberrations.

  The Zernike coefficients are ordered according to the Noll convention.

  The angle ϕ is measured counter clockwise from the local +x axis.

  The radial coordinate is the normalized dimensionless parameter ρ.

DRAFT

Term	Z (ϱ , φ)	Term	Z (ϱ , φ)
1	1	15	$\sqrt{10}\,\varrho^4\sin 4\varphi$
2	$\sqrt{4}\,\varrho\cos\varphi$	16	$\sqrt{12}\,(10\varrho^5-12\varrho^3+3\varrho)\cos\varphi$
3	$\sqrt{4}\,\varrho\sin\varphi$	17	$\sqrt{12}\,(10\varrho^5-12\varrho^3+3\varrho)\sin\varphi$
4	$\sqrt{3}\,(2\varrho^2-1)$	18	$\sqrt{12}\,(5\varrho^5-4\varrho^3)\cos 3\varphi$
5	$\sqrt{6}\,(\varrho^2\sin 2\varphi)$	19	$\sqrt{12}\,(5\varrho^5-4\varrho^3)\sin 3\varphi$
6	$\sqrt{6}\,(\varrho^2\cos 2\varphi)$	20	$\sqrt{12}\,\varrho^5\cos 5\varphi$
7	$\sqrt{8}\,(3\varrho^3-2\varrho)\sin\varphi$	21	$\sqrt{12}\,\varrho^5\sin 5\varphi$
8	$\sqrt{8}\,(3\varrho^3-2\varrho)\cos\varphi$	22	$\sqrt{7}\,(20\varrho^6-30\varrho^4+12\varrho^2-1)$
9	$\sqrt{8}\,\varrho^3\sin 3\varphi$	23	$\sqrt{14}\,(15\varrho^6-20\varrho^4+6\varrho^2)\sin 2\varphi$
10	$\sqrt{8}\,\varrho^3\cos 3\varphi$	24	$\sqrt{14}\,(15\varrho^6-20\varrho^4+6\varrho^2)\cos 2\varphi$
11	$\sqrt{5}\,(6\varrho^4-6\varrho^2+1)$	25	$\sqrt{14}\,(6\varrho^6-5\varrho^4)\sin 4\varphi$
12	$\sqrt{10}\,(4\varrho^4-3\varrho^2)\cos 2\varphi$	26	$\sqrt{14}\,(6\varrho^6-5\varrho^4)\cos 4\varphi$
13	$\sqrt{10}\,(4\varrho^4-3\varrho^2)\sin 2\varphi$	27	$\sqrt{14}\,\varrho^6\sin 6\varphi$
14	$\sqrt{10}\,\varrho^4\cos 4\varphi$	28	$\sqrt{14}\,\varrho^6\cos 6\varphi$

The formula of the zernike coefficients is $$Z(r)=\frac{r^2}{R\left(1+\sqrt{1-(1+k)\frac{r^2}{R^2}}\right)}+\sum_{i=1}^{8}\alpha_i r^{2i}+\sum_{i=1}^{28}\beta_i Z_i(\varrho,\varphi)$$ with

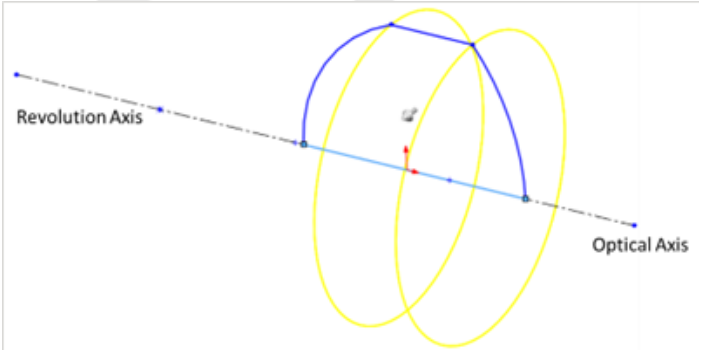
- r: variable corresponding to the position on the lens
- k: conic constant
- R: radius of curvature
- α_i: i^{th} aspherical value
- β_i: i^{th} zernike coefficient

Construction Type

Revolution

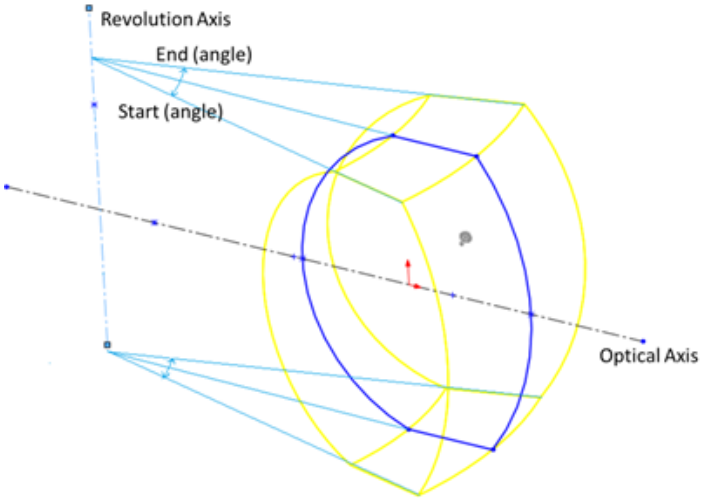
With the **Revolution** type, the Back Face and Front Face profiles are revolved around the optical axis to create a spherical lens by default or a **Custom Revolution** axis to create a toroidal lens.

Figure 81. Spherical lens (Optical axis as revolution axis)



Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Figure 82. Toroidal lens (Custom revolution axis)

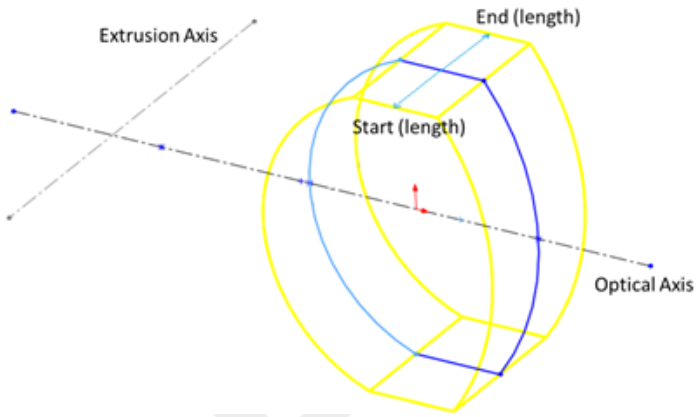


> **Note:** For a **Custom Revolution** axis, the **Start** and **End** angles must be included in the range [-180 ; 180].

Extrusion

With the **Extrusion** type, the Back Face and Front Face profiles are extruded along the extrusion axis to create a cylindrical lens.

Figure 83. Cylindrical lens



13.8.3. Creating a Projection Lens

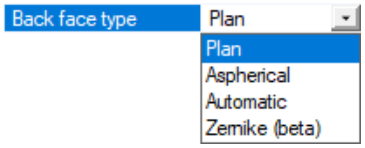
The following procedure shows how to create a Projection Lens.

> **Note:** The Zernike Face type is in BETA mode for the current release.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

To create a Projection Lens:

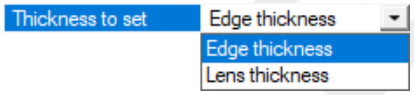
1. From the **Design** tab, click **Projection Lens** .
   A preview of the Projection Lens and associated parameters appear in the 3D view.
2. In the 3D view, click and select a point to define the **Focal point** of the lens.
3. In the 3D view, click and select a line or axis to define the **Optical axis** of the lens.
   The optical axis is used to define the light direction after passing through the lens.
4. Define the **Back Focal Length** (the distance between the focal point and the lens' back face) either by entering a value in mm or using the 3D view manipulators.
5. From the **Back face type/ Front face type** drop-down list, select which shape the back face and front face of the lens should have:



   - Select **Plan** to create a plano-concave or convex lens.
   - Select **Aspherical** to create a lens with an aspheric back/front face.
   - Select **Automatic** to create a lens that collimates rays after passing through the front face.
   - Select **Zernike (beta)** to create a lens with a Zernike back/front face.

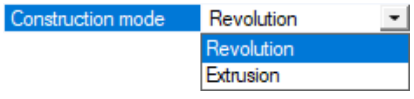
> **Note:** The Back face and Front face cannot be set as **Automatic** at the same time. A **Zernike** face is only compatible with a **Plan** face. For more information on the compatibility between the face types, refer to Understanding Projection Lens Parameters.

6. From the **Thickness to set** drop-down list, select on which parameter you want to base the thickness of the lens:



> **Note:** In case of a **Zernike** Face, the Thickness is set as **Lens Thickness**.

   - Select **Edge thickness** to define the size of the central part of the lens.
   - Select **Lens thickness** to define the size of the lens from the back to the front face of the lens.
7. Enter a value to define the thickness of the lens.
8. Define the shape of the lens:



> **Note:** If the **Back face type** or **Front face type** is set to **Automatic** or **Zernike**, the **Construction mode** not available.

   - To create a spherical lens, from the **Construction mode** drop-down list, select **Revolution**.

The back and front face profiles revolve around the optical axis.

- To create a toroidal lens:

Construction	
Construction mode	Revolution
Custom revolution	True
Revolution origin	[No selection]*
Revolution axis	[No selection]*
Reverse direction	False
Start angle	-180 °
End angle	180 °

  a. From the **Construction mode** drop-down list, select **Revolution**.
  b. Activate **Custom revolution** and click
  
  to select a custom origin and
  
  to select a revolution axis.
  c. Define the **Start angle** and **End angle** of the revolution in the range [-180° ; 180°].

- To create a cylindrical lens:

Construction	
Construction mode	Extrusion
Extrusion axis	[Default] Y Axis
Reverse direction	False
Extrusion start	-5 mm
Extrusion end	5 mm

  a. From the **Construction mode** drop-down list, select **Extrusion**.
  b. If needed, click
  
  and select a line to define an extrusion axis.
  c. Define the **Extrusion start** and **Extrusion end** points of the lens in mm.

Once the general parameters of the lens are defined, define the characteristics of the Back Face and Front Face according to their selected type.

13.8.4. Defining the Back Face and Front Face

The following section provides you with the procedures to define the characteristic of the Back Face and Front Face of a Projection Lens.

> **Note:** The Zernike Face type is in BETA mode for the current release.

13.8.4.1. Defining a Plan Face

This procedure helps you define a plan back face or a plan front face during a projection lens creation.

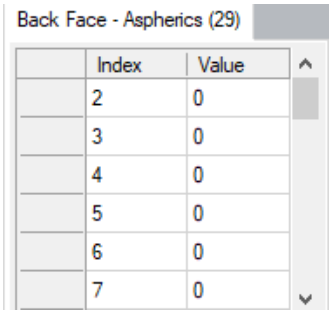
DRAFT

To create a plan face:

You must have set the **Back face type** or **Front face type** to Plan.



1. In the **Back / Front Face** section, define the **Aperture diameter** of the lens' back / front face in mm.
2. If needed, in the **Aspheric table**, define the 29 aspherical values corresponding to the aspherical coefficients.



These coefficients are used to adjust the lens or remove any construction aberration.

> **Note:** For more information on the aspherical coefficients, refer to Understanding Projection Lens Parameters.

The plan face has been defined.

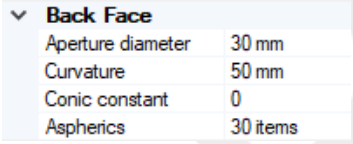
If not yet defined, define the aspherical or automatic characteristics of the other face.

13.8.4.2. Defining an Aspherical Face

This procedure helps you define an aspherical back face or an aspherical front face during a projection lens creation.

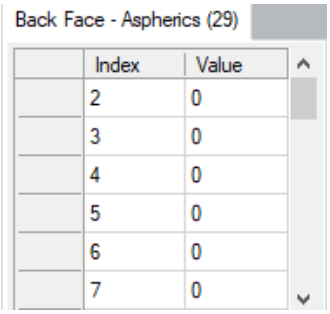
To create an aspherical face:

You must have set the **Back face type** or **Front face type** to Aspherical.



1. In the **Back / Front Face** section, define the **Aperture diameter** of the lens' back / front face in mm.
2. In **Curvature**, define the radius of curvature for the spherical part of the lens' back / front face.
3. If needed, define the **Conic constant** of the lens.
4. If needed, in the **Aspheric table**, define the 29 aspherical values corresponding to the aspherical coefficients.

DRAFT



These coefficients are used to adjust the lens or remove any construction aberration.

> **Note:** For more information on the aspherical coefficients, refer to Understanding Projection Lens Parameters.

The aspherical face has been defined.

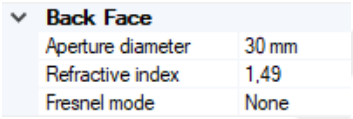
If not yet defined, define the plan, aspherical or automatic characteristics of the other face.

13.8.4.3. Defining an Automatic Face

This procedure helps you define an automatic back face or an automatic front face during a projection lens creation.

To create an automatic face:

You must have set the **Back face type** or **Front face type** to Aspherical.



1. In the **Back / Front Face** section, define the **Aperture diameter** of the lens' back / front face in mm.
2. Define the **Refractive index** of the lens' material.

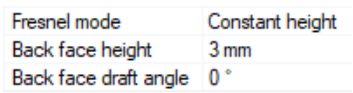
   The refractive index impacts how the face of the lens is constructed as rays should be collimated.

3. If you want to create a Fresnel lens, from the **Fresnel mode** drop-down list:
   - Select **Constant step** to create grooves of the same length.

     a. In **Back/Front face step**, define the step size.
     b. In **Back/Front face draft angle**, define a draft angle in the range [0° ; 15°] that takes into account the manufacturing constraints and ensure a safe removal of the lens from the mold.

   - Select **Constant height** to create grooves of the same height.

     a. In **Back/Front face height**, define the height of the grooves.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

b. In **Back/Front face draft angle**, define a draft angle in the range [0° ; 15°] that takes into account the manufacturing constraints and ensure a safe removal of the lens from the mold.

The automatic face has been defined.

If not yet defined, define the plan or aspherical characteristics of the other face.

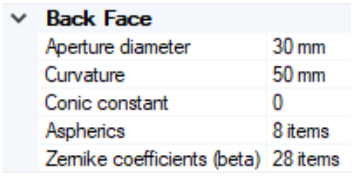
13.8.4.4. Defining a Zernike Face

This procedure helps you define a Zernike back face or a Zernike front face during a projection lens creation.

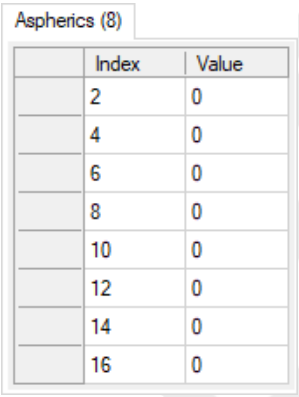
> **Note:** The Zernike Face type is in BETA mode for the current release.

To create a Zernike face:

You must have set the **Back face type** or **Front face type** to Zernike.



1. In the **Back / Front Face** section, define the **Aperture diameter** of the lens' back / front face in mm.
2. In **Curvature**, define the radius of curvature for the spherical part of the lens' back / front face.
3. If needed, define the **Conic constant** of the lens.
4. If needed, in the **Aspheric table**, define the 8 aspherical values corresponding to the aspherical coefficients.



These coefficients are used to adjust the lens or remove any construction aberration.

> **Note:** For more information on the aspherical coefficients, refer to Understanding Projection Lens Parameters.

5. If needed, in the **Zernike coefficients table**, define the 28 Zernike values corresponding to the Zernike coefficients.

DRAFT

Zernike coefficients (beta) (28)

	Index	Value
▶	1	0
	2	0
	3	0
	4	0
	5	0
	6	0

These coefficients are used to adjust the lens or remove any construction aberration.

> **Note:** For more information on the Zernike coefficients, refer to Understanding Projection Lens Parameters.

The Zernike face has been defined. The other face is automatically set as **Plan** and uses the same **Aperture Diameter** as the Zernike face.

13.9. Poly Ellipsoidal Reflector

The Poly Ellipsoidal Reflector feature allows you to create a reflector used for automotive projection modules.

13.9.1. Poly Ellipsoidal Reflector Overview

The Poly Ellipsoidal Reflector is a reflector that is mainly used in automotive projector modules to produce spread driving beams.

Poly Ellipsoidal reflectors are designed to collect light rays from a focal point and redirect them to a secondary one with defocus.

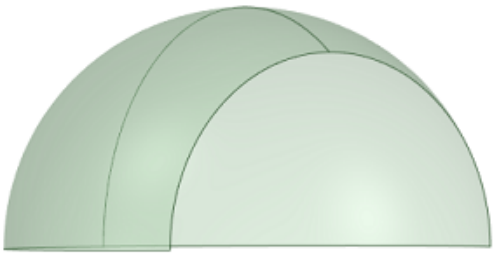
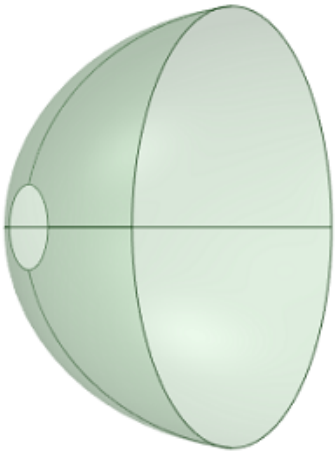


In Speos, reflectors are created in several angular sections around the optical axis.

For each of these profiles (or angular sections), control planes are defined to control the variation of the defocus from the image point.

The defocus drives the overall beam spread. A poly ellipsoid reflector without defocus produces perfectly collimated rays with no spread, that is, the beam pattern of a spotlight.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

	
LED reflector - no hole, source emitting only in upper half space	Light Bulb reflector - hole for the source, source with diffuse emission in upper and lower half space

Related concepts

Understanding Parameters of a Poly Ellipsoidal Reflector on page 620
This page describes parameters to set when creating a Poly Ellipsoidal Reflector.

Related tasks

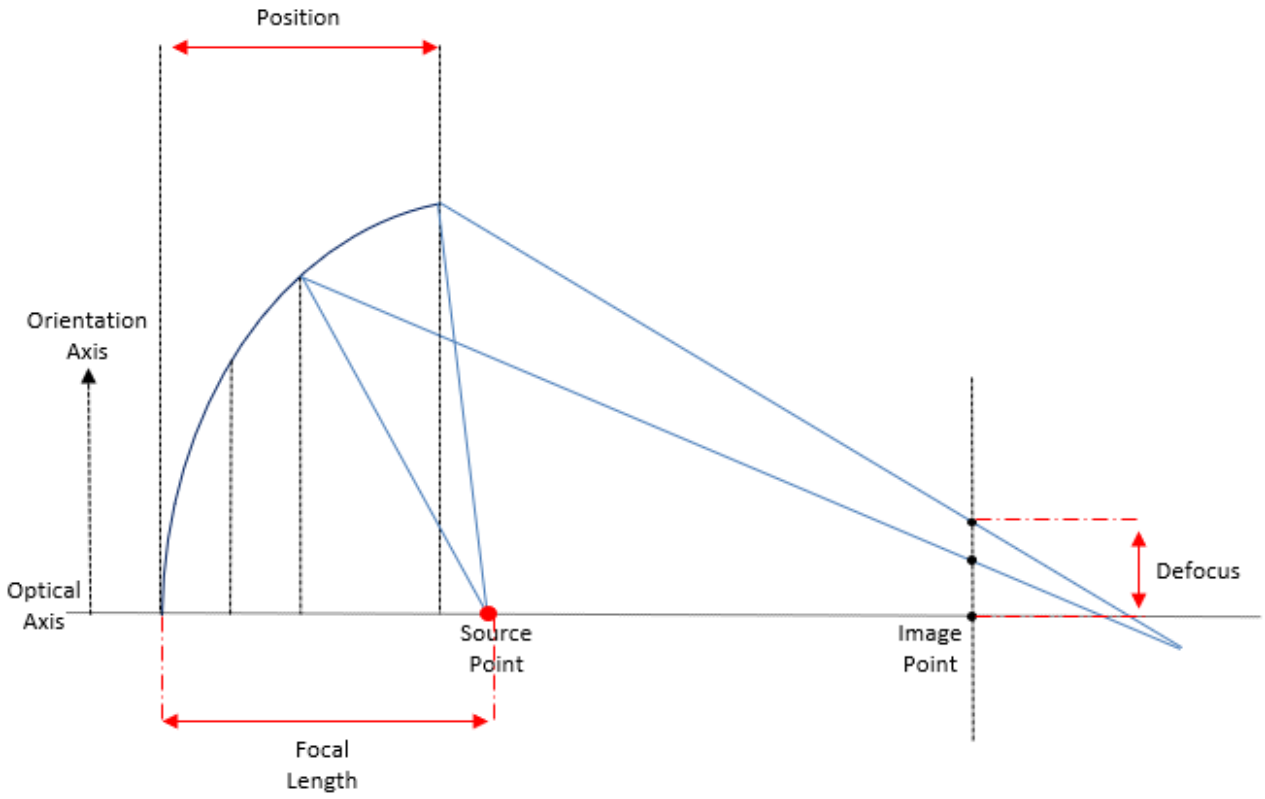
Creating a Poly Ellipsoidal Reflector on page 623
This page shows how to create a Poly ellipsoidal reflector.

13.9.2. Understanding Parameters of a Poly Ellipsoidal Reflector

This page describes parameters to set when creating a Poly Ellipsoidal Reflector.

DRAFT

Figure 84. Side view of an angular section with parameters to set to create the reflector.



Axis System

- **Source point**: this point defines the source position. The source is a simplified source that emits light from a single point and position. The source is used to calculate the poly ellipsoid profiles and illuminate the reflector.
- **Image point**: this point corresponds to the reflector's focus point, that is where the light rays converge when no defocus is applied.
- **Orientation axis**: the orientation axis is the line fixing the orientation of the reflector around the optical axis. The optical axis passes through the source point and image point. The 0 degree angular section is located in the plane defined by the optical axis and the Orientation Axis.

> **Note:** The orientation axis may not be defined on a plane normal to the optical axis.

Hole Diameter

Diameter of the hole at the bottom of the reflector. This hole is used to position the light source.

If you want to create a reflector without a hole (for example a reflector with a LED and not a bulb as light source), deactivate this option to create a closed reflector.

Focal Length

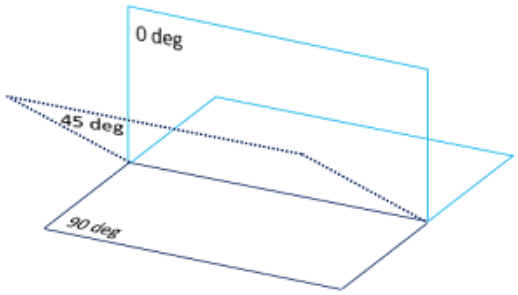
The Focal Length is the distance between the source and the back of the Poly Ellipsoid Reflector.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Angular Sections

The angular sections define how the reflector is built. You can create as many sections as needed between 0° and 90°.

Figure 85. 0 and 90 degree planes with one section added at 45°



Control Planes

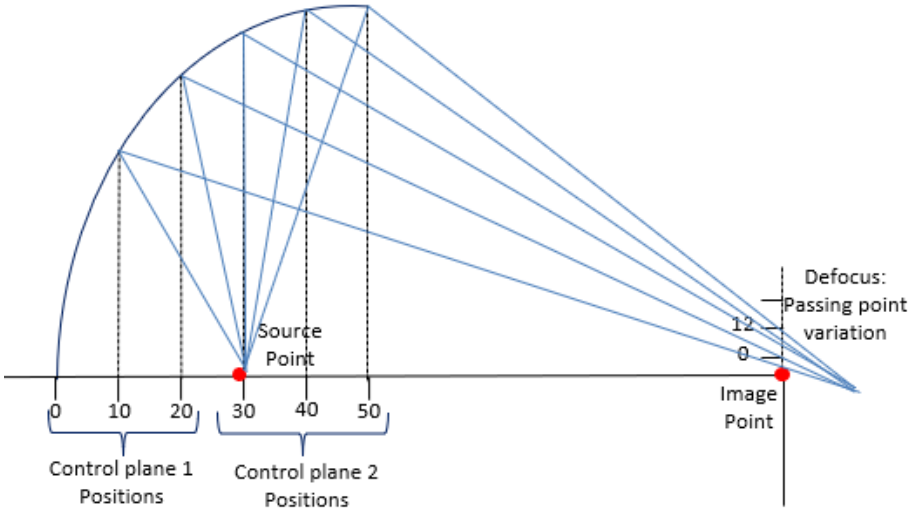
Creating control planes allows you to drive the overall beam spread.

Designing a poly ellipsoidal reflector without creating any control plane and defocus would create a pure ellipsoidal reflector that would generate a spotlight beam (once passing through the projection lens).

Each control plane is identified by its **Position** from the back of the reflector and its **Defocus** value.

- The **Position** is expressed in mm and is defined by the distance between the back of the reflector and where you want to place the control plane.
- The **Defocus** can be defined for each control plane and corresponds to the distance from the image point on image point plane.

Figure 86. Control planes defined between 0 and 50mm with defocus applied.



Related tasks

Creating a Poly Ellipsoidal Reflector on page 623
This page shows how to create a Poly ellipsoidal reflector.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Related information

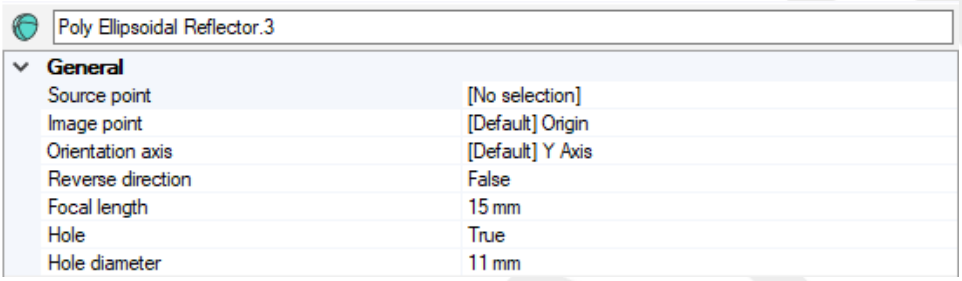
Poly Ellipsoidal Reflector Overview on page 619
The Poly Ellipsoidal Reflector is a reflector that is mainly used in automotive projector modules to produce spread driving beams.

13.9.3. Creating a Poly Ellipsoidal Reflector

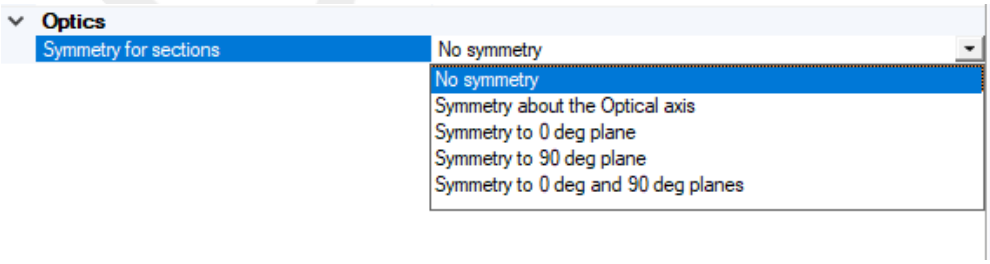
This page shows how to create a Poly ellipsoidal reflector.

To create a Poly Ellipsoidal Reflector:

1. From the **Design** tab, click **Poly Ellipsoidal Reflector**  .



2. In the 3D view, define the axis system of the reflector:
   - Click  and select a point to define the source point.
   - If you need to modify the image point (the origin point), click  and select a point in the 3D view.
   - If you need to modify the orientation axis (by default the Y axis), click  and select a line.
3. Define the **Focal length** of the reflector, that is the distance between the source point and the back of the reflector.
4. If you want to create a reflector without a hole (for example a reflector with a LED and not a bulb as light source), set **Hole** to **False**.
5. If you activated the creation of a hole, set its diameter to position the light bulb.
6. In **Optics**, define how you want to build the poly ellipsoidal reflector. From the **Symmetry for sections** drop-down list:



> **Note:** Angular sections can only be defined between 0° and 90°.

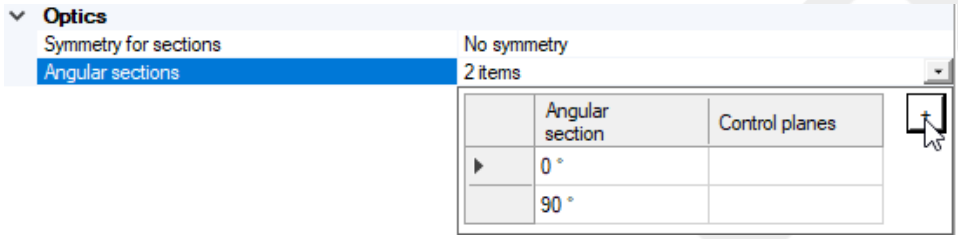
   - Select **No symmetry** to define angular sections between 0° and 90°.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

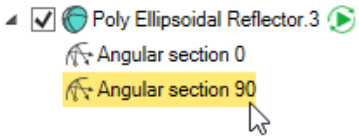
- Select **Symmetry about the Optical axis** to define only the 0° section angle.
- Select **Symmetry to 0 deg plane** to mirror the definition of the angular sections on the horizontal axis.
- Select **Symmetry to 90 deg plane** to mirror the definition of the angular sections on the vertical axis.
- Select **Symmetry to 0 deg and 90 deg planes** to create a full reflector, with mirrored definition on both horizontal and vertical axes.

7. Create as many angular sections as you need for your reflector and define their angle value.

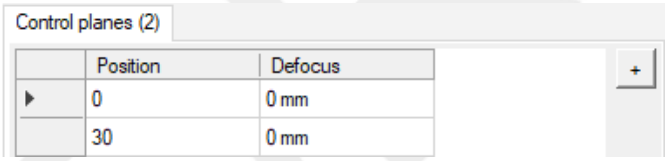
> **Note:** Only default angular sections (0° and 90°) are visible in the 3D view. But all sections created are taken into account for simulation.



8. For each angular section created, you can define control planes :
   a) From the design panel, select the angular section on which you want to define control planes.



   b) Create as many control planes as you need.
   c) Set the **Position** and **Defocus** values



> **Note:** The highest position value defines the size of the reflector. Consequently, the highest value must be the same for each angular section.

9. Click **Compute** to build the feature.

Related concepts

Understanding Parameters of a Poly Ellipsoidal Reflector on page 620
This page describes parameters to set when creating a Poly Ellipsoidal Reflector.

13.10. Freeform Lens

The Freeform Lens allows you to create a collimating lens, a user target-based lens, or a user image-based lens from any freeform surface.

13.10.1. Freeform Lens Overview

The freeform lens allows you to create a collimating lens, a user irradiance target-based lens or a user intensity target-based lens with a complex shape.

In the automotive industry, headlamps play a key role in the vehicle's signature. Therefore, designers put more and more efforts in styling the visible parts of the lens system (the front or outer face of the lenses) contained in the headlights.

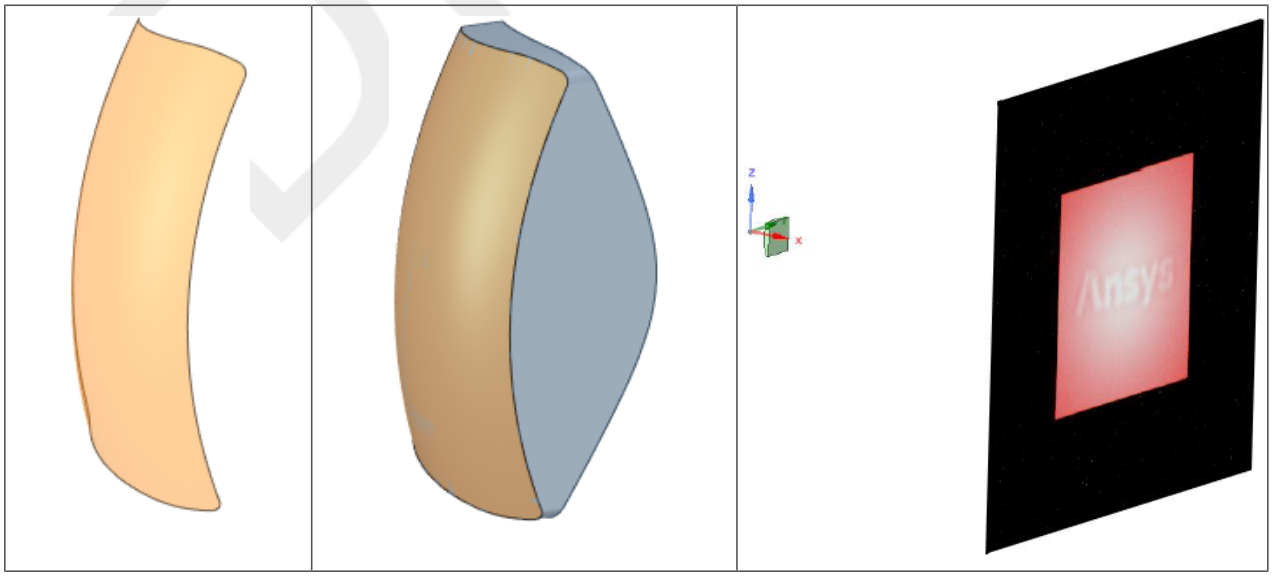
Complex or freeform shapes are used to model the front faces of projection lenses. As a result, the back face of the lens has to be adjusted by optical designers in order to compensate such complex shapes.

Freeform lenses are often made out of plastic as it is easier to shape and compatible with LED projector modules.



In Speos, the definition of the freeform lens is mainly automatic. The algorithm calculates and designs the back face of the lens based on the shape of the front face to build a lens with the expected optical behavior.

- For a collimating lens, it only requires the front face of your lens. Once selected, the lens is automatically built to meet your freeform surface profile
- For a user target-based lens, you design your own target and control the light distribution on the target.



Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Freeform Surface	Result after construction	image-based result

Related tasks

Creating a Freeform Lens on page 630
This page shows how to create a freeform lens from a freeform surface.

Creating a Freeform Lens Based on an Irradiance Target on page 631
This procedure shows how to create a freeform lens based on an irradiance target. The spread type allows you to control the light distribution on the target you define.

Creating a Freeform Lens Based on an Intensity Target on page 633
This procedure shows how to create a freeform lens based on an intensity target. The spread type allows you to control the light distribution on the target you define.

13.10.2. Understanding the Freeform Lens Parameters

The following page helps you understanding how to use the parameters of a Freeform Lens.

Maximum Threshold

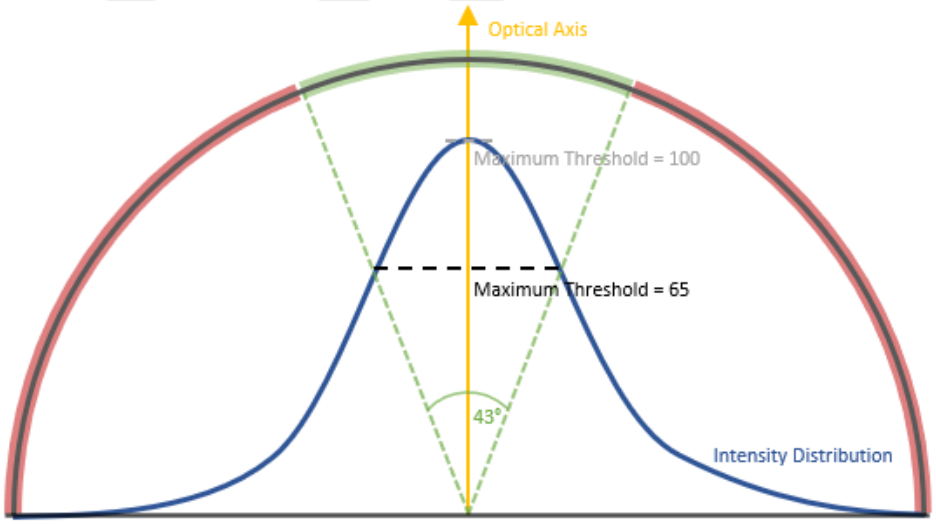
The **Maximum Threshold** is available when you create a Freeform Lens based on an Intensity Target Type with an Intensity File.

The Freeform Lens works only in refraction. However, according to the intensity distribution of the file which can display light on a half-sphere, raking rays on edges are not refracted and the Backface of the Freeform Lens cannot be designed correctly. Thus the **Maximum Threshold** has been introduced to overcome this issue.

The goal of the **Maximum Threshold** is to define a threshold above which rays will be considered in the construction of the Backface, and under which rays will not be considered, therefore removing these potential raking rays that prevent a correct design.

Maximum Threshold can be set between 1 and 100 and is considered along the optical axis.

In the schematic example below, the Maximum Threshold is set to 65 of the intensity distribution peak. This creates a 43 angle. Then, only rays included in the solid angle created by the Maximum Threshold will be used to design the back face (green part). Rays outside the solid angle (red part) will not be used, such as raking rays.



Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Resolution Factor

The **Resolution Factor** is available only in two cases:

- When you create a Freeform Lens based on an Intensity Target Type with an Intensity File Type.
- When you create a Freeform Lens based on an Irradiance Target Type with an Image.

The Backface is designed according to the Inverse Simulation principle. Rays are launched from the Target input, then the Backface is designed according to the result expected. However the Freeform Lens computation may sometimes lack rays to correctly design the Backface. Thus, the Resolution Factor parameter has been introduced to overcome the lack of rays.

The goal of the Resolution Factor is to densify the number of rays launched from the target (intensity file or image) to allow Speos to better design the Backface of the Freeform Lens.

The Resolution Factor is a multiplier of the resolution of the target.

Intensity File (Intensity Target)

In case of an Intensity File, the resolution is set to 180 on Theta and 180 on Phi.

According to the **Maximum Threshold** defined, only the rays included in the solid angle are considered to design the Backface. But this rays number may not be high enough to design correctly. Thus, the Resolution Factor multiplies the resolution of the Intensity File so that more rays are included in the solid angle to design the Backface.

In the schematic example (figure 1), the number of rays included in the solid angle is not enough.

In the schematic example (figure 2), a Resolution Factor = 2 has been applied to get a Intensity File resolution = 360. By extension, the number of rays included in the solid angle has doubled on Theta and doubled on Phi.

> **Important:** The representations are 2D, but you must consider the Resolution Factor multiplies the resolution on Theta and Phi.

> **Warning:** The higher the Resolution Factor the more performance and time needed to design the Freeform Lens by the machine.

DRAFT

Figure 87. Resolution Factor = 1

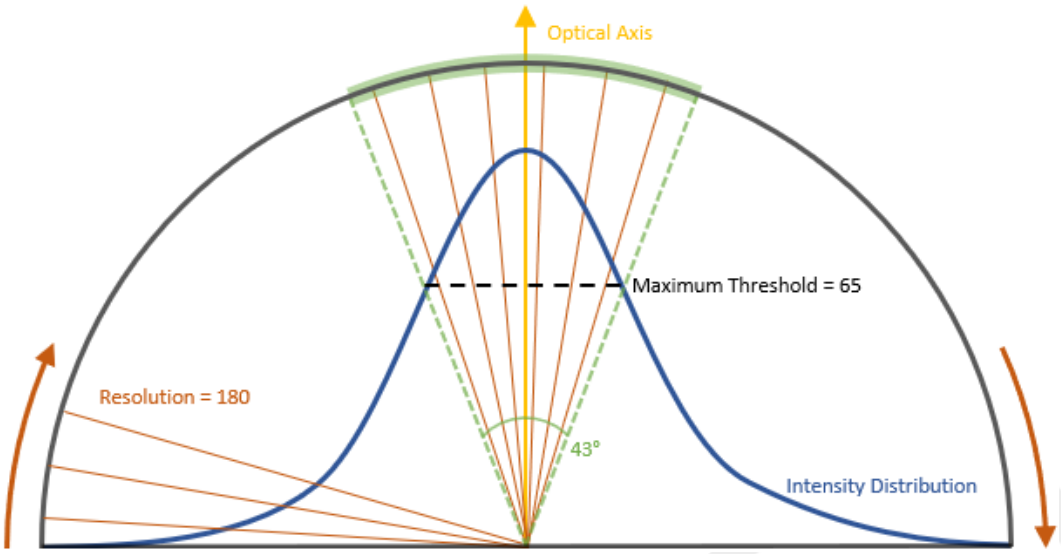


Figure 88. Resolution Factor = 2

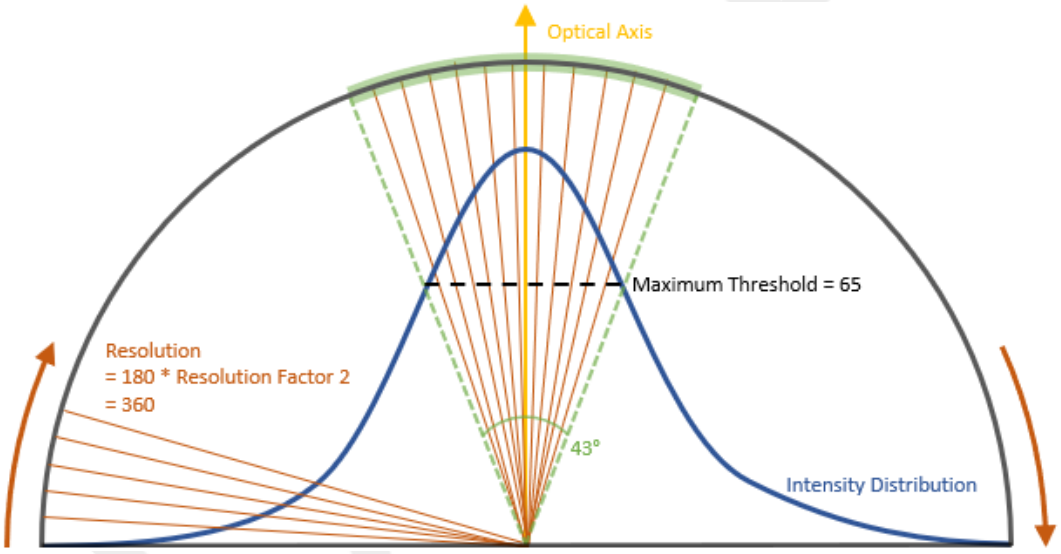
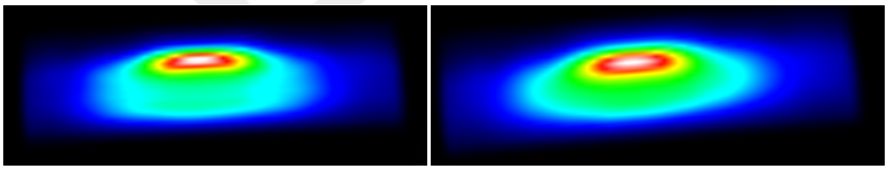


Figure 89. Real Example of Resolution Factor Impact on Intensity Target



Resolution Factor = 3 Resolution Factor = 6

Image (Irradiance Target)

In case of an Image, the resolution corresponds to the resolution of the image. For each pixel of the image, one ray is emitted. However the number of rays may not be enough sometimes.

The Resolution Factor multiplies the number of rays emitted per pixel on the vertical resolution and the horizontal resolution of the image.

In the schematic example (figure 4), the resolution of the image is 4 * 4, meaning 16 pixels. With a Resolution Factor = 1, one ray per pixel is emitted, resulting in 16 rays emitted.

In the schematic example (figure 5), the resolution of the image is 4 * 4, meaning 16 pixels. With a Resolution Factor = 2, four rays per pixel are emitted, resulting in 64 rays emitted.

> **Warning:** We recommend you to set a Resolution Factor that does exceed 1024 * 1024 pixels emitted: **Image Resolution * Resolution Factor < 1024 * 1024**.

Figure 90. Resolution Factor = 1

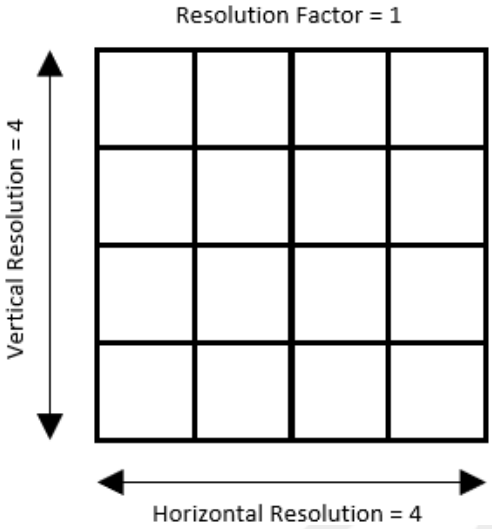
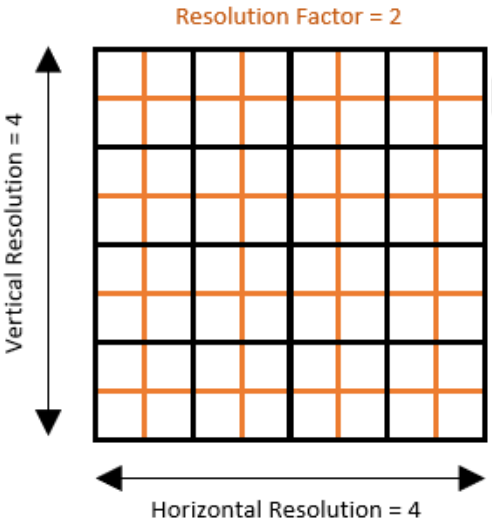
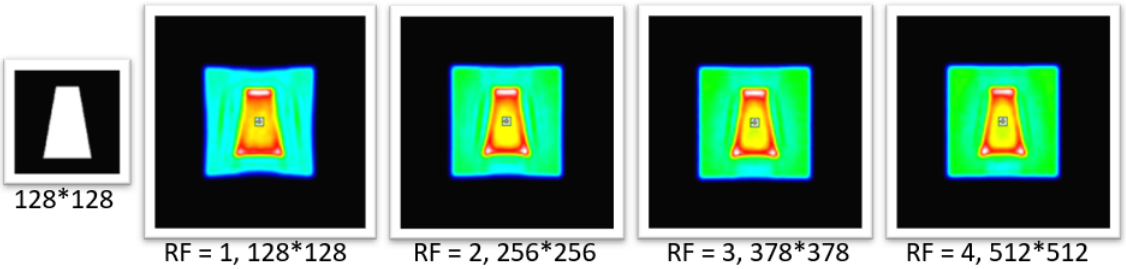


Figure 91. Resolution Factor = 2



DRAFT

Figure 92. Real Example of Resolution Factor Impact on Image Target



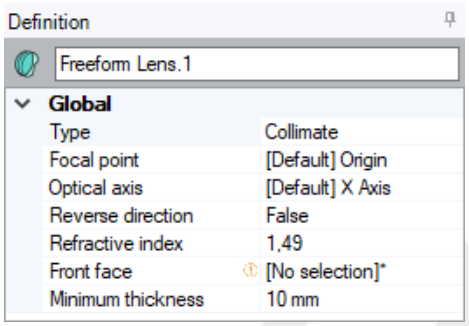
13.10.3. Creating a Freeform Lens

This page shows how to create a freeform lens from a freeform surface.

To create a Freeform Lens:

A surface corresponding to the front face of the lens must already be created.

1. From the **Design** tab, click **Freeform Lens** .



2. Define the axis system of the lens:
   - In the 3D view click
     
     select the lens' focal point and validate.
   - In the 3D view, click
     
     select a line to define the **Optical axis**.
   - or click
     
     and select a coordinate system to autofill the Axis System.
3. Define the Refractive index of the lens.

   The Refractive Index of a material is a pure number that describes how light propagates through that medium. Most transparent materials have refractive indices between 1 and 2. Here the refractive index refers to the index of the lenses' material.
4. In the 3D view, click  to select the front face of your lens (any freeform shape).

DRAFT

> **Note:** Only one surface should be selected for the freeform lens' front face. The selection of multiple surfaces is not supported.

5. In **Minimum thickness**, define a thickness threshold to be respected between the lens' front and back face. The thickness is measured along the optical axis.

> **Note:** The thickness value should not be smaller than 2mm . This value is a target value. This means the algorithm tries to generate a thickness that is the closest possible to the one defined but might not reach the exact value.

6. Click **Compute** to build the feature.

Related information

Freeform Lens Overview on page 625
The freeform lens allows you to create a collimating lens, a user irradiance target-based lens or a user intensity target-based lens with a complex shape.

13.10.4. Creating a Freeform Lens Based on an Irradiance Target

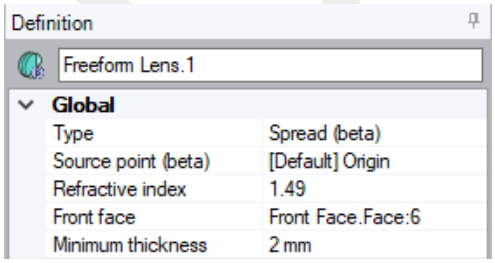
This procedure shows how to create a freeform lens based on an irradiance target. The spread type allows you to control the light distribution on the target you define.

> **Note:** The Freeform Lens based on an irradiance target definition is in BETA mode for the current release.

To create a Freeform Lens:

A surface corresponding to the front face of the lens must already be created.

1. From the **Design** tab, click **Freeform Lens**.



2. In the **Global** section, select the **Spread** Type.
3. In the 3D view, click and select a point to define the **Source point**.
4. Define the **Refractive index** of the lens.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

The Refractive Index of a material is a pure number that describes how light propagates through that medium. Most transparent materials have refractive indices between 1 and 2. Here the refractive index refers to the index of the lenses' material.

5. In the 3D view, click  to select the front face of your lens (any freeform shape).

> **Note:** Only one non-trimmed surface with exactly four sides should be selected for the freeform lens' front face. The selection of multiple surfaces is not supported.

6. In **Minimum thickness**, define a thickness threshold to be respected between the lens' front and back face. The thickness is measured along the optical axis.

> **Note:** This value is a target value. This means the algorithm tries to generate a thickness that is the closest possible to the one defined but might not reach the exact value.

7. In the **Target** section, select the **Irradiance** Target type.

Target	
Target type (beta)	Irradiance
Origin (beta)	Point
X axis (beta)	[Default] Y Axis
Y axis (beta)	[Default] Z Axis

8. Define the axis system of the target:
   a) Click  and select a point to define the **Origin** of the Target.
   b) Click  and select a line to define the horizontal axis of the sensor.
   c) Click  and select a line to define the vertical axis of the sensor.
9. In **Type**, define the uniformity of beam pattern:
   - Select **Uniform** for a uniform light spread on the Target, and set **X half size** and **Y half size** to define the size of the target.

     Each half size is generated from the target origin. A X half size of 100 mm means the X size of the target is 200 mm.

Type (beta)	Uniform
X half size (beta)	100 mm
Y half size (beta)	50 mm

   - Select **Gaussian** for a Gaussian distribution and define:

Type (beta)	Gaussian
X half size (beta)	100 mm
Y half size (beta)	50 mm
X FWHM (beta)	100 mm
Y FWHM (beta)	50 mm

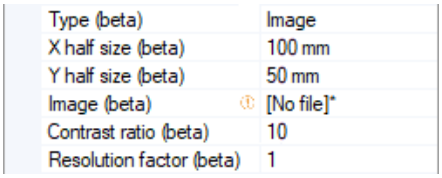
     - **X half size** and **Y half size** to define the size of the target.

       Each half size is generated from the target origin. A X half size of 100 mm means the X size of the target is 200 mm.

     - the FWHM for X and Y.

   - Select **Image** to define the light spread based on an image, and define:

DRAFT



- **X half size** and **Y half size** to define the size of the target.

  Each half size is generated from the target origin. A X half size of 100 mm means the X size of the target is 200 mm.

- the image file (*.png, *.jpg or *.bmp file).
- the **Contrast ratio** with an integer value equal or superior to 2.

  The Contrast corresponds to the ratio between the minimum illuminance value (associated in RGB=(0,0,0)) and maximum illuminance value (associated in RGB=(255,255,255)) on the photometric map.

- The **Resolution factor** with an integer value equal or superior to 1.

  The goal of the Resolution Factor is to densify the number of rays launched from the Image File target to allow to better design the Backface of the Freeform Lens.

10. Click **Compute** to build the feature.

Related information

Freeform Lens Overview on page 625
The freeform lens allows you to create a collimating lens, a user irradiance target-based lens or a user intensity target-based lens with a complex shape.

13.10.5. Creating a Freeform Lens Based on an Intensity Target

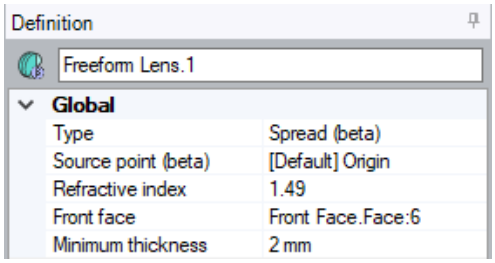
This procedure shows how to create a freeform lens based on an intensity target. The spread type allows you to control the light distribution on the target you define.

> **Note:** The Freeform Lens based on an intensity target is in BETA mode for the current release.

To create a Freeform Lens:

A surface corresponding to the front face of the lens must already be created.

1. From the **Design** tab, click **Freeform Lens**.



Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

2. In the **Global** section, select the **Spread** Type.
3. In the 3D view, click  and select a point to define the **Source point**.
4. Define the **Refractive index** of the lens.

   The Refractive Index of a material is a pure number that describes how light propagates through that medium. Most transparent materials have refractive indices between 1 and 2. Here the refractive index refers to the index of the lenses' material.
5. In the 3D view, click  to select the front face of your lens (any freeform shape).

   > **Note:** Only one non-trimmed surface with exactly four sides should be selected for the freeform lens' front face. The selection of multiple surfaces is not supported.

6. In **Minimum thickness**, define a thickness threshold to be respected between the lens' front and back face. The thickness is measured along the optical axis.

   > **Note:** This value is a target value. This means the algorithm tries to generate a thickness that is the closest possible to the one defined but might not reach the exact value.

7. In the **Target** section, select the **Intensity** Target type.

Target	
Target type (beta)	Intensity
Optical axis (beta)	[Default] X Axis
Reverse direction	True
Orientation axis (beta)	[Default] Y Axis
Reverse direction	False

8. Define the axis system of the target:
   a) Click  and select an optical axis to define the direction of the light.
   b) Click  and select an orientation axis for the target.
9. In **Type**, define the uniformity of beam pattern:
   - Select **Uniform** for a uniform light spread on the Target, and set **X half spread** and **Y half spread** to define the size of the target.

Type (beta)	Uniform
X half spread (beta)	15 °
Y half spread (beta)	20 °

   - Select **Gaussian** for a Gaussian distribution and define:

Type (beta)	Gaussian
X half spread (beta)	15 °
Y half spread (beta)	20 °
X FWHM (beta)	20 °
Y FWHM (beta)	30 °

     - **X half spread** and **Y half spread** to define the size of the target.
     - the FWHM for X and Y.
   - Select **Intensity file** to define the target intensity distribution based on an intensity file, and:

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Type (beta)	Intensity file
Intensity file (beta)	Light.ies
Contrast ratio (beta)	5
Resolution factor (beta)	2
Maximum threshold (beta)	10 %

- load a *.ies, *.ldt, *.xmp intensity file

  Note: SAE XMP (non-conoscopic) are not supported as intensity file. If you want to use it, make sure to open SAE XMP map in the **IESNA LM-63 Viewer** and convert it to IES.

- define the **Contrast ratio** with an integer value equal or superior to 2.

  The Contrast corresponds to the ratio between the minimum illuminance value (associated in RGB=(0,0,0)) and maximum illuminance value (associated in RGB=(255,255,255)) on the photometric map.

- define the **Maximum threshold**.

  The Maximum threshold corresponds to the percentage of the maximum value of the intensity file. Defining this threshold means that all values of the intensity file below it are not considered. This helps you concentrate on the useful part of the light distribution by removing the unimportant part of it.

- The **Resolution factor** with an integer value equal or superior to 1.

  The goal of the Resolution Factor is to densify the number of rays launched from the Intensity File target to allow to better design the Backface of the Freeform Lens.

10. Click **Compute** to build the feature.

Related information

Freeform Lens Overview on page 625
The freeform lens allows you to create a collimating lens, a user irradiance target-based lens or a user intensity target-based lens with a complex shape.

13.11. Micro Optical Stripes

Micro Optical Stripes are thin stripes designed on a light guide.

Note: The Micro Optical Stripes feature is in BETA mode for the current release.

13.11.1. Micro Optical Stripes Overview

The Micro Optical Stripes feature helps you create a feasible light guide by defining a Tool Bit Shape used for processing the Light Guide mold.

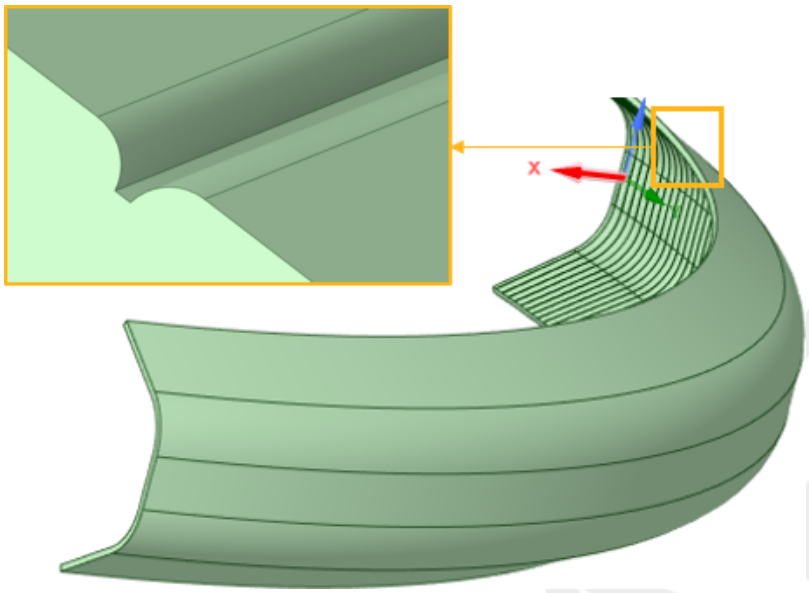
The Micro Optical Stripes feature is a light guide whose body is based on two guide curves, and the stripes are created thanks to a tool bit shape that designs the mold of the light guide. Light travels through the guide thanks to successive total internal reflections and a part of this light is extracted through reflection on the stripes.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Light Guides are optical components that are highly efficient in transmitting light. For this reason, they have various possible applications.

They are commonly used for interior design, accent lighting, backlighting, exterior lighting or tail lamps.

Figure 93. Example of a light guide with micro optical stripes



13.11.2. Understanding the Micro Optical Stripes Parameters

The following page helps you understand the parameters used to create Micro Optical Stripes.

Construction Type

The Construction types define how you want the stripes to be processed in the Support surface.

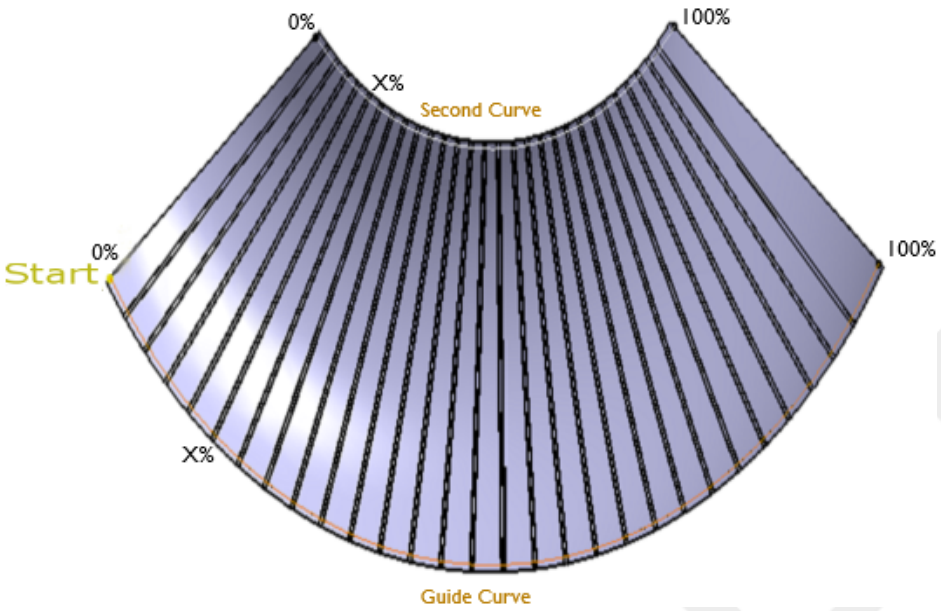
Constant Ratio

The relative position of a stripe along the second curve is equal to the relative position of the stripe along the Guide Curve according to the length of the curves.

> **Note:** When building Micro Optical Stripes based on Constant Ratio, we recommend you to define the **Support side** as **Inner surface**, or the **Projection type** as **Normal To Support Surface**. Otherwise, you may face with a constant ratio not being respected along the curves.

DRAFT

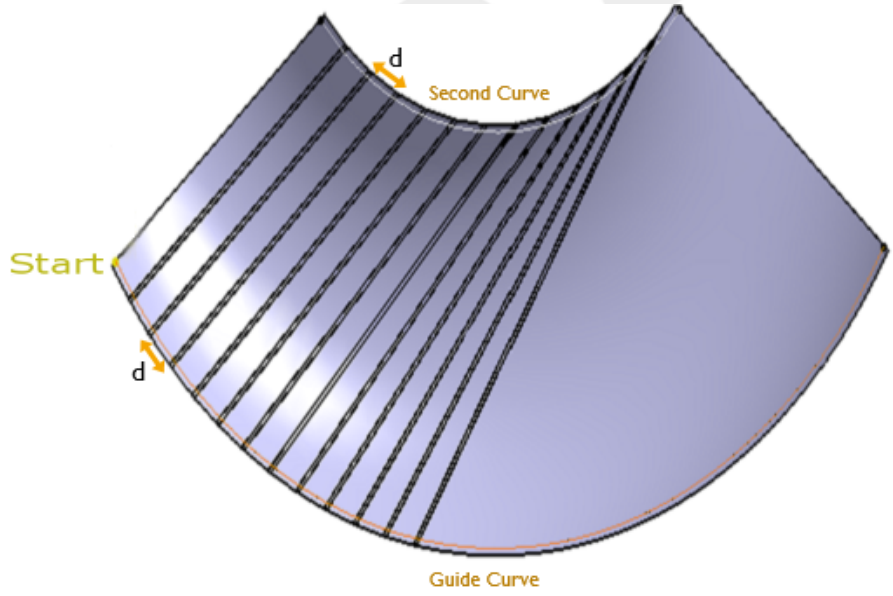
Figure 94. Construction type: Constant ratio



Same Pitch

The distance (pitch) between two stripes on the Second Curve is equal to the distance between these two stripes on the Guide Curve: $d_{Guide\ Curve} = d_{Second\ Curve}$.

Figure 95. Construction type: Same pitch



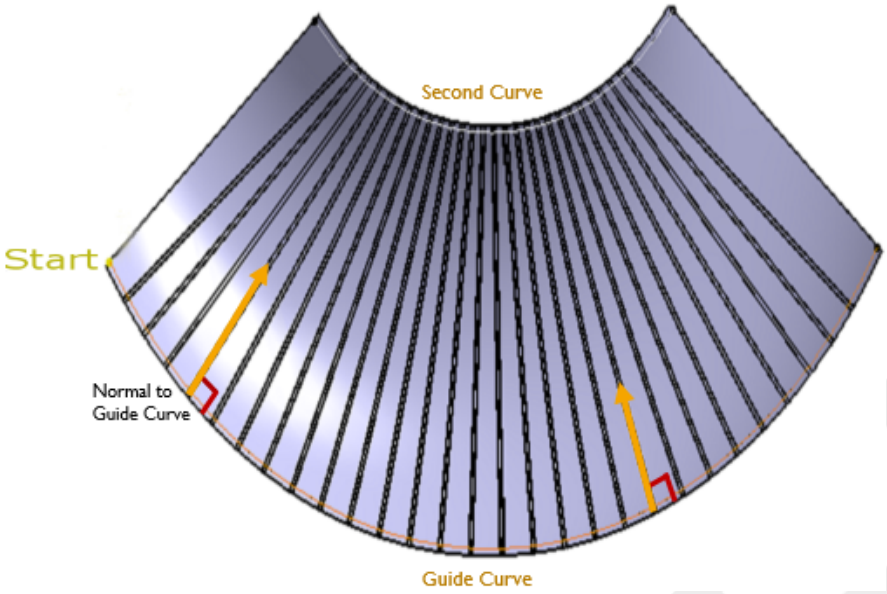
Normal to Guide Curve

Stripes are created from the Guide Curve to the Second Curve according to the normal direction to the Guide Curve at their specific starting point.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Figure 96. Construction type: Normal to guide curve



Projection Type

The Projection Type is the projection of the stripes on the support surface according to the Normal to the support surface or the Optical axis.

Figure 97. Projection of the stripes according to the optical axis

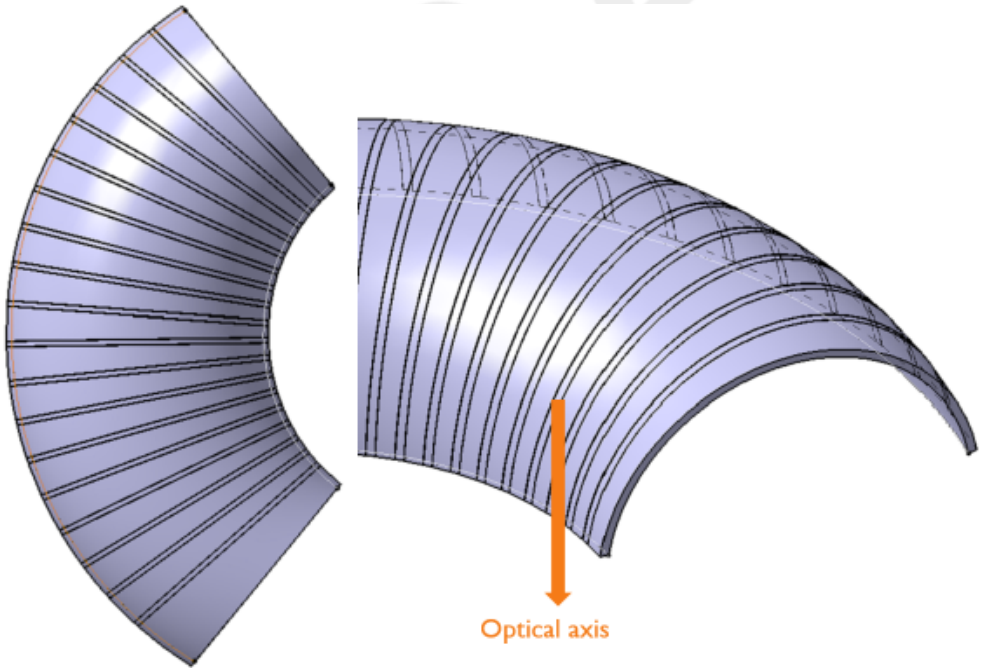
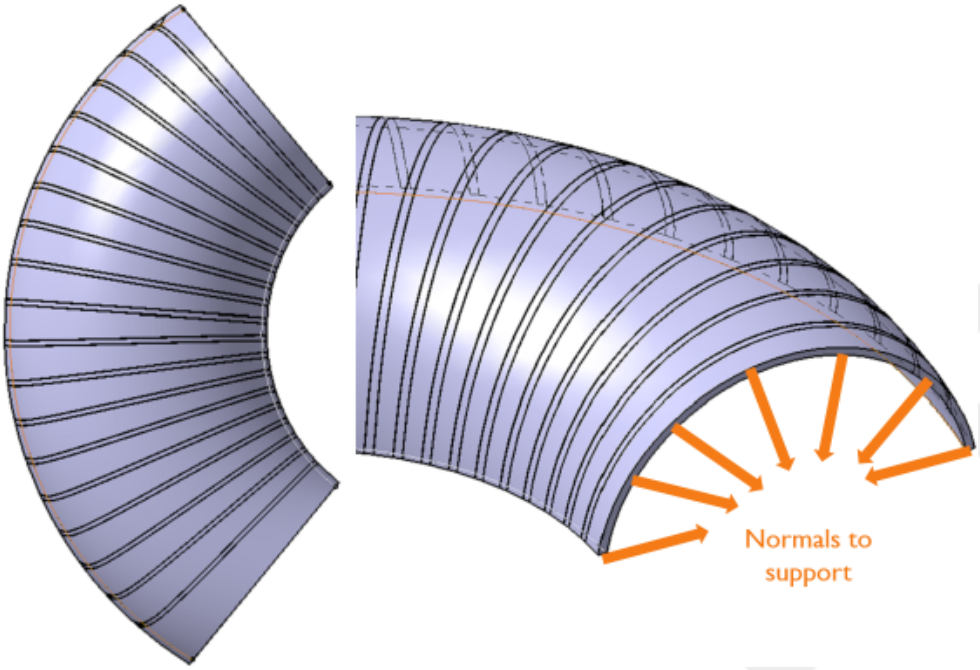


Figure 98. Projection of the stripes according to the normals to the support surface

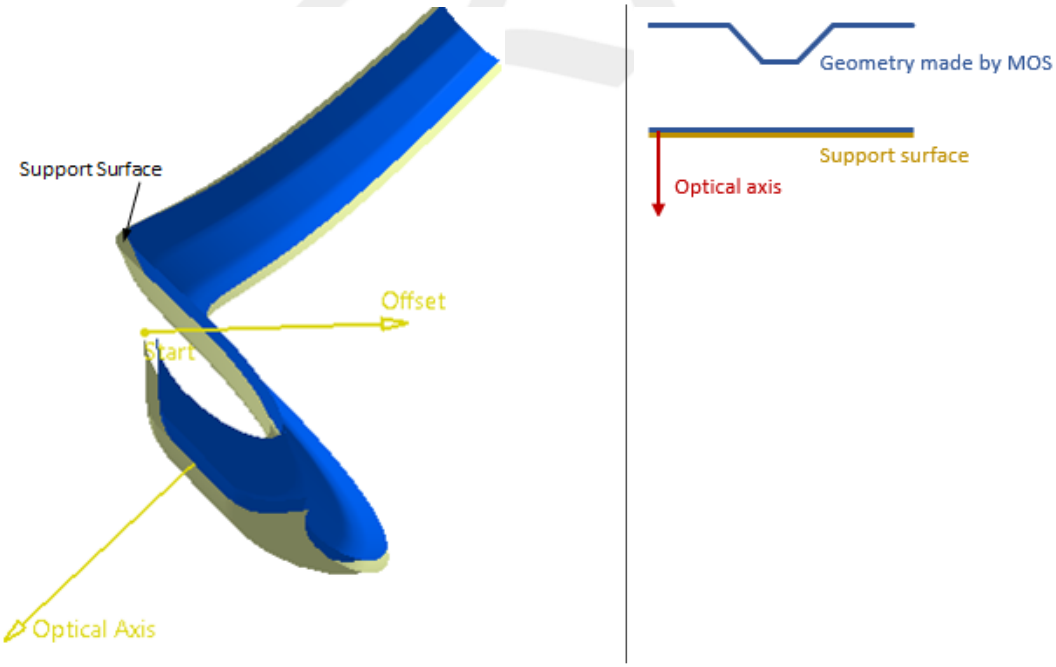


Support Side

The Support side determines on which side of the Light Guide the support surface is placed.

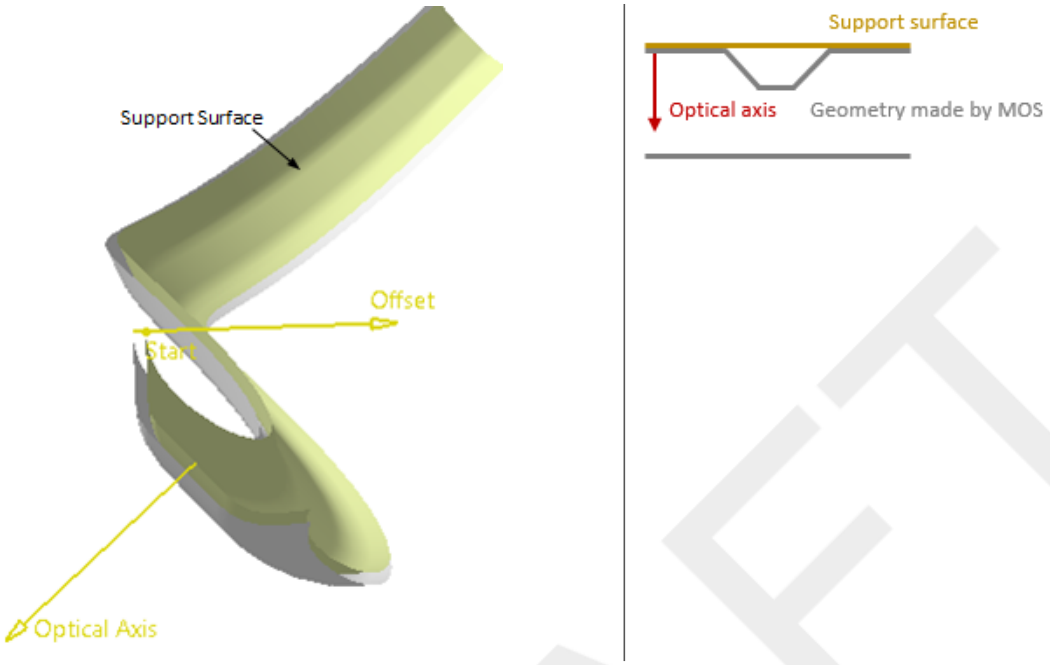
Outer Surface

The support surface is placed on the exit face of the light.



Inner Surface

The support surface is placed on the emissive face.



Tool Bit Shape and Control Points

The Tool Bit Shape defines the profile of the tool used to process the mold, and the Control Points control the stripes parameters at a specific location on the Guide curve.

> **Note:** Variation of stripes parameters is considered to be linear between two Control Points.

Figure 99. Start Side Mold Creation

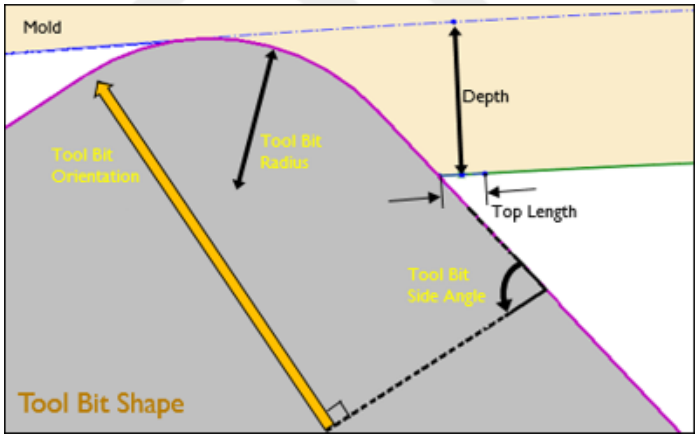


Figure 100. End Side Mold Creation

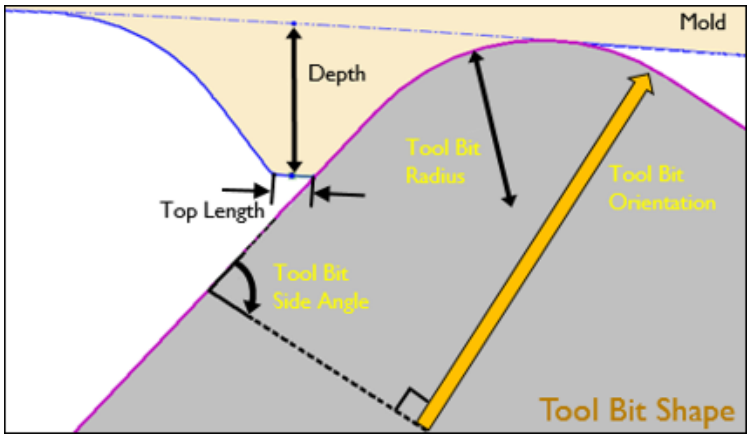
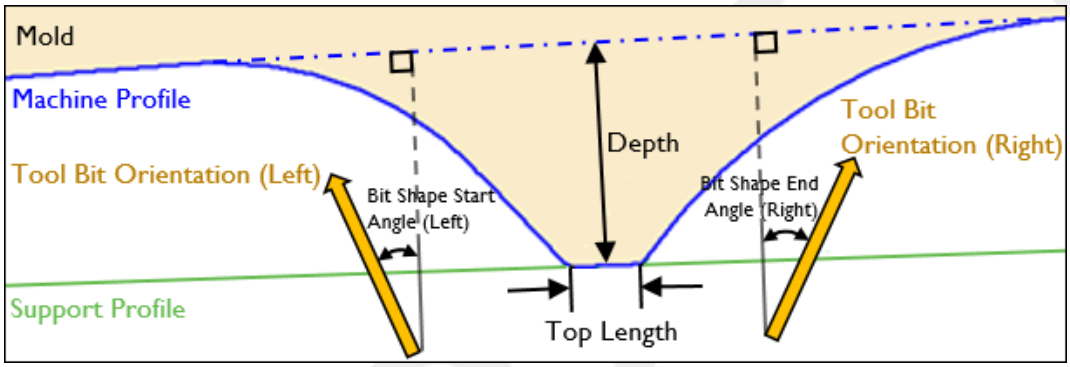


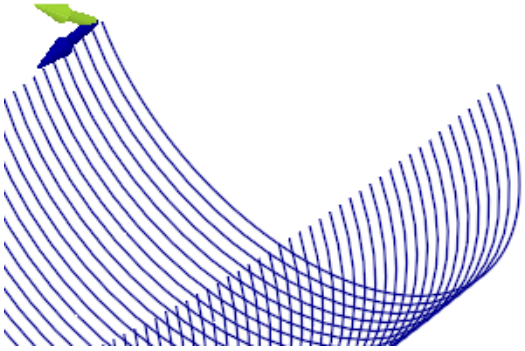
Figure 101. Mold Created



Preview Stripes

As the Micro Optical Stripes generation can take time, you can preview the stripes to see how would the stripes be generated along the guide.

When you preview the stripes, only the top curves used by the tool to create the stripes are displayed. To see the different construction curves used by the tool to create the stripes, refer to **Extract Tooling Paths** below.



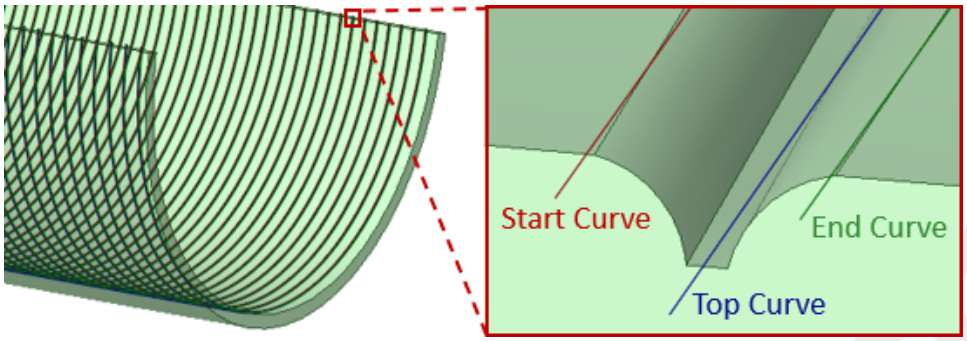
Extract Tooling Path

Extract Tooling Path is a Micro Optical Stripes functionality that allows you to generate as geometry the construction curves used by the tool to create the stripes. This is particularly useful for the communication between design teams

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

and manufacturing teams, as it helps verify how machine tool moves in 3D in detail and check manufacturing feasibility.

Extract Tooling Path generates 3 curves per stripe (top curve, start curve, end curve).



Extract as CSV File

Extract as CSF file is a Micro Optical Stripes functionality that allows you to generate a CSV containing information on the position and orientation of the machine tool. These information are useful to the manufacturing teams to check manufacturing feasibility and create the appropriate machine tool.

	A	B	C	D	E	F
1	Prism	Position (%)	Depth (mm)	TopLength (mm)	StartAngle (°)	EndAngle (°)
2	1	0	0.025	0.01	0	0
3	2	0.353678	0.025	0.01	0	0
4	3	0.707355	0.025	0.01	0	0
5	4	1.06103	0.025	0.01	0	0
6	5	1.41471	0.025	0.01	0	0
7	6	1.76839	0.025	0.01	0	0
8	7	2.12207	0.025	0.01	0	0
9	8	2.47574	0.025	0.01	0	0
10	9	2.82942	0.025	0.01	0	0
11	10	3.1831	0.025	0.01	0	0

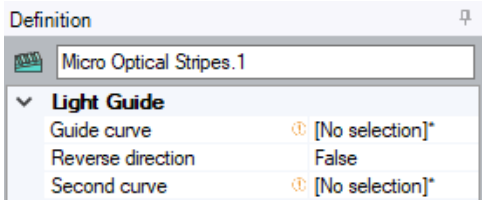
13.11.3. Creating Micro Optical Stripes

This procedures shows how to create Micro Optical Stripes from any guide curve.

To create Micro Optical Stripes:

Construction curves outlining the micro optical stripes lengths should already be sketched.

1. From the **Design** tab, click **Micro Optical Stripes**  .
2. Select the **Guide curve** and **Second curve**:

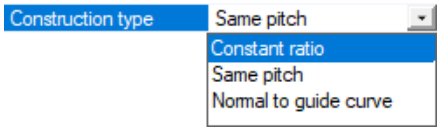


The Guide curve and the Second curve define the starting and ending points of the stripes processed in the Support surface according to the Construction type and the Projection type.

a) In the 3D view, click  and select a curve as **Guide curve**.

b) In the 3D view, click  and select a curve as **Second curve**.

3. Select the **Construction type** to define how to process the stripes in the Support surface:



- **Constant ratio**: the relative position of a stripe along the Second curve is equal to the relative position of the stripe along the Guide curve according to the length of the curves.

> **Note:** When building Micro Optical Stripes based on Constant Ratio, we recommend you to define the **Support side** as **Inner surface**, or the **Projection type** as **Normal To Support Surface**. Otherwise, you may face with a constant ratio not being respected along the curves.

- **Same pitch**: the distance (pitch) between two stripes on the Second curve is equal to the distance between these two stripes on the Guide curve.
- **Normal to guide curve**: stripes are created from the Guide curve to the Second curve according to the normal direction to the Guide curve at their specific starting point.

4. In the 3D view, click  and select a line to define the **Optical axis**.

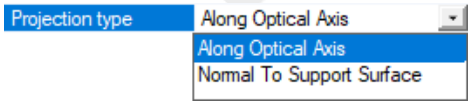
The Optical axis is used to define the orientation in which the light is going to be extracted from the light guide.

> **Note:** The Optical axis must not be collinear with any tangent to the support surface.

5. In the 3D view, click  and select the **Support surface** on which the stripes are projected according to the Projection type.

> **Note:** A face or a multifaces body can be selected as surface.

6. Select the **Projection type** to define how the stripes are projected on the support surface:

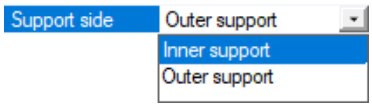


- Select **Along Optical Axis** to project the stripes according to the Optical axis.
- Select **Normal To Support Surface** to project the stripes according to the normal of the support surface.

7. In the **Support side** drop-down list, Select **Inner support** or **Outer support**.

The Support side defines on which side of the light guide the support surface is placed.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



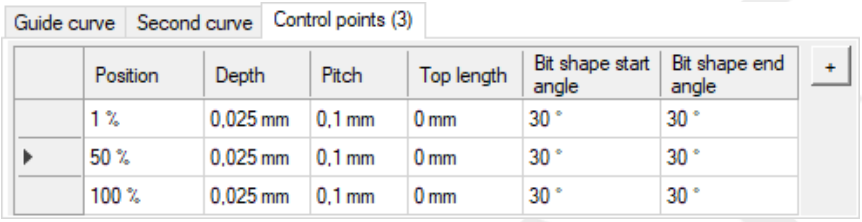
8. Define the **Thickness** of the light guide.
9. In the **Tool Bit Shape** section:



   a) In **Side angle**, define the trajectory of the tangent to the curvature in [0 ; 90].
   b) In **Radius**, define the radius of curvature of the Tool Bit in [0.005 ; infinity[.

   The Tool Bit Shape defines the profile of the tool used to process the mold.

10. In the **Control Points** section, click  to add a control point.
11. In the **Control Points** Definition panel, for each control point created:



   a) In **Position**, define the position of the control point along the light guide in percentage of the light guide.
   b) In **Depth**, define how deep the Tool Bit goes to create the mold in [0 ; infinity[.

      The depth represents the depth of the stripes on the light guide.

   c) In **Pitch**, define the distance between two consecutive stripes in [0 ; infinity[.
   d) In **Top length**, define the offset between the Start and End Tool Bit shapes in [0 ; infinity[.
   e) In **Bit shape start angle** and **Bit shape end angle**, define the angles of attach of the Tool Bit to process the mold in [0 ; 90].

12. In the **Manufacturing** section:



   a) In the 3D view, click  and select the **Drafting axis**.

      The Drafting axis is the direction in which the light guide is going to be removed from the mold.

   b) Define the **Drafting angle** in [0 ; 90[.

      The Drafting angle is the angle to respect in order to be able to remove the light guide from the mold.

13. If you want to want to preview how the stripes will be generated according to the parameters you set, in the **Design** tree, right-click the Micro Optical Stripes and click **Preview Stripes**.
    The preview shows the top curve of each stripe.
14. Click **Compute**  to build the feature.

> **Note:** Stripes on edges of the guide that cannot be entirely generated are not built at all.

DRAFT

The Micro Optical Stripes are created and appear both in the tree and in the 3D view.

13.11.4. Extracting Tooling Path

In order to facilitate communication between design teams and manufacturing teams, you can extract as geometry the paths that the machine tool will follow to create the stripes on the guide.

You must have generated Micro Optical Stripes.

1. In the **Design** tree, right-click the created **Micro Optical Stripes**.
2. Click **Extract Tooling Paths**.
   For more information, refer to the Extract Tooling Path section.

Extract Tooling Path creates 3 curves per stripe (top curve, start curve, end curve) that are generated as geometry in the Structure tree in a dedicated component named "Micro_Optical_Stripes_feature_name"_Tooling_path_"Generation_Date"

13.11.5. Exporting As CSV File

In order to facilitate communication between design teams and manufacturing teams, you can export as CSV file information that describe the position and orientation of the machine tool to create the stripes on the guide.

You must have defined the parameters of the Micro Optical Stripes.

1. In the **Design** tree, right-click the created **Micro Optical Stripes**.
2. Click **Export as CSV file** and save the file in a dedicated folder.
   For more information, refer to the Export As CSV File section.

The CSV file is generated with the machine tool information.

13.12. Post Processing

The Optical Part Design (OPD) Post Processing feature allows you to record design operations on an OPD geometry, that are repeated after each compute of the OPD geometry to avoid doing again all design operations.

> **Note:** The Post Processing is in BETA mode for the current release but is deprecated. Instead we recommend you to use the Block Recording Tool for the same capabilities.
>
> As of version 2024 R1, creating a Post Processing is no longer possible. However you can still load a previous project containing a Post-Processing feature and modify it.

13.12.1. Post Processed Optical Part Design Geometry Modification

The following section helps you modify an Optical Part Design geometry that has been post processed either by modifying the Optical Part Design feature or by modifying the Post Processing feature.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Note: The Post Processing is in BETA mode for the current release but is deprecated. Instead we recommend you to use the Block Recording Tool for the same capabilities.

As of version 2024 R1, creating a Post Processing is no longer possible. However you can still load a previous project containing a Post-Processing feature and modify it.

13.12.1.1. Modifying a Post Processed Optical Part Design Geometry

The following procedure helps you modify an Optical Part Design geometry that has been post processed.

Note: The Post Processing is in BETA mode for the current release but is deprecated. Instead we recommend you to use the Block Recording Tool for the same capabilities.

As of version 2024 R1, creating a Post Processing is no longer possible. However you can still load a previous project containing a Post-Processing feature and modify it.

To modify a post processed Optical Part Design (OPD) geometry:

1. Edit the OPD feature.
2. Modify the needed parameters.
3. In the **Design** tree, open the Post Processing feature related to the OPD feature.
4. In the 3D view, click **Compute** to apply the design operations.

Note: If you activated the Automatic Compute on the Post Processing feature, the design operations are automatically reapplied on the OPD geometry without a manual compute.

The design operations are applied on the modified OPD geometry without having to reapply them manually. Links to Speos features are kept.

You can recompute the simulation using the OPD geometry to regenerate and update the results.

13.12.1.2. Modifying a Post Processed Optical Part Design Geometry with Post Processing

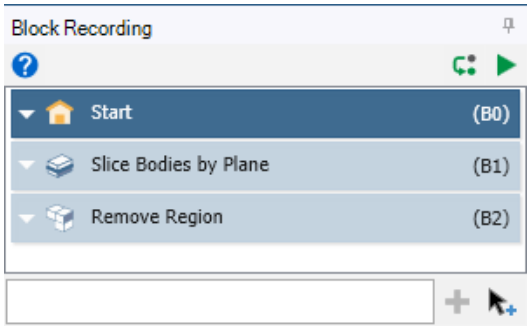
The following procedure helps you modify a post processed Optical Part Design geometry with the Post Processing feature.

Note: The Post Processing is in BETA mode for the current release but is deprecated. Instead we recommend you to use the Block Recording Tool for the same capabilities.

As of version 2024 R1, creating a Post Processing is no longer possible. However you can still load a previous project containing a Post-Processing feature and modify it.

To modify a post processed Optical Part Design (OPD) geometry with the Post Processing:

1. Edit the Post Processing feature related to the OPD feature.
2. In the 3D view, click **Start recording**  .



   The **Block Recording** panel opens with the list of design operations already applied and recorded.
3. Add or modify any SpaceClaim design operation on the OPD geometry.
4. Once you applied the design operations, open the Post Processing feature and click  to stop recording.
5. In the 3D view, click **Compute**  to apply the design operations.

> **Note:** If you activated the Automatic Compute on the Post Processing feature, the design operations are automatically reapplied on the OPD geometry without a manual compute.

The design operations are applied on the modified OPD geometry without having to reapply them manually. Links to Speos features are kept.

You can recompute the simulation using the OPD geometry to regenerate and update the results.

DRAFT

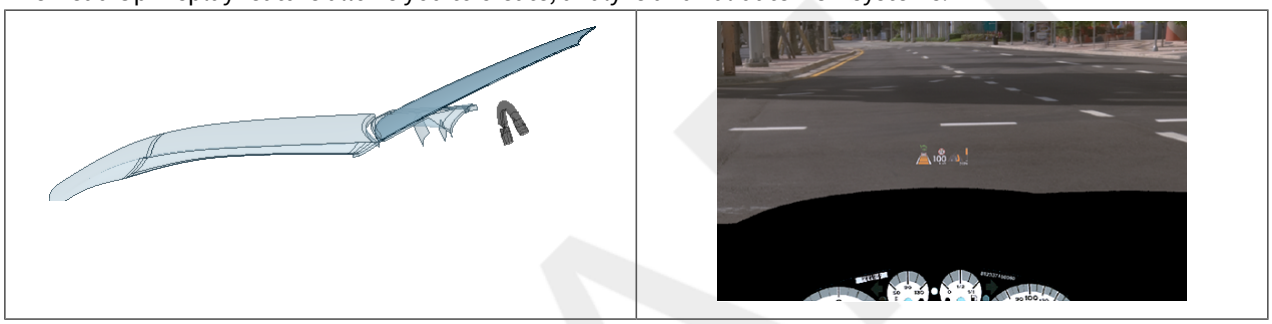
14: Head Up Display

This section describes the Head-Up Display design and analysis features.

> **Important:** This feature requires **Speos HUD Design and Analysis** add-on and Premium or Enterprise license.

14.1. Head Up Display Overview

The Head Up Display feature allows you to create, analyze and validate HUD systems.



A Head-Up Display (HUD) is a system that displays an image in the field of view of the driver. Usually this image is used to give information to the driver (the car speed for example).

From design to experience, Speos HUD feature encompasses several sub features dedicated to every step of the HUD system analysis:

- Design: HUD Optical Design (HOD) allows you to model the HUD system in the CAD interface.
- Analyze: HUD Optical Analysis (HOA) allows you to analyze the HUD system by providing a detailed report and metrics describing the overall optical system's performance.
- Experience: Visualize and experiment your system to anticipate errors and control the quality of the HUD system. Use Speos features to visualize the virtual image as perceived by the driver, appraise polarization effects, observe stray light or simulate PGUs using TFT or DLP technologies.

HUD Optical Design (HOD)

HOD creates the optical system for automotive head-up displays.

The optical shapes of mirrors and eventually combiner are optimized to provide the best virtual image quality according user defined criteria and from inputs such as:

- Eyebox: position, size, different drivers size
- Windshield: inner surface
- Relative position of optical components: distance, orientation
- Target image: distance, look over angle, look down angle, field of view

The resulting optical shapes, generated as native surfaces, are naturally compatible with geometric operations of the CAD platform. HOD provides a seamless solution to achieve a complete optical system and avoids the inevitable drawbacks caused by the usage of separate tools (accuracy losses, manual transfer operations, multiple product definitions and specific process for mold design).

DRAFT

It takes advantage of the standard capabilities of the CAD platform such as direct modeling and features associativity enabling rapid propagation of design changes and quick investigation of ‘what if’ scenarios.

Thanks to a dedicated user interface, HOD is not restricted to optical specialists and enables automotive engineers to perform feasibility studies at concept phase thanks to iteration loops with packaging, human factors and glazing departments. HOD also supports the engineering phase for optical design refinement by experts.

HUD Optical Analysis (HOA)

HOA allows you to quantify the quality of virtual image of automotive head-up displays.

From the digital mockup (PGU output, mirrors, windshield or combiner), it provides optical metrics describing head-up displays optical system performance:

- Virtual image distance, look down angle, look over angle, field of view
- Distortion, smile, trapeze, torsion, magnification, rotation, divergence, etc.
- Ghost
- Field curvature, spot size, astigmatism
- Fully automated, it performs a complete analysis with thousands of measurements for several configurations (driver size, mirror rotation) without any manual operations.

Additionally, HOA can create warping data to feed pre-distortion image correction. Warping information can also be imported. In both cases, optical metrics are calculated simulating the warping processing of the vehicle’s embedded software, reflecting the optical performance of the full HUD system.

Thanks to additional plugins supplementing HOA, analysis report can be customized with specific car manufacturer’s optical metrics definition and acceptance criteria.

14.2. Design

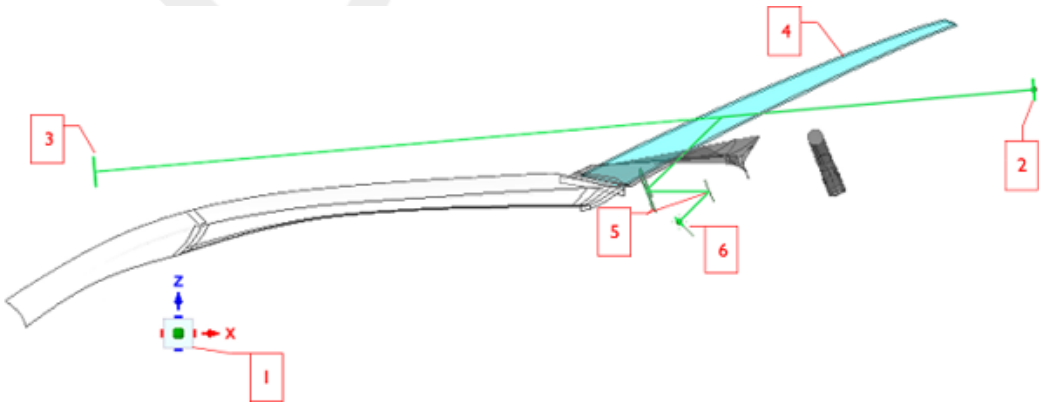
HUD Optical Design (HOD) is a tool to create the optical system for a HUD system. The optical shapes of mirrors and combiner are optimized to provide the best virtual image optical quality.

14.2.1. HUD System Overview

A Head-Up Display (HUD) is a system that displays an image in the field of view of the driver. Usually this image is used to give information to the driver (the car speed for example).

A Head-Up Display system is composed of different modules. These modules strongly interact with one another.

A complete HUD system is represented below.



DRAFT

1. The **General** settings allow to define the axis system used by the HUD system and the degree of polynomial equation used to design the mirrors.
2. The **Eyebox** is a uniform grid representing the driver's eye position. A multieyebox mode allows you to create several eyeboxes and simulate the perception of the system by drivers of different heights.
3. The **Target Image** represents the image the HUD system needs to produce. You need to define its position and size. The Virtual Image is the image the HUD system has produced.
4. The **Windshield** corresponds to the inner surface of the CAD geometry that must be selected to be considered by the HUD system.
5. The **Projector** comprises mirrors and the PGU. Mirrors are numbered following the light propagation (so they are numbered from the Picture Generation Unit (PGU) to the Eyebox).
6. The **Picture Generation Unit (PGU)** is a module that emits the light perceived by the driver after reflection on the mirrors and the windshield.

14.2.2. Understanding the HUD Optical Design Parameters

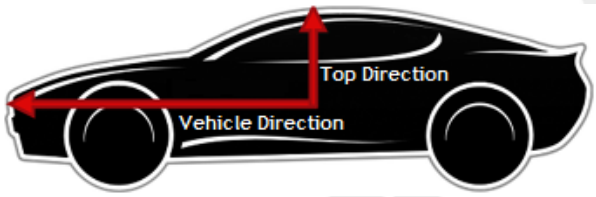
This section allows you to better apprehend the HUD Optical Design definition as it describes its key settings.

14.2.2.1. General

The General settings allow to define the axis system used by the HUD system.

You can define a custom axis system for your HUD Optical Design (HOD) system. If no axis system is set, the following default axis system is used:

- **Vehicle Direction = -X**
- **Top Direction = +Z**



14.2.2.2. Eyebox

The Eyebox is a uniform grid representing the driver's eye position. A multieyebox mode allows you to create several eyeboxes and simulate the perception of the system by drivers of different heights.

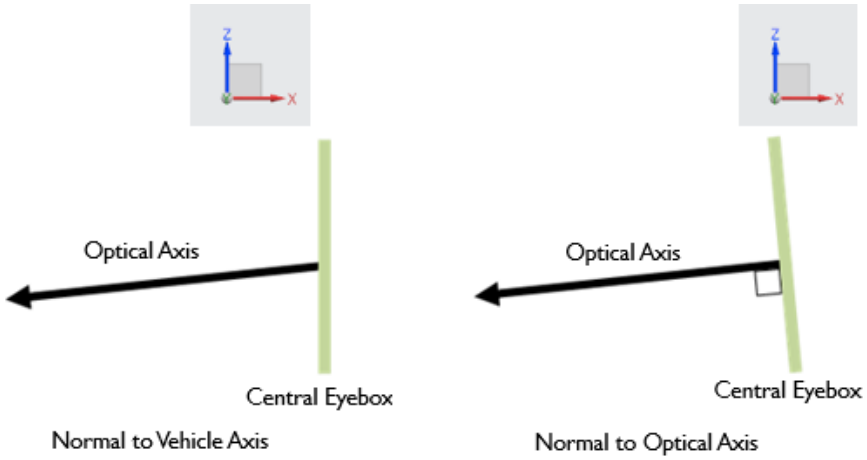
You can create various drivers' heights by specifying several eyeboxes, each one corresponding to a driver's height and you can manage their importance in the system by adjusting their **Weight**

Orientation

The Orientation corresponds to the vertical direction of the Eyebox.

- **Normal to Optical Axis** sets the vertical direction of the Eyebox as normal to the optical axis defined thanks to the Target Image section.
- **Normal to Vehicle Axis** sets the vertical direction of the Eyebox as normal to the vehicle axis defined in the General section.

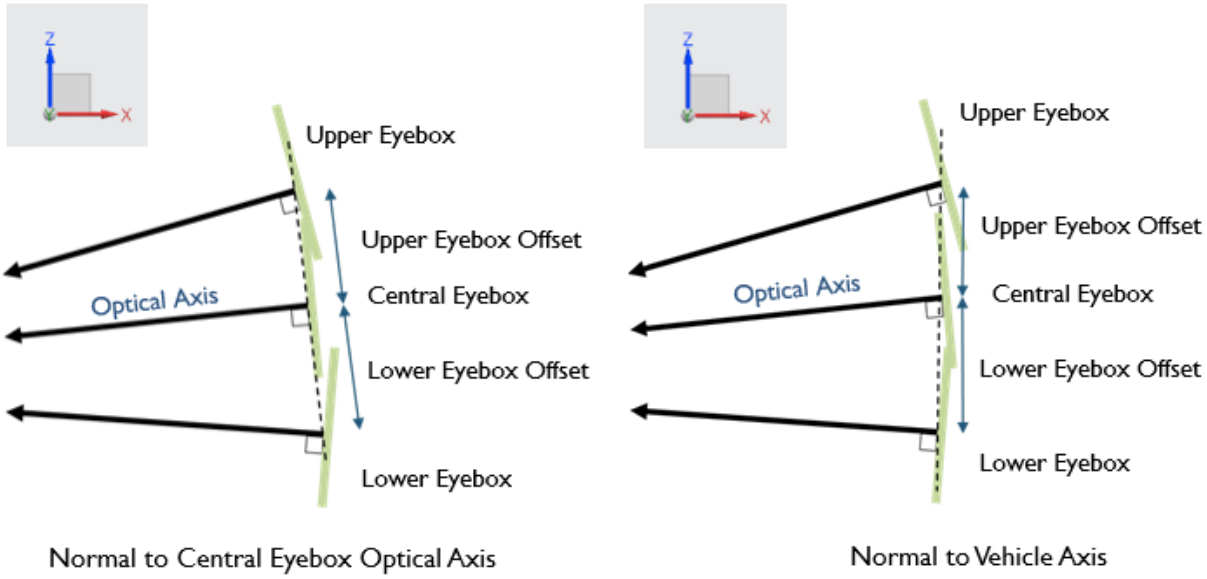
DRAFT



Position Direction

The Position Direction corresponds to the direction used to apply the offsets.

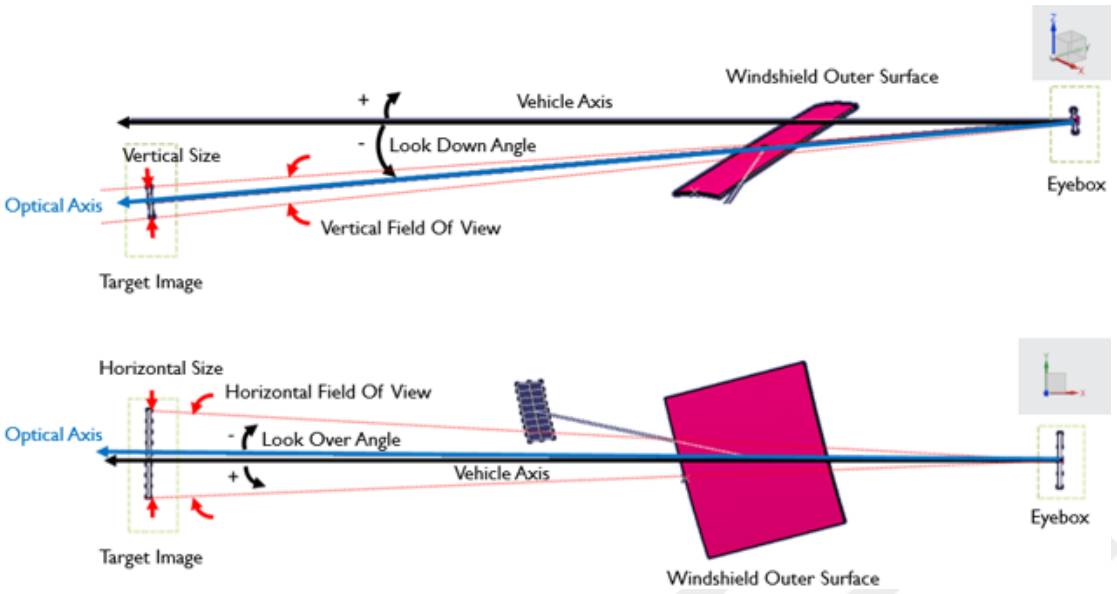
- **Normal to Central Eyebox Optical Axis** sets the position direction as normal to the nominal driver optical axis defined thanks to the Target Image section.
- **Normal to Vehicle Axis** sets the position direction as normal to the vehicle axis defined in the General section.



14.2.2.3. Target Image

The Target Image represents the image that is meant to be projected by the HUD system. You define how and where you want this image to be projected by the system.

Figure 102. Parameters to set when configuring the Target Image.

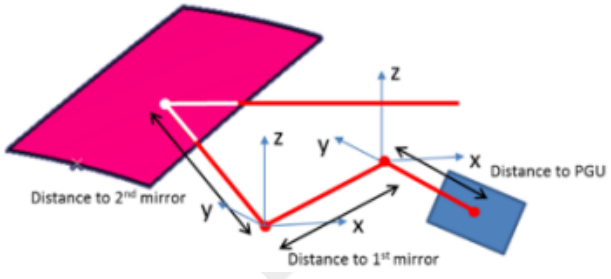


- The **Virtual Image Distance** corresponds to the distance between the center of the nominal Eyebox and the center of the Target Image.
- The **Look Over Angle** corresponds to the horizontal angle between the vehicle axis and the optical axis.
- The **Look Down Angle** corresponds to the vertical angle between the vehicle axis and the optical axis.
- The **Mode** defines how the Target Image is defined, either according to the **Size** (millimeters) or according to the **Field Of View** (degrees).

14.2.2.4. Projector

The Projector corresponds to the mirrors and the PGU. Mirrors are numbered following the light propagation (so they are numbered from the Picture Generation Unit (PGU) to the Eyebox).

Figure 103. Windshield Projector



Projector Parameters

Mirror Type

Freeform mirror type created for the optimization.

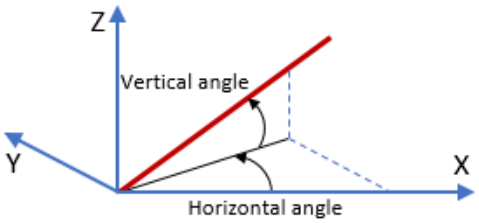
Fold mirror type used to reduce the volume of the projector.

Distance

The distance applied to a mirror or PGU corresponds to the distance separating it from the previous elements.

Horizontal Angle / Vertical Angle

the horizontal and vertical angles corresponding to the orientation of the element in the global axis system.

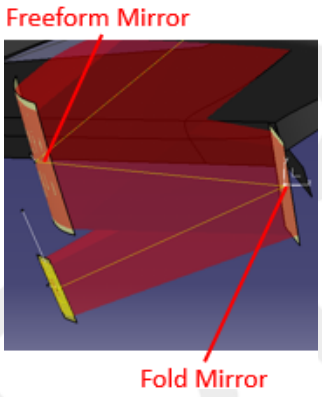
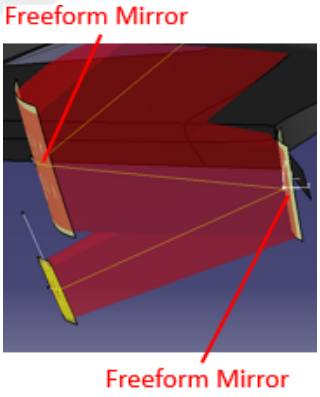


Multimirror Freeform Case

You can create HUD systems with two freeform mirrors. Using multifreeform definition allows you to gain space and obtain a higher quality of results.

The combination of the freeform mirrors helps fold the system and improve the quality of the virtual image by progressively deviating the light.

> **Tip:** In case of a multieyebox analysis, only one of the mirror can turn.

	
Monofreeform mirror HUD system	Multifreeform mirror HUD system

14.2.2.5. Manufacturing

The Manufacturing section allows you to define the degree of polynomial equation used to design the mirrors.

Manufacturing allows you to define the degree of polynomial equation used to design the mirrors.

The degrees influence the shape that the mirror takes. At the end of each computation, a surface export.txt file is generated.

The file contains the coefficients of the extended polynomial equation of the freeform mirror(s) shape created by HUD Optical Design. The file can then be shared with the manufacturers to produce the mirrors.

DRAFT

Note: The manufacturing degree can be set between 1 and 20. The default value is 6. The HOD system computation time depends on the order set here and might take a while, especially when setting high manufacturing degrees.

Note: As the file exports the equation's coefficient, the exported surfaces are unbounded.

14.2.2.6. Advanced Parameters

This page describes 4 parameters used to optimize, correct or adjust the HUD system.

The Advanced parameters are available for the sole purpose of allowing you to maximize and optimize the optical quality of its system. It is neither compulsory nor needed to use or modify the following parameters to obtain a correctly designed HUD system.

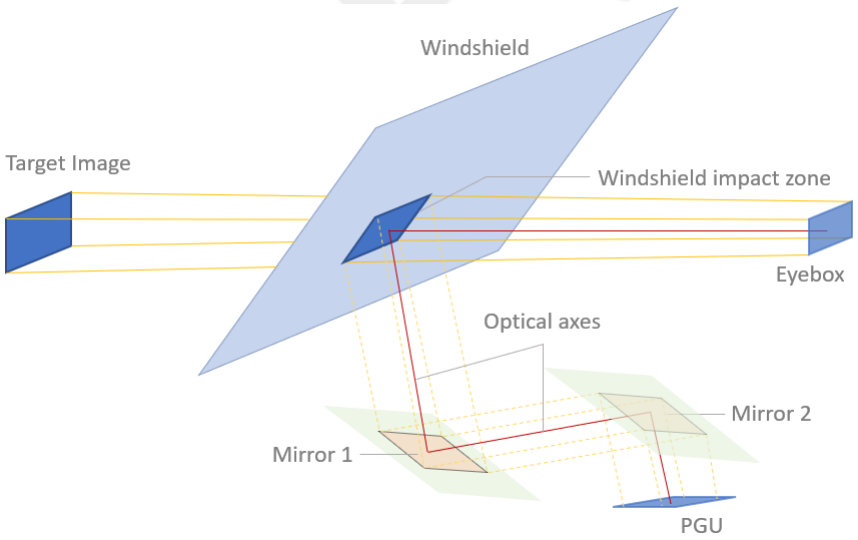
In case of problems during the conception (like a HUD calculation that is too long or results that may not be as precise as expected), they can also assist you in reaching an acceptable solution.

Note: The parameters have initial values that tend to evolve and be modified after the HUD optimization/computation.

Mirror Size Ratio

The Mirror Size Ratio optimizes the mirror's size according to how the HUD system has been designed. When the HUD system is optimized/computed, a resizing process starts.

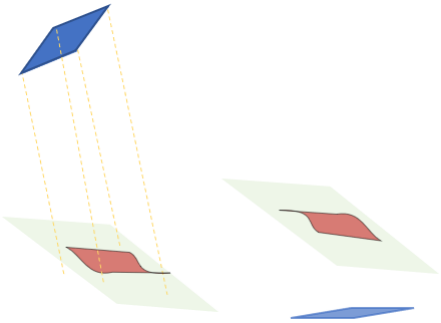
Figure 104. Mirror Size Ratio Before Optimization



Before the optimization, mirrors are considered as planar on an infinite plane. The infinite plane is orthogonal to the bisector of the optical axes defined for the two mirrors. These planar mirrors are based on the projection of the

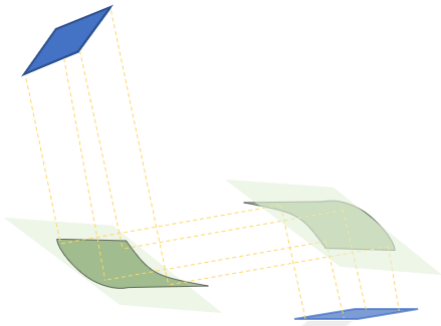
corner's rays made by the Target Image / Eyebox optical volume on the windshield (Windshield impact zone). Planar mirrors have a ratio equal to 1 and all rays intersect the mirrors.

Figure 105. Mirror Size Ratio After Optimization: Too Small



During the optimization the mirrors shapes are modified and can be warped, and some rays may not intersect the mirrors. In this context the Mirror size ratio can help you anticipate a potential set of non-intersecting rays by the mirrors due to the optimization of the mirror shapes that can be warped.

Figure 106. Mirror Size Ratio After Optimization: Optimized



Thus, in the Speos interface, the default value of the Mirror size ratio is set to 1.3 to anticipate these non-intersecting rays in most cases. However, in some case the Mirror size ratio can be increased to extend the mirrors if rays still do not intersect.

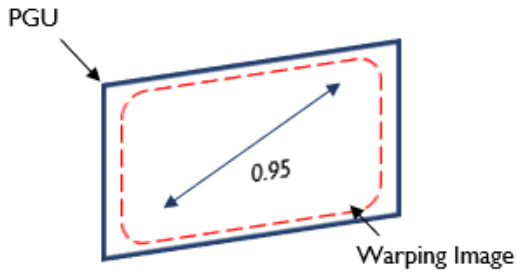
> **Note:** If the mirrors are too large, some parts of the surface will not be used by the HUD system but will still be optimized.

PGU Usage

PGU usage adjusts the ratio between the warping and the Picture Generation Unit to optimize the surface used by the warping.

By default the PGU usage is set at 0.95, that means 95% of the PGU is used for the image warping.

> **Note:** PGU usage does not impact mono freeform design. It only impacts multi freeform design.



Stopping Criterion

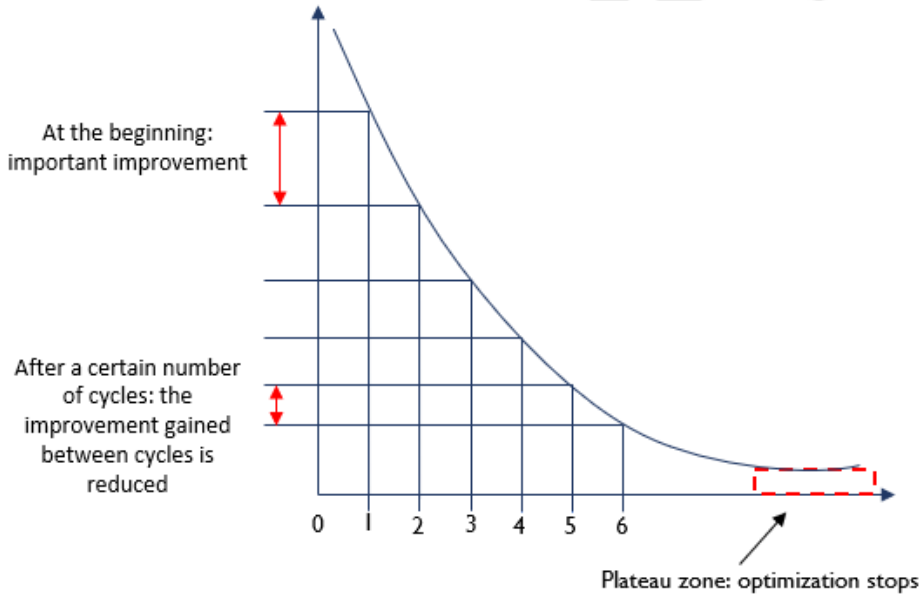
The Stopping criterion defines a threshold value representing the degree of precision to be reached for the optimization to end.

The optimization works in a principle of cycles. At each cycle, the result is optimized and the optical quality increases. After a certain number of cycles, the system becomes more and more defined and the improvement gained on a cycle is greatly reduced. When between two cycles, only 0.05% of improvement is achieved, the optimization stops.

The value is expressed in percentage and is, by default, set to 0.005 (0.05 %).

This criteria is useful to optimize computation time and results according to your needs at different stages of the development process of a HUD system. In the early stages of the design process, you can set this criterion relatively high (like 8%) to gain time during optimization. On the contrary, in the final stages of the design process, you can set this criterion low (< 0.005) so that the system is computed with precision.

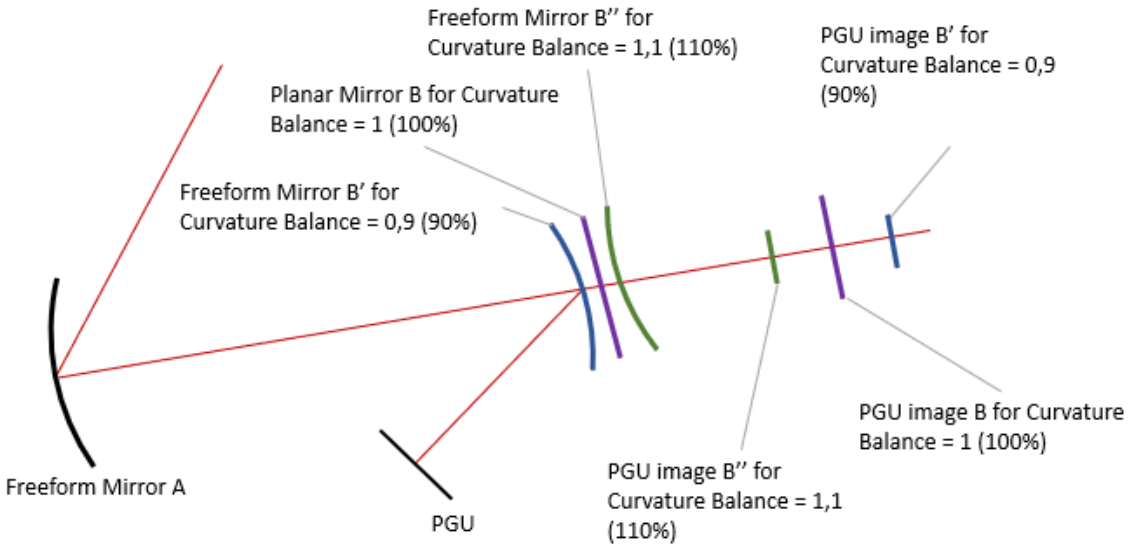
Figure 107. Improvement obtained between optimization cycles



Curvature Balance

Curvature Balance pilots the curvature of the first freeform mirror to get the best image quality. If the curvature balance is left unedited (= 0), it is automatically calculated by the algorithm based on the PGU usage.

Figure 108. Curvature balance effect on HUD.



In this figure we consider the 3 states of the Mirror B (B B' B'') as being at the same position.

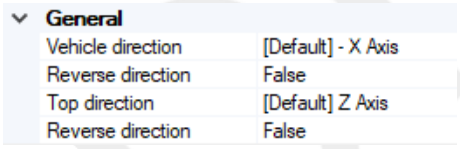
The percentage represents the percentage of the distance between Freeform Mirror A and PGU. For the Mirror B, $d_{MirrorA/PGU} = d_{MirrorA/PGU\ image\ B}$ or for the Mirror B', 90% * $d_{MirrorA/PGU} = d_{MirrorA/PGU\ image\ B'}$

14.2.3. Defining a HUD System with HUD Optical Design

The following procedure helps you define a HUD Optical Design (HOD) to create the optical system for a HUD system.

To define a HUD Optical Design system:

1. In the **Design** tab, in the **Optical Part Design** section, click **HUD Optical Design**  .
2. In the **General** section, if you want to define a custom axis system for the vehicle, define the **Vehicle Direction** and the **Top Direction**.



General	
Vehicle direction	[Default] - X Axis
Reverse direction	False
Top direction	[Default] Z Axis
Reverse direction	False

   a) Click  and select a line to define the vehicle direction.

   b) Click  and select a line to define the top direction.

   For more information on the axis system, refer to General.

3. In the **Eyebox** section, click **Center of eyebox** and select a center for the eyebox in the 3D view.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Eyebox	
Center of eyebox	[No selection]*
Orientation	Normal to optical axis
Horizontal size	130 mm
Horizontal sampling	5
Horizontal resolution	32,5 mm
Vertical size	50 mm
Vertical sampling	3
Vertical resolution	25 mm
Position direction	Normal to vehicle axis
Eyebox configurations	2 items

For more information on the Eyebox, refer to Eyebox.

4. In **Orientation**, define the vertical direction of the Eyebox:
   - Select **Normal to optical axis** to set the vertical direction as normal to the optical axis defined thanks to the **Target Image** section.
   - Select **Normal to vehicle axis** to set the vertical direction as normal to the vehicle axis defined in the **General** section.
5. Define the **Horizontal size** and **Vertical size** of the eyebox.
6. Define the **Horizontal sampling** and **Vertical sampling** of the eyebox.
   The resolution is automatically computed.
7. In **Position Direction**, define the direction used to apply the offsets:
   - Select **Normal to central eyebox optical axis** to set the position direction as normal to the nominal driver optical axis defined thanks to the **Target Image** section.
   - Select **Normal to vehicle axis** to set the position direction as normal to the vehicle axis defined in the **General** section.
8. If you want to create a multi-driver configuration and create several eyeboxes to analyze the system, In the **Eyebox configurations** table, click **Add** +.

Eyebox configurations (2) | Projectors (3)

	Name	Offset	Weight
▶	Central	0 mm	0,5
	Configuration.2	10 mm	0,5

   a) Define their **Name** and **Offset**.
      The default eyebox or nominal eyebox corresponds to the **Central** eyebox with a 0mm-offset. Each added eyebox is created with respect to the nominal eyebox at a distance corresponding to the offset.
   b) In **Weight**, specify the importance of each eyebox in the HUD system. The weight is used to indicate the importance of each eyebox in the results.
9. In the **Target Image** section, in **Virtual Image Distance** specify the distance between the center of the nominal Eyebox and the center of the Target Image.

Target Image	
Virtual image distance	2200 mm
Look over angle	0,5 °
Look down angle	-5 °
Mode	Size
Horizontal size	210 mm
Vertical size	70 mm
Horizontal field of view	5,47 °
Vertical field of view	1,82 °

For more information on the Target Image, refer to Target Image.

DRAFT

10. Set the **Look Over Angle** (horizontal angle) between the vehicle axis and the optical axis.

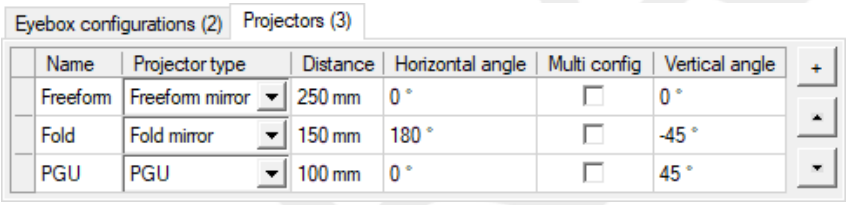
> **Note:** To define the optical axis, HUD Optical Design applies first the **Look Over Angle**, then the **Look Down Angle**.

11. Set the **Look Down Angle** (vertical angle) between the vehicle axis and the optical axis.
12. In **Mode** specify the mode used to define the Target Image:
    - Select **Size** and define the **Horizontal size** and **Vertical size** of the Target Image in millimeters.
    - Select **Field Of View** and define the **Horizontal Field of view** and **Vertical field of view** of the Target Image in degrees.
13. In the **Windshield** section, click  and select the inner surface of the windshield in the 3D view.

> **Note:** A face or a multifaces body can be selected as surface.

> **Note:** Faceted geometries (such as imported *.stl files or other file formats alike) can be used as input geometry.

14. In the **Projectors** table, click  to create new mirrors. For each element of the table:



a) In the **Projector type** column, select **Freeform mirror** to create a mirror used for the optimization, or **Fold mirror** to create a mirror used to reduce the volume of the projector
b) In the **Distance** column, define the distance separating the mirror from the previous element.
c) In **Horizontal angle** and **Vertical angle**, define the angles corresponding to the orientation of the element in the global axis system.

> **Note:** The order of the elements in the table considers the optical path from the windshield towards the Picture Generation Unit (PGU). The bottom line of the table is always the PGU.

When designing the system, a preview indicates if the construction is correct or not. A green display means the elements can be computed without error. A red display means one or more elements cannot be computed.

A green display does not mean relevant results. It only means that the system is considered as correctly defined.

For more information on the Projector, refer to Projector.

15. In the **PGU** section, define the characteristics of the PGU:

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

PGU	
Horizontal sampling	3
Vertical sampling	3
Characteristics	User Defined
Horizontal size	40,893 mm
Horizontal resolution	480
Vertical size	20,447 mm
Vertical resolution	240

- Select a predefined PGU among the available models.

  The name corresponds to the characteristics of the PGU. Example: **1.8"** = size (in inches), **2:1** = horizontal to vertical size ratio of the screen, **480x240** = PGU pixel resolution.

- Select **User Defined** and manually define the **Horizontal** and **Vertical sizes** of the PGU in millimeters, and the **Horizontal** and **Vertical resolutions** in pixel.

16. In the **Manufacturing** section, define the **X degree** and **Y degree** of polynomial equation used to design the mirrors.

Manufacturing	
X degree	6
Y degree	6

For more information on the Manufacturing, refer to Manufacturing.

17. In the **Advanced Parameters** section, if you want to maximize, optimize or adjust your HUD system, you can define:

Advanced	
Mirror size ratio	1,3
PGU Usage	0,95
Stopping criterion	0,005
Curvature Balance	0

- the **Mirror Size Ratio** to optimize the mirror's size according to how the HUD system has been designed.
- the **PGU Usage** to adjust the ratio between the warping and the Picture Generation Unit to optimize the surface used by the warping.
- the **Stopping Criterion** to define a threshold value representing the degree of precision to be reached for the optimization to end.
- the **Curvature Balance** to pilot the curvature of the first freeform mirror to get the best image quality.

For more information on the Advanced parameters, refer to Advanced Parameters.

18. If you want, click **Precompute Head Up Display**  to help determine the best optical path to choose. The optical path (position and orientation of mirrors) is optimized to reach the best optical quality for the virtual image while using the maximum surface of the PGU.

> **Note: Precompute Head Up Display** only supports one Freeform mirror. For multi-freeform mirrors, use **Compute**.

19. Click **Compute**  to optimize the HUD system.
Once the HOD optimization is done, the computed surfaces appear in the 3D view.

DRAFT

14.2.4. CNC Export (Surface Export File)

The CNC Eport (Surface Export File) is a *.txt file generated at the end of each HUD Optical Design computation, containing information relative to the HUD design.

At the end of each HUD Optical Design computation, a surface export.txt is generated.file

The file contains:

- The description of the axis system (origin, X, Y and Z axes) used for the export.
- The coefficients of the (extended) polynomial equation of the freeform mirror(s) shape. These coefficients are provided for the mirrors production.

$$z = \underbrace{\frac{cr^2}{1+\sqrt{1-(1+k)c^2r^2}}}_{Conic\ term} + \underbrace{\sum_{m,n=0}^{\infty} C_{m,n}\, x^m y^n}_{Polynomial\ term}$$

by default this term = 0

Is reflected in the text file as:

- Radius = 0
- Conic = 0

m,n from 0 to the degree defined by the user

Is reflected in the text file as:

- Coefficient $X^m\ Y^{n=}C_{m,n}$
- Normalization = 1

Warning: The exported surfaces are unbounded.

Note: The Normalization factor is used in the Polynomial term in order to make the Polynomial term dimensionless.

14.3. Analysis

This section introduces the HUD Optical Analysis (HOA) feature that allows you to perform a complete analysis of a HUD system with one or several configurations.

14.3.1. HUD Optical Analysis

HUD Optical Analysis (HOA) is a tool to analyze a head-up display system. HOA computes and displays information in the 3D view or in a report.

14.3.1.1. Setting the HUD Optical Analysis

This page describes how to define and compute a HUD Optical Analysis.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

To use the HOA feature:

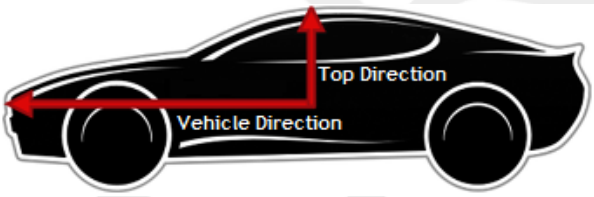
1. From the **Light Simulation** tab, click System > HUD Optical Analysis.
2. Set the:
   a. the General settings
   b. the Eyebox settings
   c. the Target Image settings
   d. the Windshield settings
   e. the Mirrors settings
   f. the PGU settings
   g. the Warping settings
   h. the Report settings
3. Run the computation.

> **Note:** To personalize the 3D view result, use the visualization properties.

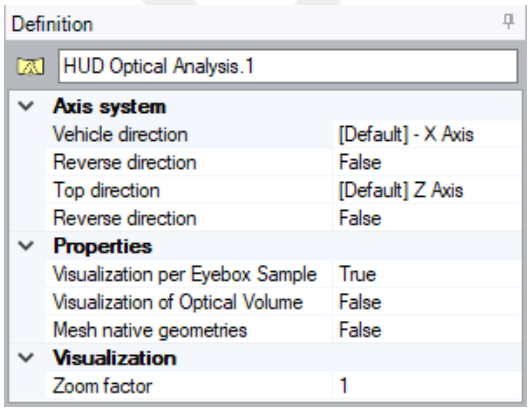
14.3.1.1.1. General

The general settings allow to specify the axis system used by HOA and the calculation type to run according to the 3D visualization you want to display.

If no axis system is set, a default axis system is used: **Vehicle direction = -X Axis** and **Top direction = +Z Axis**.



1. If you want to define a custom axis system:



- 
  and select the line that defines the vehicle direction.
- 
  and select the line that defines the top direction.

2. In **Properties**, define what parts of the results you want to visualize after computation:
   - Activate **Visualization of Optical Volume** if you want the optical volume to be displayed in the 3D view at the end of the computation. This option is useful to check if something obstructs (for example, dashboard too high) the driver field of view to see the Target Image.
   - Activate **Visualization per Eyebox Sample** if you want to access the **Eyebox sample** option after computation.

     > **Note:** To access the Eyebox sample option, activate the **Visualization per Eyebox Sample** and run the HOA computation.

Visualization	
Zoom factor	1
Eyebox sample	True
Vision mode	Binocular
Horizontal sample	1
Vertical sample	1

     - Set the **Eyebox sample** option to **True** to visualize the Target Image result from different positions on the Eyebox according to the human vision.
     - From the **Vision mode** drop-down list, select which view to display.
     - Set the horizontal and vertical sampling of the eyebox. The position (1,1) corresponds to the bottom left sample of the Eyebox.

3. Activate the **Zoom factor** option to zoom the size of the Best Focus Spot, Tangential Spot, Sagittal Spot, Static Distortion and Pixel Image in the 3D view.

14.3.1.1.2. Eyebox

The **Eyebox** section allows you to configure the eyebox(es) used by HOA.

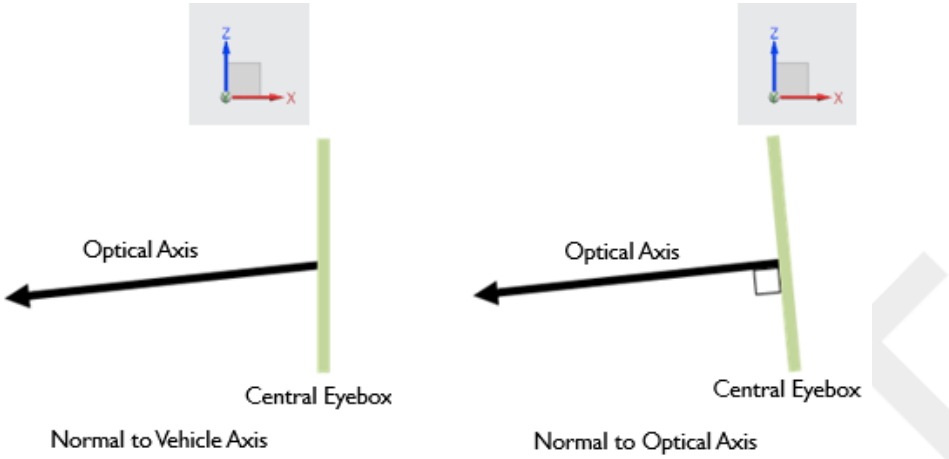
To configure the Eyebox settings:

Eyebox	
General	
Center	[No selection]*
Orientation	Normal To Optical Axis
Sampling mode	Uniform
Interpupillary distance	65 mm
Eye pupil diameter	4 mm
Eye pupil sampling	6

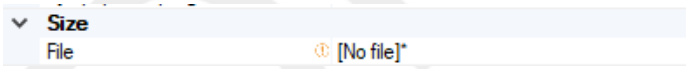
1. In the 3D view, click  and select a point to place the eyebox center.
2. In **Orientation**, define the vertical direction of the Eyebox:
   - Select **Normal to optical axis** to set the vertical direction of the Eyebox as normal to the optical axis defined in the **Target Image** section.

DRAFT

- Select **Normal to vehicle axis** to set the vertical direction of the Eyebox as normal to the vehicle axis defined in the **General** section.



3. In **Sampling mode**, select which sampling mode to use for the eyebox:
   - Select **Adaptive** if you want to define the sampling of the eyebox with a file that describes the coordinates of each sample.
   - Select **Uniform** if you want the sampling of the Eyebox to be a uniform grid. With this mode, three grids can share sample to represent the Eyebox: One Binocular grid representing the position of the eyes and two monocular grids, each representing the position of one eye. The binocular Eyebox is the union of the two monocular Eyeboxes.
     - In **Interpupillary distance**, specify the distance between the eyes.
4. In **Eye pupil diameter**, define the diameter of the pupil of the driver's eyes.
5. In **Eye pupil sampling,** define the number of samples on the pupil of the driver's eyes (samplings are on the circle of the eye).
6. In **Size**, define the sampling and size of the eyebox:
   - If you selected the **Adaptive** sampling mode, browse and load a file.



The file contains the coordinates of each sample in millimeter where **Center** is the origin point, **Vehicle direction * Top direction** is the X axis and **Orientation** is the Y axis.

```
OPTIS - Eyebox file v1.0
Example of adaptive Eyebox
15
-65     25
-32.5   25
0       25
32.5    25
65      25
-65     0
-32.5   0
0       0
32.5    0
65      0
-65     -25
-32.5   -25
0       -25
32.5    -25
65      -25
```

Number of samples

X coordinate

Y coordinate

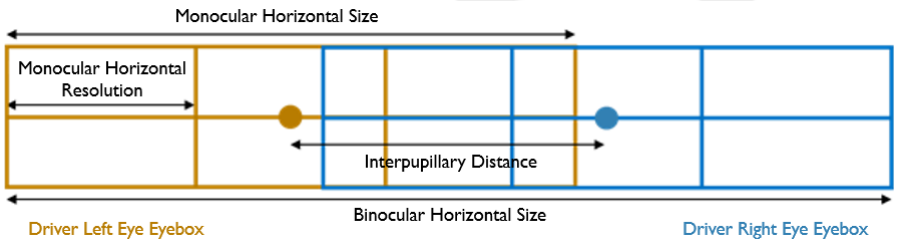
> **Note:** The file must end with an empty line. Download an . example file

- If you selected the **Uniform** sampling mode, define the sampling and size of the eyebox manually:

Size	
Binocular horizontal size	130 mm
Monocular horizontal sampling	3
Monocular horizontal size	65 mm
Monocular horizontal resolution	32.5 mm
Binocular horizontal sampling	5
Number of shared samples	3
Vertical size	50 mm
Vertical sampling	3
Vertical resolution	25 mm

  - Define the binocular horizontal size and monocular horizontal sampling of the eyebox.
  - Define the vertical size and sampling of the eyebox.

> **Note: Number of shared samples** gives the number of sampling shared between the monocular eyeboxes. The binocular and monocular Eyeboxes share the vertical parameters.

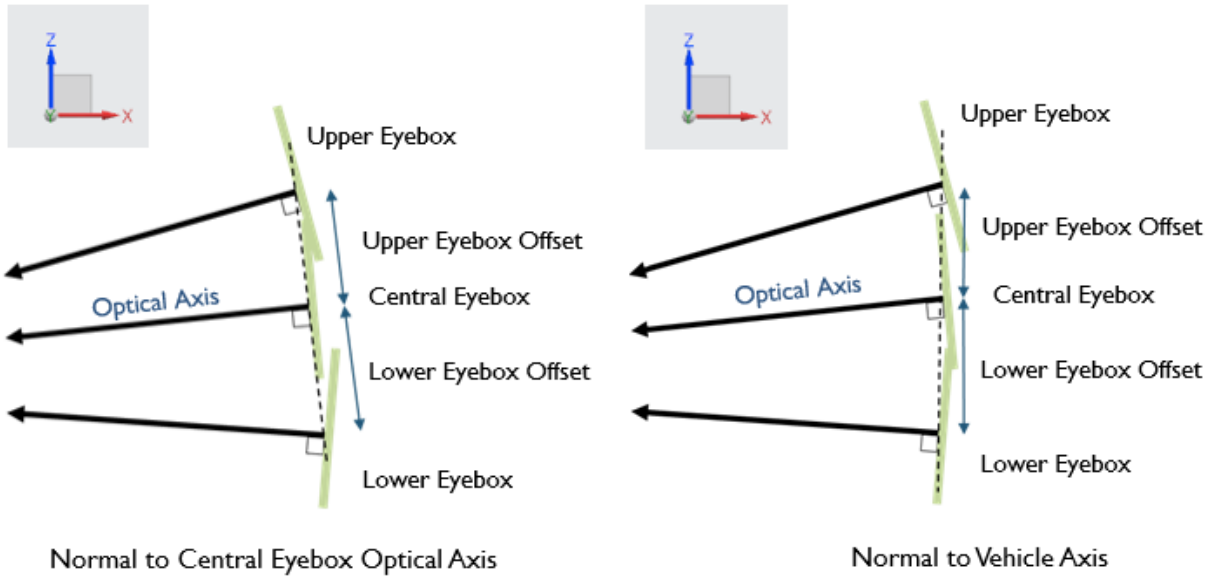


7. If you want to analyze your system according to different eyeboxes, activate the Multieyebox mode:

Eyeboxes	
Multieyebox	True
Position direction	Normal To Central Eyebox Optical Axis
Eyebox configurations	3 items

The default eyebox, nominal Eyebox, is the Central Eyebox with a 0mm-offset. Each added Eyebox is created by moving in translation from the nominal Eyebox to a distance corresponding to the **Offset**.

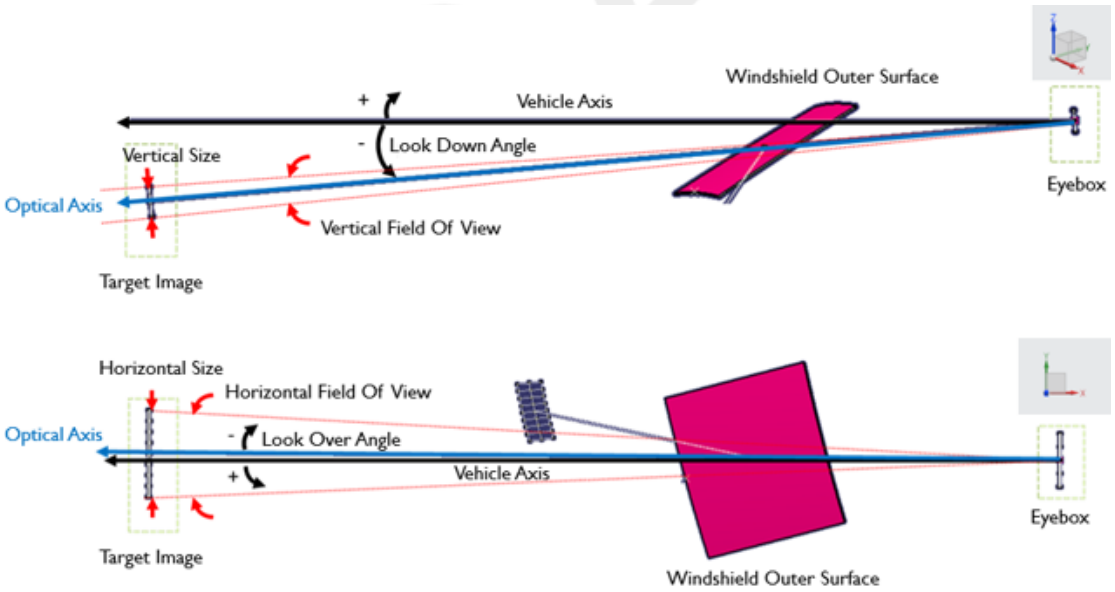
- From the **Position Direction** drop-down list, specify the direction used to apply the offsets:
  - Select **Normal to Central Eyebox Optical Axis**, to set the position direction as normal to the nominal driver optical axis defined in the **Target Image** section.
  - Select **Normal to Vehicle Axis** to set the position direction as normal to the Vehicle Axis defined in the **General** section.



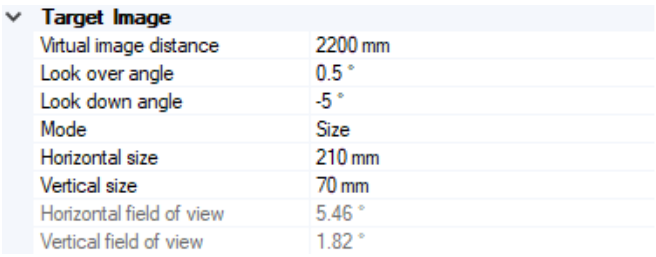
14.3.1.1.3. Target Image

The **Target Image** section allows you to specify the target image used by HOA.

Figure 109. Parameters to set when configuring the Target Image.



To configure the Target Image:



1. In **Virtual image distance**, specify the distance between the center of the nominal Eyebox and the center of the Target Image.
2. Set the **Look over angle** (the horizontal angle) between the vehicle axis and the optical axis.
3. Set the **Look down angle** (the vertical angle) between the vehicle axis and the optical axis.

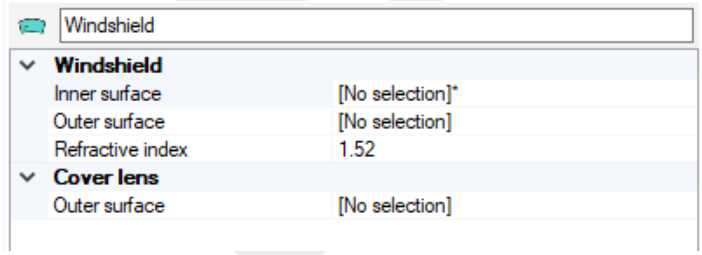
> **Note:** To define the optical axis, HOA applies first the "Look Over Angle", then the "Look Down Angle".

4. From the **Mode** drop-down list, specify the mode you to use to define the target image:
   - Select **Size to** define the size of the target image in millimeters.
   - Select **Field of view** to define the field of view of the target image in degrees.
5. According to the mode you selected:
   - Set the horizontal and vertical size of the target image in millimeters.

     Set the horizontal and vertical fields of view in degrees.

14.3.1.1.4. Windshield

The **Windshield** section allows you to specify the windshield used by HOA.



> **Note:** A face or a multifaces body can be selected as surface.

> **Note:** Faceted geometries (such as imported *.stl files or other file formats alike) can be used as input geometry.

1. In the 3D view, click 
   and select the inner surface of the windshield.
2. In the 3D view, click  and select the outer surface of the windshield. This surface is optional.

> **Note:** This parameter must be set when using the Ghost or the Overview tests.

3. Specify the **Refractive index** of the windshield.
4. If you want to select a **Cover lens**, click  and select a surface in the 3D view.

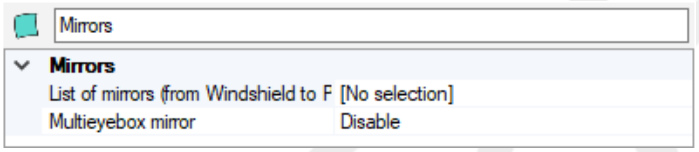
   The **Cover lens** (optional) is a geometry that works like a diaphragm to limit the light beam. It is located between the Windshield and the first mirror.

   All rays passing through the Cover Lens are considered in the results. The rays that do not pass through the Cover Lens are displayed in dotted lines in the 3D view and are not considered in the results.

14.3.1.1.5. Mirrors

The **Mirrors** section allows you to specify the mirrors used by HOA.

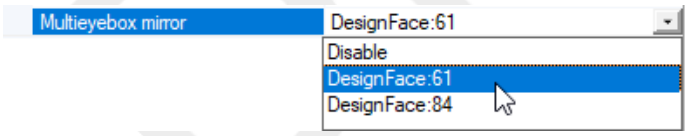
To configure the Mirrors:



1. In the 3D view, click and select the mirrors to consider in the projector.

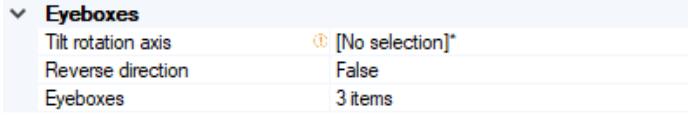
   The selection appears in the List of mirrors. The order is defined from the Windshield towards the PGU.

2. In **Multieyebox mirror**, activate the multieyebox for the selected mirror. You can activate the multieyebox for only one mirror.



> **Note:** To use this option, activate the multieyebox mode from the **Eyebox** section and create several eyeboxes.

- In **Eyeboxes**, you can apply a rotation to a mirror according to the **Tilt rotation axis** for each eyebox defined.



- In the 3D view, click  and select an axis to define the rotation axis.

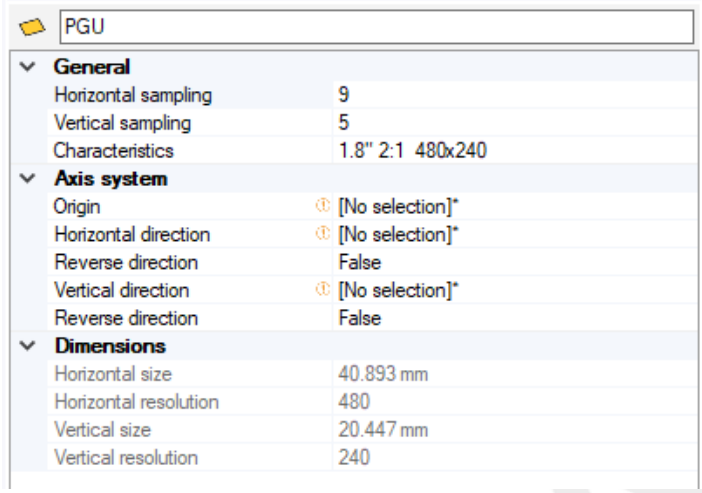
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- In **Tilt Angle**, define the rotation angle to apply to the selected eyebox.

14.3.1.1.6. PGU

The **PGU** section allows you to specify the characteristics of the PGU used by HOA.

To configure the PGU:

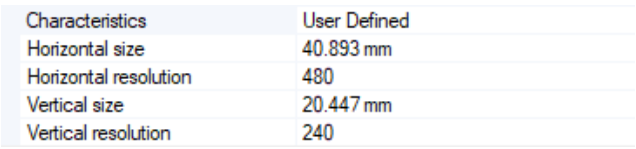


- Define the **Axis system** of the PGU:
  - In the 3D view, click and select the point to define the center of the PGU.
  - Click and select the line to define the horizontal direction of the PGU.
  - Click and select the line to define vertical direction of the PGU.

> **Note:** If you define manually one axis only, the other axis is automatically (and randomly) calculated by Speos in the 3D view. However, the other axis in the Definition panel may not correspond to the axis in the 3D view. Please refer to the axis in the 3D view.

- In **Horizontal sampling** and **Vertical sampling**, define the number of horizontal and vertical samples to use.
- From the **Characteristics** drop-down list, select which PGU characteristics are used by HOA:
  - Select a predefined PGU amongst the available models.

    The name corresponds to the characteristics of the PGU. Example: **1.8"** = size (in inches), **2:1** = horizontal to vertical size ratio of the screen, **480x240** = PGU pixel resolution.
  - Select **User Defined** to manually define the PGU and its **Dimensions**:



Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

1. Set the horizontal and vertical size of the PGU in millimeters.
2. Set the horizontal and vertical resolution of the PGU in pixels.

14.3.1.1.7. Warping

Warping is a post processing step used to correct the optical aberrations of the system to provide the optimal target image for the driver.

14.3.1.1.7.1. Understanding the Warping

Working Principle

The Picture Generation Unit (PGU) displays a deformed image to the driver. This deformation comes from the propagation of the image through the HUD system.

To solve this issue for the driver to see a rectangular image, the image displayed by the PGU is not rectangular, it is "pre-deformed" (bent beforehand).

This "pre-deformation" is called Warping.

This also corrects poor image orientation as shown in the following table.

	Image displayed by the PGU	Image seen by the driver
Without Warping		
With Warping		

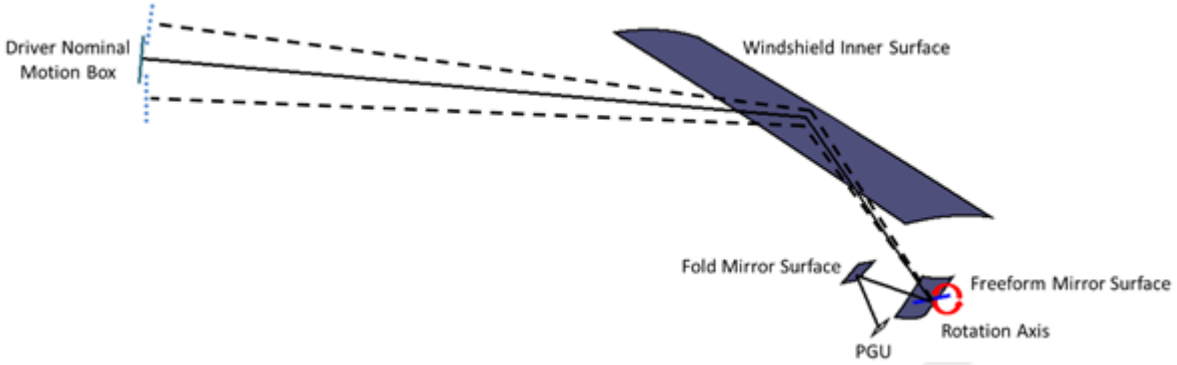
Warping Parameters

Tilt

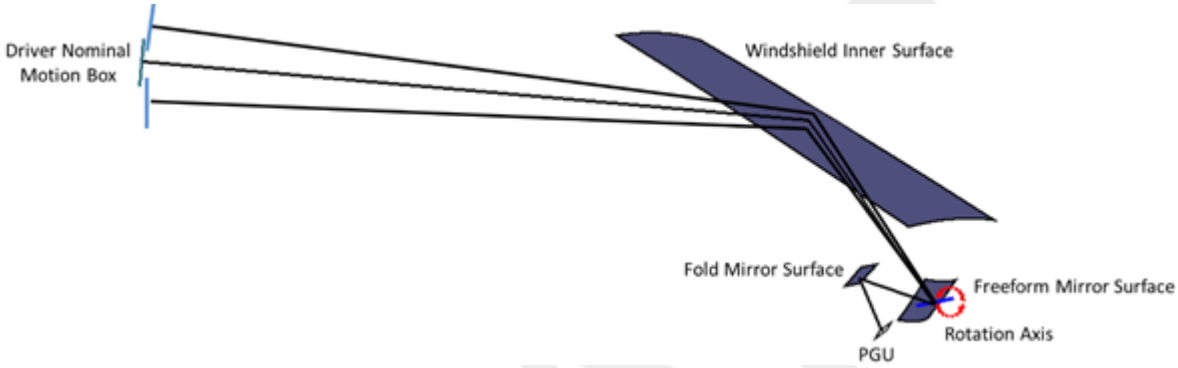
The Tilt allows you to generate a Warping file containing the samples coordinates of each driver warping. Three Tilt modes are available:

- **Predefined Mirror Tilt**: generates the Warping file containing the Warping of each driver defined in the Configurations of the Eyebox section.
- **Step Mirror Tilt**: generates the Warping file containing the Warping of each driver defined thanks to an angular range from the initial position of the rotating mirror activated to calculate each Eyebox position.

DRAFT

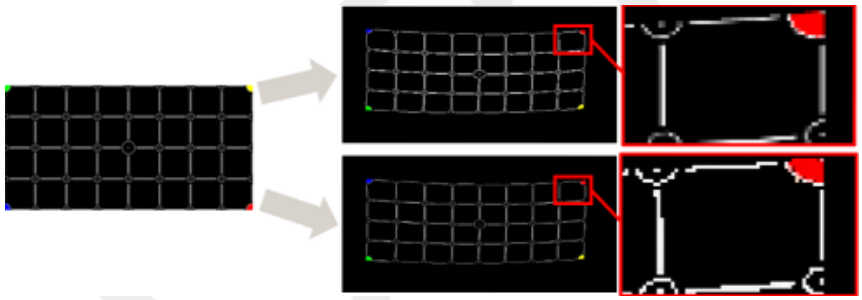


- **Free Mirror Tilt**: generates the Warping file containing the Warping of each driver defined thanks to an offset range from the center of the Central Eyebox to calculate each Mirror rotation.



Warping Algorithms

Ansys provides you with the following Warping Algorithms:

	Nearest interpolation algorithm (low quality warping)
	Bilinear interpolation algorithm (high quality warping)

The warping algorithms use interpolation to calculate the final image.

Even if an anti-aliasing is applied, lines always tend to appear smoother in the middle of the image compared to the border of the image.

14.3.1.1.7.2. Warping File

This page describes the format of warping files containing the pixel coordinates of each sample of the warping.

The order of the sample are defined as follow:

Bottom Left sample of the Target Image	...	Bottom Right sample of the Target Image
...	...	...

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Top Left sample of the Target Image	...	Top Right sample of the Target Image

> **Tip:** The origin of the pixel coordinate is defined by the PGU axis system.

Download files.example

```
ANSYS - Warping file v3.0
    Commment

    W

    X M N

    1_RotationAngle    2_RotationAngle    ...    (X-1)_RotationAngle
X_RotationAngle  -- as many as X value
    1_OffsetPosition    2_OffsetPosition    ...    (X-1)_OffsetPosition
X_OffsetPosition  -- as many as X value

    1_S(1,1)_HCoord 1_S(1,1)_VertCoord    1_S(2,1)_HCoord 1_S(2,1)_VertCoord
   ...    1_S(M,1)_HCoord 1_S(M,1)_VertCoord
    ...
    1_S(1,N)_HCoord 1_S(1,N)_VertCoord    1_S(2,N)_HCoord 1_S(2,1)_VertCoord
   ...    1_S(M,N)_HCoord 1_S(M,N)_VertCoord


    2_S(1,1)_HCoord 2_S(1,1)_VertCoord    2_S(2,1)_HCoord 2_S(2,1)_VertCoord
   ...    2_S(M,1)_HCoord 2_S(M,1)_VertCoord
    ...
    2_S(1,N)_HCoord 2_S(1,N)_VertCoord    2_S(2,N)_HCoord 2_S(2,N)_VertCoord
   ...    2_S(M,N)_HCoord 2_S(M,N)_VertCoord


    X_S(1,1)_HCoord X_S(1,1)_VertCoord    X_S(2,1)_HCoord X_S(2,1)_VertCoord
   ...    X_S(M,1)_HCoord X_S(M,1)_VertCoord
    ...
    X_S(1,N)_HCoord X_S(1,N)_VertCoord    X_S(2,N)_HCoord X_S(2,N)_VertCoord
   ...    X_S(M,N)_HCoord X_S(M,N)_VertCoord
    (empty line)
```

> **Note:** The file must end with an empty line.

- **W** is the Warping file type:
  - **0**: Predefined Mirror Tilt
  - **1**: Free Mirror Tilt
  - **2**: Step Mirror Tilt
- **X** is the number of eyeboxes:
  - defined in the **Eyebox** section when using the **Predefined Mirror Tilt** mode, or
  - defined by the **Tilt Sampling** in the Mirrors section when using the **Step or Free Mirror Tilt** mode.
- **M** is the horizontal resolution of the Warpings:
  - defined by the **Horizontal Sampling** in the PGU tab when using the **Predefined Mirror Tilt** mode, or

- defined by the **Horizontal Sampling** in the Mirrors tab when using the **Step or Free Mirror Tilt** mode.

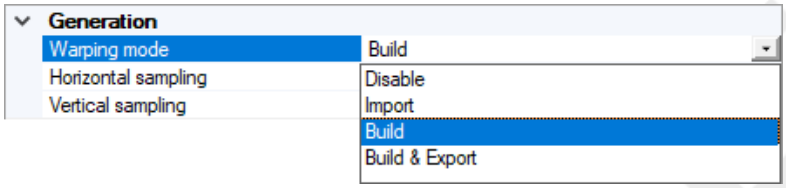
- **N** is the vertical resolution of the Warpings:
  - defined by the **Vertical Sampling** in the PGU tab when using the **Predefined Mirror Tilt** mode, or
  - defined by the **Vertical Sampling** in the Mirrors tab when using the **Step or Free Mirror Tilt** mode.

14.3.1.1.7.3. Configuring the Warping

The following procedure allows you to specify if HUD Optical Analysis (HOA) must compute the warping or not, or import a warping.

To configure a warping:

1. In the **Generation** section, select the **Warping mode**:



   - **Disable** if you do not want HOA to compute any warping. In this case, HOA analyzes the system with a rectangular grid on the PGU defined in the PGU tab.

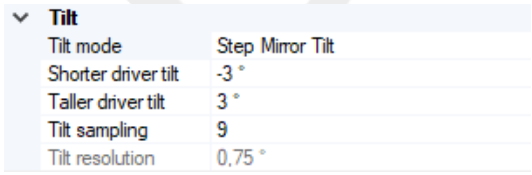
     Note: Do not use the **Disable** mode when using the Magnification or the Overview tests.

   - **Import** if you want to use an imported warping file.

     Note: For more information on the Warping file, refer to Warping File.

   - **Build** if you want HOA to compute the warping without exporting it, and define the **Horizontal sampling** and **Vertical sampling** of the warping used by HOA and sent to the plugins.
   - **Build & Export** if you want HOA to compute the warping and export it into a warping file, and define the **Horizontal sampling** and **Vertical sampling** of the warping used by HOA and the warping file.

2. If you set the **Multieyebox** to **True** in the **Eyebox** definition, in the **Tilt** section select the **Tilt mode**:



   - **Predefined Mirror Tilt** to generate a Warping file containing the Warping of each driver defined in the Configurations of the Eyebox section.
   - **Step Mirror Tilt** to generate a Warping file containing the Warping of each driver defined by rotation of the mirror.

     a. Define the angular range [**Shorter Driver Tilt** ; **Taller Driver Tilt**] of the rotating mirror to calcultate each Eyebox position

     b. Define the **Tilt Sampling**.

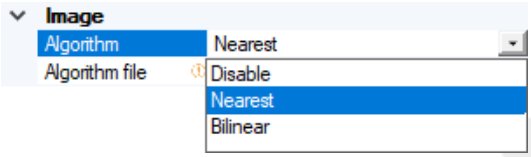
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

> **Note:** The rotating mirror is the mirror for which the multiconfiguration is activated in the Mirrors tab. The 0° corresponds to the initial position of the mirror. The rotation axis used is the Tilt Rotation Axis defined in the Mirrors tab.

- **Free Mirror Tilt** to generate a Warping file containing the Warping of each driver defined by their Eyebox position.
  - **a.** Define the offset range [**Shorter Driver Tilt** ; **Taller Driver Tilt**] of the Eyebox positions to calculate each Mirror rotation.
  - **b.** Define the **Tilt Sampling**.

> **Note:** The 0mm corresponds to the center of the Central Eyebox. The Translation axis used is the Position Direction axis defined in the Eyebox.

3. In the **Image** section, If you want to use a warping **Algorithm**:



a) Select **Nearest** for a low quality warping, or **Bilinear** for a high quality warping.
b) Select the **Algorithm file**.

> **Note:** The Algorithm file is an input image file to be pre-distorted by a provided warping algorithm or your own created warping algorithm thanks to the Image Warping APIs.

> **Note:** HOA generates one image per warping configuration in the tree.

14.3.1.1.8. Report

The **Report** section allows you to select the tests you want to compute into the report.

The tests come from the plugins (your own and/or the standard).

For a description of the different available standard test, see Plugin.

For a description of the API for your own plugin, see Application Programming Interface.

In the **Available Tests**, select the test to compute in the report and add them into the **Tests To Run**.

> **Note:** If some tests are not available, check the plugins you use.

DRAFT

14.3.1.2. Exporting a HUD Optical Analysis Simulation

The following procedure helps you export a data model representing the HUD system to a *.speos file, that you can then import in a dedicated external software to visualize and experiment your HUD system to anticipate errors and control its quality.

To export a HUD Optical Analysis (HOA) simulation:

You must have set the HUD Optical Analysis simulation.

1. In the **Speos Simulation** tree, right-click the HOA simulation you want to export.
2. Click **Export**  .
3. Browse and select a path for your ***.speos** folder.
4. Name the file and click **Save**.

The HOA simulation has been exported in the folder of the same name as the *.speos file along with the input files.

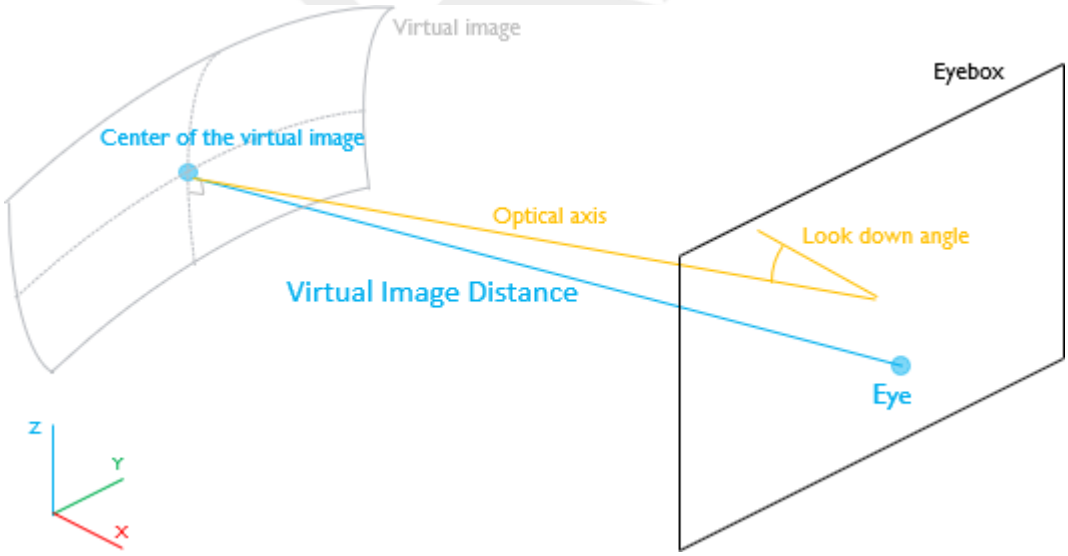
The *.speos file corresponds to the HUD model and contains all inputs from the Axis System, Eyebox, Target Image, Windshield, Mirrors, PGU, Warping.

Now you can import the *.speos file in a dedicated external software to visualize and experiment your HUD system to anticipate errors and control its quality.

14.3.1.3. Speos Plugin

14.3.1.3.1. Virtual Image Distance

The Virtual Image Distance test gives the distance between a sample of the eyebox and the center of the virtual image for this eyebox sample.



DRAFT

In detail:

For each sample EBij of the binocular eyebox, the Virtual Image Distance gives the distance from the sample EBij of the binocular eyebox to the center of the virtual image seen from the sample EBij of the binocular eyebox.

The Virtual Image Distance is expressed in millimeter.

> **Note:** In the plugin, the dynamic distortion (the virtual image seen from the sample EBij) is computed in three dimensions.

The result of the computation is displayed as a table that contains the Virtual Image Distance computed for each sample EBij of the binocular eyebox.

Virtual Image Distance

top

Plugin description	OPTIS - HOA Plugin
Test description	Virtual Image Distance

Configuration	**Upper Eyebox**	**Central Eyebox**	**Lower Eyebox**
Driver offset (mm)	40.000	0.000	-40.000
Average (mm)	2363.850	2214.880	2133.877
Standard deviation (mm)	83.819	33.010	30.242
Minimum (mm)	2257.105	2169.310	2093.238
Maximum (mm)	2525.486	2281.182	2199.902

Upper Eyebox

Virtual Image Distance (mm)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	2503.393	2437.751	2425.152	2453.588	2525.486
-0.00 mm	2354.607	2305.960	2293.344	2322.132	2384.026
-25.00 mm	2305.654	2263.356	2257.105	2282.049	2344.143

Central Eyebox

Virtual Image Distance (mm)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	2262.689	2214.079	2203.610	2225.095	2281.182
0.00 mm	2218.918	2177.374	2169.310	2190.061	2243.696
-25.00 mm	2223.021	2183.290	2176.434	2198.591	2255.843

Lower Eyebox

Virtual Image Distance (mm)	**-65.00 mm**	**-32.50 mm**	**-0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	2145.109	2102.989	2093.238	2111.373	2160.076
0.00 mm	2142.298	2103.637	2095.266	2113.705	2162.711
-25.00 mm	2172.742	2133.538	2125.718	2145.853	2199.902

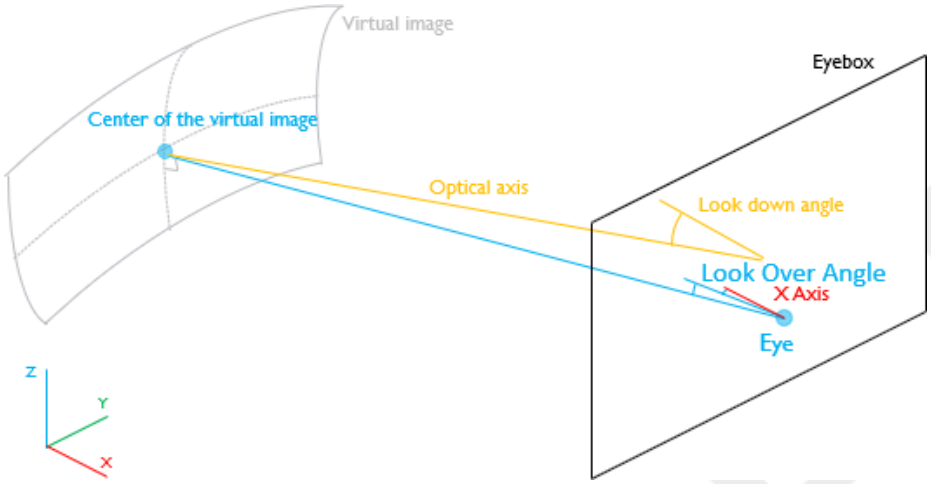
In particular, you can notice that:

- There is one table result per configuration created.
- Each table has the size of the sampling of the binocular eyebox.
- The test also displays a sum up for each configuration of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

14.3.1.3.2. Look Over Angle

The Look Over Angle test gives the angle between the Vehicle Direction and the projection, on the plane normal to the Top Direction, of the line passing through a sample of the eyebox and the center of the virtual image for this eyebox sample.



In detail:

For each sample EBij of the binocular eyebox, the Look Over Angle gives the angle, expressed in degrees, between the Vehicle Direction (you specified in the General tab) and the projection of the *optical axis* on the *horizontal plane*.

Optical axis: For each sample EBij of the binocular eyebox, the optical axis is the line between the sample EBij of the binocular eyebox and the center of the virtual image seen from the sample EBij of the binocular eyebox.

Horizontal plane: The horizontal plane is the XY plane of the direct axis system define by -X = Vehicle Direction and Z = Top Direction. Vehicle Direction and Top Direction are the vectors you specified in the General tab.

> **Note:** In the plugin, the dynamic distortion (the virtual image seen from the sample EBij) is computed in three dimensions.

The result of the computation is displayed as a table that contains the Look Over Angle computed for each sample EBij of the binocular eyebox.

DRAFT

Look Over Angle

top

Plugin description	OPTIS - HOA Plugin
Test description	Look Over Angle

Configuration	**Upper Eyebox**	**Central Eyebox**	**Lower Eyebox**
Driver offset (mm)	40.000	0.000	-40.000
Average (°)	-0.000	-0.002	0.001
Standard deviation (°)	1.185	1.192	1.201
Minimum (°)	-0.000	-0.000	0.003
Maximum (°)	1.686	-1.696	1.716

Upper Eyebox

Look Over Angle (°)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	-1.678	-0.851	-0.000	0.852	1.686
-0.00 mm	-1.675	-0.846	0.001	0.848	1.674
-25.00 mm	-1.668	-0.841	0.001	0.839	1.654

Central Eyebox

Look Over Angle (°)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	-1.696	-0.860	-0.001	0.857	1.696
0.00 mm	-1.687	-0.852	-0.000	0.850	1.680
-25.00 mm	-1.674	-0.844	-0.000	0.840	1.656

Lower Eyebox

Look Over Angle (°)	**-65.00 mm**	**-32.50 mm**	**-0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	-1.709	-0.863	0.003	0.869	1.716
0.00 mm	-1.696	-0.854	0.004	0.861	1.696
-25.00 mm	-1.678	-0.844	0.004	0.848	1.670

> **Note:** The **Minimum (°)**, **Maximum (°)**, **Look Over Angle (°)**, **Look Down Angle (°)** values are absolute values.

In particular, you can notice that:

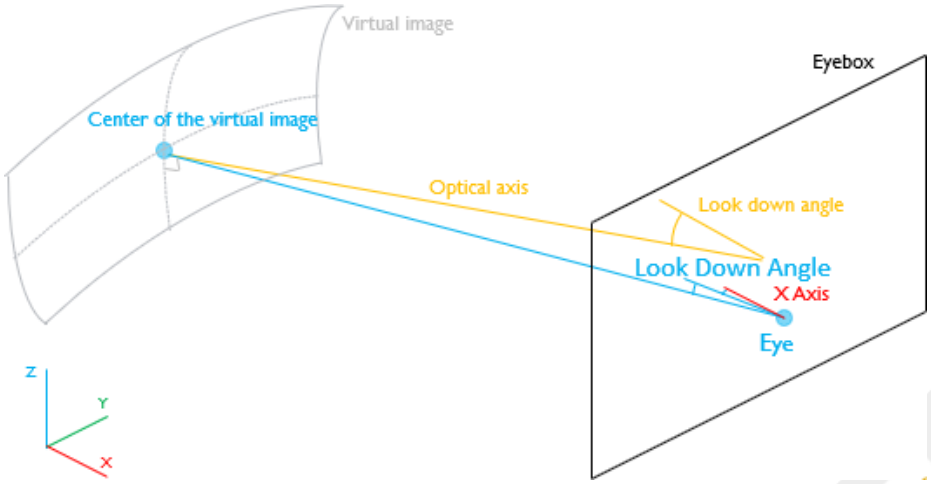
- There is one table result by configuration created.
- Each table has the size of the sampling of the binocular eyebox.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.3. Look Down Angle

The Look Down Angle test gives the angle between the plane normal to the Top Direction and the line passing through a sample of the eyebox and the center of the virtual image for this eyebox sample.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00



In detail:

For each sample EBij of the binocular eyebox, the Look Down Angle gives the angle, expressed in degrees, between the *optical axis* and the projection of the *optical axis* on the *horizontal plane*.

Optical axis: For each sample EBij of the binocular eyebox, the optical axis is the line between the sample EBij of the binocular eyebox and the center of the virtual image seen from the sample EBij of the binocular eyebox.

Horizontal plane: The horizontal plane is the XY plane of the direct axis system define by -X = Vehicle Direction and Z = Top Direction. Vehicle Direction and Top Direction are the vectors you specified in the General tab.

> **Note:** In the plugin, the dynamic distortion (the virtual image seen from the sample EBij) is computed in three dimensions.

The result of the computation is displayed as a table that contains the Look Down Angle computed for each sample EBij of the binocular eyebox.

DRAFT

Look Down Angle

top

Plugin description	OPTIS - HOA Plugin
Test description	Look Down Angle

Configuration	**Upper Eyebox**	**Central Eyebox**	**Lower Eyebox**
Driver offset (mm)	40.000	0.000	-40.000
Average (°)	-7.154	-5.003	-2.751
Standard deviation (°)	0.479	0.537	0.571
Minimum (°)	-6.551	-4.331	-2.037
Maximum (°)	-7.738	-5.667	-3.461

Upper Eyebox

Look Down Angle (°)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	-7.738	-7.738	-7.732	-7.728	-7.730
-0.00 mm	-7.170	-7.167	-7.165	-7.167	-7.177
-25.00 mm	-6.558	-6.552	-6.551	-6.557	-6.575

Central Eyebox

Look Down Angle (°)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	-5.667	-5.660	-5.654	-5.654	-5.662
0.00 mm	-5.013	-5.004	-4.999	-5.004	-5.019
-25.00 mm	-4.348	-4.335	-4.331	-4.339	-4.361

Lower Eyebox

Look Down Angle (°)	**-65.00 mm**	**-32.50 mm**	**-0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	-3.461	-3.449	-3.443	-3.445	-3.458
0.00 mm	-2.759	-2.744	-2.738	-2.744	-2.764
-25.00 mm	-2.062	-2.043	-2.037	-2.045	-2.072

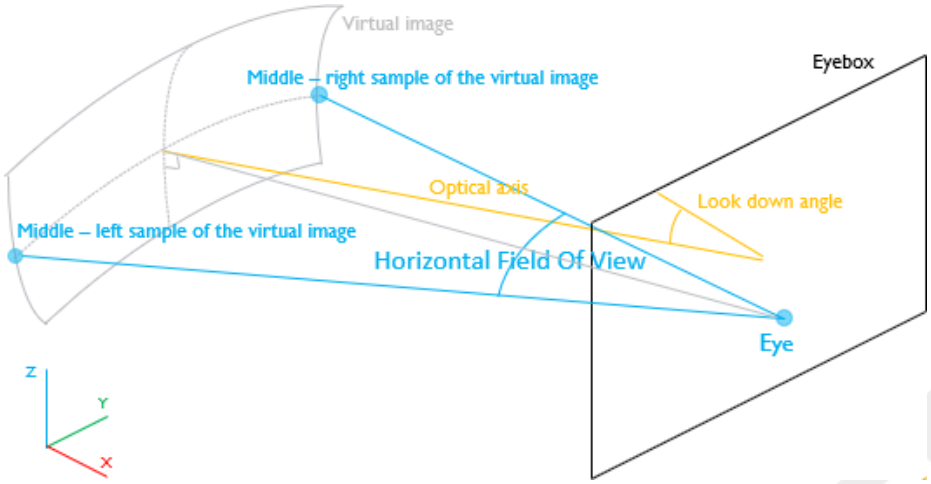
In particular, you can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the binocular eyebox.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.4. Horizontal Field of View

The Horizontal Field of View test gives the size of the virtual image, in degree, along the horizontal direction.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



In detail:

For each sample EBij of the binocular eyebox, the Horizontal Field Of View gives the angle, expressed in degrees, between the *left border line* and the >*right border line*.

Left border line: For each sample EBij of the binocular eyebox, the left border line is defined as the line between the sample EBij of the binocular eyebox and the middle left sample of the virtual image seen from the sample EBij of the binocular eyebox.

Right border line: Similarly, for each sample EBij of the binocular eyebox, the right border line is defined as the line between the sample EBij of the binocular eyebox and the middle right sample of the virtual image seen from the sample EBij of the binocular eyebox.

> **Note:** In the plugin, the dynamic distortion (the virtual image seen from the sample EBij) is computed in three dimensions.

The result of the computation is displayed as a table that contains the Horizontal Field Of View computed for each sample EBij of the binocular eyebox.

DRAFT

Horizontal Field Of View

top

Plugin description	OPTIS - HOA Plugin
Test description	Horizontal Field Of View

Configuration	**Upper Eyebox**	**Central Eyebox**	**Lower Eyebox**
Driver offset (mm)	40.000	0.000	-40.000
Average (°)	5.481	5.472	5.466
Standard deviation (°)	0.013	0.015	0.018
Minimum (°)	5.463	5.455	5.447
Maximum (°)	5.513	5.510	5.509

Upper Eyebox

Horizontal Field Of View (°)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	5.513	5.480	5.472	5.481	5.505
-0.00 mm	5.497	5.475	5.470	5.476	5.488
-25.00 mm	5.487	5.467	5.463	5.468	5.479

Central Eyebox

Horizontal Field Of View (°)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	5.510	5.474	5.462	5.467	5.485
0.00 mm	5.496	5.470	5.461	5.462	5.471
-25.00 mm	5.488	5.464	5.455	5.455	5.464

Lower Eyebox

Horizontal Field Of View (°)	**-65.00 mm**	**-32.50 mm**	**-0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	5.509	5.470	5.454	5.455	5.469
0.00 mm	5.497	5.467	5.454	5.451	5.457
-25.00 mm	5.492	5.463	5.449	5.447	5.453

In particular, you can notice that:

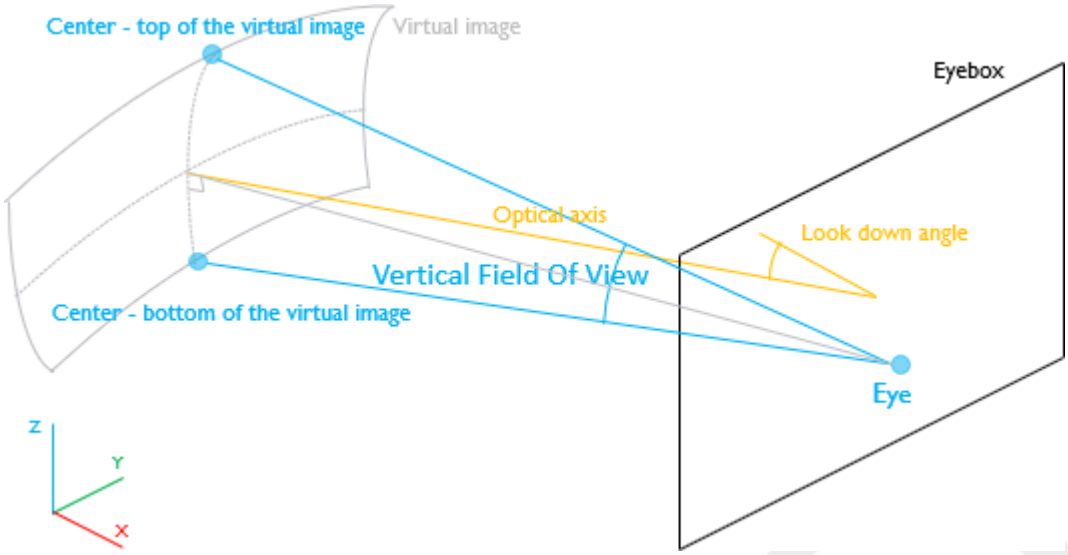
- There is one table result by configuration created.
- Each table has the size of the sampling of the binocular eyebox.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.5. Vertical Field of View

The Vertical Field of View test gives the size of the virtual image, in degree, along the vertical direction.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00



In detail:

For each sample EBij of the binocular eyebox, the "Vertical Field Of View" gives the angle, expressed in degrees, between the *bottom border line* and the *top border line*.

Bottom border line: For each sample EBij of the binocular eyebox, the bottom border line is defined as the line between the sample EBij of the binocular eyebox and the bottom middle sample of the virtual image seen from the sample EBij of the binocular eyebox.

Top border line: Similarly, for each sample EBij of the binocular eyebox, the top border line is defined as the line between the sample EBij of the binocular eyebox and the top middle sample of the virtual image seen from the sample EBij of the binocular eyebox.

> **Note:** In the plugin, the dynamic distortion (the virtual image seen from the sample EBij) is computed in three dimensions.

The result of the computation is displayed as a table that contains the Vertical Field Of View computed for each sample EBij of the binocular eyebox.

DRAFT

Vertical Field Of View

top

Plugin description	OPTIS - HOA Plugin
Test description	Vertical Field Of View

Configuration	**Upper Eyebox**	**Central Eyebox**	**Lower Eyebox**
Driver offset (mm)	40.000	0.000	-40.000
Average (°)	1.828	1.827	1.821
Standard deviation (°)	0.020	0.006	0.006
Minimum (°)	1.802	1.816	1.812
Maximum (°)	1.867	1.842	1.831

Upper Eyebox

Vertical Field Of View (°)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	1.840	1.848	1.857	1.863	1.867
-0.00 mm	1.810	1.816	1.822	1.825	1.827
-25.00 mm	1.802	1.806	1.810	1.811	1.812

Central Eyebox

Vertical Field Of View (°)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	1.825	1.831	1.837	1.840	1.842
0.00 mm	1.816	1.821	1.824	1.825	1.825
-25.00 mm	1.822	1.825	1.827	1.826	1.825

Lower Eyebox

Vertical Field Of View (°)	**-65.00 mm**	**-32.50 mm**	**-0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	1.812	1.816	1.819	1.820	1.820
0.00 mm	1.815	1.817	1.819	1.818	1.816
-25.00 mm	1.831	1.831	1.831	1.828	1.826

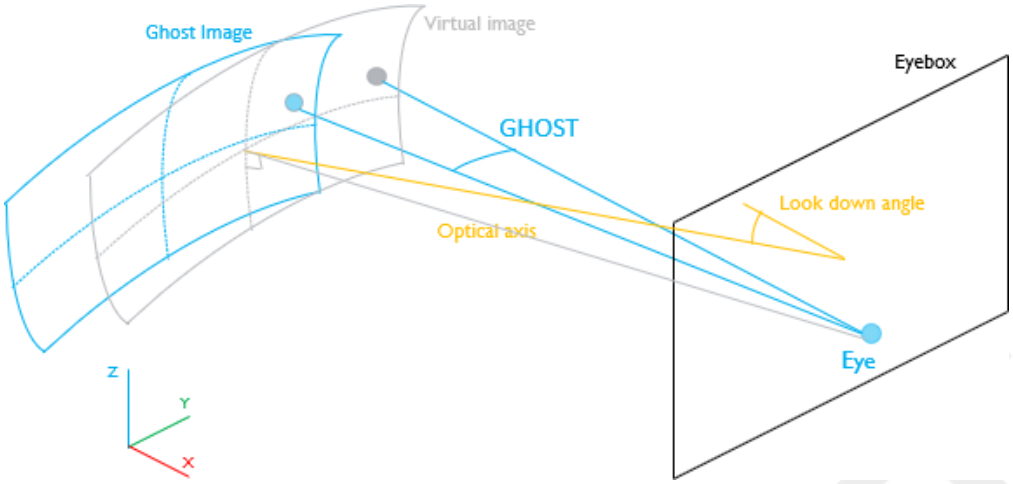
In particular, you can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the binocular eyebox.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.6. Ghost

The Ghost test gives the distance between the virtual image and the ghost image.

DRAFT



In detail:

For each sample EBij of the binocular eyebox, the "Ghost" gives the maximum, over the sample VIij of the virtual image seen from the sample EBij of the binocular eyebox, of the distance from the sample VIij of the virtual image seen from the sample EBij of the binocular eyebox to the sample VIij of the ghost image seen from the sample EBij of the binocular eyebox.

> **Note:** The "Ghost" is expressed in arcminute in the report. In Workbench interface, the "Ghost" unit is not displayed as Workbench is not able to retrieve the HOA plugin's units. However, usually the units between the report and Workbench match.

> **Note:** In the plugin, the ghost image is computed in three dimensions.

The result of the computation is displayed as a table that contains the "Ghost" computed for each sample EBij of the binocular eyebox.

> **Note:** The **Outer Surface** parameter of the Windshield tab must be set when using the Ghost test.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Ghost

top

Plugin description	OPTIS - HOA Plugin
Test description	Ghost

Configuration	**Upper Eyebox**	**Central Eyebox**	**Lower Eyebox**
Driver offset (mm)	40.000	0.000	-40.000
Average (')	2.242	0.780	1.592
Standard deviation (')	0.952	0.310	0.433
Minimum (')	1.057	0.237	0.925
Maximum (')	3.658	1.214	2.161

Upper Eyebox

Ghost (')	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	3.658	3.476	3.391	3.374	3.419
-0.00 mm	2.287	2.112	2.034	2.025	2.084
-25.00 mm	1.309	1.116	1.057	1.088	1.206

Central Eyebox

Ghost (')	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	1.214	1.050	0.989	1.000	1.083
0.00 mm	0.348	0.237	0.288	0.405	0.580
-25.00 mm	0.785	0.885	0.959	0.974	0.912

Lower Eyebox

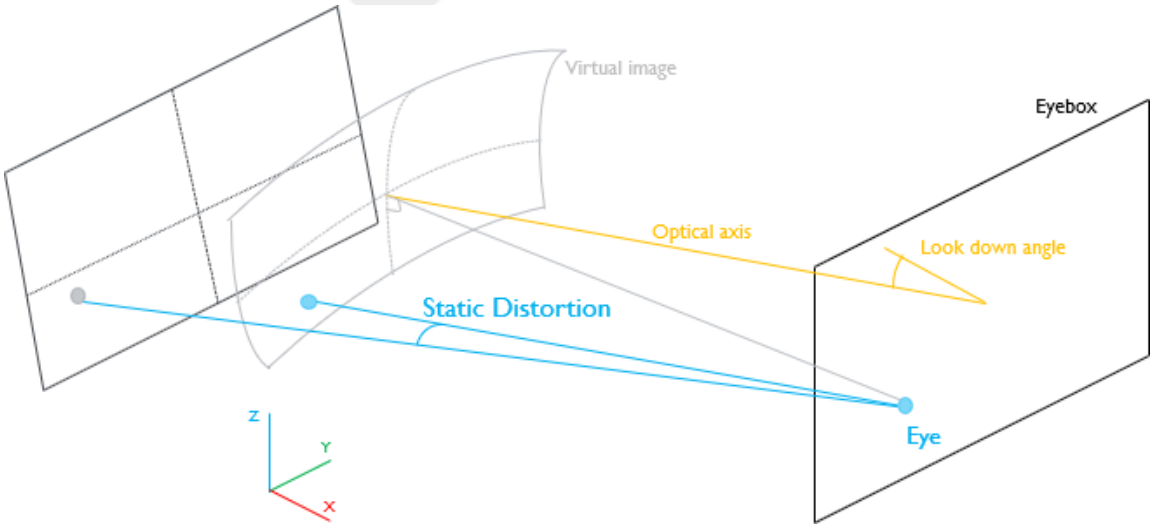
Ghost (')	**-65.00 mm**	**-32.50 mm**	**-0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	0.925	1.029	1.101	1.119	1.051
0.00 mm	1.526	1.632	1.703	1.716	1.635
-25.00 mm	1.981	2.087	2.156	2.161	2.066

In particular, you can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the virtual image.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.7. Static Distortion

The Static Distortion test gives the gap between the target image and the virtual image seen from the driver. This gap is due to the warping that is not ideal because it fits to the pixels of the PGU.



In detail:

For each sample VIij of the virtual image, the Static Distortion gives the maximum, over the sample EBij of the binocular eyebox, of the distance from the sample VIij of the virtual image seen from the sample EBij of the binocular eyebox to the sample VIij of the target image. The Static Distortion is expressed in arcminute.

The result of the computation is displayed as a table that contains the "Static Distortion" computed for each sample VIij of the virtual image.

Static Distortion top

Plugin description	OPTIS - HOA Plugin
Test description	Static Distortion

Configuration	**Upper Eyebox**	**Central Eyebox**	**Lower Eyebox**
Driver offset (mm)	40.000	0.000	-40.000
Average (')	5.794	2.963	5.467
Standard deviation (')	1.513	0.995	1.211
Minimum (')	3.164	1.792	4.230
Maximum (')	10.002	6.050	8.910

Upper Eyebox

Static Distortion (')	**-105.00 mm**	**-78.75 mm**	**-52.50 mm**	**-26.25 mm**	**0.00 mm**	**26.25 mm**	**52.50 mm**	**78.75 mm**	**105.00 mm**
35.00 mm	10.002	8.377	8.026	7.577	6.890	6.459	6.835	6.749	7.957
17.50 mm	8.816	7.215	6.952	5.991	5.743	5.538	5.950	5.766	6.107
0.00 mm	7.227	6.316	6.158	5.418	5.195	5.034	4.877	4.323	5.320
-17.50 mm	6.596	5.675	5.394	5.095	4.624	4.221	3.587	3.736	3.595
-35.00 mm	6.207	6.164	5.264	5.206	4.273	3.701	3.164	3.260	4.135

Central Eyebox

Static Distortion (')	**-105.00 mm**	**-78.75 mm**	**-52.50 mm**	**-26.25 mm**	**0.00 mm**	**26.25 mm**	**52.50 mm**	**78.75 mm**	**105.00 mm**
35.00 mm	3.124	2.208	2.666	2.817	2.920	2.679	1.887	2.293	3.562
17.50 mm	2.884	1.830	2.446	2.559	2.828	2.490	1.956	2.814	4.559
0.00 mm	3.150	2.104	2.140	2.463	2.575	2.256	2.240	3.226	4.814
-17.50 mm	3.377	2.541	2.717	2.662	2.473	1.792	2.681	4.061	5.995
-35.00 mm	4.590	3.050	2.889	2.509	2.041	2.426	3.082	4.904	6.050

Lower Eyebox

Static Distortion (')	**-105.00 mm**	**-78.75 mm**	**-52.50 mm**	**-26.25 mm**	**0.00 mm**	**26.25 mm**	**52.50 mm**	**78.75 mm**	**105.00 mm**
35.00 mm	4.776	4.695	4.264	5.123	4.642	5.235	5.562	6.175	6.909
17.50 mm	4.369	4.441	4.596	4.788	4.939	4.900	5.452	6.374	7.531
0.00 mm	4.657	4.488	4.230	4.404	4.529	5.110	5.609	7.003	8.033
-17.50 mm	5.325	4.631	4.628	4.622	4.710	5.274	5.705	7.025	8.910
-35.00 mm	6.170	5.273	4.599	4.239	4.260	5.416	6.163	7.515	8.738

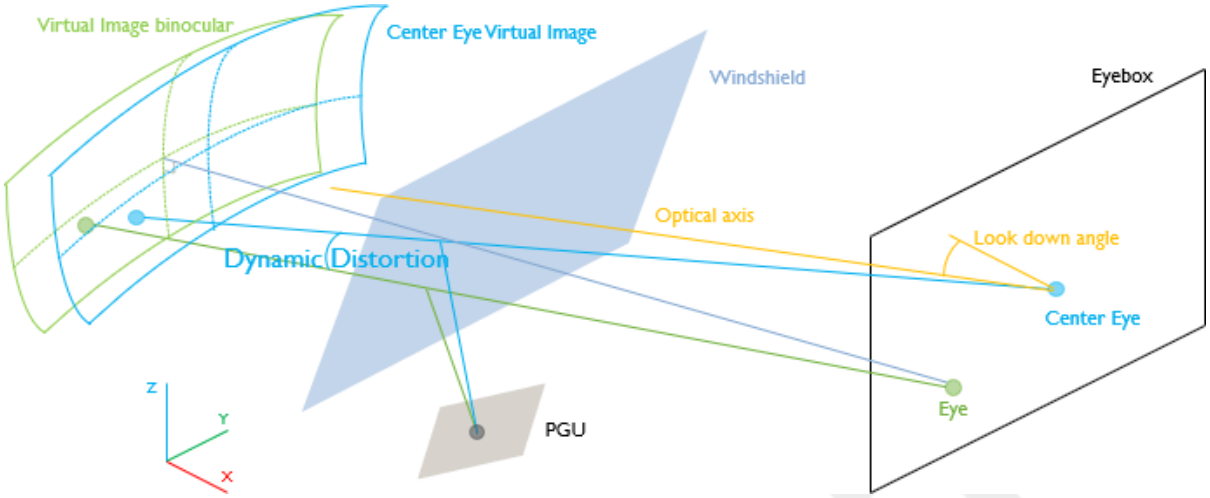
In particular, you can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the virtual image.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.8. Dynamic Distortion

The Dynamic Distortion test gives how the virtual image moves when the driver moves into the eyebox.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

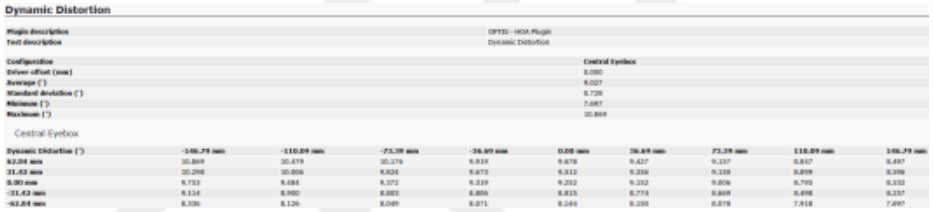


In detail:

For each sample VIij of the virtual image, the "Dynamic Distortion" gives the maximum, over the sample EBij of the binocular eyebox, of the distance from the sample VIij of the virtual image seen from the center of the binocular eyebox to the sample VIij of the virtual image seen from the sample EBij of the binocular eyebox. The "Dynamic Distortion" is expressed in arcminute.

> **Note:** In the plugin, the dynamic distortion (the virtual image seen from the sample EBij) is computed in three dimensions.

The result of the computation is displayed as a table that contains the "Dynamic Distortion" computed for each sample VIij of the virtual image.

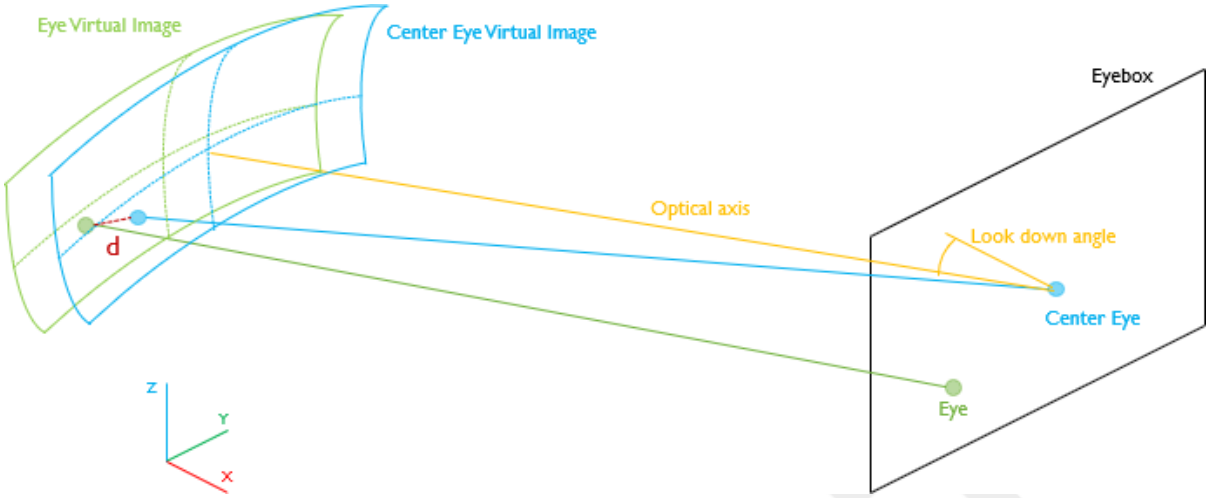


In particular, you can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the virtual image.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.9. Focus Variation

The Focus Variation corresponds to the average of the sum of the distance between VIEBij and VIEBcentereye.



In detail:

EBcentereye is the reference to calculate for each sample EBij the distance "d" between VIEBij and VIEBcentereye. Then Focus Variation for EBij corresponds to the average of the sum of all distances between VIEBij and VIEBcentereye. The "Focus Variation" is expressed in millimeters.

The result of the computation is displayed as a table that contains the "Focus Variation" computed for each sample VIij of the virtual image.

Focus Variation

Plugin description	ANSYS - HOA Plugin
Test description	Focus Variation

Configuration	**Upper Eyebox**	**Central Eyebox**	**Lower Eyebox**
Driver offset (mm)	35.000	0.000	-35.000
Average (mm)	95.233	48.805	32.468
Standard deviation (mm)	85.180	43.187	30.020
Minimum (mm)	0.000	0.000	0.000
Maximum (mm)	284.674	161.535	109.993

Upper Eyebox

Focus Variation (mm)	**-65.00 mm**	**-32.50 mm**	**-0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	284.674	199.627	169.795	172.671	199.455
0.00 mm	86.056	23.739	0.000	2.115	20.589
-25.00 mm	11.672	52.364	73.789	74.087	57.868

Central Eyebox

Focus Variation (mm)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	161.535	96.527	70.830	70.139	88.316
0.00 mm	76.787	22.666	0.000	2.579	14.571
-25.00 mm	43.754	10.017	27.147	29.314	17.900

Lower Eyebox

Focus Variation (mm)	**-65.00 mm**	**-32.50 mm**	**-0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	109.993	53.467	29.331	26.837	41.033
0.00 mm	72.863	22.500	0.000	4.102	11.550
-25.00 mm	66.963	19.279	7.872	10.086	11.144

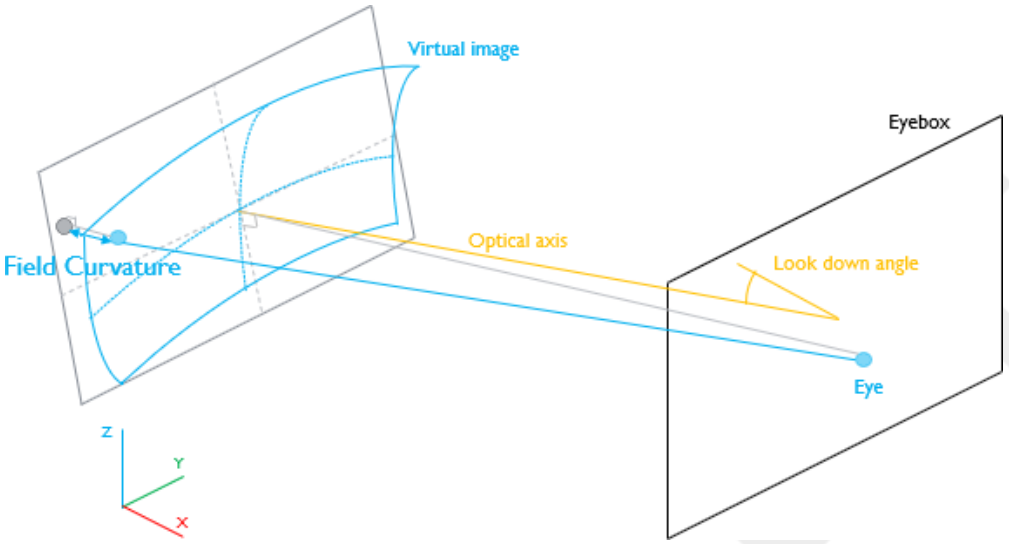
In particular, you can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the virtual image.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

14.3.1.3.10. Field Curvature

The Field Curvature test gives the distance between the virtual image and a plane normal to the optical axis at the center of the virtual image.



In detail:

For each sample VIij of the virtual image, the "Field Curvature" gives the mean of the *Field Curvature for VIij* over the EBij samples of the binocular eyebox, expressed in millimeter.

Field Curvature for VIij: For each sample EBij of the binocular eyebox, the field curvature for VIij is the distance from the sample VIij of the virtual image seen from the sample EBij of the binocular eyebox to the normal projection of this sample on the plane normal to the *optical axis* at the center of the virtual image seen from the sample EBij of the binocular eyebox.

Optical axis: For each sample EBij of the binocular eyebox, the optical axis is the line between the sample EBij of the binocular eyebox and the center of the virtual image seen from the sample EBij of the binocular eyebox.

The result of the computation is displayed as a table that contains the "Field Curvature" computed for each sample VIij of the virtual image.

DRAFT

Field Curvature

top

Plugin description	OPTIS - HOA Plugin
Test description	Field Curvature

Configuration	**Upper Eyebox**	**Central Eyebox**	**Lower Eyebox**
Driver offset (mm)	40.000	0.000	-40.000
Average (mm)	40.400	26.213	22.747
Standard deviation (mm)	28.365	22.551	20.753
Minimum (mm)	0.000	0.000	0.000
Maximum (mm)	125.054	85.131	71.909

Upper Eyebox

Field Curvature (mm)	**-105.00 mm**	**-78.75 mm**	**-52.50 mm**	**-26.25 mm**	**0.00 mm**	**26.25 mm**	**52.50 mm**	**78.75 mm**	**105.00 mm**
35.00 mm	56.660	57.841	58.312	54.170	44.619	28.943	17.934	25.389	50.496
17.50 mm	31.068	31.374	33.866	30.506	21.443	9.044	16.335	41.716	71.900
0.00 mm	16.206	9.089	11.016	8.406	0.000	15.034	36.270	63.855	93.680
-17.50 mm	24.475	14.938	11.039	11.947	19.874	34.405	55.719	81.912	112.091
-35.00 mm	43.068	33.910	28.686	30.066	37.118	50.995	71.142	96.367	125.054

Central Eyebox

Field Curvature (mm)	**-105.00 mm**	**-78.75 mm**	**-52.50 mm**	**-26.25 mm**	**0.00 mm**	**26.25 mm**	**52.50 mm**	**78.75 mm**	**105.00 mm**
35.00 mm	26.937	22.617	22.040	20.062	13.994	9.550	18.047	37.639	60.722
17.50 mm	18.827	13.058	14.364	13.207	7.081	6.662	23.086	45.395	71.326
0.00 mm	16.185	6.302	7.025	6.344	0.000	12.440	30.342	52.574	78.868
-17.50 mm	18.283	7.714	4.961	5.323	7.273	17.804	35.653	58.105	84.111
-35.00 mm	23.338	14.709	12.555	12.416	14.568	23.003	39.033	60.892	85.131

Lower Eyebox

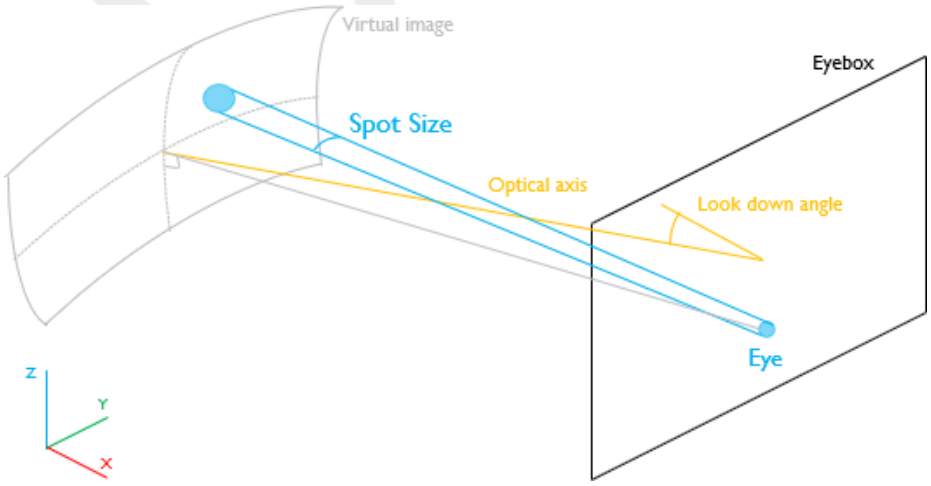
Field Curvature (mm)	**-105.00 mm**	**-78.75 mm**	**-52.50 mm**	**-26.25 mm**	**0.00 mm**	**26.25 mm**	**52.50 mm**	**78.75 mm**	**105.00 mm**
35.00 mm	25.882	16.716	10.824	7.793	7.642	15.042	29.623	48.473	69.959
17.50 mm	19.658	11.124	6.459	4.989	3.913	12.916	28.529	48.973	71.909
0.00 mm	15.998	6.483	6.011	5.592	0.000	11.222	27.245	48.251	71.777
-17.50 mm	14.148	6.538	7.901	7.627	4.517	9.158	24.978	45.899	69.619
-35.00 mm	16.151	11.446	11.989	11.708	9.752	10.851	22.741	41.535	64.072

In particular, you can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the virtual image.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.11. Spot Size

The Spot Size test gives the diameter of the spot on the virtual image.



In detail:

For each sample VIij of the virtual image, the Spot Size gives the mean, over the sample EBij of the binocular eyebox, of the diameter of the spot on the sample VIij of the virtual image seen by the sample EBij of the binocular eyebox.

The result of the computation is displayed as a table that contains the Spot Size computed for each sample VIij of the virtual image.

Spot Size

Plugin description	OPTIS - HOA Plugin
Test description	Spot Size

Configuration	**Central Eyebox**	**Lower Eyebox**	**Upper Eyebox**
Driver offset (mm)	0.000	-45.000	45.000
Average (')	0.135	0.252	0.355
Standard deviation (')	0.016	0.043	0.079
Minimum (')	0.098	0.177	0.264
Maximum (')	0.160	0.301	0.505

Central Eyebox

Spot Size (')	**-105.00 mm**	**-52.50 mm**	**0.00 mm**	**52.50 mm**	**105.00 mm**
35.00 mm	0.149	0.145	0.143	0.135	0.146
0.00 mm	0.131	0.118	0.126	0.136	0.143
-35.00 mm	0.157	0.107	0.098	0.130	0.160

Lower Eyebox

Spot Size (')	**-105.00 mm**	**-52.50 mm**	**0.00 mm**	**52.50 mm**	**105.00 mm**
35.00 mm	0.195	0.244	0.281	0.299	0.291
0.00 mm	0.177	0.232	0.273	0.297	0.289
-35.00 mm	0.180	0.205	0.240	0.279	0.301

Upper Eyebox

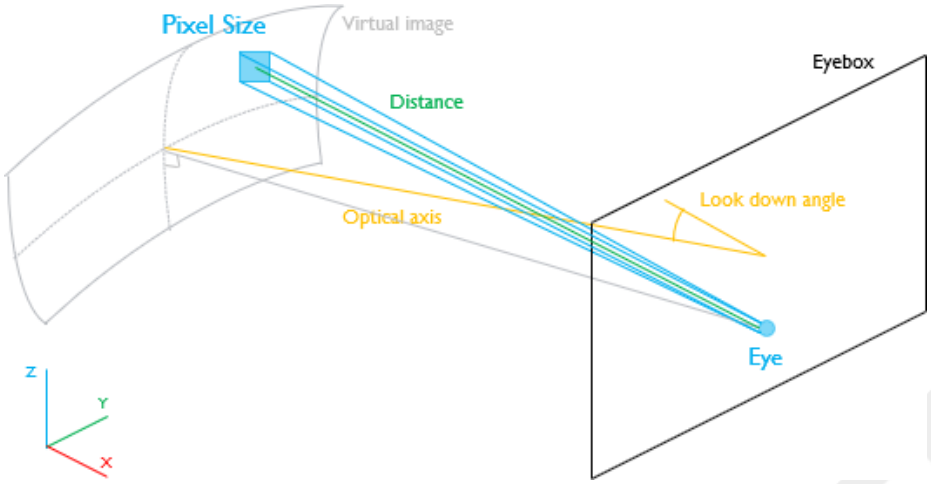
Spot Size (')	**-105.00 mm**	**-52.50 mm**	**0.00 mm**	**52.50 mm**	**105.00 mm**
35.00 mm	0.505	0.427	0.366	0.309	0.299
0.00 mm	0.476	0.386	0.319	0.273	0.272
-35.00 mm	0.469	0.377	0.305	0.264	0.271

In particular, you can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the virtual image.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.12. Pixel Size

The Pixel Size test gives the angle in minute of arc (arcmin) of the pixel on the virtual image.



Pixel Size = (Pixel Top Length + Pixel Right Length + Pixel Left Length + Pixel Bottom Length) / 4

Pixel Size = (2 * atan (Pixel Size / (2 * Distance)))

In detail:

For each sample VIij of the virtual image, the Pixel Size gives the mean in arcmin, over the sample EBij of the binocular eyebox, of the size of the pixel on the sample VIij of the virtual image seen by the sample EBij of the binocular eyebox.

The result of the computation is displayed as a table that contains the Pixel Size computed for each sample VIij of the virtual image.

DRAFT

Pixel Size

Plugin description	OPTIS - HOA Plugin
Test description	Pixel Size

Configuration	**Central Eyebox**	**Lower Eyebox**	**Upper Eyebox**
Driver offset (mm)	0.000	-45.000	45.000
Average (')	0.742	0.736	0.751
Standard deviation (')	0.008	0.008	0.008
Minimum (')	0.726	0.719	0.734
Maximum (')	0.753	0.745	0.763

Central Eyebox

Pixel Size (')	**-105.00 mm**	**-52.50 mm**	**0.00 mm**	**52.50 mm**	**105.00 mm**
35.00 mm	0.745	0.752	0.753	0.745	0.731
0.00 mm	0.742	0.749	0.750	0.742	0.727
-35.00 mm	0.740	0.747	0.747	0.740	0.726

Lower Eyebox

Pixel Size (')	**-105.00 mm**	**-52.50 mm**	**0.00 mm**	**52.50 mm**	**105.00 mm**
35.00 mm	0.739	0.745	0.745	0.737	0.722
0.00 mm	0.737	0.744	0.744	0.735	0.719
-35.00 mm	0.737	0.743	0.743	0.735	0.720

Upper Eyebox

Pixel Size (')	**-105.00 mm**	**-52.50 mm**	**0.00 mm**	**52.50 mm**	**105.00 mm**
35.00 mm	0.755	0.763	0.763	0.757	0.744
0.00 mm	0.750	0.758	0.759	0.751	0.737
-35.00 mm	0.746	0.754	0.755	0.748	0.734

In particular, you can notice that:

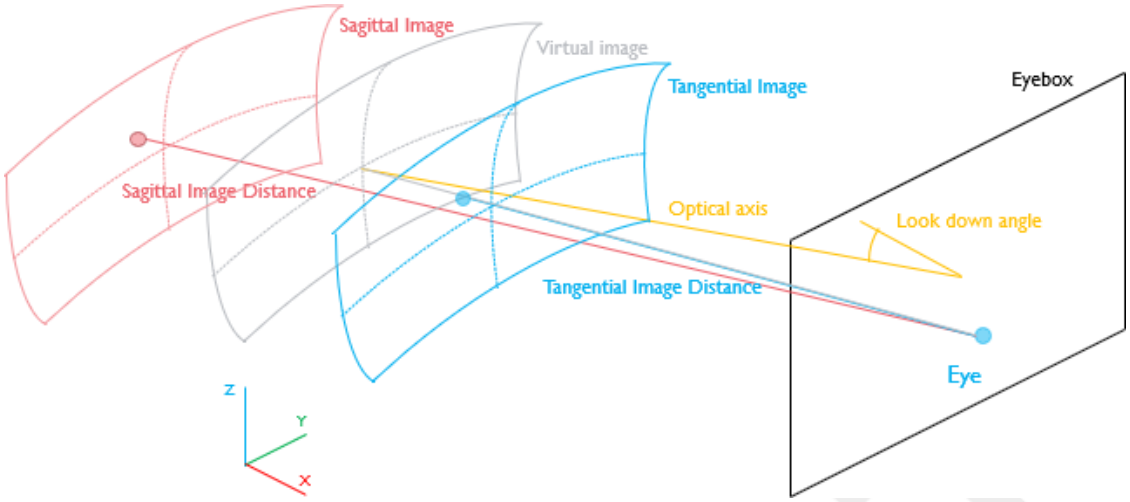
- There is one table result by configuration created.
- Each table has the size of the sampling of the virtual image.

The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.13. Astigmatism

The Astigmatism test gives the astigmatism of your system as defined on the picture.

DRAFT



Astigmatism = ç1/Tangential Image Distance - 1/Sagittal Image Distance ç

In detail:

For each sample VIij of the virtual image, the "Astigmatism" gives the mean of the *astigmatism for VIij* over the EBij samples of the binocular eyebox, expressed in dioptre.

Astigmatism for VIij : For each sample EBij of the binocular eyebox, the astigmatism for VIij is egal to Abs(1/*Tangential Image Distance for VIij* - 1/*Sagittal Image Distance for VIij*).

Tangential Image Distance for VIij : For each sample EBij of the binocular eyebox, the tangential image distance for VIij is the distance from the sample EBij of the binocular eyebox to the sample VIij of the tangential image seen from the sample EBij of the binocular eyebox. This distance is expressed in meter.

Sagittal Image Distance for VIij : For each sample EBij of the binocular eyebox, the sagittal image distance for VIij is the distance from the sample EBij of the binocular eyebox to the sample VIij of the sagittal image seen from the sample EBij of the binocular eyebox. This distance is expressed in meter.

> **Note:** In the plugin, the dynamic distortion of the sagittal and tangential images is computed in three dimensions.

The result of the computation is displayed as a table that contains the "Astigmatism" computed for each sample VIij of the virtual image.

DRAFT

Astigmatism

top

Plugin description	OPTIS - HOA Plugin
Test description	Astigmatism

Configuration	**Upper Eyebox**	**Central Eyebox**	**Lower Eyebox**
Driver offset (mm)	40.000	0.000	-40.000
Average (δ)	0.045	0.021	0.037
Standard deviation (δ)	0.009	0.002	0.005
Minimum (δ)	0.034	0.015	0.025
Maximum (δ)	0.065	0.025	0.043

Upper Eyebox

Astigmatism (δ)	**-105.00 mm**	**-78.75 mm**	**-52.50 mm**	**-26.25 mm**	**0.00 mm**	**26.25 mm**	**52.50 mm**	**78.75 mm**	**105.00 mm**
35.00 mm	0.065	0.059	0.054	0.050	0.046	0.042	0.039	0.037	0.038
17.50 mm	0.063	0.057	0.051	0.047	0.043	0.039	0.037	0.035	0.036
0.00 mm	0.063	0.055	0.049	0.045	0.041	0.037	0.035	0.034	0.036
-17.50 mm	0.063	0.055	0.049	0.044	0.039	0.036	0.034	0.034	0.036
-35.00 mm	0.063	0.056	0.050	0.044	0.040	0.037	0.035	0.035	0.038

Central Eyebox

Astigmatism (δ)	**-105.00 mm**	**-78.75 mm**	**-52.50 mm**	**-26.25 mm**	**0.00 mm**	**26.25 mm**	**52.50 mm**	**78.75 mm**	**105.00 mm**
35.00 mm	0.024	0.024	0.023	0.023	0.023	0.022	0.021	0.021	0.022
17.50 mm	0.022	0.021	0.021	0.021	0.021	0.021	0.021	0.021	0.022
0.00 mm	0.021	0.020	0.019	0.019	0.020	0.020	0.020	0.021	0.022
-17.50 mm	0.022	0.019	0.018	0.017	0.018	0.019	0.020	0.021	0.023
-35.00 mm	0.025	0.020	0.017	0.015	0.015	0.017	0.020	0.022	0.024

Lower Eyebox

Astigmatism (δ)	**-105.00 mm**	**-78.75 mm**	**-52.50 mm**	**-26.25 mm**	**0.00 mm**	**26.25 mm**	**52.50 mm**	**78.75 mm**	**105.00 mm**
35.00 mm	0.029	0.032	0.035	0.038	0.040	0.041	0.042	0.042	0.041
17.50 mm	0.027	0.030	0.034	0.037	0.040	0.042	0.042	0.042	0.041
0.00 mm	0.025	0.029	0.033	0.036	0.039	0.041	0.042	0.042	0.041
-17.50 mm	0.025	0.029	0.032	0.035	0.037	0.040	0.042	0.042	0.042
-35.00 mm	0.026	0.028	0.030	0.032	0.035	0.038	0.040	0.042	0.043

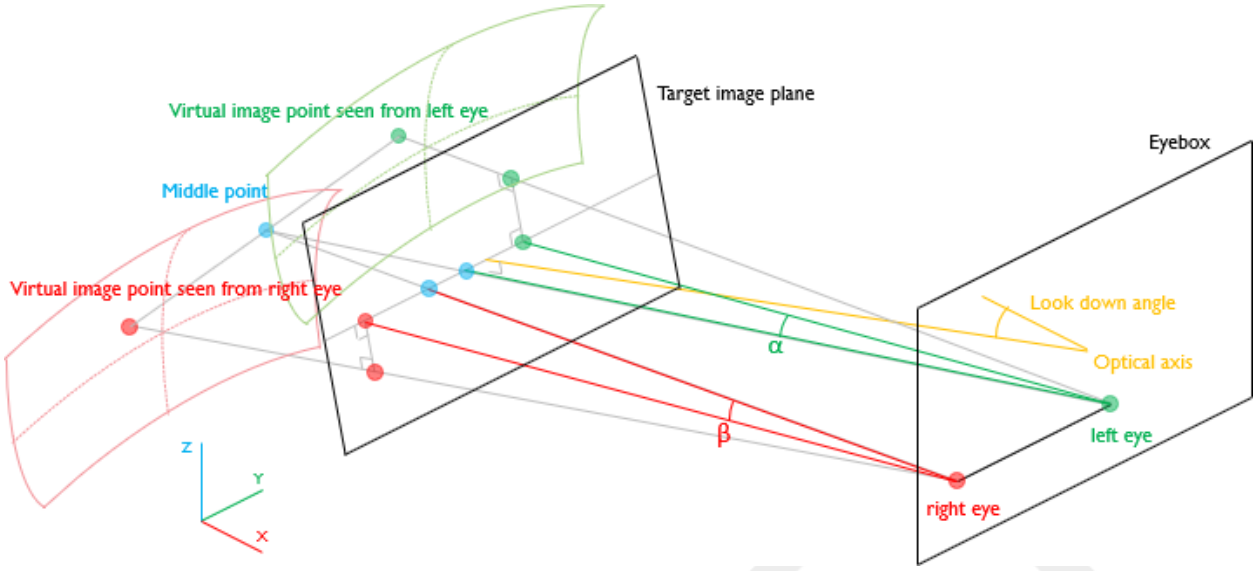
In particular, you can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the virtual image.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.14. Convergence

The Convergence test gives the horizontal position difference a point of the virtual image seen by the left eye and the same point seen by the right eye.

DRAFT



Convergence = β + α

In detail:

For each sample EBij of the monocular eyebox, the "Convergence" gives, in milliradian, the maximum, over the sample VIij of the virtual image, of the sum of the *delta left* and the *delta right*.

Delta right: Similar to *delta left* but for the right eye.

Delta left: For each sample EBij of the left monocular eyebox, the delta left is the angle between the *left optical axis* and *left horizontal virtual image axis*.

left optical axis: For each sample EBij of the left monocular eyebox, the left optical axis is the axis between the sample VIij of the target image and the sample EBij of the left monocular eyebox.

left horizontal virtual image axis: For each sample EBij of the left monocular eyebox, the left horizontal virtual image axis is the axis between the projection of the sample VIij of the virtual image seen from the sample EBij of the left monocular eyebox onto the target image plane and then projected on the horizontal axis of the target image.

> **Note:** In the plugin, the dynamic distortion (the virtual image seen from the sample EBij) is computed in three dimensions.

The result of the computation is displayed as a table that contains the "Convergence" computed for each sample EBij of the monocular eyebox.

DRAFT

Convergence

top

Plugin description	OPTIS - HOA Plugin
Test description	Convergence

Configuration	**Upper Eyebox**	**Central Eyebox**	**Lower Eyebox**
Driver offset (mm)	40.000	0.000	-40.000
Average (mrad)	9.411	3.820	6.720
Standard deviation (mrad)	5.355	0.591	0.970
Minimum (mrad)	3.406	2.870	5.003
Maximum (mrad)	20.452	4.663	7.966

Upper Eyebox

Convergence (mrad)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**
25.00 mm	12.916	14.658	20.452
-0.00 mm	5.853	6.836	10.333
-25.00 mm	3.406	3.816	6.424

Central Eyebox

Convergence (mrad)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**
25.00 mm	2.989	3.672	3.831
0.00 mm	4.475	4.663	3.488
-25.00 mm	4.254	4.134	2.870

Lower Eyebox

Convergence (mrad)	**-65.00 mm**	**-32.50 mm**	**-0.00 mm**
25.00 mm	7.525	7.176	5.687
0.00 mm	7.966	7.309	5.607
-25.00 mm	7.398	6.809	5.003

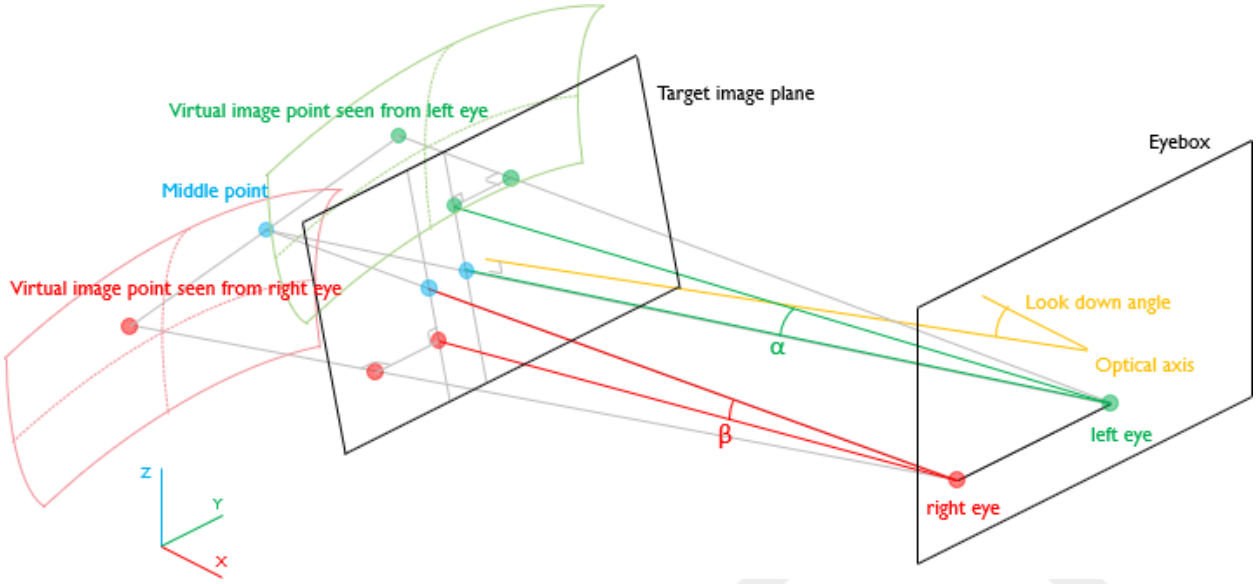
In particular, you can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the monocular eyebox.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.15. Dipvergence

The Dipvergence test gives the vertical position difference a point of the virtual image seen by the left eye and the same point seen by the right eye.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



Dipvergence = β + α

In detail:

For each sample EBij of the monocular eyebox, the Dipvergence gives, in milliradian, the maximum, over the sample VIij of the virtual image, of the sum of the *delta left* and the *delta right*.

Delta right: Similar to *delta left* but for the right eye.

Delta left: For each sample EBij of the left monocular eyebox, the delta left is the angle between the *left optical axis* and *left vertical virtual image axis*.

left optical axis: For each sample EBij of the left monocular eyebox, the left optical axis is the axis between the sample VIij of the target image and the sample EBij of the left monocular eyebox.

left vertical virtual image axis: For each sample EBij of the left monocular eyebox, the left vertical virtual image axis is the axis between the projection of the sample VIij of the virtual image seen from the sample EBij of the left monocular eyebox onto the target image plane and then projected on the vertical axis of the target image (and the sample EBij of the left monocular eyebox).

> **Note:** In the plugin, the dynamic distortion (the virtual image seen from the sample EBij) is computed in three dimensions.

The result of the computation is displayed as a table that contains the Dipvergence computed for each sample EBij of the monocular eyebox.

DRAFT

Dipvergence

top

Plugin description	OPTIS - HOA Plugin
Test description	Dipvergence

Configuration	Upper Eyebox	Central Eyebox	Lower Eyebox
Driver offset (mm)	40.000	0.000	-40.000
Average (mrad)	4.036	1.289	2.571
Standard deviation (mrad)	2.029	0.186	0.461
Minimum (mrad)	2.144	1.055	1.914
Maximum (mrad)	7.280	1.617	3.388

Upper Eyebox

Dipvergence (mrad)	-65.00 mm	-32.50 mm	0.00 mm
25.00 mm	6.763	6.546	7.280
-0.00 mm	2.939	2.408	2.791
-25.00 mm	2.144	2.288	3.165

Central Eyebox

Dipvergence (mrad)	-65.00 mm	-32.50 mm	0.00 mm
25.00 mm	1.359	1.055	1.221
0.00 mm	1.538	1.395	1.146
-25.00 mm	1.129	1.617	1.138

Lower Eyebox

Dipvergence (mrad)	-65.00 mm	-32.50 mm	-0.00 mm
25.00 mm	3.388	3.134	2.813
0.00 mm	2.596	2.380	1.914
-25.00 mm	2.111	2.659	2.140

In particular, you can notice that:

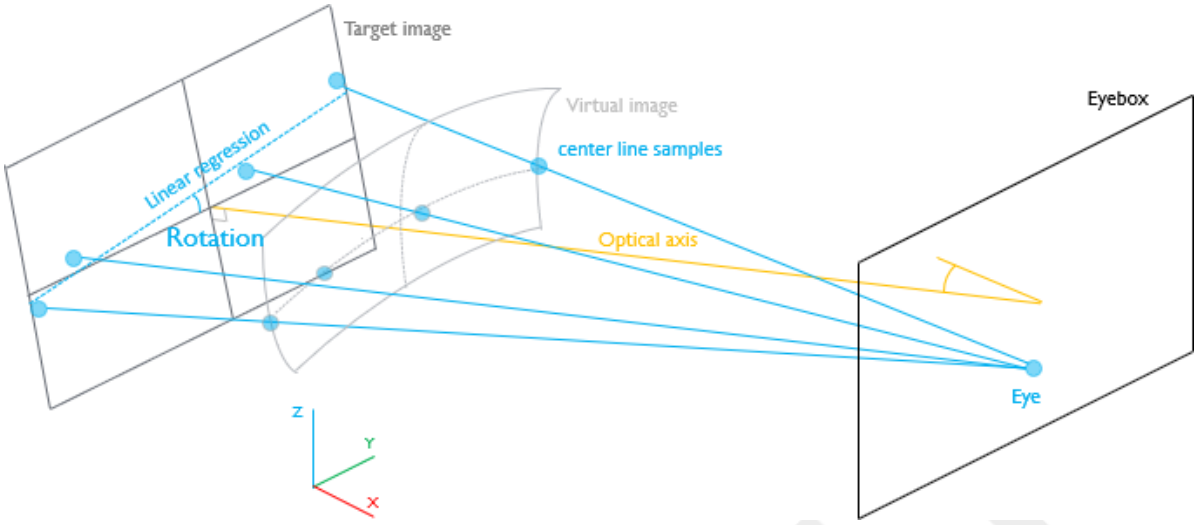
- There is one table result by configuration created.
- Each table has the size of the sampling of the monocular eyebox.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.16. Rotation

The Rotation test gives the angle between the horizontal line of the virtual image and the horizontal line of the target image.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00



In detail:

For each sample EBij of the binocular eyebox, the "Rotation" gives the angle, expressed in degrees, between the *horizontal line of the virtual image* and the *horizontal line of the target image*.

Horizontal line of the virtual image: For each sample EBij of the binocular eyebox, the horizontal line of the virtual image is the result of a linear regression over the sample of the horizontal center line of the virtual image seen from the sample EBij projected, with a normal projection, on the target image.

Horizontal line of the target image: The horizontal line of the target image in the line between the center of the target image and the middle right sample of the target image.

> **Note:** In the plugin, the dynamic distortion (the virtual image seen from the sample EBij) is computed in three dimensions.

The result of the computation is displayed as a table that contains the "Rotation" computed for each sample EBij of the binocular eyebox.

DRAFT

Rotation

top

Plugin description	OPTIS - HOA Plugin
Test description	Rotation

Configuration	**Upper Eyebox**	**Central Eyebox**	**Lower Eyebox**
Driver offset (mm)	40.000	0.000	-40.000
Average (°)	0.249	0.201	0.189
Standard deviation (°)	0.225	0.177	0.099
Minimum (°)	0.006	0.008	0.047
Maximum (°)	0.718	0.545	0.368

Upper Eyebox

Rotation (°)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	0.029	0.104	0.104	0.070	0.505
-0.00 mm	0.118	0.006	0.112	0.314	0.652
-25.00 mm	0.144	0.089	0.281	0.482	0.718

Central Eyebox

Rotation (°)	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	0.008	0.095	0.098	0.065	0.463
0.00 mm	0.141	0.055	0.048	0.233	0.531
-25.00 mm	0.229	0.026	0.146	0.333	0.545

Lower Eyebox

Rotation (°)	**-65.00 mm**	**-32.50 mm**	**-0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	0.081	0.202	0.225	0.081	0.284
0.00 mm	0.241	0.193	0.115	0.047	0.314
-25.00 mm	0.368	0.205	0.058	0.110	0.307

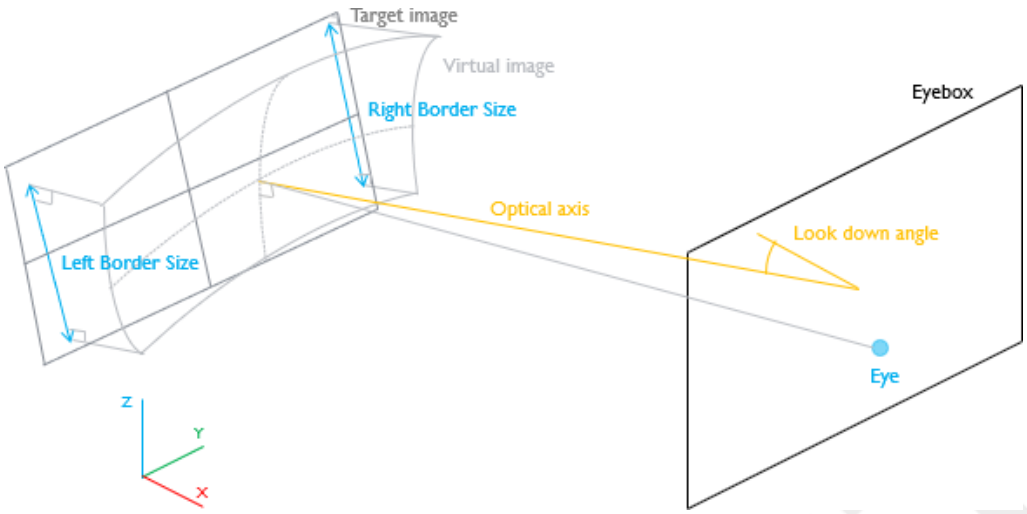
In particular, you can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the binocular eyebox.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.17. Horizontal Trapezoid

The Horizontal Trapezoid test gives the ratio between the size of the left border of the image and the size of the right border of the image.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



Horizontal Trapezoid = Left border size / Right border size

In detail:

For each sample EBij of the binocular eyebox, the Horizontal Trapezoid gives the ratio, *left border size* / *right border size*.

Left border size: For each sample EBij of the binocular eyebox, the left border size is the distance, in millimeter, from the top left sample of the virtual image seen from the sample EBij of the binocular eyebox and projected on the target image to the bottom left sample of the virtual image seen from the sample EBij of the binocular eyebox and projected on the target image.

Right border size: For each sample EBij of the binocular eyebox, the right border size is the distance, in millimeter, from the top right sample of the virtual image seen from the sample EBij of the binocular eyebox and projected on the target image to the bottom right sample of the virtual image seen from the sample EBij of the binocular eyebox and projected on the target image.

> **Note:** In the plugin, the dynamic distortion (the virtual image seen from the sample EBij) is computed in three dimensions.

The result of the computation is displayed as a table that contains the Horizontal Trapezoid computed for each sample EBij of the binocular eyebox.

DRAFT

Horizontal Trapezoid

top

Plugin description	OPTIS - HOA Plugin
Test description	Horizontal Trapezoid

Configuration	Upper Eyebox	Central Eyebox	Lower Eyebox
Driver offset (mm)	40.000	0.000	-40.000
Average	1.040	1.031	1.031
Standard deviation	0.011	0.009	0.010
Minimum	1.022	1.016	1.015
Maximum	1.068	1.048	1.051

Upper Eyebox

Horizontal Trapezoid	-65.00 mm	-32.50 mm	0.00 mm	32.50 mm	65.00 mm
25.00 mm	1.068	1.045	1.046	1.050	1.037
-0.00 mm	1.055	1.034	1.035	1.038	1.024
-25.00 mm	1.047	1.028	1.032	1.036	1.022

Central Eyebox

Horizontal Trapezoid	-65.00 mm	-32.50 mm	0.00 mm	32.50 mm	65.00 mm
25.00 mm	1.048	1.030	1.031	1.033	1.020
0.00 mm	1.047	1.028	1.029	1.030	1.016
-25.00 mm	1.046	1.028	1.030	1.033	1.019

Lower Eyebox

Horizontal Trapezoid	-65.00 mm	-32.50 mm	-0.00 mm	32.50 mm	65.00 mm
25.00 mm	1.043	1.027	1.028	1.029	1.017
0.00 mm	1.048	1.029	1.029	1.029	1.015
-25.00 mm	1.051	1.031	1.032	1.035	1.019

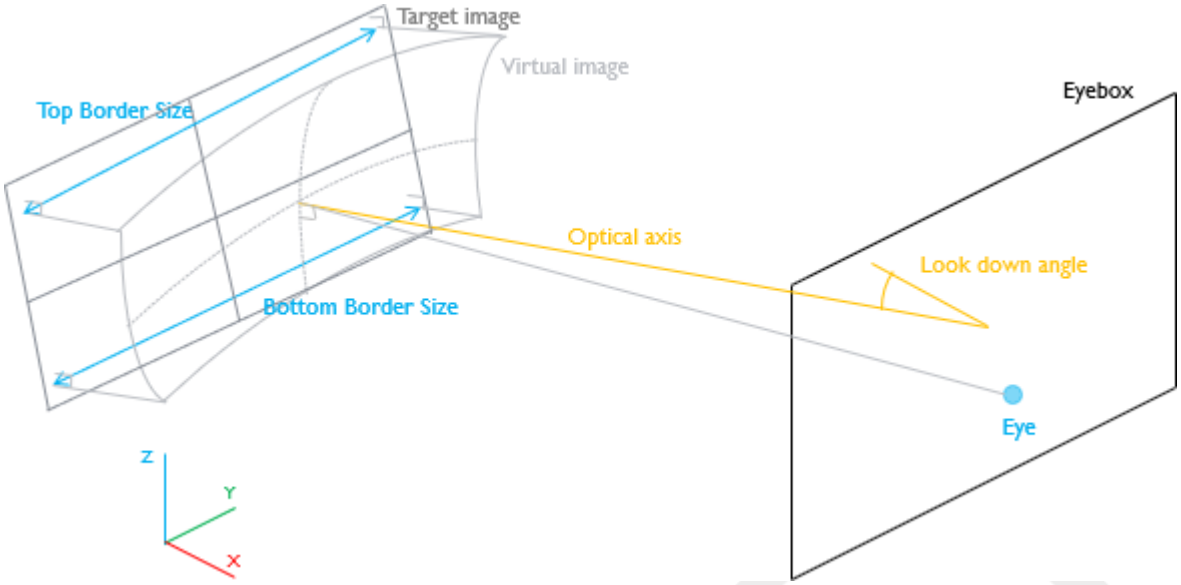
In particular, you can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the binocular eyebox.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.18. Vertical Trapezoid

The Vertical Trapezoid test gives the ratio between the size of the top border of the image and the size of the bottom border of the image.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



Vertical Trapezoid = Top border size / Bottom border size

In detail:

For each sample EBij of the binocular eyebox, the Vertical Trapezoid gives the ratio, *top border size* / *bottom border size*.

Top border size: For each sample EBij of the binocular eyebox, the top border size is the distance, in millimeter, from the top left sample of the virtual image seen from the sample EBij of the binocular eyebox and projected on the target image to the top right sample of the virtual image seen from the sample EBij of the binocular eyebox and projected on the target image.

Bottom border size: For each sample EBij of the binocular eyebox, the bottom border size is the distance, in millimeter, from the bottom left sample of the virtual image seen from the sample EBij of the binocular eyebox and projected on the target image to the bottom right sample of the virtual image seen from the sample EBij of the binocular eyebox and projected on the target image.

> **Note:** In the plugin, the dynamic distortion (the virtual image seen from the sample EBij) is computed in three dimensions.

The result of the computation is displayed as a table that contains the Vertical Trapezoid computed for each sample EBij of the binocular eyebox.

DRAFT

Vertical Trapezoid

top

Plugin description	OPTIS - HOA Plugin
Test description	Vertical Trapezoid

Configuration	Upper Eyebox	Central Eyebox	Lower Eyebox
Driver offset (mm)	40.000	0.000	-40.000
Average	1.038	1.011	0.997
Standard deviation	0.022	0.014	0.012
Minimum	1.005	0.986	0.972
Maximum	1.075	1.036	1.017

Upper Eyebox

Vertical Trapezoid	-65.00 mm	-32.50 mm	0.00 mm	32.50 mm	65.00 mm
25.00 mm	1.075	1.063	1.060	1.061	1.067
-0.00 mm	1.041	1.036	1.034	1.033	1.032
-25.00 mm	1.016	1.016	1.015	1.011	1.005

Central Eyebox

Vertical Trapezoid	-65.00 mm	-32.50 mm	0.00 mm	32.50 mm	65.00 mm
25.00 mm	1.036	1.028	1.025	1.024	1.026
0.00 mm	1.017	1.014	1.011	1.009	1.006
-25.00 mm	0.997	0.999	0.998	0.993	0.986

Lower Eyebox

Vertical Trapezoid	-65.00 mm	-32.50 mm	-0.00 mm	32.50 mm	65.00 mm
25.00 mm	1.017	1.011	1.008	1.006	1.006
0.00 mm	1.003	1.000	0.998	0.995	0.991
-25.00 mm	0.984	0.987	0.987	0.982	0.972

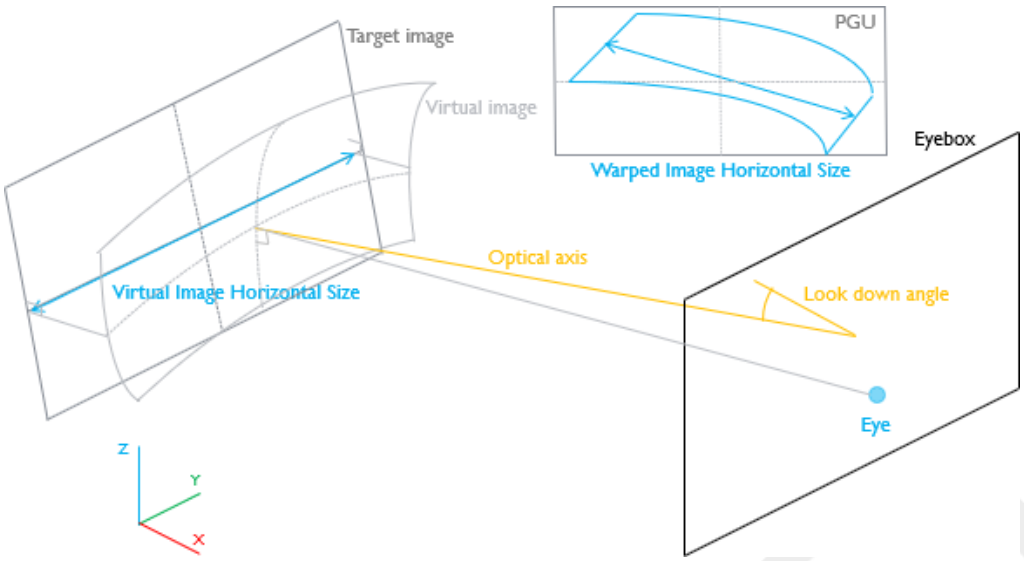
In particular, you can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the binocular eyebox.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.19. Horizontal Magnification

The Horizontal Magnification test gives the ratio between the horizontal size of the virtual image and the horizontal size of the warping.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



Horizontal Magnification = Virtual image horizontal size / horizontal size

In detail:

For each sample EBij of the binocular eyebox, the Horizontal Magnification gives the ratio, *virtual image horizontal size* / *warping horizontal size*.

Virtual image horizontal size: For each sample EBij of the binocular eyebox, the virtual image horizontal size is the distance, in millimeter, from the middle left sample of the virtual image seen from the sample EBij of the binocular eyebox and projected on the target image to the middle right sample of the virtual image seen from the sample EBij of the binocular eyebox and projected on the target image.

Warping horizontal size: The warping horizontal size is the distance, in millimeter, from the middle left sample of the warping to the middle right sample of the warping.

> **Note:** In the plugin, the dynamic distortion (the virtual image seen from the sample EBij) is computed in three dimensions.

The result of the computation is displayed as a table that contains the Horizontal Magnification computed for each sample EBij of the binocular eyebox.

DRAFT

Horizontal Magnification

top

Plugin description	OPTIS - HOA Plugin
Test description	Horizontal Magnification

Configuration	**Upper Eyebox**	**Central Eyebox**	**Lower Eyebox**
Driver offset (mm)	40.000	0.000	-40.000
Average	6.561	6.126	5.869
Standard deviation	0.259	0.117	0.096
Minimum	6.227	5.983	5.749
Maximum	7.134	6.384	6.091

Upper Eyebox

Horizontal Magnification	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	6.884	6.726	6.726	6.863	7.134
-0.00 mm	6.482	6.356	6.355	6.468	6.698
-25.00 mm	6.351	6.230	6.227	6.338	6.570

Central Eyebox

Horizontal Magnification	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	6.233	6.108	6.097	6.189	6.384
0.00 mm	6.105	5.993	5.983	6.072	6.260
-25.00 mm	6.114	5.996	5.986	6.082	6.289

Lower Eyebox

Horizontal Magnification	**-65.00 mm**	**-32.50 mm**	**-0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	5.891	5.779	5.761	5.831	5.990
0.00 mm	5.871	5.764	5.749	5.822	5.986
-25.00 mm	5.951	5.833	5.817	5.900	6.091

In particular, you can notice that:

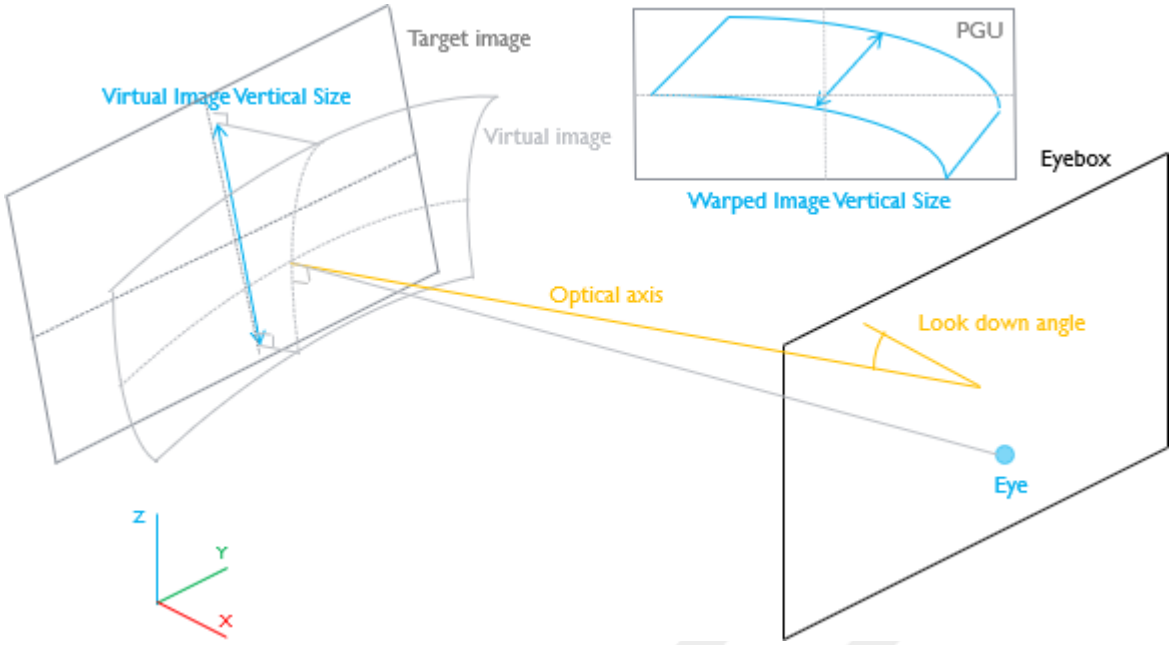
- There is one table result by configuration created.
- Each table has the size of the sampling of the binocular eyebox.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.20. Vertical Magnification

The Vertical Magnification test gives the ratio between the vertical size of the virtual image and the vertical size of the warping.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00



Vertical Magnification = Virtual image vertical size / Warping vertical size

In detail:

For each sample EBij of the binocular eyebox, the Vertical Magnification gives the ratio, *virtual image vertical size* / *warping vertical size*.

Virtual image vertical size: For each sample EBij of the binocular eyebox, the virtual image vertical size is the distance, in millimeter, from the top middle sample of the virtual image seen from the sample EBij of the binocular eyebox and projected on the target image to the bottom middle sample of the virtual image seen from the sample EBij of the binocular eyebox and projected on the target image.

Warping vertical size: The warping vertical size is the distance, in millimeter, from the top middle sample of the warping to the bottom middle sample of the warping.

> **Note:** In the plugin, the dynamic distortion (the virtual image seen from the sample EBij) is computed in three dimensions.

The result of the computation is displayed as a table that contains the Vertical Magnification computed for each sample EBij of the binocular eyebox.

DRAFT

Vertical Magnification

top

Plugin description	OPTIS - HOA Plugin
Test description	Vertical Magnification

Configuration	Upper Eyebox	Central Eyebox	Lower Eyebox
Driver offset (mm)	40.000	0.000	-40.000
Average	5.450	5.014	4.786
Standard deviation	0.202	0.072	0.071
Minimum	5.196	4.914	4.691
Maximum	5.876	5.175	4.933

Upper Eyebox

Vertical Magnification	-65.00 mm	-32.50 mm	0.00 mm	32.50 mm	65.00 mm
25.00 mm	5.718	5.607	5.611	5.697	5.876
-0.00 mm	5.396	5.299	5.297	5.366	5.520
-25.00 mm	5.296	5.203	5.196	5.256	5.403

Central Eyebox

Vertical Magnification	-65.00 mm	-32.50 mm	0.00 mm	32.50 mm	65.00 mm
25.00 mm	5.073	4.992	4.989	5.046	5.175
0.00 mm	5.008	4.923	4.914	4.965	5.090
-25.00 mm	5.042	4.952	4.937	4.985	5.112

Lower Eyebox

Vertical Magnification	-65.00 mm	-32.50 mm	-0.00 mm	32.50 mm	65.00 mm
25.00 mm	4.774	4.700	4.691	4.735	4.842
0.00 mm	4.801	4.718	4.702	4.743	4.853
-25.00 mm	4.896	4.802	4.781	4.818	4.933

In particular, yopu can notice that:

- There is one table result by configuration created.
- Each table has the size of the sampling of the binocular eyebox.
- The test also displays a sum up, for each configuration, of the Average, the Standard deviation, the Minimum and the Maximum of the value contained in the table.

14.3.1.3.21. Overview

The Overview test gives an overview of all the tests of the OPTIS plugin.

In detail:

For each test of the plugin, the Overview gives the average value of the test.

DRAFT

Overview

top

Plugin description	OPTIS - HOA Plugin
Test description	Overview

	Upper Eyebox average	Central Eyebox average	Lower Eyebox average
Virtual Image Distace (mm)	2363.850	2214.880	2133.877
Look Over Angle (°)	-0.000	-0.002	0.001
Look Down Angle (°)	-7.154	-5.003	-2.751
Horizontal Field of View (°)	5.481	5.472	5.466
Vertical Field of View (°)	1.828	1.827	1.821
Astigmatism (δ)	0.045	0.021	0.037
Dynamic Distortion (mm)	3.721	2.326	3.450
Horizontal Magnification ()	6.561	6.126	5.869
Vertical Magnification ()	5.450	5.014	4.786
Horizontal Trapezoid ()	1.040	1.031	1.031
Vertical Trapezoid ()	1.038	1.011	0.997
Rotation (°)	0.249	0.201	0.189
Field Curvature (mm)	77.157	67.594	150.483
Convergence (mrad)	65.169	80.282	101.053
Dipvergence (mrad)	64.750	80.074	100.887
Ghost (mm)	1.236	0.102	0.892
Spot Size (mm)	0.362	0.084	0.238
Static Distortion (mm)	0.213	0.205	0.205

> **Note:** There is one row per created configuration.

The Ghost and the Magnification tests require some settings to be set. Refer to these chapters for more details.

14.3.1.3.22. Detailed Overview

The Detailed Overview test gives a detailed overview of all the tests of the Speos plugin.

In detail:

For each test of the plugin, the Overview gives the average, the standard deviation, the minimum and the maximum value of the test.

Detailed Overview

Plugin description

Test description

	Upper Eyebox average	Upper Eyebox standard deviation	Upper Eyebox minimum	Upper Eyebox maximum	Centr averag
Virtual Image Distace (mm)	2363.850	83.819	2257.105	2525.486	2214.8
Look Over Angle (°)	-0.000	1.185	-0.000	1.686	-0.002
Look Down Angle (°)	-7.154	0.479	-6.551	-7.738	-5.003
Horizontal Field of View (°)	5.481	0.013	5.463	5.513	5.472
Vertical Field of View (°)	1.828	0.020	1.802	1.867	1.827
Astigmatism (δ)	0.045	0.009	0.034	0.065	0.021
Dynamic Distortion (mm)	3.721	1.016	2.380	6.687	2.326
Horizontal Magnification ()	6.561	0.259	6.227	7.134	6.126
Vertical Magnification ()	5.450	0.202	5.196	5.876	5.014
Horizontal Trapezoid ()	1.040	0.011	1.022	1.068	1.031
Vertical Trapezoid ()	1.038	0.022	1.005	1.075	1.011
Rotation (°)	0.249	0.225	0.006	0.718	0.201
Field Curvature (mm)	77.157	37.019	4.189	140.076	67.594
Convergence (mrad)	65.169	25.086	31.839	104.210	80.282
Dipvergence (mrad)	64.750	24.489	31.695	100.680	80.074
Ghost (mm)	1.236	0.074	1.064	1.343	0.102
Spot Size (mm)	0.362	0.104	0.226	0.577	0.084
Static Distortion (mm)	0.213	0.103	0.002	0.366	0.205

There are four rows per created configuration.

14.3.1.4. Speos Plugin Examples

The following section presents you HUD Optical Analysis plugins and the metrics they use.

Plugins are available under C:\Program Files\ANSYS Inc\v212\Optical Products\Plugins

In order for the plugins to work, you need to install the Visual C++ 2019 Redistributable Package.

> **Important:** All plugin *.dll files are loaded when starting a Speos session. That means the files are locked and no action can be performed while Speos is opened.

14.3.1.4.1. Plugin - Ansys Metrics

Description

This plugin contains the default metrics delivered by Ansys with the HUD Optical Analysis (HOA) product.

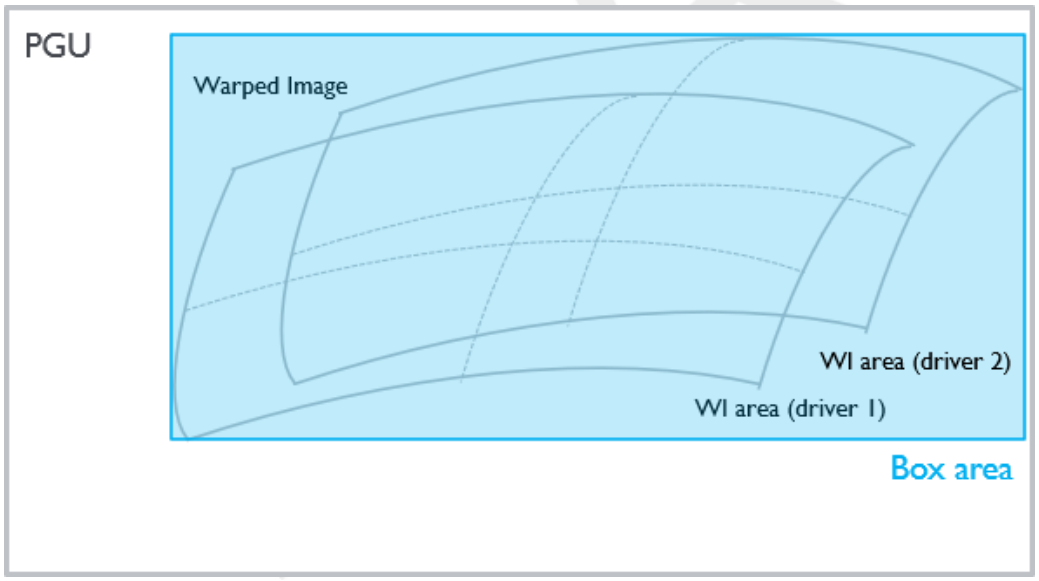
It provides methods to qualify HUD's virtual image metrics:

- Virtual Image Distance
- Look Over Angle
- Look Down Angle
- Horizontal Field Of View

- Vertical Field Of View
- Ghost
- Static Distortion
- Dynamic Distortion
- Focus Variation
- Field Curvature
- Spot Size
- Pixel Size
- Astigmatism
- Convergence
- Dipvergence
- Rotation
- Horizontal Trapezoid
- Vertical Trapezoid
- Horizontal Magnification
- Vertical Magnification
- PGU Usage (see below)
- Overview
- Detailed Overview

PGU Usage

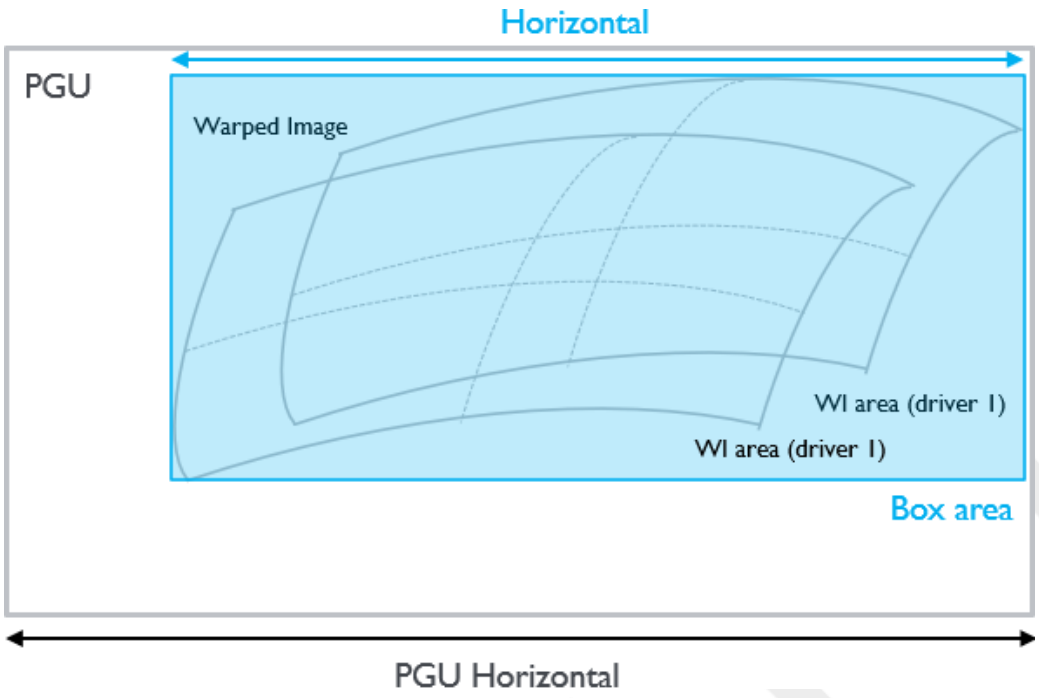
Bounding Box Definition



Bounding Box = Box area / PGU area * 100

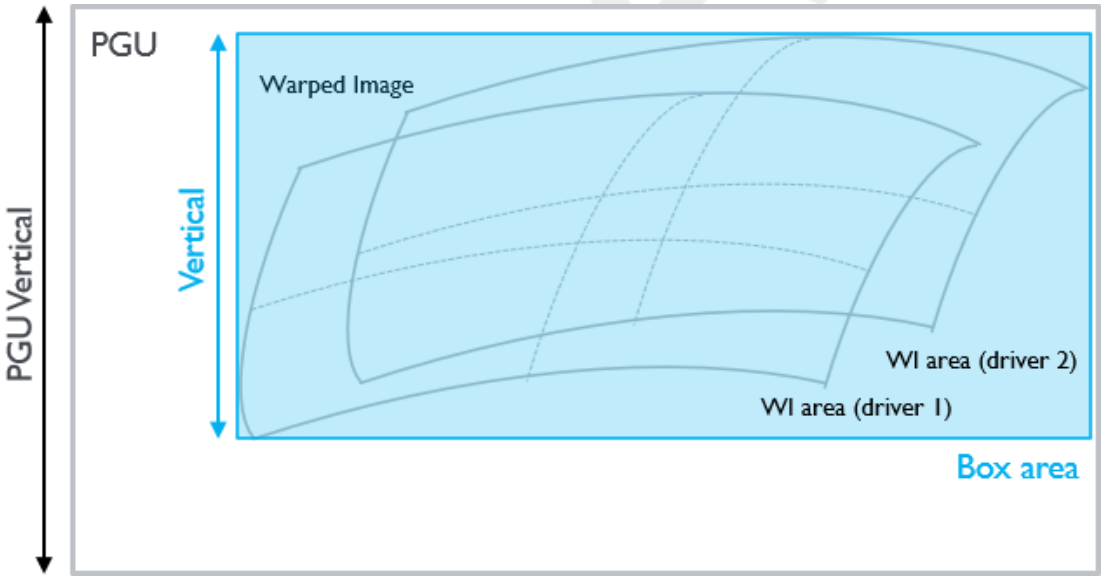
DRAFT

Horizontal Definition



Horizontal = Horizontal / PGU Horizontal * 100

Vertical Definition



Vertical = Vertical / PGU Vertical * 100

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Report

Figure 110. Normal Warping Image

PGU Usage

Plugin description	ANSYS - HOA Plugin			
Test description	PGU Usage			
	Upper Eyebox	**Central Eyebox**	**Lower Eyebox**	**All Eyeboxes**
Bounding Box area (%)	36.50	36.85	37.19	37.33
Horizontal size (%)	66.62	66.50	66.37	66.62
Vertical size (%)	54.79	55.41	56.04	56.04

Figure 111. Warping Image Outside of the PGU (red value)

PGU Usage

Plugin description	ANSYS - HOA Plugin			
Test description	PGU Usage			
	Upper Eyebox	**Central Eyebox**	**Lower Eyebox**	**All Eyeboxes**
Bounding Box area (%)	54.79	55.41	56.04	56.04
Horizontal size (%)	100.00	100.00	100.00	100.00
Vertical size (%)	54.79	55.41	56.04	56.04

14.3.1.4.2. Plugin - Projector Image

Description

This plugin is used to minimize the distance between the virtual image and the target image, and optimize the field curvature of the virtual image by ajusting the PGU's location in a HUD system with the Hud Optical Analysis (HOA).

It provides methods to qualify HUD’s virtual image metrics:

- Field Curvature
- Virtual Image Distance from Target Image

You can find a use case of the Projector Image here.

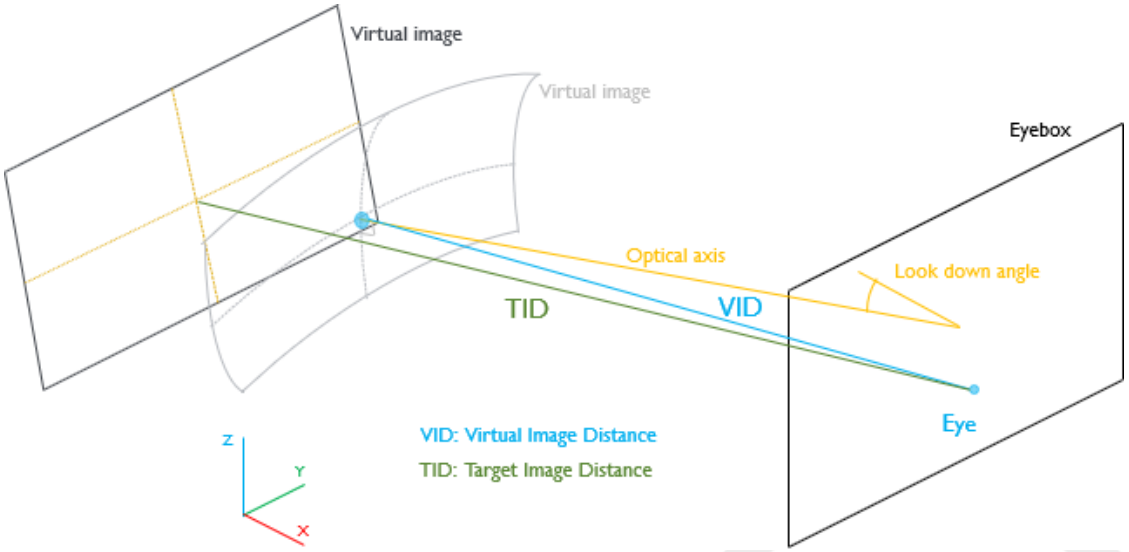
Projector and Projector Image

A projector can be used to display an image and can illuminate directly the windshield to simulate the Head Up Display system. The orientation and position of projector image are quite critical to obtain correct virtual image orientation and position seen from predefined eyebox.

Speos can help you build this optimized projector position and orientation using HUD Optical Analysis Plugin.

DRAFT

Virtual Image Distance from Target Image

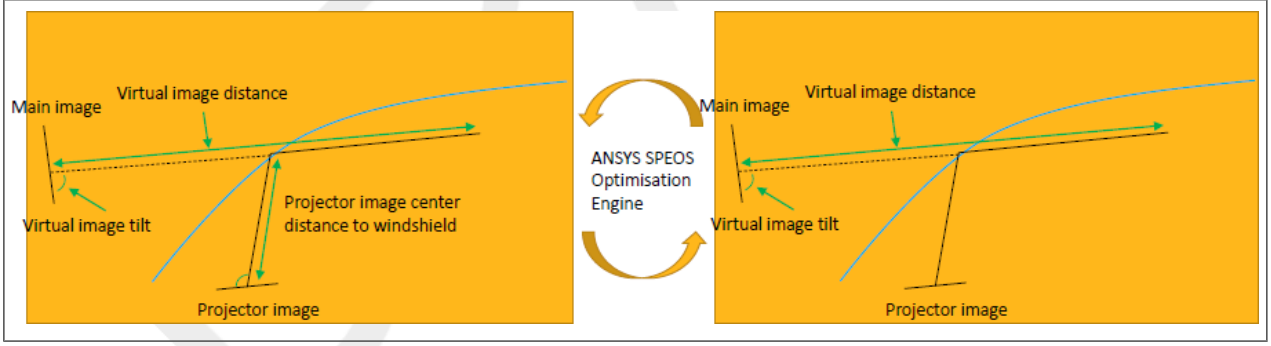


Unit: millimeters (mm)

Virtual image distance From Target Image = VID - TID

Essential for Optimizing Projector Image (Example)

Defining variables:	Defining targets:
• Projector image center (Geometrical parameter) • Projector image orientation (Geometrical parameter)	• Virtual image distance (output of HUD Optical Analysis metrics) • Virtual image tilt (output of HUD Optical Analysis metrics)



Known Issue

Non-consistent units

Values under the Report node of a HUD Optical Analysis in the Simulation tree are expressed in meter (m).

Values in the definition of the HUD Optical Analysis and in the HTML Report are expressed in millimeter (mm).

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

14.3.1.4.3. Plugin - Wedge Angle Determination

Description

This plugin is used to optimize the wedge angle of the windshield to minimize the ghost image in a HUD system with the Hud Optical Analysis (HOA) product.

It provides methods to qualify HUD's virtual image metrics Ghost.

You can find a use case of the Projector Image here.

Ghost Image

When the image hits the inner surface of the windshield, it is split into two images

- The main image that is reflected by the inner surface of the windshield
- The ghost image that goes through the inner surface of the windshield is reflected by the outer surface of the windshield

As a result, the driver sees two images that give an impression of blur

Figure 112. Image seen by the driver



Wedge Angle

To solve the ghost issue, one solution is to superpose both images seen by the driver.

To do that, the windshield is built with an angle between its two faces as shown on the image. This angle is called the "wedge angle".

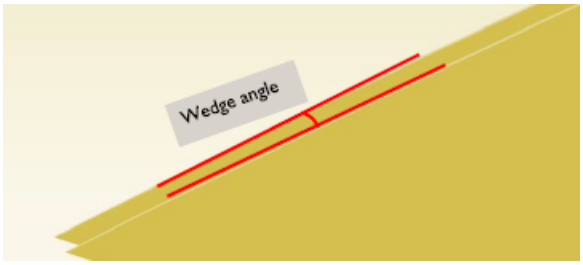




Image produced with a windshield without wedge angle



Image produced with a windshield with a wedge angle

DRAFT

Ghost analysis and optimization with Speos

Speos can analyze ghost images of Head Up Display to generate ghost report. Moreover it can help you optimize the ghost as a target by taking the wedge angle as a variable. Example:

> **Note:** The "Ghost" is expressed in arcminute by default in the report. In Speos and Workbench, the "Ghost" is always expressed in degrees.

Ghost

Plugin description	OPTIS - HOA Plugin
Test description	Ghost

Configuration	**Central Eyebox**
Driver offset (mm)	0.000
Average (')	6.145
Standard deviation (')	0.196
Minimum (')	5.814
Maximum (')	6.453

Central Eyebox

Ghost (')	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	5.814	5.861	5.913	5.961	6.003
0.00 mm	6.077	6.114	6.164	6.211	6.252
-25.00 mm	6.272	6.313	6.362	6.409	6.453

Ghost before wedge optimization

Ghost

Plugin description	OPTIS - HOA Plugin
Test description	Ghost

Configuration	**Central Eyebox**
Driver offset (mm)	0.000
Average (')	1.561
Standard deviation (')	0.415
Minimum (')	0.856
Maximum (')	2.295

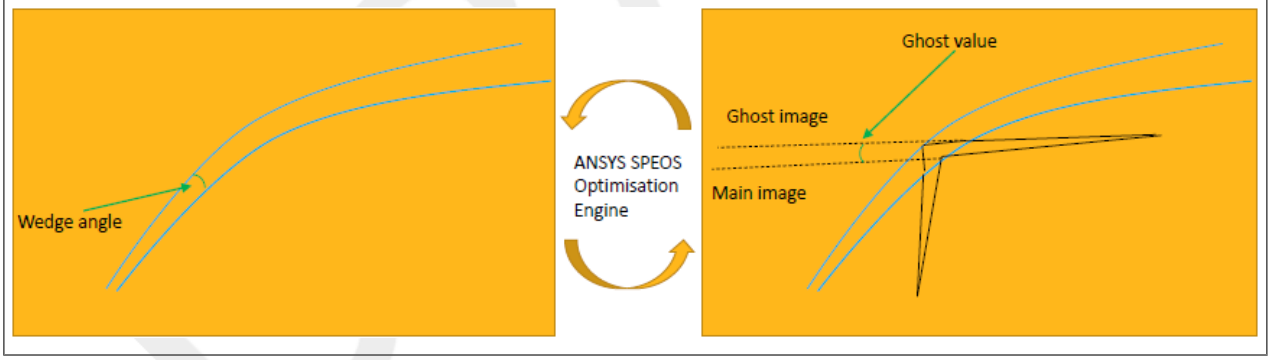
Central Eyebox

Ghost (')	**-65.00 mm**	**-32.50 mm**	**0.00 mm**	**32.50 mm**	**65.00 mm**
25.00 mm	1.918	1.460	1.151	0.953	0.856
0.00 mm	1.983	1.461	1.258	1.257	1.272
-25.00 mm	2.295	1.889	1.862	1.882	1.912

Ghost after wedge optimization

Essential for Optimizing Wedge Angle (Example)

Defining variables: windshield wedge angle (Geometrical parameter)	Defining targets: customized ghost value (output of HUD Optical Analysis metrics)



14.3.2. Results

14.3.2.1. Eyebox

The Eyebox result displays the eyebox you specified in the Eyebox tab.

If you specified a Uniform eyebox, it displays a rectangle with the size and the sampling of your binocular eyebox.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

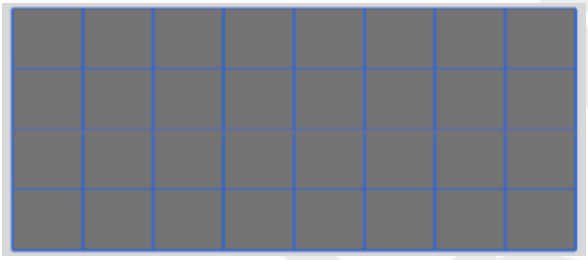


If you specified an Adaptive eyebox, it displays the sample described in the .OPTEyeBox file you specified.



14.3.2.2. Target Image

The Target Image result displays the target image of your system. For the position and size of the target image, refer to the Target Image settings. For the sampling, refer to PGU settings.

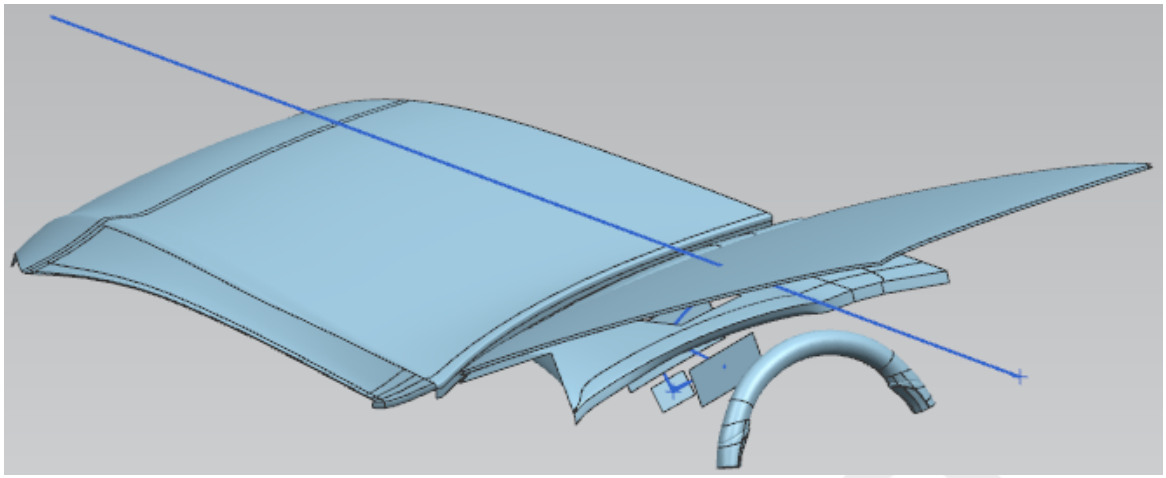


14.3.2.3. Optical Axis

The Optical Axis result displays the optical axis of your system. It starts from the center of the eyebox and reaches the center of the virtual image.

To define the center of the virtual image, the center of the target image is projected onto the PGU, fit to the pixels and reprojected on the target image space.

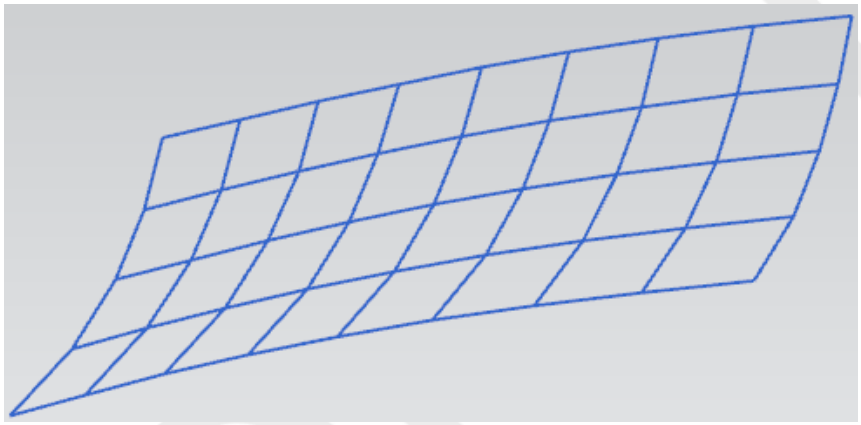
DRAFT



14.3.2.4. Best Focus Virtual Image

The Best Focus Virtual Image result displays where your system produces the best focus surface of your virtual image (also directly called virtual image).

This is where the driver will perceive the virtual image.



14.3.2.4.1. Understanding the Best Focus Point Result

In some case, HOA simulation can return no focus point even if light rays are crossing the eye pupil, which leads to an incorrect HUD system.

Description

The location of a focus point on HOA relies on a set of rays conjugating a point of the PGU and a set of support rays passing through the contour of the eye pupil (let's call him a research cone). The HOA best focus location is then defined as the location along the research cone where the section of the cone is the smallest and most circular.

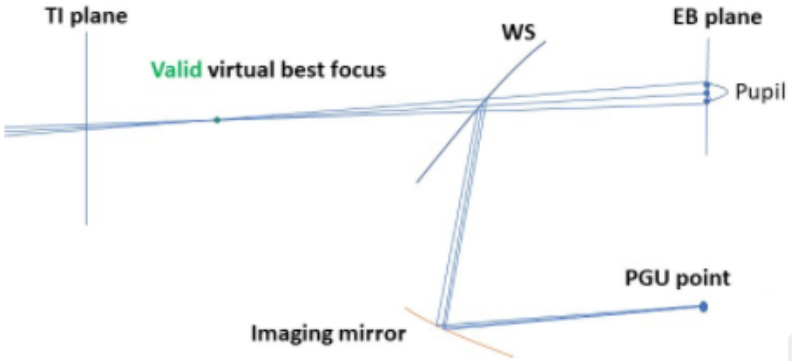
From that definition, the geometry of the research cone, which depends on the imaging mirror and windshield, can lead to the three results.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Focus Point Cases

Case 1: Virtual Focus Point

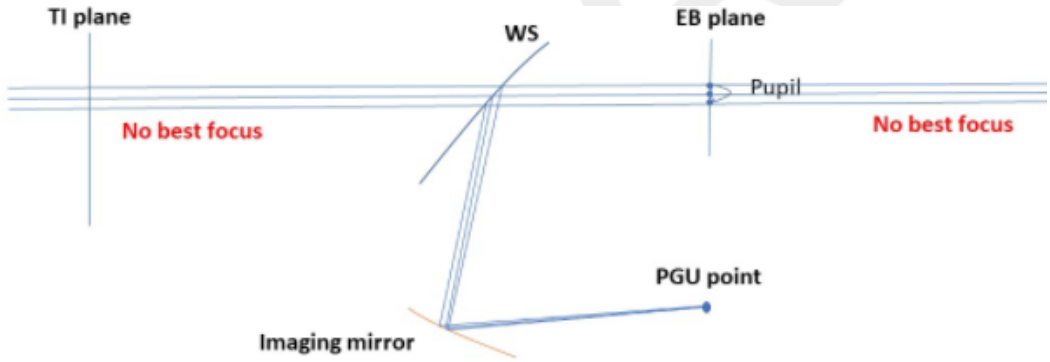
This is the expected behavior if the HUD mirrors are correctly designed.



Case 2: No Focus Point

The research cone rays are in this case parallel and no focus position can be determined (actually an infinite number of solution relatively to the HOA best focus definition).

> **Note:** This type of case may lead to the following warning message: "Image point not found".

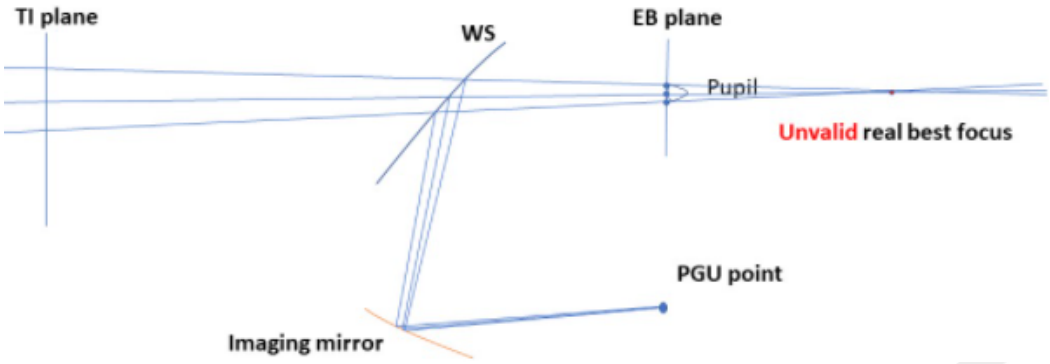


Case 3: Real Focus Point

The point is typically behind the eyebox or between the windshied and the eyebox.

In this case the focus point is determined but incorrect respectively to a consistent HUD design.

> **Note:** This type of case may lead to the following warning message: "Image point not found".



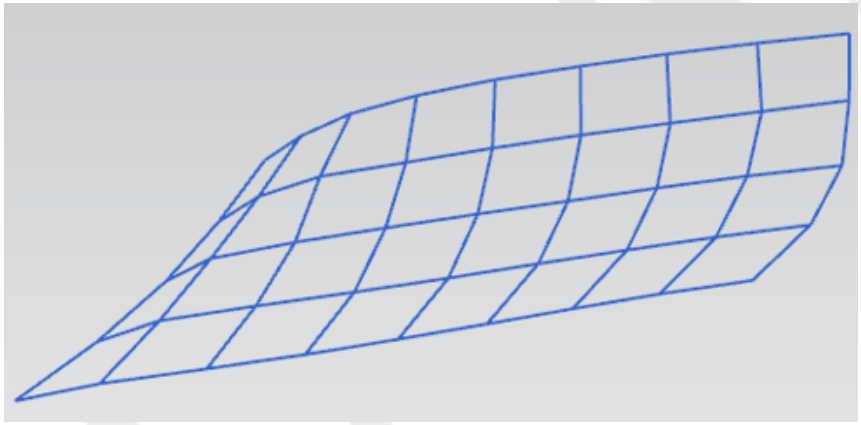
Conclusion

In case 1, the virtual point is well displayed at the end of HOA simulation, while in case 2 and case 3 no best focus is displayed since no result/inconsistent result is found with respect to the HOA best focus search criterion.

If you encounter the case 2 or 3, make a luminance analysis of your system to see what human eyes would see. Then, modify your HUD system with HOD and analyze it with HOA by iterating until you get a correct system.

14.3.2.5. Tangential Virtual Image

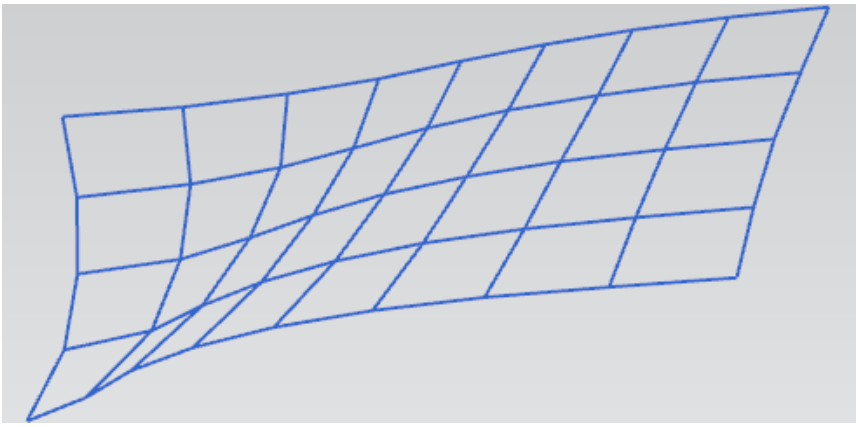
The Tangential Virtual Image result displays where your system produces the tangential surface of your virtual image.



14.3.2.6. Sagittal Virtual Image

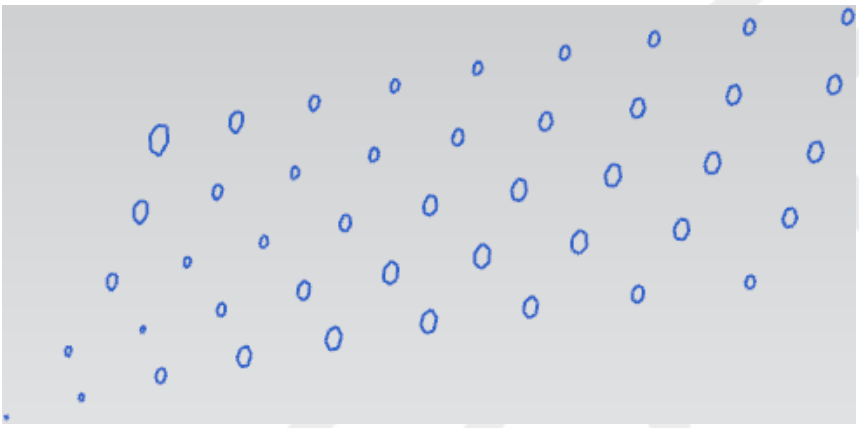
The Sagittal Virtual Image result displays where your system produces the sagittal surface of your virtual image.

DRAFT



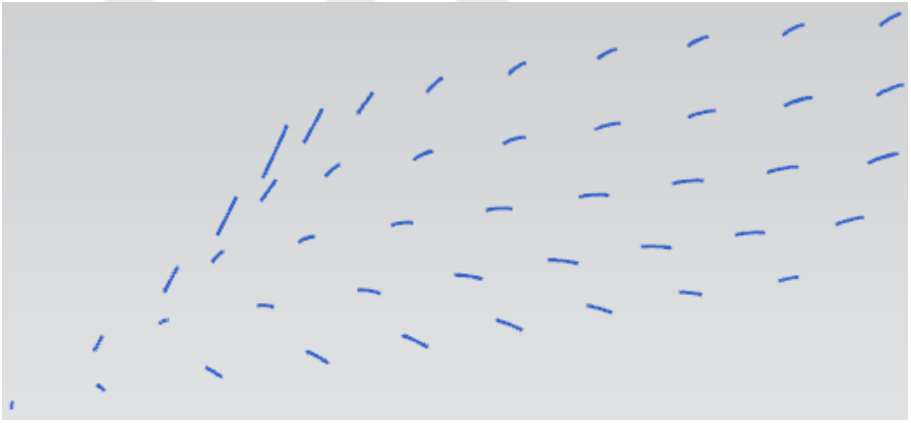
14.3.2.7. Best Focus Spot

The Best Focus Spot result displays the spot size on the best focus surface.



14.3.2.8. Tangential Spot

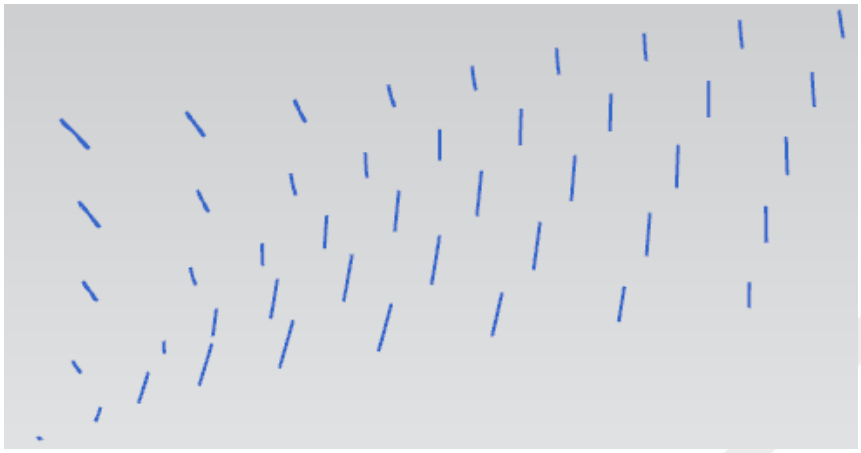
The Tangential Spot result displays the spot size on the tangential surface.



Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

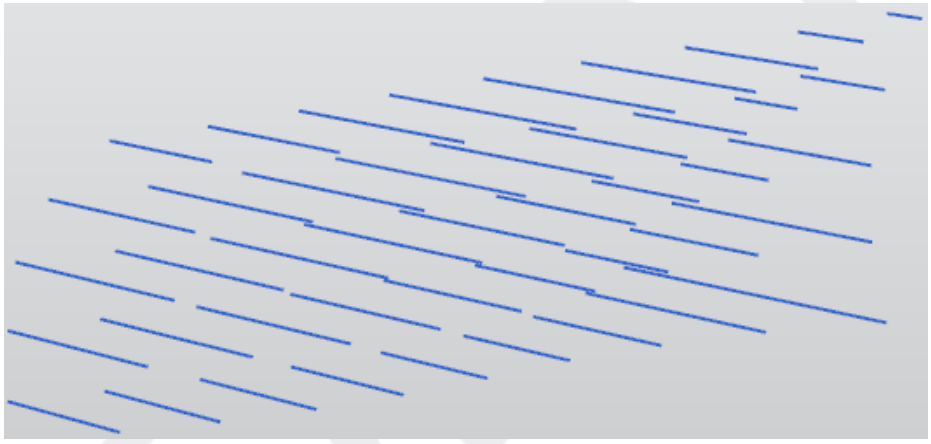
14.3.2.9. Sagittal Spot

The Sagittal Spot result displays the spot size on the sagittal surface.



14.3.2.10. Astigmatism

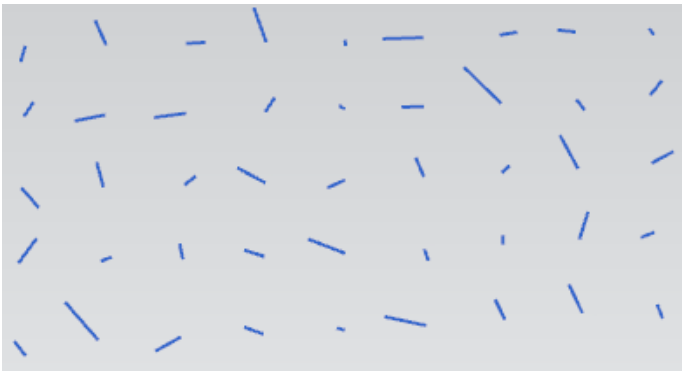
The Astigmatism result displays the difference between the tangential surface and the sagittal surface.



14.3.2.11. Static Distortion

The Static Distortion result displays the difference between the target image and the virtual image. This difference is due to the warping that must fit the pixel of the PGU and cannot exactly reach the target image.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



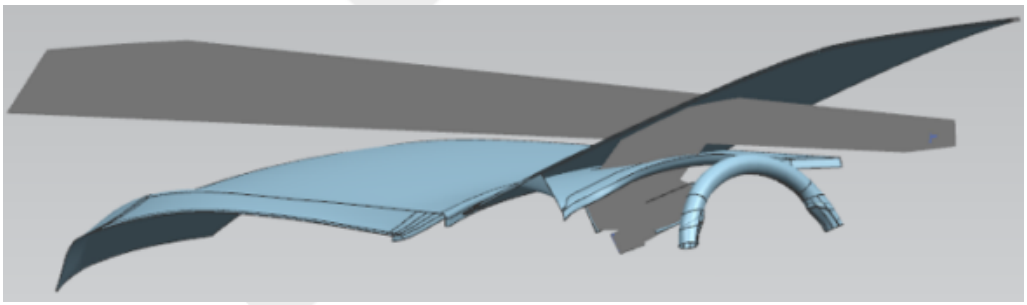
14.3.2.12. Dynamic Distortion

The Dynamic Distortion result shows how the virtual image moves when the driver moves into the eyebox.



14.3.2.13. Optical Volume

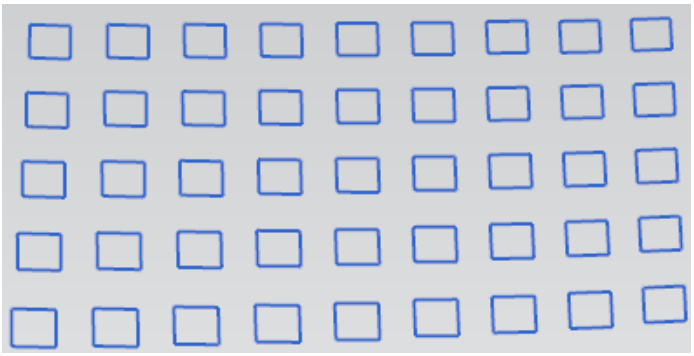
The Optical Volume result shows the volume needed by the light to produce an image without vignetting.



14.3.2.14. Pixel Image

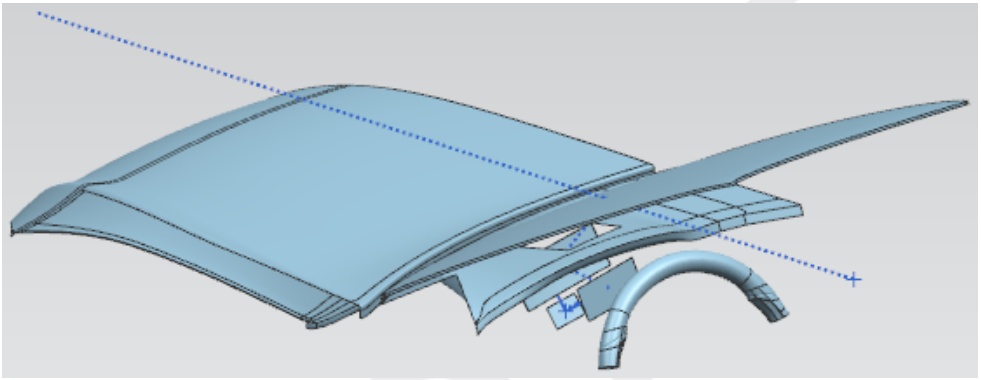
The Pixel Image result displays the size and the shape of the pixel of the PGU projected into the virtual image space.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



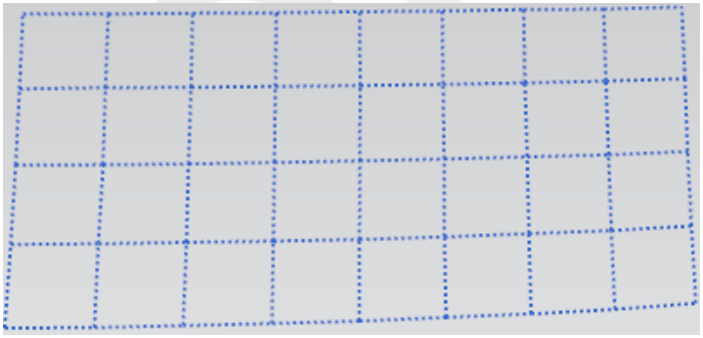
14.3.2.15. Ghost Image Optical Axis

The Ghost Image Optical Axis result displays the optical axis of your system for the ghost image. An outer surface must be specified from the Windshield settings for HOA to compute this result.



14.3.2.16. Ghost Image

The Ghost Image result displays the ghost image of your system. An outer surface must be specified from the Windshield settings for HOA to compute this result.



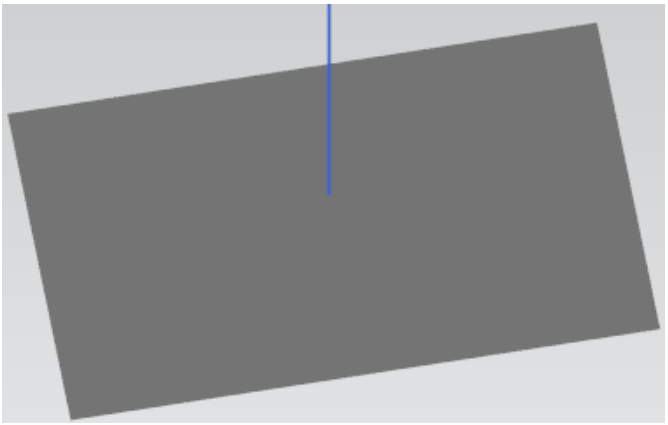
14.3.2.17. PGU

The Picture Generation Unit (PGU) of your system is also displayed.

You can hide/show the PGU using the Speos Navigator.

You can define the size, the position and the orientation of the PGU from the PGU settings.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

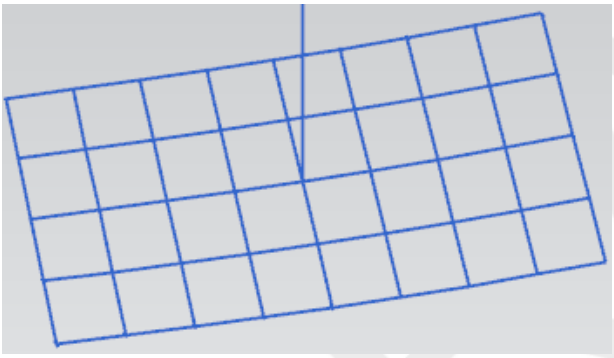


14.3.2.18. Warping

The warping is also displayed.

You can hide/show the warping using the Speos Navigator.

The warping is a grid on your PGU representing how the image must be deformed before being displayed by the PGU.



You can define the sampling of the grid that represent the warping from the PGU settings.

14.3.2.19. Visualizing a Speos360 Result

> **Note:** *Speos360 results* appear in the specification tree.

From the specification tree, double-click the **.speos360** file to open it using the **Virtual Reality Lab**.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



Speos360 result

14.3.3. HOA Tests APIs

The Application Programming Interface (API) of HUD Optical Analysis allows you to create plugins. A plugin is a .dll file that contains your own test and validity criteria.

To get a full example of the API use, download the .Plugin

To create your own plugin, include the header file HOA_Plugin_Ansys.h (stored in the Plugin example) in your project and implement each function described in this section.

To install your plugin, copy your .dll file into the *Plugins* folder of the Speos installation folder:

- ..\Optical Products\Plugins on a server installation.
- %programdata%\Ansys\v2XX\Optical Products\Plugins\ on a local installation.

14.3.3.1. Test APIs

14.3.3.1.1. GetPluginType

Description

GetPluginType specifies the type of your plugin. For a HOA Plugin, the type is "HOA".

Syntax

```
int GetPluginType(wchar_t*& owszType);
```

Parameters

Output

owszType: type of plugin (HOA for a HOA Plugin).

Return

DRAFT

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
int GetPluginType(wchar_t*& owszType)
{
     owszType = L"HOA";
     return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.1.2. GetPluginGUID

Description

GetPluginGUID returns a valid and unique "Globally Unique IDentifier" (GUID).

Each plugin must have a different GUID.

> **Note:** In Microsoft Visual Studio, you can use "Tools / Create GUID" to create a valid and unique GUID.

Syntax

```
int GetPluginGUID(wchar_t*& owszGUID);
```

Parameters

Output

owszGUID: the valid and unique GUID.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
int GetPluginGUID(wchar_t*& owszGUID)
{
     owszGUID = L"{EA9CC8A1-DD98-4AFE-8739-8448EFB51C24}";
     return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.1.3. GetPluginDescription

Description

GetPluginDescription returns information about the plugin. This description appears in the Graphical User Interface (GUI) and in the report.

Syntax

```
int GetPluginDescription(wchar_t*& owszDescription);
```

Parameters

Output

owszDescription: Description of the plugin.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
int GetPluginDescription(wchar_t*& owszDescription)
{
     owszDescription = L"OPTIS - HOA Plugin";
     return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.1.4. GetErrorDescription

Description

GetErrorDescription returns the description of an error identified by its identification number.

Syntax

```
int GetErrorDescription(const int inError, const wchar_t*& owszDescription);
```

Parameters

Input

inError: the identification number of the error returned in the description. The identification number is different from 0 and is returned when an error occurs in a function.

Output

owszDescription: the description of the error identified by the identification number *inError*.

DRAFT

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_ERROR_UNKNOWN_ERROR 1
#define OPT_PLUGIN_ERROR_NO_TEST 2
#define OPT_PLUGIN_ERROR_UNKNOWN_TEST 3
#define OPT_PLUGIN_ERROR_INVALID_DATA 4
#define OPT_PLUGIN_ERROR_UNKNOWN_SECTION 5
std::map<int, std::wstring> gmErrorDescription;
      gmErrorDescription[OPT_PLUGIN_ERROR_UNKNOWN_ERROR] = L"PLUGIN ERROR :
Unknown error";
      gmErrorDescription[OPT_PLUGIN_ERROR_NO_TEST] = L"PLUGIN ERROR : No test
 defined";
      gmErrorDescription[OPT_PLUGIN_ERROR_UNKNOWN_TEST] = L"PLUGIN ERROR :
Test index out of bounds";
      gmErrorDescription[OPT_PLUGIN_ERROR_INVALID_DATA] = L"PLUGIN ERROR :
Empty or null data";
      gmErrorDescription[OPT_PLUGIN_ERROR_UNKNOWN_SECTION] = L"PLUGIN ERROR :
 Section index out of bounds";
int GetErrorDescription(const int inError, const wchar_t*& owszDescription)
{
     if (inError >= 0 && inError < (int)gmErrorDescription.size())
     {
owszDescription = gmErrorDescription[inError].c_str();
     }
     else
     {
owszDescription = gmErrorDescription[OPT_PLUGIN_ERROR_UNKNOWN_ERROR].c_str();
return OPT_PLUGIN_ERROR_UNKNOWN_ERROR;
     }
     return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.1.5. GetTestNumber

Description

GetTestNumberreturns the number of test the plugin contains.

Syntax

```
int GetTestNumber(unsigned int& ounTestNb);
```

Parameters

Output

ounTestNb: number of test the plugin contains.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
int GetTestNumber(unsigned int& ounTestNb)
{
     ounTestNb = 4;
     return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.1.6. GetTestDescription

Description

GetTestDescription returns the description or the name of the test. This description appears in the Graphical User Interface (GUI) and in the report.

Syntax

```
int GetTestDescription(const unsigned int iunTestIndex, wchar_t*&
owszDescription);
```

Parameters

Input

iunTestIndex: the identification number of the test returned the description.

Output

owszDescription: the description of the test identified by the number "iunTestIndex".

> **Note:** To write the *owszDescription* in Japanese on an English operating system, convert the Japanese string in unicode characters. For example: **gvTests[0].Description = L"□□□**"; becomes **gvTests[0].Description = L"\u65E5\u672C\u8A9E";**

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_ERROR_UNKNOWN_TEST 3
std::map<int, std::wstring> gmTestDescription;
     gmTestDescription[0] = L"Virtual Image Distance";
     gmTestDescription[1] = L"Look Down Angle";
```

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

```
     gmTestDescritpion[2] = L"Look Over Angle";
int GetTestDescription(const unsigned int iunTestIndex, wchar_t*&
owszDescription)
{
     if (iunTestIndex < gmTestDescription.size())
     {
owszDescription = const_cast<wchar_t*>(gmTestDescription[iunTestIndex].c_str());

     }
     else
     {
return OPT_PLUGIN_ERROR_UNKNOWN_TEST;
     }
     return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.1.7. GetNeededData

Description

GetNeededData asks the plugin which data each test needs. The plugin answers "true" for yes or "false" for no if the test number "iunTestIndex" needs or not the data "iwsParameterName".

HOA automatically sends the result of the computation to the plugin, but some specific data take more time to be computed. By default, this kind of data are not computed. If you need it, you have to specify it using the "GetNeededData" function.

Syntax

```
int GetNeededData(const wchar_t* iwszParameterName, const unsigned int
iunTestIndex, bool& obNeeded);
```

Parameters

Input

iwszParameterName: Data name HOA ask the plugin. See the list of possible values bellow.

iunTestIndex: Number of the test HOA asks.

Output

obNeeded:

- True if your test number "iunTestIndex" needs the data "iwszParameterName".
- False if not.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_ERROR_UNKNOWN_TEST 3
std::map<int, std::wstring> gmTestDescription;
     gmTestDescription[0] = L"Virtual Image Distance";
     gmTestDescription[1] = L"Look Down Angle";
     gmTestDescription[2] = L"Look Over Angle";
     gmTestDescritpion[3] = L"My own test";
std::map<int, std::vector<std::wstring> > gmNeededData;
     gmNeededData[3].pushback(L"KeyWord1");
     gmNeededData[3].pushback(L"KeyWord2");
int GetNeededData(const wchar_t* iwszParameterName, const unsigned int
iunTestIndex, bool& obNeeded)
{
     if (iunTestIndex < gmTestDescription.size())
     {
obNeeded = std::find(gmNeededData[iunTestIndex].begin(),
gmNeededData[iunTestIndex].end(), std::wstring(iwszParameterName)) !=
gmNeededData[iunTestIndex].end();
     }
     else
     {
return OPT_PLUGIN_ERROR_UNKNOWN_TEST;
     }
     return OPT_PLUGIN_NO_ERROR;
}
```

List of the possible value of "iwszParameterName"

Parameter Name	Description
(Empty)	This function is provisional

14.3.3.1.8. SetDataDouble

Description

SetDataDouble gives the data to the plugin.

Syntax

```
int SetDataDouble(const wchar_t* iwszParameterName, const unsigned int
iunTableSize, const double* ipTable);
```

Parameters

Input

- *iwszParameterName*: Data name the "ipTable" contains. See the list of possibles values below.
- *iunTableSize*: Size of the table"ipTable".
- *ipTable*: A table that contains the data.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_ERROR_INVALID_DATA 4
struct DataDouble
{
     unsigned int Size;
     double* Table;
};
std::map<std::wstring, DataDouble> gmDatas;
int SetDataDouble(const wchar_t* iwszParameterName, const unsigned int
iunTableSize, const double* ipTable)
{
     if (iunTableSize != 0 && ipTable != NULL)
     {
DataDouble theData;
theData.Size = iunTableSize;
theData.Table = ipTable;
std::wstring parameterName(iwszParameterName);
gmDatas[parameterName] = theData;
     }
     else
     {
return OPT_PLUGIN_ERROR_INVALID_DATA;
     }
     return OPT_PLUGIN_NO_ERROR;
}
```

List of the possible values of "iwszParameterName"

Only the parameters that HOA can compute are sent to the plugin.

For example in "Uniform" eyebox mode, the parameters relative to the adaptive eyebox are not sent. If you do not specify an "Outer Surface" in the "Windshield/Combiner" tab, the parameters relative to the ghost are not sent.

Legend:

- iEB: represents the coordinate of the point on the horizontal axis of the Eyebox.
- jEB: represents the coordinate of the point on the vertical axis of the Eyebox.
- kMEB: indicates the Eyebox to consider in case of a multieyebox mode.
- iPGU: represents the coordinate of the point on the horizontal axis of the Virtual Image.
- jPGU: represents the coordinate of the point on the vertical axis of the Virtual Image.
- iTI: represents the coordinate of the point on the horizontal axis of the Target Image.
- jTI: represents the coordinate of the point on the vertical axis of the Target Image.
- iW: represents the coordinate of the point on the horizontal axis of the Warping grid.
- jW: represents the coordinate of the point on the vertical axis of the Warping grid.
- LREye: indicates the Eyebox (Left or Right Eyebox) to consider in case of a monocular eyebox.
- kCFG: indicates the configuration to consider in case of a multiconfiguration analysis.

In case of "Validation" parameters:

- 0 means the parameter is valid
- 1 means the parameter is extrapolated (not valid)

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

- 2 means the parameter is not valid.

DRAFT

	Parameter name	Unit	Description	Organization
Eyebox	EyeBoxCenterX	mm	User defined X coordinate of the eyebox origin point.	kCFG
	EyeBoxCenterY	mm	User defined Y coordinate of the eyebox origin point.	kCFG
	EyeBoxCenterZ	mm	User defined Z coordinate of the eyebox origin point.	kCFG
	EyeBoxInterpupillaryDistance	mm	"Interpupillary Distance" entered in the "Eyebox" tab.	kCFG
	EyeBoxNumberOfSharedSamples		"Number of Shared Sample" displayed in the "Eyebox" tab.	kCFG
	EyeBoxBinocularHorizontalSize	mm	"Binocular Horizontal Size" entered in the "Eyebox" tab.	kCFG
	EyeBoxMonocularHorizontalSize	mm	"Monocular Horizontal Size" displayed in the "Eyebox" tab.	kCFG
	EyeBoxVerticalSize	mm	"Vertical Size" entered in the "Eyebox" tab.	kCFG
	EyeBoxMonocularHorizontalSampling		"Monocular Horizontal Sampling" entered in the "Eyebox" tab.	kCFG
	EyeBoxBinocularHorizontalSampling		"Binocular Horizontal Sampling" displayed in the "Eyebox" tab.	kCFG
	EyeBoxVerticalSampling		"Vertical Sampling" entered in the "Eyebox" tab.	kCFG
	EyeBoxHorizontalSpacing	mm	Horizontal spacing between two samples of the eyebox.	kCFG
	EyeBoxVerticalSpacing	mm	Horizontal spacing between two samples of the eyebox.	kCFG
	EyeBoxBinocular2DX	mm	Sample along X direction of the eyebox for both eyes in the 2D space of the current EB.	iEB, jEB, kMEB, kCFG
	EyeBoxBinocular2DY	mm	Sample along Y direction of the eyebox for both eyes in the 2D space of the current EB.	iEB, jEB, kMEB, kCFG
	EyeBoxBinocular3DX	mm	Sample along X direction of the eyebox for both eyes in the 3D global space.	iEB, jEB, kMEB, kCFG
	EyeBoxBinocular3DY	mm	Sample along Y direction of the eyebox for both eyes in the 3D global space.	iEB, jEB, kMEB, kCFG
	EyeBoxBinocular3DZ	mm	Sample along Z direction of the eyebox for both eyes in the 3D global space.	iEB, jEB, kMEB, kCFG
	EyeBoxMonocular2DX	mm		iEB, jEB, kMEB, LREye, kCFG

DRAFT

	Parameter name	Unit	Description	Organization
			Sample along X direction of the eyebox for one eye in the 2D space of the current EB.	
	EyeBoxMonocular2DY	mm	Sample along Y direction of the eyebox for one eye in the 2D space of the current EB.	iEB, jEB, kMEB, LREye
	EyeBoxMonocular3DX	mm	Sample along X direction of the eyebox for one eye in the 3D global space.	iEB, jEB, kMEB, LREye, kCFG
	EyeBoxMonocular3DY	mm	Sample along Y direction of the eyebox for one eye in the 3D global space.	iEB, jEB, kMEB, LREye, kCFG
	EyeBoxMonocular3DZ	mm	Sample along Z direction of the eyebox for one eye in the 3D global space.	iEB, jEB, kMEB, LREye, kCFG
	EyeBoxNumberOfConfigurations		Number of configurations entered in the "Eyebox" tab.	kCFG
	EyeBoxOffsetOfConfigurations	mm	Offset of configurations entered in the "Eyebox" tab.	kMEB, kCFG
	EyeBoxSamplingMode		"Sampling Mode" entered in the "Eyebox" tab.	kCFG 0 = Uniform, 1 = Adaptive
	EyeBoxAdaptiveSampling		Number of samples the adaptive eyebox contains.	kCFG
	EyeBoxAdaptive2DX	mm	Sample along the X direction of the adaptive eyebox in the 2D space of the current EB.	iEB, kMEB, kCFG
	EyeBoxAdaptive2DY	mm	Sample along the Y direction of the adaptive eyebox in the 2D space of the current EB.	iEB, kMEB, kCFG
	EyeBoxAdaptive3DX	mm	Sample along the X direction of the adaptive eyebox in the 3D global space.	iEB, kMEB, kCFG
	EyeBoxAdaptive3DY	mm	Sample along the Y direction of the adaptive eyebox in the 3D global space.	iEB, kMEB, kCFG
	EyeBoxAdaptive3DZ	mm	Sample along the Z direction of the adaptive eyebox in the 3D global space.	iEB, kMEB, kCFG

DRAFT

	Parameter name	Unit	Description	Organization
Target Image	TargetLookOverAngle	radians	User defined look over angle of the target.	kCFG
	TargetLookDownAngle	radians	User defined look down angle of the target.	kCFG
	TargetHorizontalFieldOfView	radians	User defined horizontal field of view of the target.	kCFG
	TargetVerticalFieldOfView	radians	User defined vertical field of view of the target.	kCFG
	TargetImage2DX	mm	Sample along the X direction of the target image in the 2D space of the target image.	iTI, jTI, kMEB, kCFG
	TargetImage2DY	mm	Sample along the Y direction of the target image in the 2D space of the target image.	iTI, jTI, kMEB, kCFG
	TargetImage3DX	mm	Sample along the X direction of the target image in the 3D global space.	iTI, jTI, kMEB, kCFG
	TargetImage3DY	mm	Sample along the Y direction of the target image in the 3D global space.	iTI, jTI, kMEB, kCFG
	TargetImage3DZ	mm	Sample along the Z direction of the target image in the 3D global space.	iTI, jTI, kMEB, kCFG

DRAFT

	Parameter name	Unit	Description	Organization
Virtual Image	VirtualImageDistance	mm	Distance between the eyebox center and the target image center.	kCFG
	VirtualImageMonocular3DX	mm	Sample along the X direction of the virtual image for one eye in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	VirtualImageMonocular3DY	mm	Sample along the Y direction of the virtual image for one eye in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	VirtualImageMonocular3DZ	mm	Sample along the Z direction of the virtual image for one eye in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	VirtualImageValidationMonocular		Sample validation of the virtual image for one eye.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	VirtualImageBinocular3DX	mm	Sample along the X direction of the virtual image for both eyes in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	VirtualImageBinocular3DY	mm	Sample along the Y direction of the virtual image for both eyes in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	VirtualImageValidationBinocular		Sample validation of the virtual image for both eyes.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	VirtualImageBinocular3DZ	mm	Sample along the Z direction of the virtual image for both eyes in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	VirtualImageAdaptive3DX	mm	Sample along the X direction of the virtual image for the adaptive eyebox in the 3D global space.	iEB, iPGU, jPGU, kMEB, kCFG
	VirtualImageAdaptive3DY	mm	Sample along the Y direction of the virtual image for the adaptive eyebox in the 3D global space.	iEB, iPGU, jPGU, kMEB, kCFG
	VirtualImageAdaptive3DZ	mm	Sample along the Z direction of the virtual image for the adaptive eyebox in the 3D global space.	iEB, iPGU, jPGU, kMEB, kCFG
	VirtualImageValidationAdaptive		Sample validation of the virtual image for the adaptive eyebox.	iEB, iPGU, jPGU, kMEB, kCFG

DRAFT

	Parameter name	Unit	Description	Organization
Sagittal Image	SagittalImageMonocular3DX	mm	Sample along the X direction of the sagittal image for one eye in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	SagittalImageMonocular3DY	mm	Sample along the Y direction of the sagittal image for one eye in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	SagittalImageMonocular3DZ	mm	Sample along the Z direction of the sagittal image for one eye in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	SagittalImageBinocular3DX	mm	Sample along the X direction of the sagittal image for both eyes in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	SagittalImageBinocular3DY	mm	Sample along the Y direction of the sagittal image for both eyes in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	SagittalImageBinocular3DZ	mm	Sample along the Z direction of the sagittal image for both eyes in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	SagittalImageAdaptive3DX	mm	Sample along the X direction of the sagittal image for the adaptive eyebox in the 3D global space.	iEB, iPGU, jPGU, kMEB, kCFG
	SagittalImageAdaptive3DY	mm	Sample along the Y direction of the sagittal image for the adaptive eyebox in the 3D global space.	iEB, iPGU, jPGU, kMEB, kCFG
	SagittalImageAdaptive3DZ	mm	Sample along the Z direction of the sagittal image for the adaptive eyebox in the 3D global space.	iEB, iPGU, jPGU, kMEB, kCFG
	SagittalImageValidationMonocular		Sample validation of the sagittal image for the monocular eyebox.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	SagittalImageValidationBinocular		Sample validation of the sagittal image for the binocular eyebox.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	SagittalImageValidationAdaptive		Sample validation of the sagittal image for the adaptive eyebox.	iEB, iPGU, jPGU, kMEB, kCFG

DRAFT

	Parameter name	Unit	Description	Organization
Tangential Image	TangentialImageMonocular3DX	mm	Sample along the X direction of the tangential image for one eye in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	TangentialImageMonocular3DY	mm	Sample along the Y direction of the tangential image for one eye in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	TangentialImageMonocular3DZ	mm	Sample along the Z direction of the tangential image for one eye in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	TangentialImageBinocular3DX	mm	Sample along the X direction of the tangential image for both eyes in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	TangentialImageBinocular3DY	mm	Sample along the Y direction of the tangential image for both eyes in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	TangentialImageBinocular3DZ	mm	Sample along the Z direction of the tangential image for both eyes in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	TangentialImageAdaptive3DX	mm	Sample along the X direction of the tangential image for the adaptive eyebox in the 3D global space.	iEB, iPGU, jPGU, kMEB, kCFG
	TangentialImageAdaptive3DY	mm	Sample along the Y direction of the tangential image for the adaptive eyebox in the 3D global space.	iEB, iPGU, jPGU, kMEB, kCFG
	TangentialImageAdaptive3DZ	mm	Sample along the Z direction of the tangential image for the adaptive eyebox in the 3D global space.	iEB, iPGU, jPGU, kMEB, kCFG
	TangentialImageValidationMonocular		Sample validation of the tangential image for the monocular eyebox.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	TangentialImageValidationBinocular		Sample validation of the sagittal image for the monocular eyebox.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	TangentialImageValidationAdaptive		Sample validation of the sagittal image for the monocular eyebox.	iEB, iPGU, jPGU, kMEB, kCFG

DRAFT

	Parameter name	Unit	Description	Organization
Ghost	GhostMonocular3DX	mm	Sample along the X direction of the ghost image for one eye in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	GhostMonocular3DY	mm	Sample along the Y direction of the ghost image for one eye in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	GhostMonocular3DZ	mm	Sample along the Z direction of the ghost image for one eye in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	GhostValidationMonocular		Sample validation of the ghost image for one eye.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	GhostBinocular3DX	mm	Sample along the X direction of the ghost image for both eyes in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	GhostBinocular3DY	mm	Sample along the Y direction of the ghost image for both eyes in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	GhostBinocular3DZ	mm	Sample along the Z direction of the ghost image for both eyes in the 3D global space.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	GhostValidationBinocular		Sample validation of the ghost image for both eyes.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	GhostAdaptive3DX	mm	Sample along the X direction of the ghost image for the adaptive eyebox in the 3D global space.	iEB, iPGU, jPGU, kMEB, kCFG
	GhostAdaptive3DY	mm	Sample along the Y direction of the ghost image for the adaptive eyebox in the 3D global space.	iEB, iPGU, jPGU, kMEB, kCFG
	GhostAdaptive3DZ	mm	Sample along the Z direction of the ghost image for the adaptive eyebox in the 3D global space.	iEB, iPGU, jPGU, kMEB, kCFG
	GhostValidationAdaptive		Sample validation of the ghost image for the adaptive eyebox.	iEB, iPGU, jPGU, kMEB, kCFG

DRAFT

	Parameter name	Unit	Description	Organization
PGU	PGUHorizontalSize	mm	"Horizontal Size" displayed or entered in the "PGU" tab.	kCFG
	PGUVerticalSize	mm	"Vertical Size" displayed or entered in the "PGU" tab.	kCFG
	PGUHorizontalResolution		"Horizontal Resolution" displayed or entered in the "PGU" tab.	kCFG
	PGUVerticalResolution		"Vertical Resolution" displayed or entered in the "PGU" tab.	kCFG
	PGUHorizontalSampling		"Horizontal Sampling" entered in the "PGU" tab.	kCFG
	PGUVerticalSampling		"Vertical Sampling" entered in the "PGU" tab.	kCFG
	PGU2DX	mm	Warping sampling coordinates, along the X direction in the PGU 2D space, used in simulation.	iPGU, jPGU, kCFG
	PGU2DY	mm	Warping sampling coordinates, along the Y direction in the PGU 2D space, used in simulation.	iPGU, jPGU, kCFG
	PGU3DX	mm	Warping sampling coordinates, along the X direction in the 3D global space, used in simulation.	iPGU, jPGU, kCFG
	PGU3DY	mm	Warping sampling coordinates, along the Y direction in the 3D global space, used in simulation.	iPGU, jPGU, kCFG
	PGU3DZ	mm	Warping sampling coordinates, along the Z direction in the 3D global space, used in simulation.	iPGU, jPGU, kCFG
Mirror	MirrorTiltAngle	rad	Tilt angle of the configuration entered in the "Mirrors" tab.	kMEB, kCFG

DRAFT

	Parameter name	Unit	Description	Organization
Warping	WarpingMode		"Mode" entered in the "Warping" tab.	kCFG 1 = Disable 2 = Import 3 = Build & Export 4 = Build
	WarpingHorizontalSampling		"Horizontal Sampling" entered in the "Warping" tab.	kCFG
	WarpingVerticalSampling		"Vertical Sampling" entered in the "Warping" tab.	kCFG
	WarpingHorizontalPixel	mm	Horizontal size of a pixel on the PGU.	kCFG
	WarpingVerticalPixel	mm	Vertical size of a pixel on the PGU.	kCFG
	Warping2DX	mm	Warping sampling coordinates, along the X direction in the PGU 2D space, used to generate the .OPTWarping file (fit to pixels).	iW, jW, kMEB, kCFG
	Warping2DY	mm	Warping sampling coordinates, along the Y direction in the PGU 2D space, used to generate the .OPTWarping file (fit to pixels).	iW, jW, kMEB, kCFG
	Warping3DX	mm	Warping sampling coordinates, along the X direction in the 3D global space, used to generate the .OPTWarping file (fit to pixels).	iW, jW, kMEB, kCFG
	Warping3DY	mm	Warping sampling coordinates, along the Y direction in the 3D global space, used to generate the .OPTWarping file (fit to pixels).	iW, jW, kMEB, kCFG
	Warping3DZ	mm	Warping sampling coordinates, along the Z direction in the 3D global space, used to generate the .OPTWarping file (fit to pixels).	iW, jW, kMEB, kCFG
	RawWarping2DX	mm	Warping sampling coordinates along the X direction in the PGU 2D space before the fitting of samples to the nearest pixels.	iW, jW, kMEB, kCFG
	RawWarping2DY	mm	Warping sampling coordinates along the Y direction in the PGU 2D space before the fitting of samples to the nearest pixels.	iW, jW, kMEB, kCFG
	RawWarping3DX	mm	Warping sampling coordinates along the X direction in the 3D global space before the fitting of samples to the nearest pixels.	iW, jW, kMEB, kCFG
	RawWarping3DY	mm		iW, jW, kMEB, kCFG

DRAFT

	Parameter name	Unit	Description	Organization
			Warping sampling coordinates along the Y direction in the 3D global space before the fitting of samples to the nearest pixels.	
	RawWarping3DZ	mm	Warping sampling coordinates along the Z direction in the 3D global space before the fitting of samples to the nearest pixels.	iW, jW, kMEB, kCFG
	WarpingPixelX	Pixel	Sample along the X direction of the warping in the 2D space of the PGU (fit to pixels).	iW, jW, kMEB, kCFG
	WarpingPixelY	Pixel	Sample along the Y direction of the warping in the 2D space of the PGU (fit to pixels).	iW, jW, kMEB, kCFG
	WarpingPixelByMirrorX	Pixel	Sample along the X direction of the warping in the 2D space of the mirror (fit to pixels).	iW, jW, kMEB, kCFG
	WarpingPixelByMirrorY	Pixel	Sample along the Y direction of the warping in the 2D space of the mirror (fit to pixels).	iW, jW, kMEB, kCFG
Reference	TopDirection	mm	X ,Y,Z of the "Top Direction" entered in the "Axis System" tab.	i, kCFG
	VehicleDirection	mm	X ,Y,Z of the "Vehicle Direction" entered in the "Axis System" tab.	i, kCFG

DRAFT

	Parameter name	Unit	Description	Organization
Pixel Size	PixelSizeMonocular3DX		Size of the pixel of the virtual image for one eye.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG, n n=0,1,2,3 for the corners (according to the PGU axes): 0: up left 1: up right 2: bottom right 3: bottm left
	PixelSizeMonocular3DY		Size of the pixel of the virtual image for one eye.	iEB, jEB, iPGU, jPGU, kMEB, LREye, n, kCFG
	PixelSizeMonocular3DZ		Size of the pixel of the virtual image for one eye.	iEB, jEB, iPGU, jPGU, kMEB, LREye, n, kCFG
	PixelSizeBinocular3DX		Size of the pixel of the virtual image for both eyes.	iEB, jEB, iPGU, jPGU, kMEB, n, kCFG
	PixelSizeBinocular3DY		Size of the pixel of the virtual image for both eyes.	iEB, jEB, iPGU, jPGU, kMEB, n, kCFG
	PixelSizeBinocular3DZ		Size of the pixel of the virtual image for both eyes.	iEB, jEB, iPGU, jPGU, kMEB, n, kCFG
	PixelSizeAdaptive3DX		Size of the pixel of the virtual image for the adaptive eyebox.	iEB, iPGU, jPGU, kMEB, n, kCFG
	PixelSizeAdaptive3DY		Size of the pixel of the virtual image for the adaptive eyebox.	iEB, iPGU, jPGU, kMEB, n, kCFG
	PixelSizeAdaptive3DZ		Size of the pixel of the virtual image for the adaptive eyebox.	iEB, iPGU, jPGU, kMEB, n, kCFG
Sharpness	SpotDiameterMonocular	mm	Diameter of the best focus spot for one eye.	iEB, jEB, iPGU, jPGU, kMEB, LREye, kCFG
	SpotDiameterBinocular	mm	Diameter of the best focus spot for both eyes.	iEB, jEB, iPGU, jPGU, kMEB, kCFG
	SpotDiameterAdaptive	mm	Diameter of the best focus spot for the adaptive eyebox.	iEB, iPGU, jPGU, kMEB, kCFG
Configuration	ConfigurationValidation		Validation of each configuration of the analysis.	kCFG
	NumberOfConfigurations		Number of configurations of the analysis.	kCFG

DRAFT

14.3.3.1.9. SetDataString

Description

SetDataString gives the data to the plugin.

Syntax

```
int SetDataString(const wchar_t* iwszParameterName, const unsigned int
iunTableSize, const wchar_t** ipTable);
```

Parameters

Input

- *iwszParameterName*: data name the "ipTable" contains. See the list of possible values below.
- *iunTableSize*: Size of the table "ipTable".
- *ipTable*: A table that contains the data.

Return

(*int*) : returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_ERROR_INVALID_DATA  4
struct DataString
{
     unsigned int Size;
     wchar_t** Table;
};
std::map<std::wstring, DataString> gmDatas;
int SetDataString(const wchar_t* iwszParameterName, const unsigned int
iunTableSize, const wchar_t** ipTable)
{
     if (iunTableSize != 0 && ipTable != NULL)
     {
DataString theData;
theData.Size = iunTableSize;
theData.Table = ipTable;
std::wstring parameterName(iwszParameterName);
gmDatas[parameterName] = theData;
     }
     else
     {
return OPT_PLUGIN_ERROR_INVALID_DATA;
     }
     return OPT_PLUGIN_NO_ERROR;
}
```

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

List of the possible value of "iwszParameterName"

	Parameter name	Unit	Description	Organization
Eyebox	EyeBoxNameOfConfigurations		Name of configurations entered in the "Eyebox" tab	kCFG
Other	OutputFolder		Output folder path.	kCFG
Configuration	NameOfConfigurations		Name of the configurations.	kCFG

14.3.3.1.10. RunTest

Description

RunTest is where you compute your test.

Syntax

```
int RunTest(const unsigned int iunTestIndex, int (*pProgress)(const double&));
```

Parameters

Input

iunTestIndex: the identification number of the test to run.

pProgress: you can specify the progression of your test using this function to send a double between 0 and 1 (0 for 0% and 1 for 100%).

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
int RunTest(const unsigned int iunTestIndex, int (*pProgress)(const double&))
{
     switch (iunTestIndex)
     {
case 0:
    pProgress(0);
    [...]
    pProgress(1);
break;
case 1:
    [...]
break;
case 2:
    [...]
break;
     }
```

```
     return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.1.11. GetReportSectionNumber

Description

GetReportSectionNumber returns the number of section in the report a test needs.

Syntax

```
int GetReportSectionNumber(const unsigned int iunTestIndex, unsigned int&
ounReportSectionNb);
```

Parameters

Input

iunTestIndex: the identification number of the test for which HOA asks the number of section needed.

Output

ounReportSectionNb: the number of section in the report the test "iunTestIndex" needs.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
int GetReportSectionNumber(const unsigned int iunTestIndex, unsigned int&
ounReportSectionNb)
{
     switch (iunTestIndex)
     {
case 0:
    ounReportSectionNb = 1;
break;
case 1:
    ounReportSectionNb = 3;
break;
case 2:
    ounReportSectionNb = 1;
break;
     }
     return OPT_PLUGIN_NO_ERROR;
}
```

DRAFT

14.3.3.1.12. GetReportSection

Description

GetReportSection returns the information to create a section in the report. A section is composed of a title and a table.

Syntax

```
int GetReportSection(const unsigned int iunTestIndex, const unsigned int
iunSectionIndex, wchar_t*& owszSectionTitle, unsigned int& ounSectionLineNb,
unsigned int& ounSectionRowNb, wchar_t**& owszDescription);
```

Parameters

Input

- *iunTestIndex*: the identification number of the test for which HOA asks you information about the section.
- *iunSectionIndex*: the index of the section for which HOA asks information.

Output

- *owszSectionTitle*: the title of the section.
- *ounSectionLineNb*: the number of line the table must have in this section.
- *ounSectionRowNb*: the number of row the table must have in this section.
- *owszDescritpion*: the content of the table.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
int GetReportSection(const unsigned int iunTestIndex, const unsigned int
iunSectionIndex, wchar_t*& owszSectionTitle, unsigned int& ounSectionLineNb,
unsigned int& ounSectionRowNb, wchar_t**& owszDescription)
{
     switch (iunTestIndex)
     {
case 0:
    switch (iunSectionIndex)
    {
        case 0:
            owszSectionTitle = L"The title of my first section";
            ounSectionLineNb = 2;
            ounSectionRowNb = 1;
            wchar_t[100] V1 = L"First value";
            wchar_t[100] V2 = L"Second value";
            wchar_t** description = new wchar_t*[2];
            description[0] = V1;
            description[1] = V2;
            owszDescritpion = description;
        break;
```

```
        case 1:
            [...]
        break;
    }
break;
case 1:
    [...]
break;
case 2:
    [...]
break;
     }
     return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.1.13. GetDataDouble

Description

GetDataDouble returns the parameters and their values and displays them in the tree.

Syntax

```
GetDataDouble(const unsigned int iunIndex, COptString& owszParameterName,
COptString& owszParameterType, double& odValue) const;
```

Parameters

Input

iunIndex: index of the parameter returned unique for each one.

owszParameterName: name of the parameter to be displayed in the tree.

owszParameterType: type of the parameter to be displayed in the tree:

- Length (meter)
- Plane angle (radian)

odValue: value of the parameter to be displayed in the tree.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_ERROR_UNKNOWN_TEST 3
std::vector
int GetDataDouble(const unsigned int iunIndex, wchar_t*& iwszParameterName,
wchar_t*& iwszParameterType, double& odValue)
{
   // Check if the test index is not outside of the test vector
   if (iunIndex < gvDisplayData.size())
   {
```

```
      DisplayData& temp = gvDisplayData.at(iunIndex);
      iwszParameterName = const_cast<wchar_t*>(temp.Name.c_str());
      iwszParameterType = const_cast<wchar_t*>(temp.Unit.c_str());
      odValue = temp.Value;
   }
   else
   {
      return OPT_PLUGIN_ERROR_UNKNOWN_TEST;
   }
   return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.1.14. GetDataDoubleNumber

Description

GetDataDoubleNumber returns the number of parameters to be displayed in the tree.

Syntax

```
GetDataDoubleNumber(unsigned int& ounDataDoubleNb);
```

Parameters

Output

ounDataDoubleNb: number of parameters returned.

Example

To get a full example of the API use, download the .Plugin

```
#define OPT_PLUGIN_NO_ERROR 0
std::vector
OPT_HOA_API int GetDataDoubleNumber(unsigned int& ounDataDoubleNb)
{
   // Check if input and output vector are not empty
   ounDataDoubleNb = (unsigned int)gvDisplayData.size();
   return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.2. Image Warping APIs

14.3.3.2.1. GetPluginType

Description

GetPluginType specifies the type of plugin. For a HIW Plugin, the type is "HIW".

Syntax

```
int GetPluginType(wchar_t*& owszType);
```

Parameters

Output

owszType: type of plugin (HIW for a HIW Plugin)

Return

(*int*): returns the identification number of the error if an error occurs, or 0 if no error occurs.

Example

```
#define OPT_PLUGIN_NO_ERROR 0
int GetPluginType(wchar_t*& owszType)
{
     owszType = L"HIW";
     return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.2.2. GetPluginGUID

Description

GetPluginGUID returns a valid and unique "Globally Unique IDentifier" (GUID).

Each plugin must have a different GUID.

> **Note:** In Microsoft Visual Studio, you can use "Tools / Create GUID" to create a valid and unique GUID.

Syntax

```
int GetPluginGUID(wchar_t*& owszGUID);
```

Parameters

Output

owszGUID: the valid and unique GUID.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

```
#define OPT_PLUGIN_NO_ERROR 0
int GetPluginGUID(wchar_t*& owszGUID)
```

```
{
     owszGUID = L"{12E5E507-6E81-4E58-BCFA-01D283C22506}";
     return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.2.3. GetPluginDescription

Description

GetPluginDescription returns information about the plugin. This description appears in the Graphical User Interface (GUI) and in the report.

Syntax

```
int GetPluginDescription(wchar_t*& owszDescription);
```

Parameters

Output

owszDescription: Description of the plugin.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

```
#define OPT_PLUGIN_NO_ERROR 0
int GetPluginDescription(wchar_t*& owszDescription)
{
     owszDescription = L"HIW Warping Plugin";
     return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.2.4. GetErrorDescription

Description

GetErrorDescription returns the description of an error identified by its identification number.

Syntax

```
int GetErrorDescription(const int inError, const wchar_t*& owszDescription);
```

Parameters

Input

inError: the identification number of the error returned in the description. The identification number is different from 0 and is returned when an error occurs in a function.

> **Note:** Negative error number refers to an OpenCL error code. Refer to OpenCL documentation for further details.

Output

owszDescription: the description of the error identified by the identification number *inError*.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_ERROR_OPENCL_ERROR 1
#define OPT_PLUGIN_ERROR_NO_ALGO 2
#define OPT_PLUGIN_ERROR_UNKNOWN_ALGO 3
#define OPT_PLUGIN_ERROR_INVALID_DATA 4
#define OPT_PLUGIN_ERROR_UNKNOWN_SECTION 5
#define OPT_PLUGIN_ERROR_UNKNOWN_PARAMETER 6
#define OPT_PLUGIN_ERROR_KERNEL_LOADING 7
#define OPT_PLUGIN_ERROR_UNKNOWN_ENV_VAR 8
#define OPT_PLUGIN_ERROR_WARPING_FILE_LOADING 9
#define OPT_PLUGIN_ERROR_MISSING_INPUT_PARAMETERS 10
#define OPT_PLUGIN_ERROR_ROTATION_DRIVER_HEIGHT 11
#define OPT_PLUGIN_ERROR_CPU_ALLOC 12
#define OPT_PLUGIN_ERROR_PLUGIN_NOT_INITIALIZED 13
#define OPT_PLUGIN_ERROR_UNKNOWN_ERROR 14

std::map<int, std::wstring> gmErrorDescription;

gmErrorDescription[OPT_PLUGIN_ERROR_OPENCL_ERROR] = L"OpenCL error";
gmErrorDescription[OPT_PLUGIN_ERROR_NO_ALGO] = L"No algorithm defined";
gmErrorDescription[OPT_PLUGIN_ERROR_UNKNOWN_ALGO] = L"Algo index out of bounds";
gmErrorDescription[OPT_PLUGIN_ERROR_INVALID_DATA] = L"Empty or inconsistent
data";
gmErrorDescription[OPT_PLUGIN_ERROR_UNKNOWN_SECTION] = L"Section index out of
 bounds";
gmErrorDescription[OPT_PLUGIN_ERROR_UNKNOWN_PARAMETER] = L"Unknown parameter";
gmErrorDescription[OPT_PLUGIN_ERROR_KERNEL_LOADING] = L"Cannot load kernel
file";
gmErrorDescription[OPT_PLUGIN_ERROR_UNKNOWN_ENV_VAR] = L"Environment variable
 to plugin path not found";
gmErrorDescription[OPT_PLUGIN_ERROR_WARPING_FILE_LOADING] = L"Cannot load
warping file";
gmErrorDescription[OPT_PLUGIN_ERROR_MISSING_INPUT_PARAMETERS] = L"Missing one
 or several input parameters";
gmErrorDescription[OPT_PLUGIN_ERROR_ROTATION_DRIVER_HEIGHT] = L"Mirror rotation
 and driver's height values do not match warping file data";
gmErrorDescription[OPT_PLUGIN_ERROR_CPU_ALLOC] = L"Error allocating on CPU";
gmErrorDescription[OPT_PLUGIN_ERROR_PLUGIN_NOT_INITIALIZED] = L"Plugin is not
 initialized. Please call Init() before any API calls.";
gmErrorDescription[OPT_PLUGIN_ERROR_UNKNOWN_ERROR] = L"Unknown error";

OPT_HIW_API int GetErrorDescription(const int inError, const wchar_t*&
```

```
owszDescription)
{
     if (inError==0)
return OPT_PLUGIN_NO_ERROR;

// Check if error code is not out of bounds
     if (inError > 1 && inError <= (int)gmErrorDescription.size())
     {
owszDescription = gmErrorDescription[inError].c_str();
return OPT_PLUGIN_NO_ERROR;
     }
     else
     {
owszDescription = clGetErrorString(inError);
return OPT_PLUGIN_NO_ERROR;
     }
}
```

14.3.3.2.5. GetAlgoNumber

Description

GetAlgoNumber returns the number of algorithms the plugin contains.

Syntax

```
int GetAlgoNumber(unsigned int& ounAlgoNb);
```

Parameters

Output

ounAlgoNb: number of algorithms the plugin contains.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

```
#define OPT_PLUGIN_NO_ERROR 0
OPT_HIW_API int GetAlgoNumber(unsigned int& uiAlgoNb)
{
     // Check if input and output vector are not empty
     if (!gvAlgos.empty())
     {
uiTestNb = (unsigned int)gvAlgos.size();
     }
     else
     {
return OPT_PLUGIN_ERROR_NO_ALGO;
     }
     return OPT_PLUGIN_NO_ERROR;
}
```

DRAFT

14.3.3.2.6. GetAlgoDescription

Description

GetAlgoDescription returns the description or the name of the algorithm. This description appears in the Graphical User Interface (GUI) and in the report.

Syntax

```
int GetAlgoDescription(const unsigned int iunAlgoIndex, wchar_t*&
owszDescription);
```

Parameters

Input

iunAlgoIndex: the identification number of the algorithm returned in the description:

- 0: nearest algorithm, low quality warping.
- 1: bilinear algorithm, high quality warping.

Output

owszDescription: the description of the algorithm identified by the number "iunAlgoIndex".

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_ERROR_UNKNOWN_TEST 3

std::map<int, std::wstring> gmAlgoDescription;

gmTestDescription[0] = L" Bilinear ";
     gmTestDescription[1] = L" Bicubic ";
     gmTestDescritpion[2] = L" Polynomial";

OPT_HIW_API int GetAlgoDescription(const unsigned int iunAlgoIndex, wchar_t*&
 owszDescription)
{
     // Check if the algo index is not outside of the algo vector
     if (iunAlgoIndex < gvAlgos.size())
     {
owszDescription =
const_cast<wchar_t*>(gvAlgos[iunAlgoIndex].Description.c_str());
     }
     else
     {
return OPT_PLUGIN_ERROR_UNKNOWN_ALGO;
     }
     return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.2.7. SetDataDouble

Description

SetDataDouble gives the data to the plugin.

Syntax

```
int SetDataDouble(const wchar_t* iwszParameterName, const unsigned int
iunTableSize, const double* ipTable);
```

Parameters

Input

- *iwszParameterName*: Data name the "ipTable" contains. See the list of possibles values bellow.
- *iunTableSize*: Size of the table "ipTable".
- *ipTable*: A table that contains the data.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_ERROR_INVALID_DATA 4

struct DataDouble

{
     unsigned int Size;
     double* Table;
};

std::map<std::wstring, DataDouble> gmDatas;

int SetDataDouble(const wchar_t* iwszParameterName, const unsigned int
iunTableSize, const double* ipTable)
{
     if (iunTableSize != 0 && ipTable != NULL)
     {
DataDouble theData;
theData.Size = iunTableSize;
theData.Table = ipTable;
std::wstring parameterName(iwszParameterName);
gmDatas[parameterName] = theData;
     }
     else
     {
return OPT_PLUGIN_ERROR_INVALID_DATA;
     }
     return OPT_PLUGIN_NO_ERROR;
}
```

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

List of the possible value of "iwszParameterName"

Only the parameters that HIW can compute are sent to the plugin.

For example in "Uniform" eyebox mode, the parameters relative to the adaptive eyebox are not sent. If you do not specify an "Outer Surface" in the "Windshield/Combiner" tab, the parameters relative to the ghost are not sent.

Parameter name	Unit	Description
DriverHeight	mm	Defines the driver's height
RotationMirror		Mirror's rotation angle

14.3.3.2.8. SetDataUnsignedInt

Description

SetDataUnsignedInt gives the data to the plugin.

Syntax

```
int SetDataUnsignedInt(const wchar_t* iwszParameterName, const unsigned int
iunTableSize, const unsigned int* ipTable);
```

Parameters

Input

- *iwszParameterName*: Data name the "ipTable" contains. See the list of possibles values bellow.
- *iunTableSize*: Size of the table "ipTable".
- *ipTable*: A table that contains the data.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_ERROR_INVALID_DATA 4

struct DataUnsignedInt

{
     unsigned int Size;
     unsigned int* Table;
};
std::map<std::wstring, DataUnsignedInt> gmDatas;
int SetDataUnsignedInt(const wchar_t* iwszParameterName, const unsigned int
iunTableSize, const unsigned int* ipTable)
{
     if (iunTableSize != 0 && ipTable != NULL)
     {
DataUnsignedInt theData;
```

DRAFT

```
theData.Size = iunTableSize;
theData.Table = ipTable;
std::wstring parameterName(iwszParameterName);
gmDatas[parameterName] = theData;
     }
     else
     {
return OPT_PLUGIN_ERROR_INVALID_DATA;
     }
     return OPT_PLUGIN_NO_ERROR;
}
```

List of the possible value of "iwszParameterName"

Parameter name	Unit	Description	Organization
InputImageHorizontalResolution		Horizontal resolution of image to be warped.	
InputImageVerticalResolution		Vertical resolution of image to be warped.	
PGUHorizontalResolution		Horizontal resolution of PGU.	
PGUVerticalResolution		Vertical resolution of PGU.	

14.3.3.2.9. SetDataUnsignedChar

Description

SetDataUnsignedChar gives the data to the plugin.

Syntax

```
int SetDataUnsignedChar(const wchar_t* iwszParameterName, const unsigned int
iunTableSize, const unsigned char* ipTable);
```

Parameters

Input

- *iwszParameterName*: Data name the "ipTable" contains. See the list of possibles values bellow.
- *iunTableSize*: Size of the table "ipTable".
- *ipTable*: A table that contains the data.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

DRAFT

Example

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_ERROR_INVALID_DATA 4

struct DataUnsignedChar

{
     unsigned int Size;
     unsigned char* Table;
};
std::map<std::wstring, DataUnsignedChar> gmDatas;
int SetDataUnsignedChar(const wchar_t* iwszParameterName, const unsigned int
iunTableSize, const unsigned char* ipTable)
{
     if (iunTableSize != 0 && ipTable != NULL)
     {
DataUnsignedInt theData;
theData.Size = iunTableSize;
theData.Table = ipTable;
std::wstring parameterName(iwszParameterName);
gmDatas[parameterName] = theData;
     }
     else
     {
return OPT_PLUGIN_ERROR_INVALID_DATA;
     }
     return OPT_PLUGIN_NO_ERROR;
}
```

List of the possible value of "iwszParameterName"

Parameter name	Unit	Description	Organization
OutputImage		Pixels array containing warping image result.	RGB
InputImage		Pixels array containing image to be warped.	RGB

14.3.3.2.10. SetDataString

Description

SetDataString gives the data to the plugin.

Syntax

```
int SetDataString(const wchar_t* iwszParameterName, const unsigned int
iunTableSize, const wchar_t** ipTable);
```

Parameters

Input

- *iwszParameterName*: data name the "ipTable" contains. See the list of possible values below.
- *iunTableSize*: Size of the table "ipTable".
- *ipTable*: A table that contains the data.

Return

(*int*) : returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_ERROR_INVALID_DATA 4
struct DataString
{
     unsigned int Size;
     wchar_t** Table;
};
std::map<std::wstring, DataString> gmDatas;
int SetDataString(const wchar_t* iwszParameterName, const unsigned int
iunTableSize, const wchar_t** ipTable)
{
     if (iunTableSize != 0 && ipTable != NULL)
     {
DataString theData;
theData.Size = iunTableSize;
theData.Table = ipTable;
std::wstring parameterName(iwszParameterName);
gmDatas[parameterName] = theData;
     }
     else
     {
return OPT_PLUGIN_ERROR_INVALID_DATA;
     }
     return OPT_PLUGIN_NO_ERROR;
}
```

List of the possible value of "iwszParameterName"

	Parameter name	Unit	Description	Organization
Other	WarpingDataPath		Absolute path of xarping data file.	

14.3.3.2.11. Init

Description

Init initializes the HIW plugin and should be called once before any other HIW API call.

Syntax

```
OPT_HIW_API int Init();
```

Parameters

Return

(*int*) : returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_INIT_ERROR 1

int Init()
{
     If (!InitErrorDescription(...)) return OPT_PLUGIN_INIT_ERROR;
     If (!InitAlgorithms(...)) return OPT_PLUGIN_INIT_ERROR;

return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.2.12. Run

Description

Run executes the requested algorithm computation with the given inputs and writes down the result into the output array parameter "OutputImage".

Syntax

```
OPT_HIW_API int Run(const unsigned int iunAlgoIndex);
```

Parameters

Input

iunAlgoIndex: Algorithm to be used for the warping computation.

- 0: nearest algorithm, low quality warping.
- 1: bilinear algorithm, high quality warping.

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_INIT_ERROR 1

struct DataUnsignedInt
{
     unsigned int ValuesSize; // size of uint table
     const unsigned int* Values; // pointer to uint table
};
```

DRAFT

```
int Run(const unsigned int iunAlgoIndex)
{
     DataUnsignedInt& theData = gmDatasUint[L"OutputImage"];
     theData.Values = ComputeWarping(iunAlgoIndex);

return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.2.13. Release

Description

Release deallocates all resources allocated by the HIW plugin and should be called once before freeing the library.

Syntax

```
OPT_HIW_API int Release();
```

Parameters

Return

(*int*): returns the identification number of the error if an error occurs or 0 if no error occurs.

Example

```
#define OPT_PLUGIN_NO_ERROR 0
#define OPT_PLUGIN_RELEASE_ERROR 2

int Release()
{
     If (!ClearAlgorithms(...)) return OPT_PLUGIN_ RELEASE_ERROR;

return OPT_PLUGIN_NO_ERROR;
}
```

14.3.3.3. Debugging a Plugin

The following procedure helps you debug a HUD Optical Analysis plugin with Visual Studio.

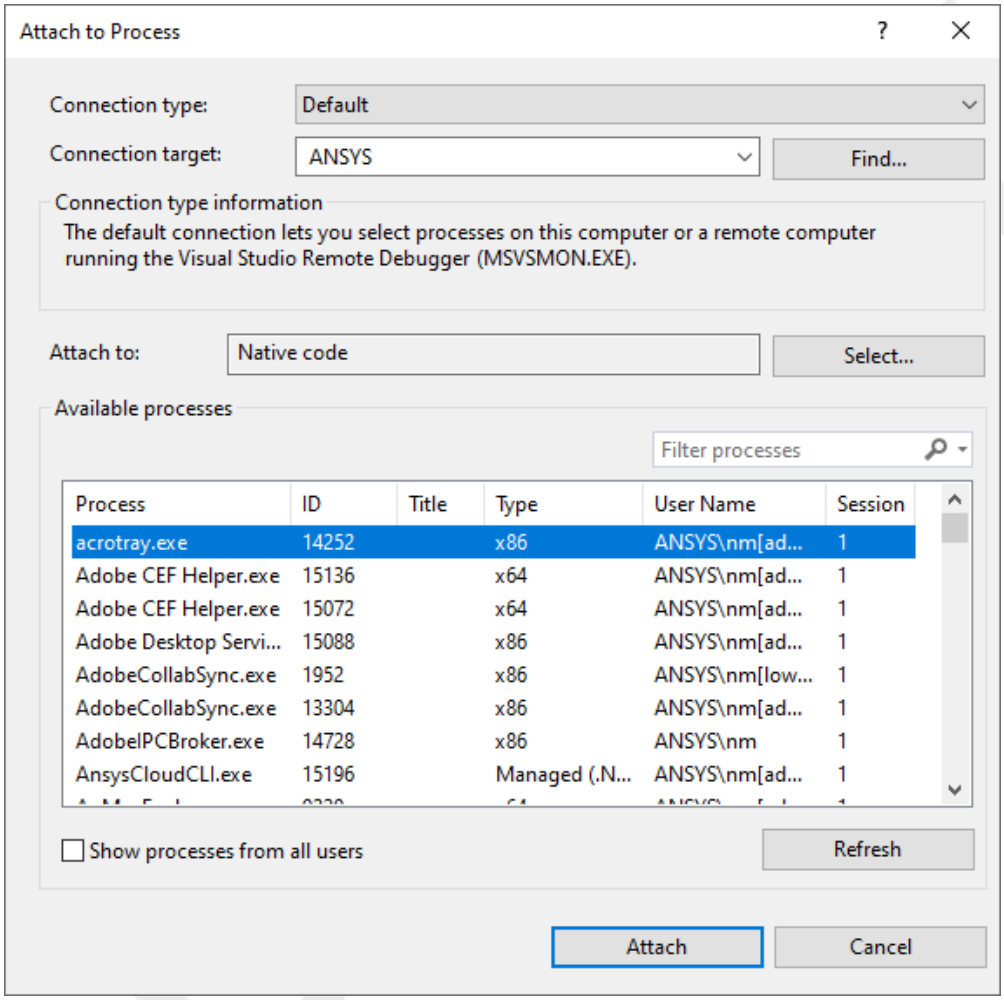
To debug a plugin:

1. Open **Visual Studio**.
2. Load the source code of your plugin.
3. Set **Visual Studio** to **Debug x64**.
4. In the Build tab, click **Build Solution** to create the *.dll file.
5. Copy the *.dll file and its *.pdb file to `C:\ProgramData\Ansys\v2XX\Optical Products\Plugins`.

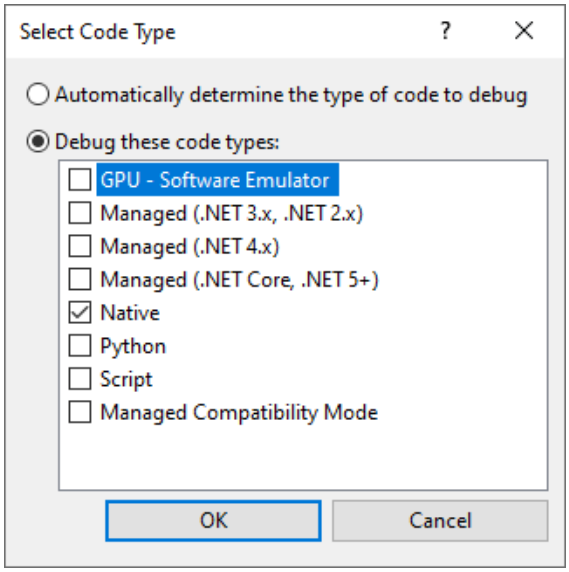
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Note: The *.pdb file is used to link the debugger to the source code by Visual Studio to enable debugging.

6. Start Speos.
7. Load the project containing the HUD Optical Analysis (HOA).
8. In **Visual Studio**, click **Debug**, **Attach to Process**.
   The **Attach to Process** window opens.
9. In **Attach to**, click **Select**.



   The **Select Code Type** window opens.
10. Select **Degug these code types** and check **Native**.



11. Click **OK**.
12. In the **Attach to Process** window, select **SpaceClaim.exe**.
13. In Speos, run the HOA with the plugin to debug, and keep Visual Studio open at the same time.

You can now debug or modify your plugin while running a HOA.

DRAFT

15: Optimization

Optimization is an experiment process that helps to find the best solution for an optical system. It mostly consists in trying to achieve an expected result thanks to parameter variation.

15.1. Optimization with Ansys Workbench

The following section presents you how to run an optimization with Ansys Workbench.

15.1.1. Speos in Ansys Workbench

This page quickly introduces Ansys Workbench, the tool used by Speos to perform result and system optimization.

> **Note:** If you need more information on Ansys Workbench functioning and behavior, refer to Ansys Workbench User's Guide.

Ansys Workbench

Ansys Workbench is the central software environment for performing mechanical, thermal, electromagnetic and optical analyses with Ansys engineering simulation solutions.

Ansys Workbench uses building blocks called *systems*. These systems make up a flowchart-like diagram that represent the data flow through your project.

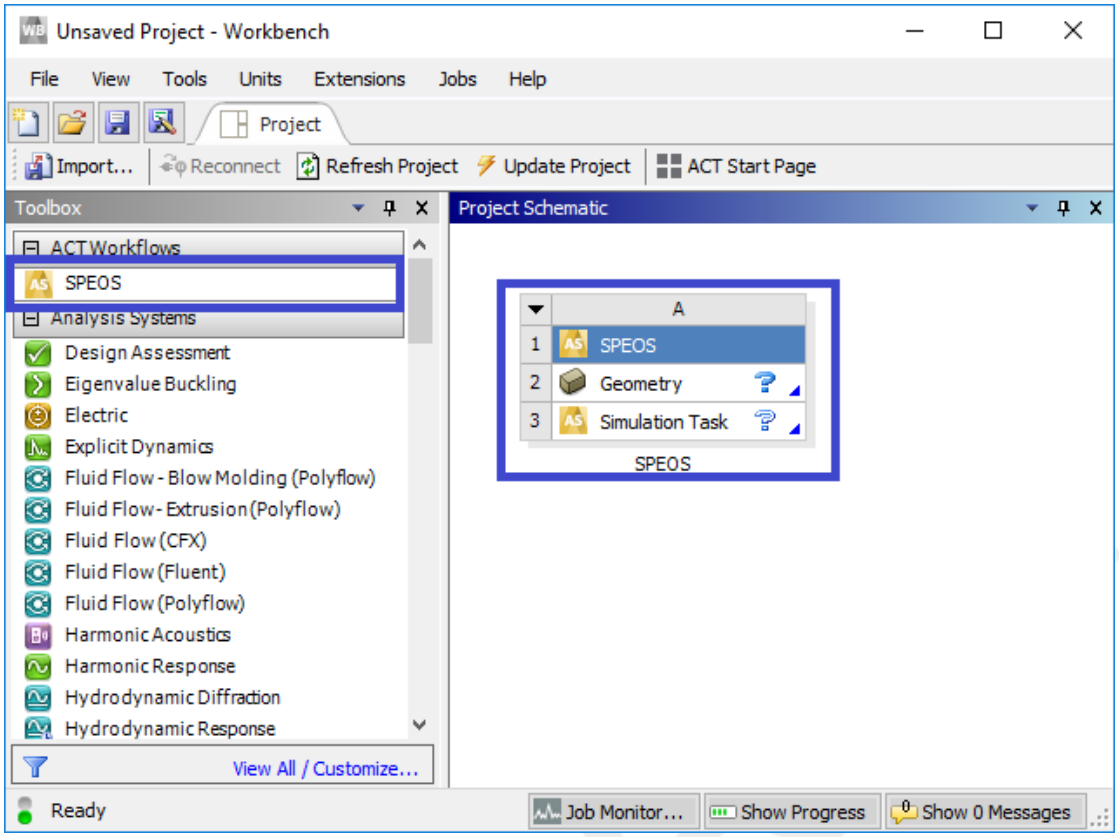
Each system is a block of one or more components called cells, which represent the sequential steps necessary for the specific type of analysis.

One or several systems can be added to your analysis. Then, you can decide to work with a standalone system or make systems interact with each other.

To create a project, a *system* must be dragged and dropped in the **Project Schematic** view.

DRAFT

Speos in Workbench



Speos building block or system is directly integrated into Ansys Workbench and can be used to automate simulation processes, understand and optimize a design or create multi-physics analyses.

> **Note:** The Speos block is not compatible with DesignModeler.

This system creates a bridge between Speos and Ansys Workbench allowing the software to exchange input and output data.

To perform an Speos analysis in Ansys Workbench, you must work through the cells of the Speos system to define inputs, specify project parameters, run simulations, and investigate the results.

Related tasks

Creating a Speos system in Ansys Workbench on page 770
This page shows the main steps to create a Speos system in Ansys Workbench. The Speos system in Ansys Workbench allows you to create a link between the two software. This link/bridge allows the software to exchange input and output data.

Related information

Optimization Tools on page 775
The following section introduces how you can optimize Speos results with Ansys Workbench.

15.1.2. Creating a Speos system in Ansys Workbench

This page shows the main steps to create a Speos system in Ansys Workbench. The Speos system in Ansys Workbench allows you to create a link between the two software. This link/bridge allows the software to exchange input and output data.

Note: If you need more information on Ansys Workbench functioning and behavior, refer to Ansys Workbench User's Guide.

Note: As a project cannot mix *.scdoc data and *.scdocx data, a Workbench project created with *.scdoc data should be recreated after the *.scdoc file conversion to *.scdocx file

For more information, refer to SCDOC File Import.

To create a Speos system in Ansys Workbench:

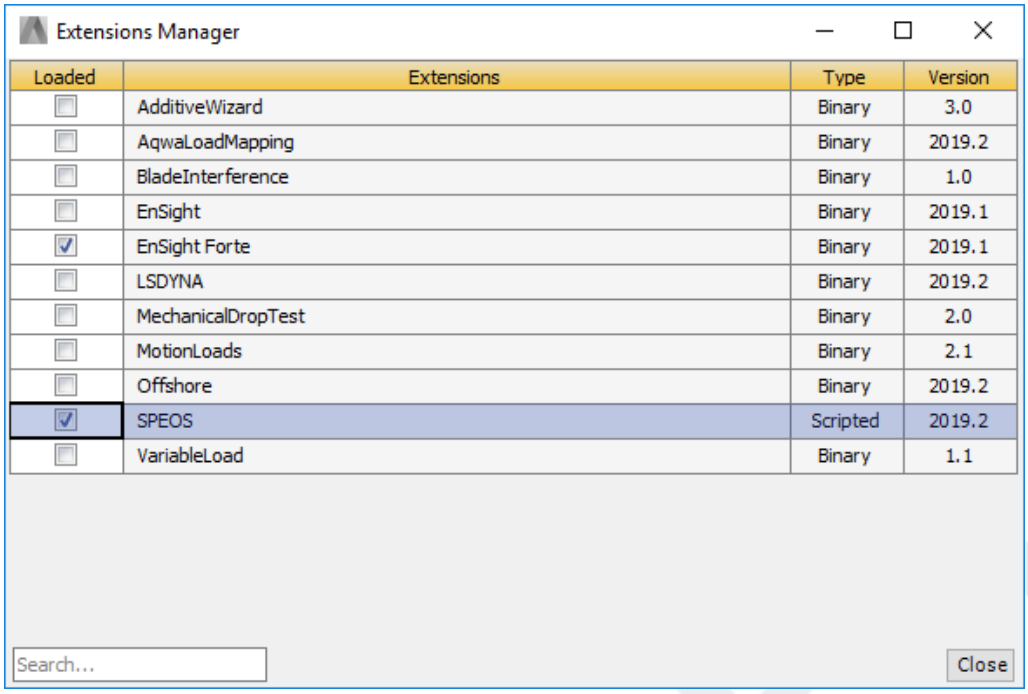
The assembly, all parts used in the assembly, and the Speos input files folder must be located at the same directory level. Plus, all file used by the assembly and parts must be in this same Speos input files folder.

1. From **Windows Start Menu**, launch **Workbench with Speos**.

   CAUTION: If you launch **Workbench** (not **Workbench with Speos**), the Speos environment will not be loaded when you open a SpaceClaim session from the geometry system or cell.

2. Before working, make sure Workbench with Speos runs in foreground, and the Geometry Editor is set to SpaceClaim Direct Modeler:
   a) Select the **Tools** tab and click **Options**.
   b) Select the **Solution Process** section.
   c) In the **Default Update Option** drop-down list, if not defined, select **Run in Foreground**.
   d) Now, select the **Geometry Import section**.
   e) In the **Preferred Geometry Editor** drop-down list, if not defined, select **SpaceClaim Direct Modeler**.
   f) Click **OK**.
3. If **Speos** extension is not displayed in the **Toolbox,** click **Extensions**> **Manage Extensions...**
4. From the list, select **Speos** and click **Close** to import the extension in Workbench.

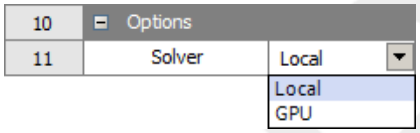
DRAFT



5. Drag and drop **Speos** extension from the **Toolbox** to the **Project Schematic** pane to create a standalone project.

> **Note:** Project dependencies are not preserved when duplicating a Speos block.

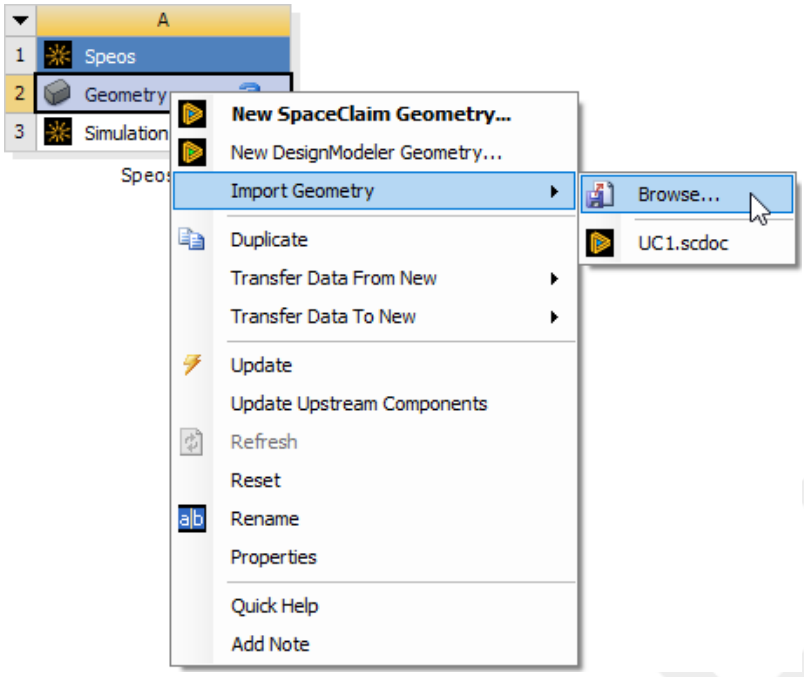
6. If you want to modify the solver to compute the simulation:



   a) in the **Speos** system, select the **Simulation Task** cell.
   b) In the **Properties** panel of the Simulation Task, select the **Solver** to use between **Local** (CPU) and **GPU**

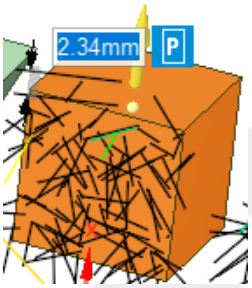
7. To import the Speos project, right-click **Geometry** > **Import Geometry**.
   You can select a previously used design (*.scdoc file) or browse to a specific location to retrieve a project.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

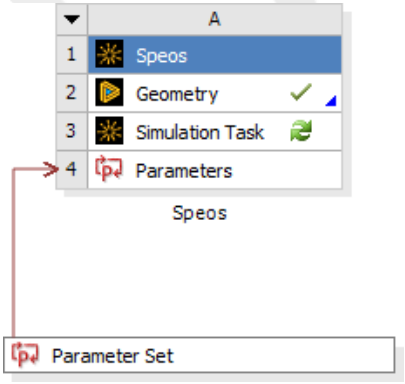


Your geometry and project is imported in Ansys Workbench.

8. If you want to import a geometry parameter (like a dimension parameter), double-click the **Geometry** cell to open SpaceClaim.
   a) Modify a parameter on a geometry.
   SpaceClaim proposes you to add the parameter in Ansys Workbench.



   b) Click **Add as parameter** P.
   The geometry parameters is linked and imported in Ansys Workbench.



9. Double-click Speos **Simulation Task** to open Speos.
10. In Speos, define the Speos input parameters to import in Ansys Workbench:
    a) From Speos tree, select a feature (a source for example).

DRAFT

b) Click the **Workbench** tab and click **Publish Parameters** 

.

c) From the list, select the Speos parameters to be imported (for a source for example, its **flux** or **total angle**).

> **Note:** Publish Parameters are not compatible with the control points positions (%) of the Light Guide feature.

11. In Speos, define the Speos output parameters to import in Ansys Workbench:

a) If not already done, create a simulation and **Compute** it.

b) Open the result and define **Measures**.

c) Select **File** > **Export template**.

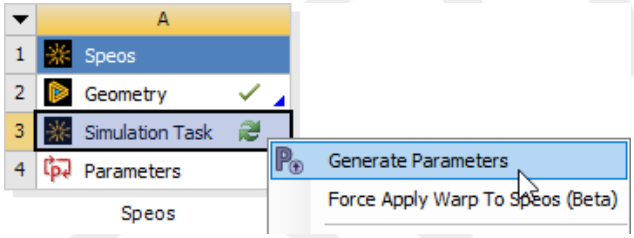
d) In Speos, use the template as XML template in the sensor that generated the result.
This will define the measures as output parameters.

e) Save the current project.

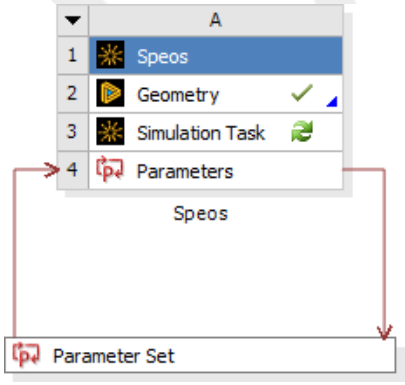
All output data/parameters (measure performed on the simulation results) are automatically imported in Ansys Workbench.

> **Note:** Once the **Publish Parameters** are defined, you can close Speos or leave it open and running in the background.

12. In Ansys Workbench, right-click **Simulation Task** and click **Generate Parameters**  to import Speos input parameters.



The input parameters are imported into Ansys Workbench. The loop is created between the two software.



The project is created, you can now interact and adjust parameters directly from Workbench to observe result variation.

> **Tip:** If you need to refresh or modify input data, repeat the procedure from step 6. Modify input publish parameters and regenerate the import.

Related information

Speos Parameters' Variation on page 775
This section introduces the **Table of Design Points** optimization tool and describes how to manually modify input parameters to observe result variation.

Speos Direct Optimization on page 779
Ansys DesignXplorer uses optimization methods such as Direct Optimization to study and optimize an Speos design.

15.1.3. Linking Static Structural Solution to Speos Geometry

You can create a connection between the Solution cell of a Static Structural system and the Geometry cell of a Speos system. When the link is created, *.scdoc and *.obj files or *.pmdb and *.obj files containing geometry, mesh, and deformed mesh information are transferred. From this information, warping is applied to the geometry.

To create the link:

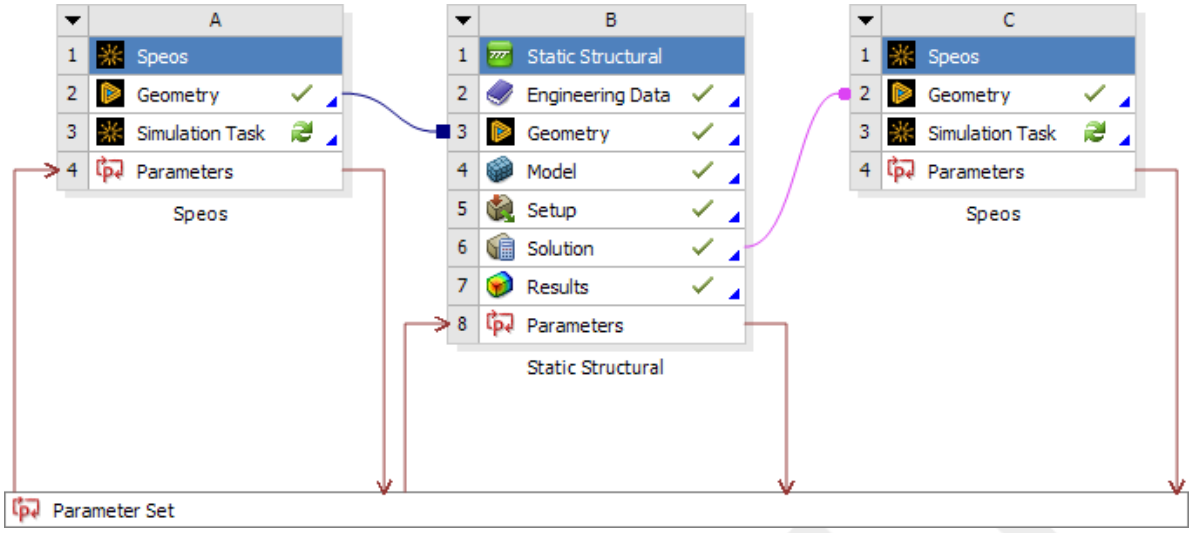
1. To add a Static Structural system, drag the system from the **Toolbox** to the **Project Schematic** or double-click the system in the **Toolbox**.
2. Right-click the **Solution** cell and select **Transfer Data to New** > **SPEOS**.

> **Note:** When defining the project, make sure to create named selections on nominal project (Block A). These named selections will be replaced by deformed geometries in the block (Block C) after mechanical block.

> **Note:** Only Bodies can be deformed. Surfaces are not supported.

The Speos system is added and the connection is made.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



15.1.4. Optimization Tools

The following section introduces how you can optimize Speos results with Ansys Workbench.

> **Note:** To have detailed and extensive information on Workbench optimization processes and tools, refer to Ansys Workbench User's Guide .

15.1.4.1. Speos Parameters' Variation

This section introduces the **Table of Design Points** optimization tool and describes how to manually modify input parameters to observe result variation.

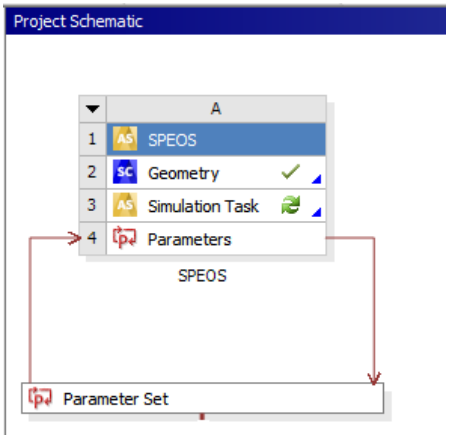
15.1.4.1.1. Table of Design Points

This page introduces the **Table of Design Points** optimization tool.

> **Note:** If you need more information on Ansys Workbench table of design points, refer to Ansys Workbench User's Guide .

The table of design points is directly accessible from the **Parameters Set** tab.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00



Ansys Workbench Structure

Outline of All Parameters

	A	B	C	D
1	ID	Parameter Name	Value	Unit
2	Input Parameters			
3	SPEOS (A1)			
4	P1	Flux/FluxValueLuminous	60	cd*sr
5	P2	Flux/FluxValueLuminous	270	cd*sr
6	P3	Flux/FluxValueLuminous	20	cd*sr
*	New input parameter			
8	Output Parameters			
9	SPEOS (A1)			
10	P4	Area_1/x.Average	0,29725	
11	P5	Area_1/y.Average	0,26861	
*	New output parameter			
13	Charts			

Outline of Input Parameters

The **Table of Design Points** allows you to manually modify input parameters to automatically regenerate output results.

A design point represents a configuration with its set of values.

You can create various design points and launch simulations for each one of these configurations through Workbench to observe result variation.

Ansys Workbench triggers Speos simulation when you update a design point and displays the corresponding results in the current session.

Inputs and outputs are confronted side to side in the table of design points.

Figure 113. Table of Design Points

Table of Design Points

	A	B	C	D	E	F	G	H
1	Name	P1 - Flux/FluxValueLuminous	P2 - Flux/FluxValueLuminous	P3 - Flux/FluxValueLuminous	P4 - Area_1/x.Average	P5 - Area_1/y.Average	Retain	Retained Data
2	Units	cd*sr	cd*sr	cd*sr				
3	DP 0 (Current)	60	270	20	0,29725	0,26861	☑	✓
4	DP 1	100	300	20	0,33786	0,28276	☐	
5	DP 2	54,231	244,04	18,077	⚡	⚡	☐	
6	DP 3	54,692	271,04	19,41	⚡	⚡	☐	
7	DP 4	55,154	257,54	20,744	⚡	⚡	☐	
8	DP 5	55,615	284,54	18,521	⚡	⚡	☐	
9	DP 6	56,077	250,79	19,855	⚡	⚡	☐	
10	DP 7	56,538	277,79	21,188	⚡	⚡	☐	
11	DP 8	57	264,29	18,966	⚡	⚡	☐	
12	DP 9	57,462	291,29	20,299	⚡	⚡	☐	
13	DP 10	57,923	247,41	21,632	⚡	⚡	☐	

Related tasks

Varying Speos parameters in Ansys Workbench on page 777
This page describes how to manually drive and vary input parameters thanks to Ansys Workbench **Table of Design Points**.

Related information

Speos in Ansys Workbench on page 768
This page quickly introduces Ansys Workbench, the tool used by Speos to perform result and system optimization.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

15.1.4.1.2. Varying Speos parameters in Ansys Workbench

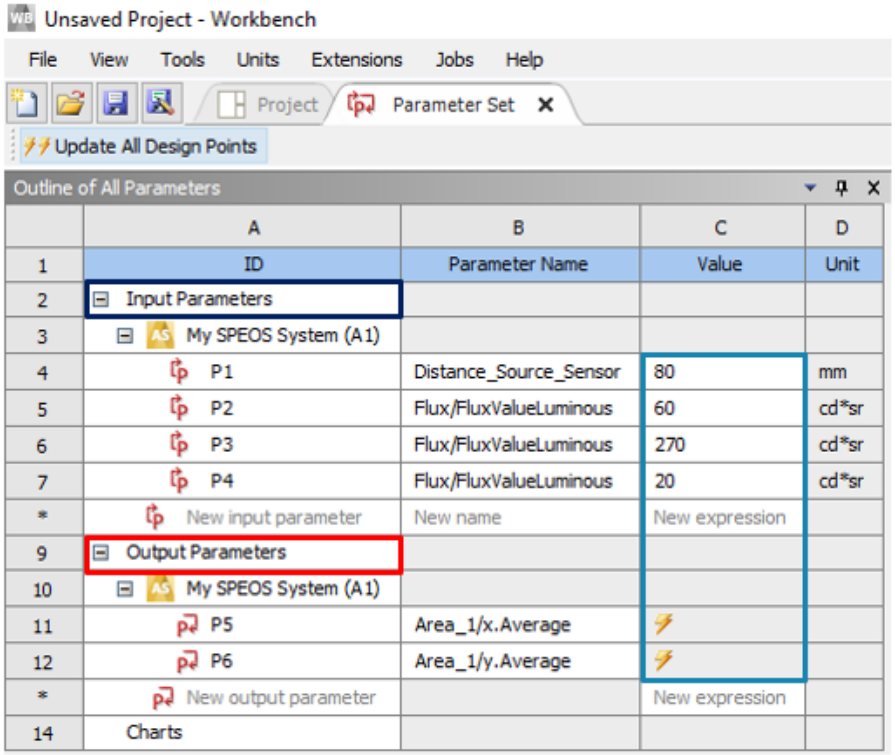
This page describes how to manually drive and vary input parameters thanks to Ansys Workbench **Table of Design Points**.

> **Note:** If you need more information on Ansys Workbench table of design points, refer to Ansys Workbench User's Guide.

To vary Speos Parameters in Ansys Workbench:

A Speos system should already be created .

1. Double-click the **Parameter Set** under the Speos system to access Speos parameters and/or geometry parameters. The **Parameter Set** tab appears and contains the imported input and output Speos parameters. Values displayed in the **Outline of All Parameters** correspond to the parameters of the current row of the **Table of Design Points**.



	A	B	C	D
1	ID	Parameter Name	Value	Unit
2	Input Parameters			
3	My SPEOS System (A1)			
4	P1	Distance_Source_Sensor	80	mm
5	P2	Flux/FluxValueLuminous	60	cd*sr
6	P3	Flux/FluxValueLuminous	270	cd*sr
7	P4	Flux/FluxValueLuminous	20	cd*sr
*	New input parameter	New name	New expression	
9	Output Parameters			
10	My SPEOS System (A1)			
11	P5	Area_1/x.Average		
12	P6	Area_1/y.Average		
*	New output parameter		New expression	
14	Charts			

Table of Design Points

	A	B	C	D	E	F	G	H	I	J
1	Name	P1 - Distance_Source_Sensor	P2 - Flux /FluxValueLuminous	P3 - Flux /FluxValueLuminous	P4 - Flux /FluxValueLuminous	P5 - Area_1/x .Average	P6 - Area_1/y .Average	Retain	Retained Data	Note
2	Units	mm	cd*sr	cd*sr	cd*sr					
3	DP 0 (Current)	80	60	270	20			☑		
*								☐		

2. In the **Table of Design Points**, in case of a large number of Design Points, make sure to deactivate the **Retain** option. Otherwise, you may face with an increasing time for the Design Points update.
3. In the **Project** tab, click **Update Project** to trigger the import of Speos output values.

> **Note:** By default, output parameters are not updated. A lightning bolt  indicates when a design point needs an update.

4. Change or create new input parameters by editing the values to generate new output values:
   a) If you only want to modify values, double-click the design point cell and modify an input (increase the source flux for example).
   b) If you want to add a new design point, type a value in a cell of the table's last row.

   A lightning bolt  is displayed to indicate that output values need to be updated for the newly created point or for modified design points.

   > **Tip:** Right-click a row to be able to duplicate/delete a design point or to show and edit the **Update Order** column. The **Update Order** column allows you to number the design points to prioritize their update by creating a running order.

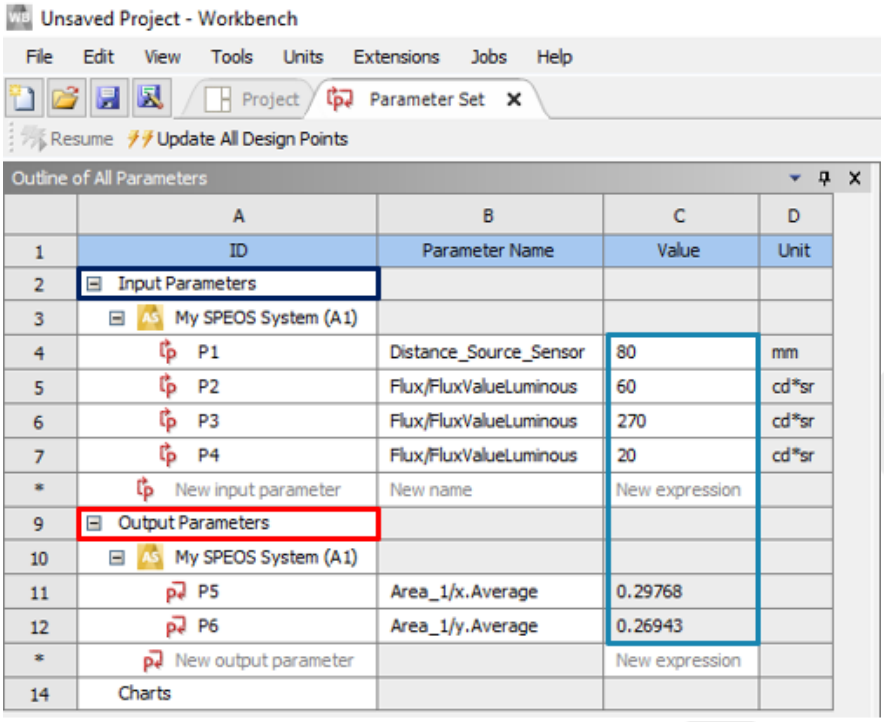
5. Once values have been modified, update the design points:

   > **CAUTION:** We recommend you not to change the units in the Table of Design Points. Units are not converted alongside in Speos.

   - To update a single design point, right-click the point and click **Update Selected Design Point**.
   - To update designated design points, hold the CTRL key and select the row you want to update, then right-click and click **Update Selected Design Points**.
   - To update all design points at once, click **Update All Design Points**

DRAFT



	A	B	C	D
1	ID	Parameter Name	Value	Unit
2	Input Parameters			
3	My SPEOS System (A1)			
4	P1	Distance_Source_Sensor	80	mm
5	P2	Flux/FluxValueLuminous	60	cd*sr
6	P3	Flux/FluxValueLuminous	270	cd*sr
7	P4	Flux/FluxValueLuminous	20	cd*sr
*	New input parameter	New name	New expression	
9	Output Parameters			
10	My SPEOS System (A1)			
11	P5	Area_1/x.Average	0.29768	
12	P6	Area_1/y.Average	0.26943	
*	New output parameter		New expression	
14	Charts			

Ansys Workbench triggers Speos simulations to compute the design points and displays the corresponding output parameters values in the Table of Design points.

> **Note:** Units displayed in the **Output Parameters** may sometimes appear inconsistent with the unit used for simulation (example: "radians" instead of "degrees"")

Related information

Table of Design Points on page 775
This page introduces the **Table of Design Points** optimization tool.

Speos in Ansys Workbench on page 768
This page quickly introduces Ansys Workbench, the tool used by Speos to perform result and system optimization.

15.1.4.2. Speos Direct Optimization

Ansys DesignXplorer uses optimization methods such as Direct Optimization to study and optimize an Speos design.

15.1.4.2.1. Speos Direct Optimization with Ansys DesignXplorer

Direct Optimization allows you to automate the optimization of a system.

> **Note:** If you need more information about Direct Optimization in Ansys DesignXplorer , you can refer to Ansys DesignXplorer User's Guide .

Ansys DesignXplorer is a powerful application in Ansys Workbench for studying and optimizing a Speos design starting from Speos input and output parameters.

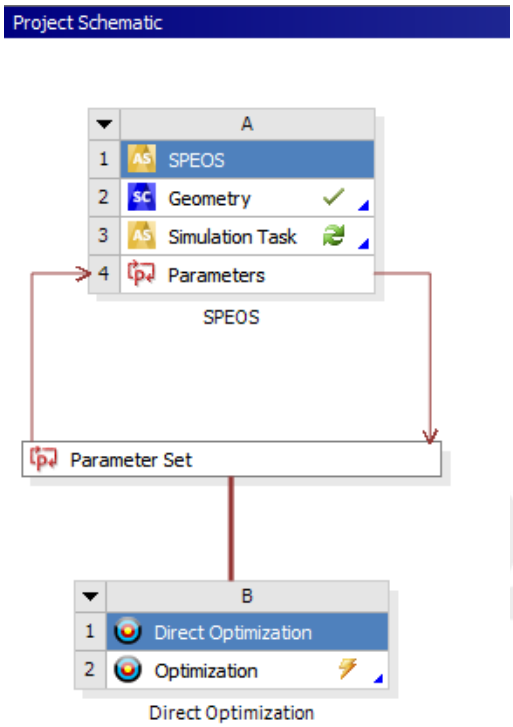
The application uses a deterministic method based on Design of Experiments (DOE) and several optimization methods including Direct Optimization.

Direct Optimization is a goal-driven optimization system that generates arrays of design points and finds out solutions candidates. Responses can be studied, quantified, and graphed in Ansys Workbench.

Presentation

Direct Optimization allows you to automate the search of the best candidate values to reach a desired result.

To create a **Direct Optimization** system, drag the system from the **Toolbox** and drop it under the Speos system so that the parameters are automatically linked.



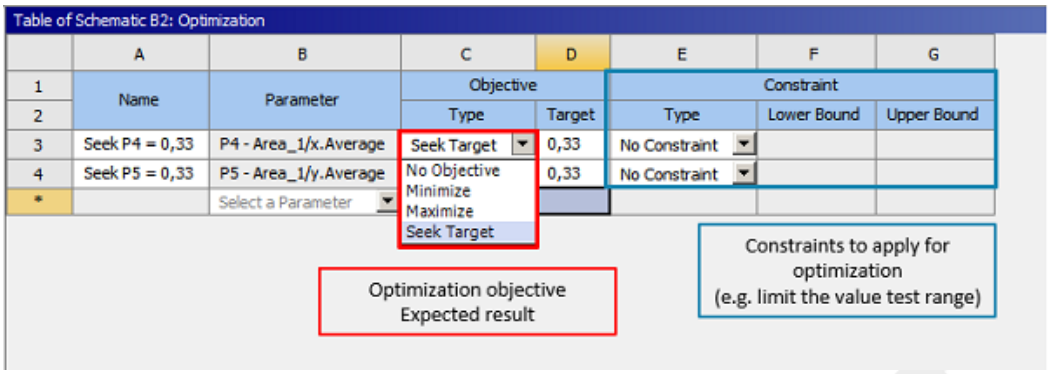
Optimization Process

A target or an action to be performed on the results is defined. You can ask the optimizer to seek a target (define a target value to reach), maximize or minimize a result.

To reduce the optimizer field of action, you can also define constraints like an upper and/or a lower bound to limit the field of measure.

> **Note:** Defining constraints can help gain time as it reduces the possible candidate values to test.

Figure 114. Optimization Definition Table



The system will then basically compute and test out a large number of configurations and parameters set to find the best candidate values to reach the defined target.

To compute these configurations, the optimizer creates design points with specific parameters set.

At the end of the computation, the best candidate values to reach the expected result are selected.

Related tasks

Creating a Direct Optimization in Ansys Workbench on page 781
This page describes how to use Direct Optimization that allows you to automate the search of the best candidate values to reach a desired result.

Related information

Speos in Ansys Workbench on page 768
This page quickly introduces Ansys Workbench, the tool used by Speos to perform result and system optimization.

Speos Parameters' Variation on page 775
This section introduces the **Table of Design Points** optimization tool and describes how to manually modify input parameters to observe result variation.

15.1.4.2.2. Creating a Direct Optimization in Ansys Workbench

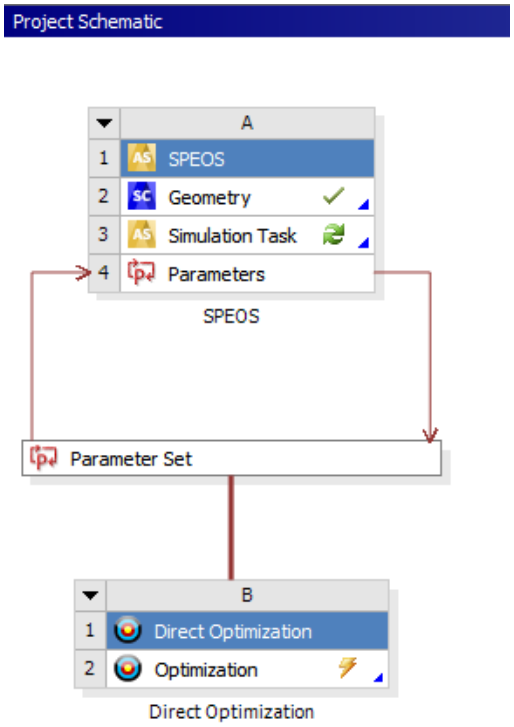
This page describes how to use Direct Optimization that allows you to automate the search of the best candidate values to reach a desired result.

> **Note:** If you need more information about Direct Optimization in Ansys DesignXplorer , you can refer to Ansys DesignXplorer User's Guide .

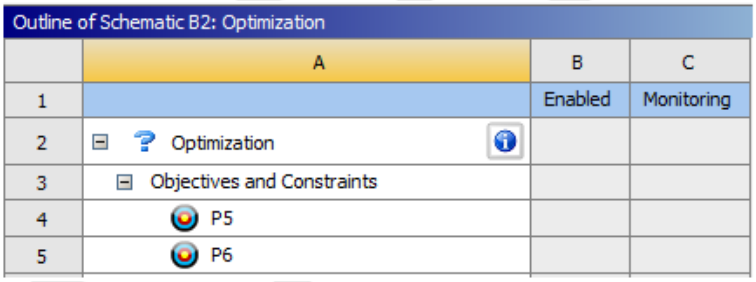
To create a Direct Optimization in Ansys Workbench:

1. Drag the **Direct Optimization** system from the **Toolbox** and drop it under the Speos system so that the parameters are automatically linked.

DRAFT



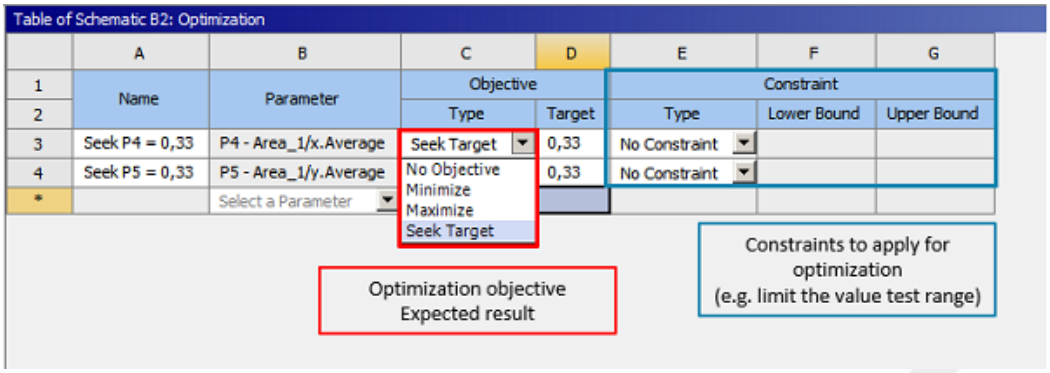
2. Double-click the corresponding cell to access the Direct Optimization parameters.
3. Define targets (parameters to be optimized):
   a) Right-click the **Objective and Constraints** in the **Outlines of Schematic Optimization** panel and select **Insert Objective on**.
   b) Select the parameters to be optimized.



   The parameters to be optimized appear under **Objectives and Constraints**.

4. To define the objective and constraints to apply to the created targets, click a target.

   Objectives and constraints appear in the **Table of Schematic Optimization**.



5. From the **Objective Type** drop-down list, define the action to perform on an output parameter:
   - Select **No objective** if you do not want to specify any action to be performed on the current target.
   - Select **Minimize** to achieve the lowest possible value for the output parameter.
   - Select **Maximize** to achieve the highest possible value for the output parameter.
   - Select **Seek Target** to achieve an output parameter value that is close to the objective target.
6. From the **Constraints Type** drop-down list, define the constraint to be applied to the output parameter:
   - Select **No Constraint** if you do not want to specify any constraint.
   - Select **Values=Bound** to specify a lower bound and obtain an output parameter value that is close to that bound.
   - Select **Values >= Lower Bound** to specify a lower bound and obtain an output parameter that is above that bound.
   - Select **Values <= Upper Bound** to specify an upper bound and obtain an output parameter below that bound.
   - Select **Lower Bound <= Values <= Upper Bound** to specify lower and upper bounds and obtain an output parameter within the defined range.
7. When the Optimization is set, click Update to launch the computation.

The optimization iterates over Design Points and refines values until finding the best candidate values to reach the expected result.

Related information

Speos in Ansys Workbench on page 768
This page quickly introduces Ansys Workbench, the tool used by Speos to perform result and system optimization.

Speos Direct Optimization with Ansys DesignXplorer on page 779
Direct Optimization allows you to automate the optimization of a system.

15.2. Optimization with Speos

The following section presents you how to run an optimization with the embedded optimization feature in Speos.

DRAFT

15.2.1. Optimization Overview

The following page present you the Speos Optimization feature and its capabilities.

The Speos Optimization feature helps you get the best of your optical system by allowing you to find the ideal values to set according to target values you want to reach. The Optimization feature gives you flexibility in your analysis providing you with different optimization modes and different types of variables possible.

Types of Optimization

The Optimization feature provides you with three optimization modes:

- Random Search algorithm is a global optimization method based on random.
- Design of Experiment allows you to strictly define the values of the variables you define through the use of an Excel File based on the selected variables.
- Plugin allows you to use an optimization algorithm you created yourself to add more flexibility in your analysis.

Types of Variables

The Optimization feature provides you with three variable types according to where they come from.

- Simulation Variable

  The Simulation Variables correspond to the Speos Light Simulation parameters which correspond to the numerical parameters of the Speos features from the Light Simulation tab used in the Speos simulation you select for the optimization.

- Design Variable

  The Design Variables correspond to the Optical Part Design parameters which correspond to the numerical parameters of the Optical Part Design features geometries from the Design tab used in the Speos simulation you select for the optimization.

- Document Variable

  Document Variables correspond to the input parameters that you can create into the SpaceClaim Groups panel (Driving Dimension, Script Parameter).

Target

Target correspond to the output elements on which you want to measure/evaluate/assess the impact of the variables defined.

Basic Workflow

1. Create a Direct or an Inverse Simulation to analyze your optical system.
2. Run the simulation to generate the results.
3. In the XMP result, define **measures** that you want to use as targets.
4. Create the Optimization in Speos.
5. Add variables.
6. Define the variable boundaries (min max).
7. Add targets that come from the measures you created in the XMP result.
8. Define the target values and their weight.
9. Run the optimization.
10. Analyze the results to see which is the optimal value found according the optimization parameters set.

DRAFT

15.2.2. Understanding Optimization Parameters

This page describes the different parameters to set when creating an Optimization.

Merit Function

The **Merit function** allows you to define the convergence process of the optimization.

- **Minimize** allows you to get the simulation measurement as close as possible to the target value.
- **Maximize** allows you to get the simulation measurement as far as possible from the target value.

The Merit function formula is $MeritFunction = \sqrt{\text{Weight}_1 \cdot (\text{Measure}_1 - \text{Target}_1)^2 + \text{Weight}_2 \cdot (\text{Measure}_2 - \text{Target}_2)^2 + \ldots}$

With:

- Target: Optimization target
- Measure: Measured value of the target
- Weight: level of importance of the target according to the other targets

Mode

Random Search Algorithm

The **Random search** algorithm is a global optimization method based on random.

The Random search algorithm gives comparable results from one optimization run to another and can address all types of cases.

However, be aware that the random search algorithm has its own termination criteria. If this criteria is met before the termination criteria that you defined, the optimization stops. Indeed, at each optimization run, the algorithm compares the Merit function with the best result found. If after 15 runs, the result is the same, then the search area is divided by two and another run of optimization starts. When the search area is lower than 5% of the original search area, the optimization stops as it becomes inefficient to run optimizations due to convergence time.

Design of Experiment

Design of Experiment allows you to strictly define the values of the variables you define through an Excel File generated after the variables selection.

	A	B	C	D
1	Iteration number	Green/Flux/Value	Red/Flux/Value	Blue/Flux/Value
2	1	100	100	100
3	2	80	80	80
4	3	90	90	90
5	4	110	110	110
6	5	120	120	120
7				

Unlike the Random Search and the Plugin modes, the Design of Experiment is not an optimization algorithm as you do not have to define a Merit function and Stop Conditions because it only depends on the number of variable values you defined in the Excel file.

Plugin

The Plugin mode allows you to use an optimization algorithm you created yourself.

DRAFT

The Plugin mode is dedicated to advanced users who know how to create an optimization algorithm.

For more information on the Plugin mode, refer to Optimization Plugin. Optimization Plugin chapter will help you understand how to create a plugin algorithm with a complete optimization plugin example.

Variables

Simulation Variables

The Simulation Variables correspond to the Speos Light Simulation parameters which correspond to the numerical parameters of the Speos features from the Light Simulation tab used in the Speos simulation you select for the optimization.

Design Variables

The Design Variables correspond to the Optical Part Design parameters which correspond to the numerical parameters of the Optical Part Design features from the Design tab used in the Speos simulation you select for the optimization

Document Variables

Document Variables correspond to the input parameters that you can create into the SpaceClaim Groups panel.

They can be:

- A Driving Dimension which is a parameter that has effect on the size or the position of the element.
- A Script Parameter that you created and which is used into a SpaceClaim script.

The classical workflow to create Script Parameters would be:

1. Create a script group in the **Groups** panel.
2. Write down the script.

   You can help yourself using the script **Record** tool that allows you to record parameters as a script in the script editor window.
3. Create and name **Script Parameters** in the **Groups** panel.
4. Edit the script to replace the parameters with the script parameters.

For a complete understanding of how to create Driving Dimensions and Script Parameters, refer to the SpaceClaim User Guide:

- Refer to **Using Driving Dimensions in Ansys** to understand the different ways of creating driving dimensions.
- Refer to **Creating a Parameter Group** to understand how to add and define a script parameter to a script.

> **Note:** Note that unlike Simulation Variables and Design Variables, the Document Variables appearing in the Document Variables list do not necessarily impact the simulation. Therefore, make sure to add Document Variables that are related to your optical system or your simulation.
>
> Example: A Driving Dimension modifying the size of a geometry that is not included in the simulation. Setting the Driving Dimension as Document Variable will have no impact on the optimization.

Targets

Target correspond to the output elements on which you want to measure/evaluate/assess the impact of the variables defined.

DRAFT

These targets correspond to measures you created in the XMP results generated from the simulation that you want to include into the optimization. These measures correspond then to your initial measure generated with the configuration of the simulation before the optimization.

Weight: the weight corresponds to the level of importance of the target according to the other targets. The higher the number the more weight the target.

Keep Intermediate Results

Keep intermediate results allows you to keep the results corresponding to the different optimization iterations.

A subfolder is created with the name of the optimization function in the **SPEOS output files** directory. There is one result saved per iteration.

Use Maximum Time

Use maximum time allows you to define the maximum time in seconds to spend on the optimization process.

In case the time is over and the last simulation run has not yet finished, the optimization process finishes the simulation before stopping.

Use Maximum Number of Simulations

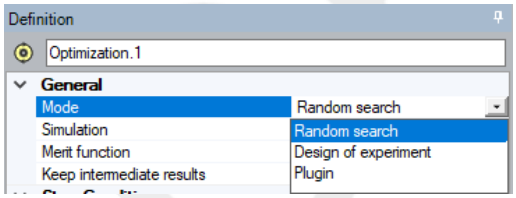
Use maximum number of simulations allows you to define the maximum number of simulations to run before stopping the stopping the optimization process.

15.2.3. Creating an Optimization

The following procedure helps you create an optimization according to the optimization mode defined.

To create an optimization:

1. From the **Light Simulation** tab, click **Optimization**  .
2. In the **General** section, in the **Mode** drop-down list, select the optimization mode to use:



- **Random search**: the optimization algorithm based on the Merit function.
- **Design of experiment**: allows you to use an generated excel file based on the variable you selected so that you can directly define all values for each variable.
- **Plugin**: allows you to add your own optimization plugin for more flexibility.

3. In the 3D view, click  and select the simulation to be optimized.

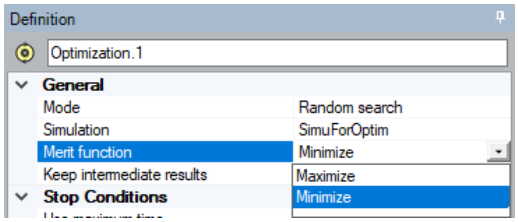
Follow the next procedure for Random search, Design of experiment, or Plugin according to the optimization mode selected.

DRAFT

15.2.3.1. Defining the Random Search Optimization

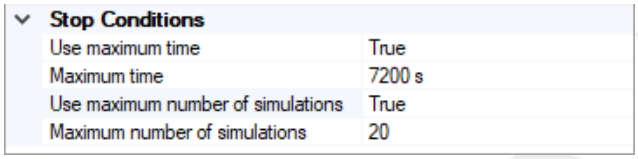
You must have selected the Random search mode in the **Optimization** feature.

1. In the **Merit function** drop-down list, define how you want the optimization to converge:



   - Select **Minimize** to get the simulation measurement as close as possible to the target value.
   - Select **Maximize** to get the simulation measurement as far as possible from the target value.

2. Define if you want to **Keep intermediate results**. That means the result of each iteration will be saved into the **SPEOS output files** folder.
3. In the **Stop Conditions** section, define how to stop the optimization:



   - If you want the optimization to stop after a certain time spent, set **Use maximum time** to **True** and define the **Maximum time** in seconds.

     Important: In case the time is over and the last simulation run has not yet finished, the optimization process finishes the simulation before stopping.

   - If you want the optimization to stop after a certain number of simulations run, set **Use maximum number of simulations** to **True** and define the **Maximum number of simulations**.
   - You can set both stop conditions to **True**. In this case, the first stop condition to be reached will end the optimization.

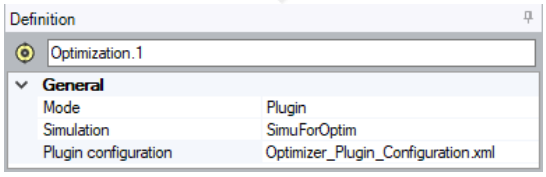
Now you can add and define the different Simulation, Design, Document variables and Targets.

15.2.3.2. Defining the Optimization Plugin

You must have selected the Plugin mode in the **Optimization** feature.

You must have created an optimization plugin. For more information on how to create a plugin, refer to Optimization Plugin.

In the **Plugin configuration** field, browse and load the XML plugin configuration file.



DRAFT

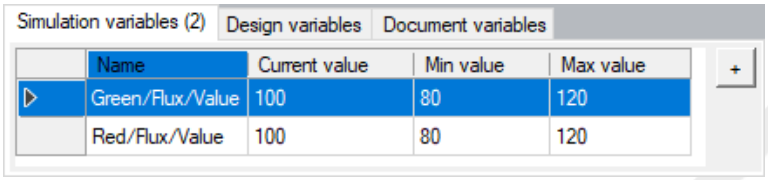
Note: For more information on how to create a XML plugin configuration file, refer to Configuring the XML Optimizer Plugin Configuration File.

Now you can add and define the different Simulation, Design, Document variables and Targets.

15.2.3.3. Defining the Design of Experiment

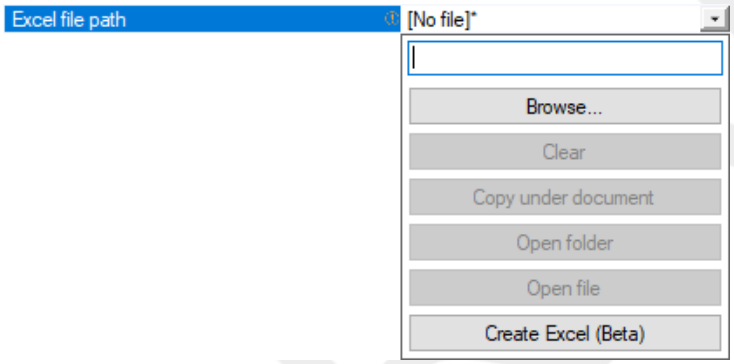
You must have selected the Design of experiment mode in the **Optimization** feature.

1. In the Variables panel, add the variable you want to vary.



   To add variables according to their type, refer to Adding and Defining Simulation Variables, Design Variables and Document Variables.

2. Once variables are defined, in the **Excel file path** drop-down list, click **Create Excel**.



   Important: You must add the variables first in the Variables panel. Otherwise the Excel file will not generate the table of the variable values.

3. Save the Excel file in the **SPEOS input files** folder.
   The Excel file automatically opens.
4. In the Excel file, the first column corresponds to the iterations to run and their order, and all following columns correspond to the each variable you added:

	A	B	C	D
1	Iteration number	Green/Flux/Value	Red/Flux/Value	Blue/Flux/Value
2	1	100	100	100
3	2	80	80	80
4	3	90	90	90
5	4	110	110	110
6	5	120	120	120
7				

   a) In the **Iteration number column**, insert a line for each iteration to run and number them.

b) In the following variable columns, define for each iteration the value of the variables.

> **Note:** The **Min value** and **Max value** of the variables in the Variables panel do not need to be set as the optimization process will take the values defined in the Excel file.

5. Save and close the Excel file.

As you already defined all variables, you can now directly run the optimization.

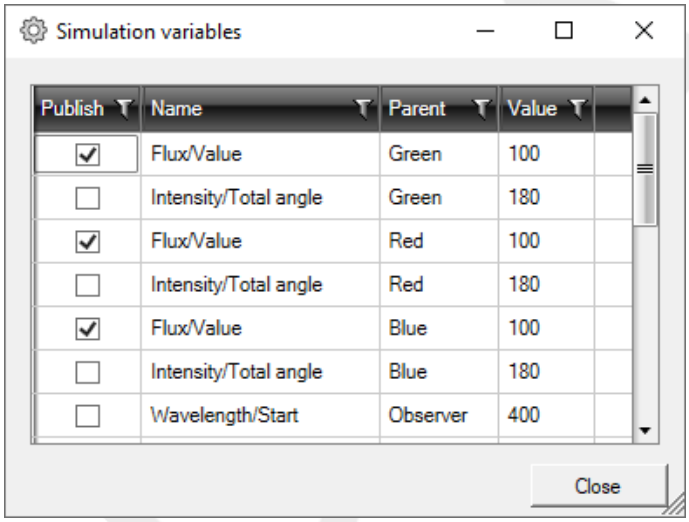
15.2.4. Adding and Defining Variables

The following procedures help you define the variables, that correspond to parameters you want to vary, according to their type.

15.2.4.1. Adding and Defining Simulation Variables

You must have added a simulation to the optimization.

1. In the **Optimization** definition, in the Variables panel, select the **Simulation variables** tab.
2. Click + to open the **Simulation variables** list.



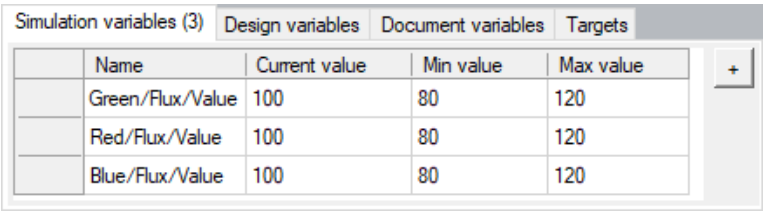
The list shows you the available numerical parameters that you can add as variable that come from the Speos Light Simulation features included in the simulation.

3. Check the parameters to vary.

> **Tip:** In case of a simulation containing a large number of numerical parameters, you can use the filter tool to help you quickly find the parameters.

4. Close the **Simulation variables** list.
   Variables appear into the Simulation variables tab.
5. In the **Simulation variables** tab, for each variable, set the **Min value** and **Max value** to define the range of values the variable can take.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

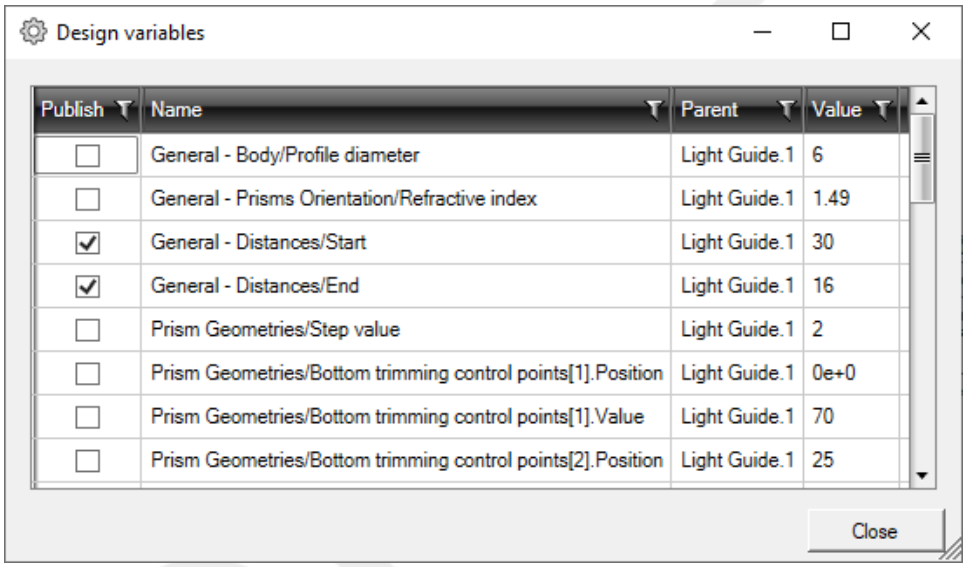


Now you can add other variable types as Design variables or Document variables or directly add the optimization Targets.

15.2.4.2. Adding and Defining Design Variables

You must have added a simulation to the optimization.

1. In the **Optimization** definition, in the Variables panel, select the **Design variables** tab.
2. Click + to open the **Design variables** list.

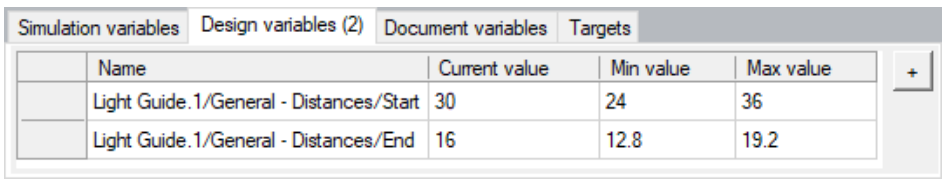


The list shows you the available numerical parameters that you can add as variable that come from the Optical Part Design features included in the simulation.

3. Check the parameters to vary.

> **Tip:** In case of a simulation containing a large number of numerical parameters, you can use the filter tool to help you quickly find the parameters.

4. Close the **Design variables** list.
   Variables appear into the Design variables tab.
5. In the **Design variables** tab, for each variable, set the **Min value** and **Max value** to define the range of values the variable can take.



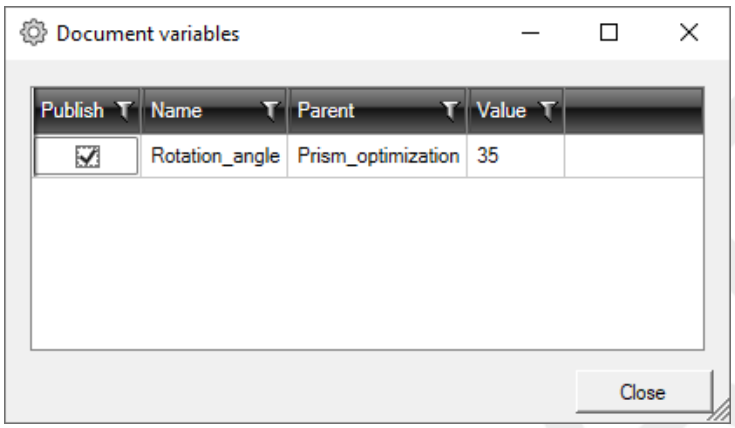
Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Now you can add other variable types as Simulation variables or Document variables or directly add the optimization Targets.

15.2.4.3. Adding and Defining Document Variables

> **Note:** We recommend you to add a simulation to the optimization. You can however define document variables even if no simulation is selected yet.

1. In the **Optimization** definition, in the Variables panel, select the **Document variables** tab.
2. Click + to open the **Document variables** list.



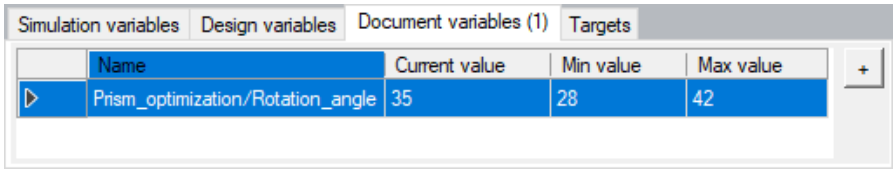
> **Note:** Unlike Simulation Variables and Design Variables, the Document Variables appearing in the Document Variables list do not necessarily impact the simulation. Therefore, make sure to add Document Variables that are related to your optical system or your simulation.

The list shows you the available numerical parameters that you can add as variable that come from the Driving Dimension parameters and/or Script parameters created.

3. Check the parameters to vary.

> **Tip:** In case of a large number of numerical parameters, you can use the filter tool to help you quickly find the parameters.

4. Close the **Document variables** list.
   Variables appear into the Document variables tab.
5. In the **Document variables** tab, for each variable, set the **Min value** and **Max value** to define the range of values the variable can take.

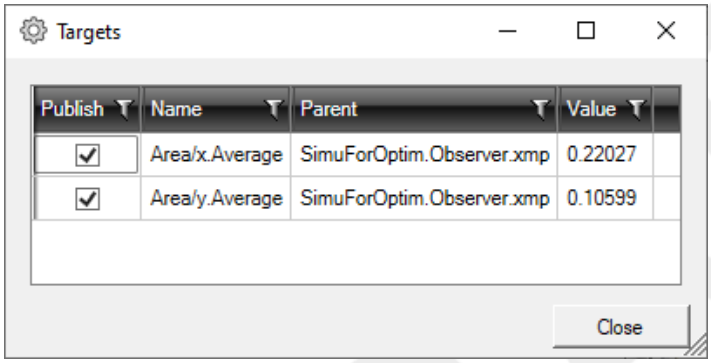


Now you can add other variable types as Design variables or Simulation variables or directly add the optimization Targets.

15.2.5. Adding and Defining Targets

The following procedure helps you define the targets that correspond to the output elements on which you want to measure/evaluate/assess the impact of the variables defined.

1. Create measures that will be used as targets in the optimization:
   a) In Speos, **Compute** the simulation to be used in the optimization.
   b) Open the result and define **Measures** in the Lab.
   c) Select **File** > **Export template**.
   d) In Speos, use the template as XML template in the sensor that generated the result.
   This will define the measures as targets in the optimization.
2. In the **Optimization** definition, in the Variables panel, select the **Targets** tab.
3. Click to open the **Document variables** list.

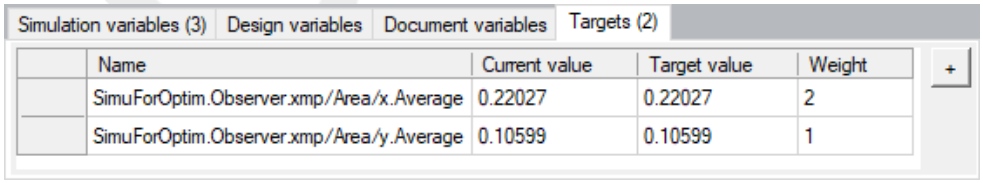


The list shows you the available measures that you can add as target that come from the results of the simulation.

4. Check the measures to target.

> **Tip:** In case of a large number of measures, you can use the filter tool to help you quickly find them.

5. Close the **Targets** list.
   Targets appear into the Targets tab.
6. In the **Targets** tab, for each target, set the **Target value** to reach (Merit function minimize) or from which to move away (Merit function maximize) and its **Weight**.



The weight corresponds to the level of importance of the target according to the other targets. The higher the number the more weight the target.

Now you can add variable types if not yet done as Design variables, Simulation variables or Document variables.

15.2.6. Running the Optimization

The following procedure shows you how to run an optimization (Random Search, Plugin optimization, Design of Experiment).

You must have set the target(s) and variable(s) in case of Random search and Plugin optimization, or set the variables in case of a Design of Experiment.

1. In the **Simulation** tree, right-click the optimization feature and select **Compute** or **GPU Compute** to run the optimization.
2. At the end of the optimization process, Speos asks you if you want to replace the initial variables value with the best solution found.
   - Click **Yes** to replace.
   - Click **No** to keep the initial values.

An HTML report is generated at the end of the optimization process under the Optimization feature node.

In case of a Random search optimization, if you set **Keep intermediate results** to **True**, then a folder is created in the **SPEOS output files** folder with the name of the Optimization feature containing all iteration results.

15.2.7. Reading the HTML Report (Random Search)

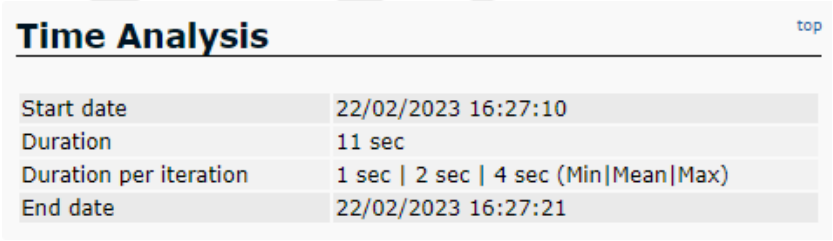
The following page lists and describes all the information contained in an HTML report that includes the summary of all the variables, targets, parameters, and the results of all the evaluations that have been performed during the process, and the best solution.

You can find the HTML report under the SPEOS output files folder of the project.

> **Note:** The following HTML report corresponds to the report for a Random Search optimization.

Time Analysis

The **Time Analysis** section provides you with the date and time of the initialization/termination and duration of the optimization.



Time Analysis top

Start date	22/02/2023 16:27:10
Duration	11 sec
Duration per iteration	1 sec | 2 sec | 4 sec (Min|Mean|Max)
End date	22/02/2023 16:27:21

Variables

Variables section sums up all the variables used in the simulation as well as their Min value and Max value.

> **Note:** Do not consider the Precision column as it is not used.

Variables

top

Name	Minimum	Maximum	Precision
Green/Flux/Value	80	120	0
Red/Flux/Value	80	120	0
Blue/Flux/Value	80	120	0

Targets

Targets section sums up all the targets select, targets value, and weight.

Targets

top

Name	Description	Target	Weight
SimuForOptim.Observer.xmp/Area/x.Average		0.220272	2
SimuForOptim.Observer.xmp/Area/y.Average		0.105991	1

Parameters

Parameters sums up the parameters defined according to the Optimization mode selected.

In case of a Random search optimization the Merit function is developed.

Parameters

Merit Function

Operator	Minimization
Formula	sqrt(2*(([SimuForOptim.Observer.xmp/Area/x.Average]-0.220272)/0.220272)^2 + 1*((
Algorithm	Random Search

Stop Conditions

Optimization iterations greater than 5

Evaluations

Evaluations lists for each iteration all the variable values used, the values found for each target and the Merit function value.

The best solution is highlighted in green.

In case of a Random search optimization, if **Keep intermediate results** is set to **True**, then you have direct link to download the XMP results from the HTML Report.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Evaluations

Index	Green/Flux/Value	Red/Flux/Value	Blue/Flux/Value	SimuForOptim.Observer.xmp/Area/x.Average	SimuForOptim.Obse
1	100	100	100	0.220435	0.
2	97.90331253	98.85626622	99.7755933	0.219668	0.
3	95.61053157	104.6373186	95.73257892	0.225455	0
4	118.2905912	103.8741957	87.21136076	0.232664	0.
5	108.1740852	109.6185879	90.72130163	0.232583	0.

Results

Results repeat the best solution found (highlighted in green in the Evaluations section).

Results

Stop condition encountered: Optimization iterations greater than 5
Engine Status (0):

Best solution

Index	Green/Flux/Value	Red/Flux/Value	Blue/Flux/Value	SimuForOptim.Observer.xmp/Area/x.Average	SimuForOptim.Obse
1	100	100	100	0.220435	0.

15.2.8. Optimization Plugin

The following chapter will help you understand how to create your own optimization algorithm with a complete guided optimization plugin example.

> **Note:** The Plugin mode is dedicated to advanced users who know how to create an optimization algorithm.

15.2.8.1. Optimization Plugin Overview

In the Optimization Plugin section, you learn how to create an optimization plugin using Visual Studio through a full example.

Integrated Development Environment

In the plugin example provided, Visual studio is used with the .NET Framework 4.7.2 development tools for the ease of use to create a plugin for an optimization.

> **Note:** Of course, you can use another development environment. If you do so, make sure to use the .NET framework 4.7.2 tools.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

Plugin Example Tutorial

In the plugin example, you will:

- take the input variables and increment them with a given amount.
- have the configuration run (this is done automatically).
- receive the new values for the targets that you received.
- loop until a certain number of iteration is reached.
- when this is done, output a report containing the target values for each iteration.

Therefore, you will have to implement the following elements:

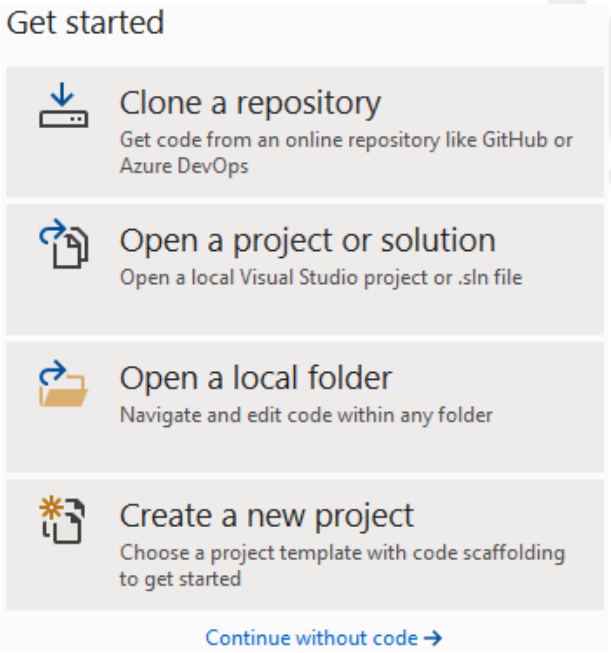
- read input arguments for the increment value and the number of iteration.
- register the variables that you want to increment.
- register the targets that you want to have in the report.
- implement the incrementation of the variables.
- implement the reception of the new measures.
- implement the check to finish the iteration loop.
- output a report file.

15.2.8.2. Creating a Project in Visual Studio

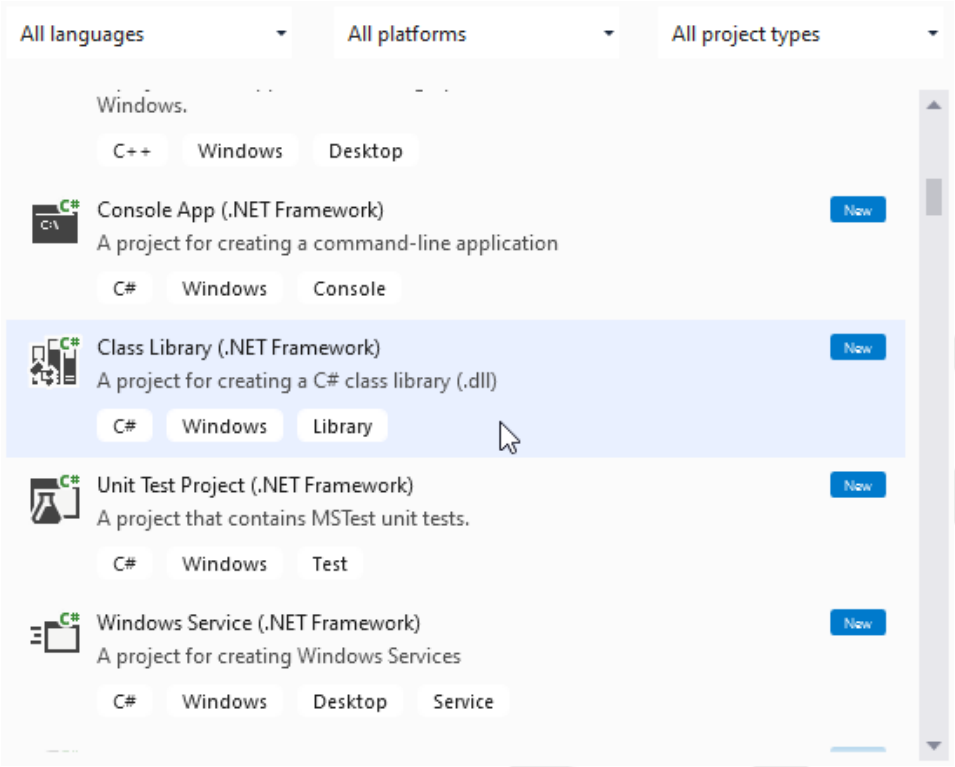
The following procedure helps you creating the project in Visual Studio for the optimization plugin example.

You must have installed the .NET Framework 4.7.2 development tools.

1. Open **Visual Studio**.
2. In the **Get Started** section, click **Create a new project**.

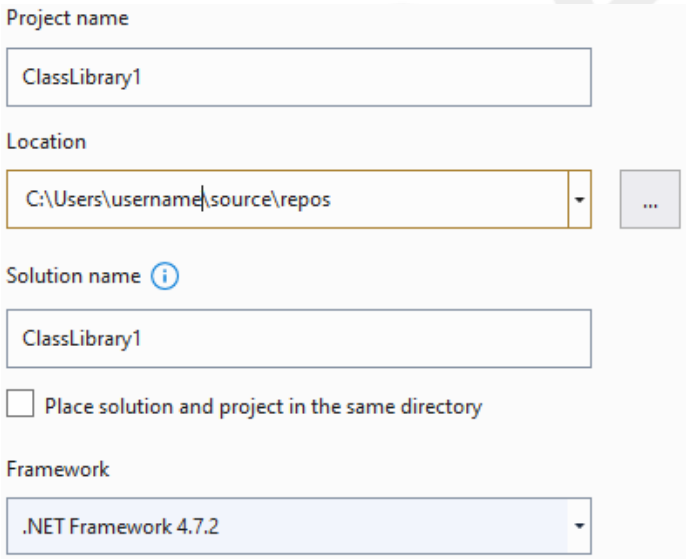


3. In the **Create a new project** window, select **Class Library (.NET Framework)**.



The Class Library (.NET Framework) lets you create a *.dll file using the .NET framework.

4. Click **Next**.
5. In the **Configure your new project**, fill in the different fields:

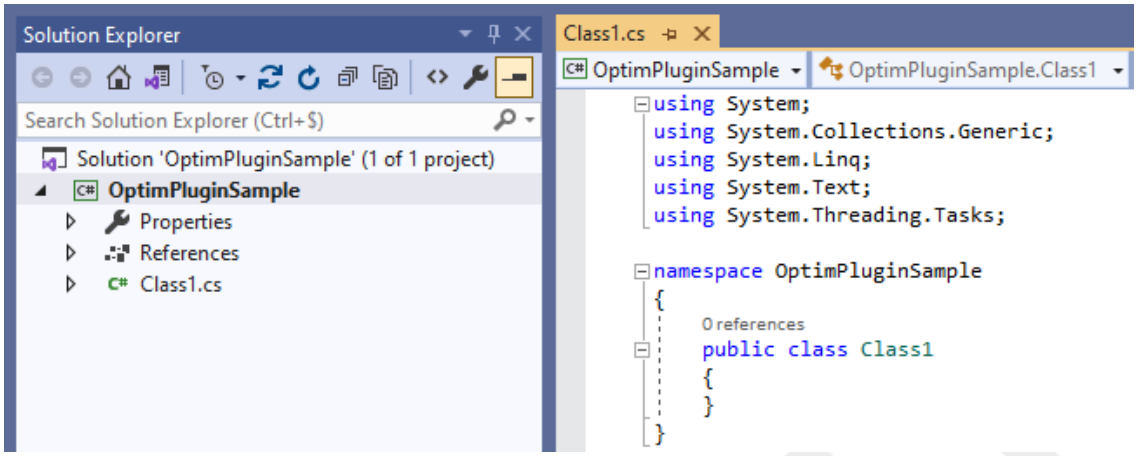


- **Project name** is the name of your *.dll.
- **Location** is the folder where your files are created.
- **Solution name**is the name of your whole solution (a solution can have multiple projects).
- **Framework** is the version of the .NET Framework to use. In your case .NET Framework 4.7.2.

> **Note:** For the plugin example, the project and solution will have the same name: OptimPluginSample.

6. Click **Create**.

The project is created with a default class in it.



15.2.8.3. Creating an Optimization Plugin

The following section guides you step by step in the creation of an optimization plugin providing you specific code at each step.

15.2.8.3.1. Optimization Plugin Template

The following code corresponds to the basic scaffolding of the optimization plugin.

In the Plugin Complete page, you can see the full plugin example written based on this scaffolding.

```
namespace OptimPluginSample
{
    public class PluginSample
    {
        private readonly string _arguments;
        public PluginSample(string arguments)
        {
            _arguments=arguments;
        }
        /// <summary>
        ///     Callback called upon starting the optimization loop (optional)

        /// <summary>
        public void StartRun()
        {
        }
        /// <summary>
        ///     Callback called upon starting a new iteration in the
optimization loop
        /// <summary>
        /// <param name="iteration">the iteration that is starting</param>
        /// <returns> whether we should continue on this iteration or
not</returns>
        public bool StartIteration(int iteration)
        {
            return false;
        }
```

```
        /// <summary>
        ///     Callback called upon ending an iteration   (optional)
        /// <summary>
        /// <param name="iteration">the iteration that is ending</param>
        public void EndIteration(int iteration)
        {
        }
        /// <summary>
        ///     Called to return the new value to set for this parameter
        /// <summary>
        public double GetNewValue(string parameterId)
        {
            return 0;
        }
        /// <summary>
        ///     Called to register a new variable as input
        /// <summary>
        public void AddVariable(string parameterId, string parameterUserName,
 double startingValue, double min, double max)
        {
        }
        /// <summary>
        ///    Called to register a new target as output (optional)
        /// <summary>
        public void AddTarget(string parameterId, string parameterUserName,
double startingValue, double targetValue, double weight)
        {
        }
        /// <summary>
        ///     Callback called after the simulation's update to update the
values of the targets
        /// <summary>
        /// <param name="parameterId"/>
        /// <param name="value"/>
        public void SetMeasures(string parameterId, double value)
        {
        }
        /// <summary>
        ///     Callback called to inform the simulation details
        /// <summary>
        /// <param name="simulationName"/>
        /// <param name="reportPath"/>
       public void SetSimulationInfos(string simulationName, string reportPath)

        {
        }
    }
}
```

15.2.8.3.2. Reading Input Arguments for the Incrementation Value and Iteration Number

1. Reading input arguments for the increment value and the number of iteration
2. Registering the variables to increment
3. Registering the targets to have in the report
4. Implementing the incrementation of the variables
5. Implementing the reception of the new measures

6. Implementing the check to finish the iteration loop
7. Outputting a report file

The plugin must implement a constructor that takes a string for arguments. For the example, a simple comma separated list is used to keep everything simple.

This code provides you with the arguments of your optimization.

To do this, you have to change the scaffolding constructor to have the following:

```
private double _incrementValue = 0;
private int _iterationNumber = 0;

public PluginSample(string arguments)
{
    // splint the list of parameters
    var argumentsList=  arguments.Split(',');

    // check that the argument list represents the number of arguments that
we need
    if(argumentsList.Length != 2)
    {
        throw new ArgumentException("We should only have two arguments");
    }

    // try to read the first argument as the increment value
    if (double.TryParse(argumentsList[0], out var incrementValue))
    {
        _incrementValue = incrementValue;
    }
    else
    {
       throw new ArgumentException("first argument cannot be read as a double
 for increment value");
    }

    // try the read the second argument as the iteration number
    if (int.TryParse(argumentsList[1], out var iterationNumber))
    {
        _iterationNumber = iterationNumber;
    }
    else
    {
        throw new ArgumentException("second argument cannot be read as a int
for iteration number");
    }
}
```

15.2.8.3.3. Registering the Variables to Increment

1. Reading input arguments for the increment value and the number of iteration
2. Registering the variables to increment
3. Registering the targets to have in the report
4. Implementing the incrementation of the variables
5. Implementing the reception of the new measures
6. Implementing the check to finish the iteration loop
7. Outputting a report file

Then, you need to receive the variables you care about:

Note: The `parameterId` is an unique string that identify the parameter, it has no other purpose.

Note: The `min` and `max` parameters are not used in the sample, but they represent the values displayed in the definition panel, if you need them in your implementation.

```
// dictionary containing a map from a parameter unique id and its display name,
 for readability in a report
private IDictionarystring<string, string> _variableNames = new
Dictionarystring<string, string>();
// dictionary containing a map from a parameter unique id and its current
value, to send to Speos
private IDictionarystring<string, double> _variableValues = new
Dictionary<string, double>();

//.... it is recommended for cleaner reading to leave the declaration of fields
 at the top of the file

/// <summary>
///     Called to register a new variable as input
/// </summary>
public void AddVariable(string parameterId, string parameterUserName, double
startingValue, double min, double max)
{
    // add the parameter id to the two dictionaries
    _variableNames.Add(parameterId, parameterUserName);
    _variableValues.Add(parameterId, startingValue);
}
```

15.2.8.3.4. Registering the Targets to Have in the Report

1. Reading input arguments for the increment value and the number of iteration
2. Registering the variables to increment
3. Registering the targets to have in the report
4. Implementing the incrementation of the variables
5. Implementing the reception of the new measures
6. Implementing the check to finish the iteration loop
7. Outputting a report file

Now, as the variables, you need to register the targets:

Note: `weight` is not in the sample, but this is the same value that appears in the definition panel.

```
private IDictionary<string, string> _targetNames = new Dictionary<string,
string>();
private IDictionary<string, double> _targetValues = new Dictionary<string,
double>();

//.... it is recommended for cleaner reading to leave the declaration of fields
 at the top of the file
```

```
/// <summary>
///    Called to register a new target as output (optional)
/// </summary>
public void AddTarget(string parameterId, string parameterUserName, double
startingValue, double targetValue, double weight)
{
    _targetNames.Add(parameterId, parameterUserName);
    _targetValues.Add(parameterId, startingValue);
```

15.2.8.3.5. Implementing the Incrementation of the Variables

1. Reading input arguments for the increment value and the number of iteration
2. Registering the variables to increment
3. Registering the targets to have in the report
4. Implementing the incrementation of the variables
5. Implementing the reception of the new measures
6. Implementing the check to finish the iteration loop
7. Outputting a report file

At the beginning of the iteration (in the `StartIteration()` method) you modify the value in the `_variableValues` dictionary.

Then, you need to return this value for the parameter id when the asked with the `GetNewValue()` method.

```
/// <summary>
///     Callback called upon starting a new iteration in the optimization loop
/// </summary>
/// <param name="iteration">the iteration that is starting</param>
/// <returns> whether we should continue on this iteration or not</returns>
public bool StartIteration(int iteration)
{
    foreach(var key in _variableValues.Keys)
    {
        _variableValues[key] = _variableValues[key] + _incrementValue;
    }

    return true; // we want to continue looping on this iteration
}

// ....

/// <summary>
///     Called to return the new value to set for this parameter
/// </summary>
public double GetNewValue(string parameterId)
{
    return _variableValues[parameterId];
}
```

15.2.8.3.6. Implementing the Reception of the New Measures

1. Reading input arguments for the increment value and the number of iteration
2. Registering the variables to increment
3. Registering the targets to have in the report

4. Implementing the incrementation of the variables
5. Implementing the reception of the new measures
6. Implementing the check to finish the iteration loop
7. Outputting a report file

Here, you need to set the new value of the target and save them to create the report.

```
// list of the values for each iteration
private IList<IDictionary<string,double>> _iterationResults = new
List<IDictionary<string,double>>();

/// ....

/// <summary>
///     Callback called upon ending an iteration   (optional)
/// </summary>
/// <param name="iteration">the iteration that is ending</param>
public void EndIteration(int iteration)
{
    // copying the list of values for this iteration
    _iterationResults.Add(_targetValues.ToDictionary(entry => entry.Key, entry
 => entry.Value));
}

/// ....

/// <summary>
///     Callback called after the simulation's update to update the values of
 the targets
/// </summary>
/// <param name="parameterId"></param>
/// <param name="value"></param>
public void SetMeasures(string parameterId, double value)
{
    _targetValues[parameterId] = value;
}
```

15.2.8.3.7. Implementing the Check to Finish the Iteration Loop

1. Reading input arguments for the increment value and the number of iteration
2. Registering the variables to increment
3. Registering the targets to have in the report
4. Implementing the incrementation of the variables
5. Implementing the reception of the new measures
6. Implementing the check to finish the iteration loop
7. Outputting a report file

The way it is currently setup, this will loop infinitely because the implementation of `StartIteration` returns true all the time.

Therefore, you need to check at the beginning if you want to end or not the iterations.

```
/// <summary>
///     Callback called upon starting a new iteration in the optimization loop
/// </summary>
/// <param name="iteration">the iteration that is starting</param>
/// <returns> whether we should continue on this iteration or not</returns>
```

```
public bool StartIteration(int iteration)
{
    // Start : code added to end the loop on the wanted iteration
    if(iteration >= _iterationNumber)
    {
        return false;
    }
    // End : code added to end the loop on the wanted iteration
    foreach(var key in _variableValues.Keys)
    {
        _variableValues[key] = _variableValues[key] + _incrementValue;
    }

    return true; // we want to continue looping on this iteration
}
```

15.2.8.3.8. Outputting a Report File

1. Reading input arguments for the increment value and the number of iteration
2. Registering the variables to increment
3. Registering the targets to have in the report
4. Implementing the incrementation of the variables
5. Implementing the reception of the new measures
6. Implementing the check to finish the iteration loop
7. Outputting a report file

Retrieving the Report File

First, you need to retrieve the report file by implementing the `SetSimulationInfos()` method.

> **Note:** The report is a HTML file report. However, you can write in other files, but they will not be displayed in the Speos tree.

```
private string _simulationName = string.Empty;
private string _reportPath = string.Empty;

// ...

/// <summary>
///     Callback called to inform the simulation details
/// </summary>
/// <param name="simulationName"></param>
/// <param name="reportPath"></param>
public void SetSimulationInfos(string simulationName, string reportPath)
{
    reportPath = reportPath ?? string.Empty;
    simulationName= simulationName ?? string.Empty;
}
```

Creating the WriteReport method to Write the Report

Then, you need to create a new method named `WriteReport` to write the report.

The `WriteReport` method creates an html table with the values of the targets for each iteration.

```
private void WriteReport()
{
    var stringBuilder = new StringBuilder();
    stringBuilder.AppendLine("<html><body><table>");

    stringBuilder.AppendLine("<tr><th>Iteration Number</th>");

    var parameterIdOrder = new string[_targetNames.Count];

    var idx = 0;
    foreach(var names in _targetNames)
    {
        stringBuilder.AppendLine($"<th>{names.Value}</th>");
        parameterIdOrder[idx] = names.Key;
        idx++;
    }
    stringBuilder.AppendLine("<tr/>");

    idx = 0;
    foreach (var iterationResult in _iterationResults)
    {
        stringBuilder.AppendLine("<tr>");
        stringBuilder.AppendLine($"<td>{idx}</td>");
        foreach(var id in parameterIdOrder)
        {
            stringBuilder.AppendLine($"<td>{iterationResult[id]}</td>");
        }
        stringBuilder.AppendLine("<tr/>");
        idx++;
    }

    stringBuilder.AppendLine("</table></body></html>");

    File.WriteAllText(_reportPath, stringBuilder.ToString());
}
```

Calling the WriteReport method Before Stopping the Iteration Loop

Finally, you need to call the `WriteReport` before stopping the iteration loop. Therefore you call it before returning false in the `StartIteration` method.

```
/// <summary>
///     Callback called upon starting a new iteration in the optimization loop
/// </summary>
/// <param name="iteration">the iteration that is starting</param>
/// <returns> whether we should continue on this iteration or not</returns>
public bool StartIteration(int iteration)
{
    if(iteration >= _iterationNumber)
    {
        // Start : new line to write the report
        WriteReport();
        // End : new line to write the report
        return false;
    }
    foreach(var key in _variableValues.Keys)
    {
        _variableValues[key] = _variableValues[key] + _incrementValue;
```

```
    }

    return true; // we want to continue looping on this iteration
}
```

15.2.8.3.9. Optimization Plugin Complete

The following code corresponds to the full optimization plugin example written based on the scaffolding from the Optimization Plugin Template page.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

namespace OptimPluginSample
{
    public class PluginSample
    {
        private double _incrementValue = 0;
        private int _iterationNumber = 0;

        private IDictionary<string, string> _variableNames = new
Dictionary<string, string>();
        private IDictionary<string, double> _variableValues = new
Dictionary<string, double>();

        private IDictionary<string, string> _targetNames = new
Dictionary<string, string>();
        private IDictionary<string, double> _targetValues = new
Dictionary<string, double>();

        private IList<IDictionary<string,double>> _iterationResults = new
List<IDictionary<string,double>>();

        private string _simulationName = string.Empty;
        private string _reportPath = string.Empty;

        public PluginSample(string arguments)
        {
            // splint the list of parameters
            var argumentsList=  arguments.Split(',');

           // check that the argument list represents the number of arguments
 that we need
            if(argumentsList.Length != 2)
            {
                throw new ArgumentException("We should only have two
arguments");
            }

            // try to read the first argument as the increment value
            if (double.TryParse(argumentsList[0], out var incrementValue))
            {
                _incrementValue = incrementValue;
            }
            else
```

```
            {
                throw new ArgumentException("first argument cannot be read as
 a double for increment value");
            }

            // try the read the second argument as the iteration number
            if (int.TryParse(argumentsList[1], out var iterationNumber))
            {
                _iterationNumber = iterationNumber;
            }
            else
            {
                throw new ArgumentException("second argument cannot be read
as a int for iteration number");
            }
        }

        /// <summary>
        ///     Callback called upon starting the optimization loop (optional)

        /// </summary>
        public void StartRun()
        {

        }

        /// <summary>
        ///     Callback called upon starting a new iteration in the
optimization loop
        /// </summary>
        /// <param name="iteration">the iteration that is starting</param>
        /// <returns> whether we should continue on this iteration or
not</returns>
        public bool StartIteration(int iteration)
        {
            if(iteration >= _iterationNumber)
            {
                WriteReport();
                return false;
            }
            var keys = _variableValues.Keys.ToArray();
            foreach (var key in keys)
            {
               _variableValues[key] = _variableValues[key] + _incrementValue;

            }

            return true; // we want to continue looping on this iteration
        }
        /// <summary>
        ///     Callback called upon ending an iteration   (optional)
        /// </summary>
        /// <param name="iteration">the iteration that is ending</param>
        public void EndIteration(int iteration)
        {
          _iterationResults.Add(_targetValues.ToDictionary(entry => entry.Key,
 entry => entry.Value));
        }

        /// <summary>
        ///     Called to return the new value to set for this parameter
```

```
        /// </summary>
        public double GetNewValue(string parameterId)
        {
            return _variableValues[parameterId];
        }

        /// <summary>
        ///     Called to register a new variable as input
        /// </summary>
        public void AddVariable(string parameterId, string parameterUserName,
 double startingValue, double min, double max)
        {
            _variableNames.Add(parameterId, parameterUserName);
            _variableValues.Add(parameterId, startingValue);
        }

        /// <summary>
        ///    Called to register a new target as output (optional)
        /// </summary>
        public void AddTarget(string parameterId, string parameterUserName,
double startingValue, double targetValue, double weight)
        {
            _targetNames.Add(parameterId, parameterUserName);
            _targetValues.Add(parameterId, startingValue);
        }

        /// <summary>
        ///     Callback called after the simulation's update to update the
values of the targets
        /// </summary>
        /// <param name="parameterId"></param>
        /// <param name="value"></param>
        public void SetMeasures(string parameterId, double value)
        {
            _targetValues[parameterId] = value;
        }

        /// <summary>
        ///     Callback called to inform the simulation details
        /// </summary>
        /// <param name="simulationName"></param>
        /// <param name="reportPath"></param>
       public void SetSimulationInfos(string simulationName, string reportPath)

        {
            _reportPath = reportPath ?? string.Empty;
            _simulationName= simulationName ?? string.Empty;
        }

        private void WriteReport()
        {
            var stringBuilder = new StringBuilder();
            stringBuilder.AppendLine("<html><body><table>");

            stringBuilder.AppendLine("<tr><th>Iteration Number</th>");

            var parameterIdOrder = new string[_targetNames.Count];

            var idx = 0;
            foreach(var names in _targetNames)
            {
```

```
                stringBuilder.AppendLine($"<th>{names.Value}</th>");
                parameterIdOrder[idx] = names.Key;
                idx++;
            }
            stringBuilder.AppendLine("<tr/>");

            idx = 0;
            foreach (var iterationResult in _iterationResults)
            {
                stringBuilder.AppendLine("<tr>");
                stringBuilder.AppendLine($"<td>{idx}</td>");
                foreach(var id in parameterIdOrder)
                {
                 stringBuilder.AppendLine($"<td>{iterationResult[id]}</td>");

                }
                stringBuilder.AppendLine("<tr/>");
                idx++;
            }

            stringBuilder.AppendLine("</table></body></html>");

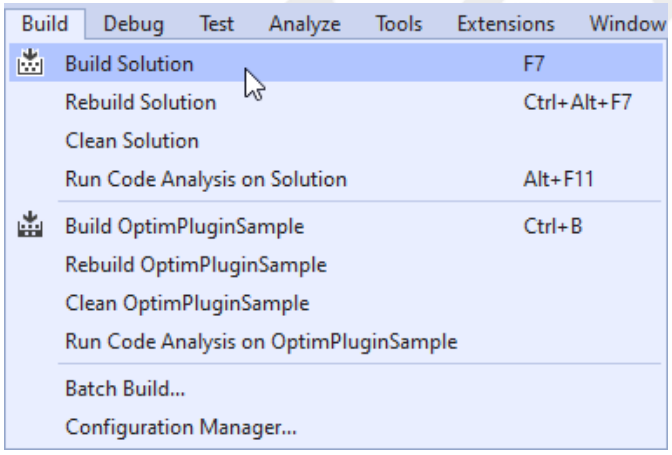
            File.WriteAllText(_reportPath, stringBuilder.ToString());
        }

    }
}
```

15.2.8.4. Compiling the Project with Visual Studio

The following procedure helps you compile the project in Visual Studio once you are done writing the optimization plugin.

In Visual Studio, in the main ribbon, select **Build** > **Build Solution**



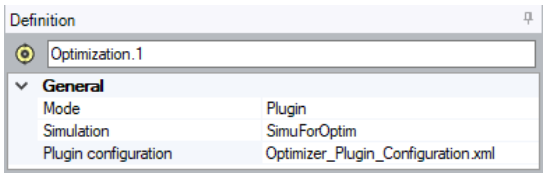
The *.dll is created in the folder you specified when creating the project.

15.2.8.5. Configuring the XML Optimizer Plugin Configuration File

The following page helps you configure the XML file to provide to the Speos Optimization feature that will give the information on which optimization plugin to use.

Optimization Feature Definition

The XML configuration file is to be provided in the **Plugin configuration** field of the **Optimization** feature definition.



XML File Format

```
<OptimizerPluginConfig>
    <!-- the path to the dll file -->
    <PluginPath>[path to you dll]/OptimPluginSample.dll</PluginPath>
    <!-- the name of the class that you created, including namespace -->
    <PluginClass>OptimPluginSample.PluginSample</PluginClass>
    <!-- the string that will be passed to your class constructor -->
    <Arguments>4.2,3</Arguments>
</OptimizerPluginConfig>
```

- The **PluginPath** corresponds to the full path to your *.dll file.
- The **PluginClass** corresponds to the full name of the class, including the namespace: [namespace].[class name]
- The **Arguments** correspond to the string that is passed to the constructor of your class.

  Arguments can input other formats that you want to read. For more information, refer to the **Arguments** section below.

Arguments

CSV Data

As in the plugin example, you can use CSV line data:

```
<OptimizerPluginConfig>
    <PluginPath>...</PluginPath>
    <PluginClass>...</PluginClass>
    <Arguments>value1,value2,value3</Arguments>
</OptimizerPluginConfig>
```

XML String

If you want to have a XML string as input, you must integrate it inside a CDATA tag:

```
<OptimizerPluginConfig>
    <PluginPath>...</PluginPath>
    <PluginClass>...</PluginClass>
    <Arguments><![CDATA[<test><test2>value<test2/><test3>other
value<test3/><test/>]]></Arguments>
</OptimizerPluginConfig>
```

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

JSON

You can apply JSON:

```
<OptimizerPluginConfig>
    <PluginPath>...</PluginPath>
    <PluginClass>...</PluginClass>
    <Arguments>{"test2": "value", "test3": "other value"}</Arguments>
</OptimizerPluginConfig>
```

15.3. Optimization with optiSLang

The following section presents a brief introduction of optiSLang and the integration of Speos in optiSLang.

15.3.1. Speos in optiSLang

This page quickly introduces optiSLang, a Process Integration and Design Optimization software from which Speos can benefit to optimize an optical system.

> **Note:** If you need more information on optiSLang functioning and behavior, refer to **optiSLang User's Guide**.

optiSLang

optiSLang is a software platform for Computer-Aided Engineering-based (CAE) optimization in virtual prototyping. Based on design variations or measurement and observation points, you can perform efficient variation analyses with minimal user input and few solver calls.

It supports you with:

- Calibration of virtual models to physical tests
- Analysis of parameter sensitivity and importance
- Metamodeling
- Optimization of product performance
- Quantification of product robustness and reliability also referred as to Uncertainty Quantification (UQ)
- Robust Design Optimization (RDO) also referred as to Design for Six Sigma (DFSS)

optiSLang also includes a powerful simulation workflow environment. The software is the perfect tool for simulation driven workflow generation using parametric models for sensitivity analysis, optimization, and robustness evaluation.

- Sensitivity analysis helps you understand the design, focus on key parameters, check your response variation’s forecast quality and automatically generate your optimum metamodel.
- Optimization helps improve your design performance.
- Robustness evaluation helps you verify the design robustness regarding scattering material parameters, production tolerances and varying environmental conditions.

DRAFT

Speos Design Optimization inside optiSLang

Presentation

From the version 2023 R2, a wizard-based Speos Integration has been integrated into Ansys optiSLang.

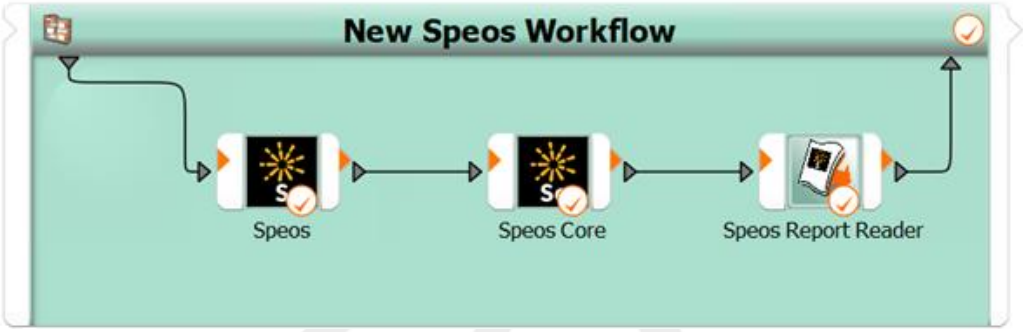
The **Solver wizard** helps you easily:

- connect to your Speos project
- define the parameters and parameter ranges for the variation analysis
- define the Speos simulations to be exported and solved
- create the workflow for an automated Speos simulation.

The automated Speos workflow consists of 3 nodes:

- the **Ansys Speos** node (input node), that updates the geometry based on the parameter values and export the Speos simulation file
- the **Ansys Speos Core** node (solver node), that launches and processes the simulation
- the **Ansys Speos report reader** node (output node), that extract response values from the Speos simulation report

Once you finished setting the Speos workflow, you can define the criteria in the parametric system and follow up to set up the variation analysis (example: sensitivity analysis, optimization, etc.) using the available wizards.



Generic Workflow

1. In Speos define the parameters to use in optiSLang using **Publish Parameters**.
2. In optiSLang, drag the **Solver Wizard** and drop it to the Scenery.
3. Select the Ansys Speos solver paradigma and open a *.scdocx project.
4. Define the **Ansys Speos** solver node.
5. Define the **Ansys Speos Core** solver node.
6. Define the **Ansys Speos Report Reader** solver node.

15.3.2. Defining the Speos Parameters to Use in optiSLang

The following procedure helps you select the parameters that you want to use in optiSLang for your design optimization.

To define the parameters:

1. In Speos, in your *.scdocx project, select the **Workbench** tab.

2. Click **Publish Parameters** 

.
3. Select an Optical Part Design feature or a Light Simulation feature (Material, Source, Sensor, Simulation). In the Publish Parameters panel, a list of the possible parameters you can use in optiSLang appears.

Publish Parameters

Publish	Name	Parent	Value
☐	Global/Refractive index	Freeform Lens.1	1.49
☑	Global/Minimum thickness	Freeform Lens.1	2
☐	Target/Intensity file (beta)	Freeform Lens.1	.\SPEOS input files\Light.ies
☐	Target/Contrast ratio (beta)	Freeform Lens.1	5
☐	Target/Resolution factor (beta)	Freeform Lens.1	2
☐	Target/Maximum threshold (beta)	Freeform Lens.1	10

4. Check the parameters to be used in optiSLang.
5. Save your *.scdocx project.
6. Keep Speos opened.

Once you selected the parameters, you can open optiSLang to integrate a Speos solver node.

DRAFT

16: Automation

16.1. What is Automation

This page provides a brief overview of automation in Speos.

General Description

Automation refers to the process of generating tools meant to automate the execution of tasks.

You can use Automation as follows:

- to generate and define any Speos object.
- to post process simulation results.

Technical Description

The interface is described using the IDL (Interface Description Language).

The exposed data are called Properties.

The exposed functions are called Methods.

Development Tools

If you want to use Automation, you need to use IronPython Script language.

You can use the Speos built-in **Script Editor** to create or edit scripts.

16.2. Methodology

This page helps you understand the Speos API methodology in order to help you generate automation scripts.

Presentation

Speos APIs are based on the Speos user interface. This means that for any feature/item currently available in the GUI, an associated automation function is available. As the automation functions are derived from the GUI, they are completely aligned with the actions that you would have to perform in the software.

Three interfaces are available to declare the API functions:

- **SpeosSim** allows you to access all the Light Simulation features
- **SpeosDes** allows you to access all the Optical Part Design features
- **SpeosAsm** allows you to access the geometry update features

Interfaces are systematically called on feature creation.

```
#Interface declaration
3DTexture = SpeoSim.Component3DTexture.Create()
OpticalSurface = SpeosDes.TIRLens.Create()
```

DRAFT

HTML Resource File

All currently available APIs are described in the Speos API Docs. This file provides you with a description and the syntax of the methods name. Several script examples are provided in the file.

The HTML file provides a modular presentation of the Speos API functions:

- The first column provides you with the **method** and **property** name. The method/property hierarchy is simply expressed in the file structure. The name of a section refers to a "parent" method (i.e. a feature level). All dependent methods are listed below it.
- The second column provides you with the method/property **Description**.
- The third column provides the method's **Syntax**. The syntax corresponds to a template for the use of the methods with its return value and arguments.
  - The return value is used as an intermediary step in the method calculation. These values are, therefore, both indicating which value is expected as a result of the function and what type of input is expected by the method.
  - Arguments are always declared with parenthesis () and correspond to variables that must be provided to obtain the method's result.

    In Speos, arguments often correspond to axis system definitions.

In addition to the HTML resource file, some common cross functional and more specific APIs are provided in the **Methods** section.

Related reference

Methods on page 818
The following section gathers cross-functional methods and specific methods that are not covered in the Speos API Docs.

Related information

Creating a Script on page 816
This page shows how to create a script group in Speos. A script group has to be created to use Speos APIs.

16.3. Creating a Script

This page shows how to create a script group in Speos. A script group has to be created to use Speos APIs.

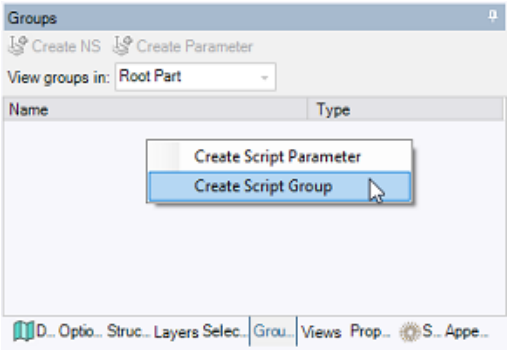
To create a Script Group:

1. Access the **Groups** panel.
2. Right-click in the panel and click **Create Script Group**.

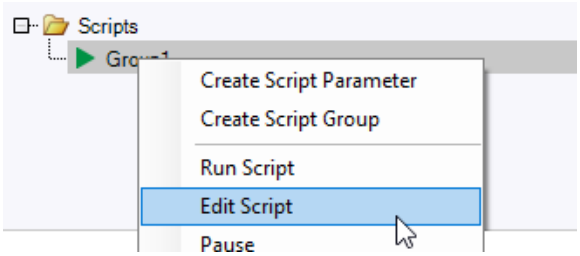
   The Group appears in the panel under the Scripts folder.

> **Note:** Scripts Groups can be created on the root part or on the active part level.

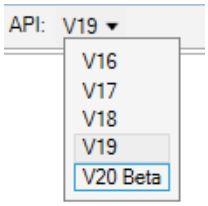
DRAFT



3. Right-click the **Group** and click **Edit Script** top open the built-in script editor.



4. From the **Script Editor**, if you want to use the new SpaceClaim methods available, select the latest SpaceClaim API version available.



> **Note:** Do not confuse the SpaceClaim API version and the Speos API version. Unlike the SpaceClaim API version, the Speos API version is always the latest version available.

> **Note:** If your script version is not the latest version available, and you want to use geometry objects retrieved from a Speos object selection attribute (i.e. Items, LinkedObjects), you need to convert the retrieved geometry objects thanks to the method ConvertToScriptVersion.

The Script Group is created and ready to be used. You can now load an existing script or create a script from scratch using Speos APIs.

> **Note:** When you run a script on SpaceClaim in headless mode (without User Interface), the rendering calculations are performed in all cases.
>
> If you do not want to perform the rendering calculations in headless mode, create the environment variable **SPEOS_DISABLE_RENDERING_WHEN_HEADLESS**, and set it to **1**.

DRAFT

Related reference

Methodology on page 815
This page helps you understand the Speos API methodology in order to help you generate automation scripts.

Methods on page 818
The following section gathers cross-functional methods and specific methods that are not covered in the Speos API Docs.

16.4. Forbidden Values Management

The following page presents you how Speos handles the forbidden values when they are defined in scripts.

Forbidden values correspond to values that do not meet the Speos standards.

Out of Range Values

If a value set in the script is out of the standard range in which it should be, then the previous value is defined.

Example

```
material = SpeosSim.Material.Create() (reflectance =100)
material.SOPReflectance =50; //ok
material.SOPReflectance =150; // keeps the previous value since it should be
lower than 100
```

Decimal Values

If a value set in the script uses decimals whereas it only allows integer, then the value is rounded up or down to the nearest whole number.

Example

```
material = SpeosSim.Material.Create()
material.SOPReflectance =15,4; // set the value to 15 since it should be an
integer
```

16.5. Methods

The following section gathers cross-functional methods and specific methods that are not covered in the Speos API Docs.

> **Note:** The Speos Simulation and Design objects currently do not correctly take into account dimension functions (such as "MM()"). Specify the dimension, not the unit. (e.g. use irradiance_x=5 instead of irradiance_x=MM(5)).

DRAFT

16.5.1. Generic Methods

This section gathers standard functions that can be used across almost any Speos feature and that are not listed in the API resource file.

16.5.1.1. Common Methods

Common methods are cross-functional standard functions.

- **Create** creates a new feature: `newMaterial = SpeosSim.Material.Create()`
- **Find** looks for a custom object by its name and directly gets the corresponding object in the active component: `existingMaterial = SpeosSim.Material.Find("Plastic")`
- **Clone** copies the object: `copiedMaterial = existingMaterial.Clone()`
- **Delete** deletes the oject: `copiedMaterial.Delete()`
- **Name** gets or modifies the name of the feature: `plasticMaterial.Name = "Plastic"`
- **Visible** gets or modifies the visibility of the feature: `existingSource.Visible = False`
- **Subject** returns the underlying CustomObject: `Selection.Create(existingMaterial.Subject)`
- **StatusInfo** returns the warnings or error messages associated to the feature. If there are no errors, the returned string is empty: `print inverseSimulation.StatusInfo`
- **PublishAllParameterstoWorkbench** exposes the attributes of Speos elements so that Ansys Workbench can access and drive them: `existingSource.PublishAllParametersToWorkbench(True)`
- **Compute** is used to trigger the generation of any object: `inverseSimulation.Compute()`

> **Note:** Computation is not automatic on features during the script process as Compute events are executed at the end of the "Script" Command. So if you want your features to be up to date, you need to call explicitly the Compute() method on objects that are usually supposed to update automatically in the interactive session.

16.5.1.2. Set and Get Methods

Setting and getting values can be easily done, mainly though the `set` and `print` functions. Most values of parameters are numeric, strings or booleans.

- **print** gets the value of a specific parameter: `print irradianceSensor.YResolution`
- Setting values is done by assigning a value, string or boolean to a Speos object.

```
#Example of irradiance sensor definition
irradianceSensor = SpeosSim.SensorIrradiance.Create()
print irradianceSensor.XStart
print irradianceSensor.XEnd
print irradianceSensor.XResolution
irradianceSensor.XIsMirrored = True
irradianceSensor.XEnd = 15
irradianceSensor.XResolution = 500
print irradianceSensor.YStart
print irradianceSensor.YEnd
print irradianceSensor.YResolution
irradianceSensor.YIsMirrored = True
irradianceSensor.YEnd = 15
irradianceSensor.YResolution = 500
```

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

```
print irradianceSensor.XMPTemplateFile
irradianceSensor.XMPTemplateFile = ".\\SPEOS input file\\xmpTemplate.xml"
```

16.5.1.3. List (Enum) Parameters

Parameters that are defined though a choice list (usually a **Type** list in the Speos interface, can be changed using their corresponding `enum`. For instance, there are three different types of volume properties (opaque, optic and library) and three types of surface properties (mirror, optical polished, library) in the Material definition:

```
#Example of the use of enum to define optical properties
newMaterial = SpeosSim.Material.Create()
newMaterial.Name = "Plastic"
newMaterial.VOPType = SpeosSim.Material.EnumVOPType.Opaque
newMaterial.SOPType = SpeosSim.Material.EnumSOPType.Library
newMaterial.SOPLibrary = ".\\SPEOS input files\\Plastic.simplescattering"
```

For lists that are not predefined, for example the possible locations of a Natural Light Ambient Source, the complete list of `enums` can be obtained though a `Get` method and parsed to find the useful value.

```
naturalLight = SpeosSim.SourceAmbientNaturalLight.Create()
locations = naturalLight.GetLocationPossibleValues()
locationLabel = ""
for locationLabel in locations:
 print locationLabel
 if locationLabel.Contains("Boston"):
 break
naturalLight.Location = locationLabel
```

16.5.1.4. Geometry Selection Methods

Geometry selections are required for almost any Speos object definition. They can be performed using the SpaceClaim selection.

- Generic object selection can be performed using the SpaceClaim

```
Geometries can be set using the SpaceClaim Selection:
geometry1 = GetRootPart().Bodies[3]
geometry2 = GetRootPart().Bodies[8]
geometry3 = GetRootPart().Bodies[10]
newMaterial.VolumeGeometries.Set(Selection.Create([geometry1, geometry2,
geometry3]))
```

```
Selection.Create(GetRootPart().Bodies[3]).SetActive()
Selection.Create(GetRootPart().Bodies[8]).AddToActive()
Selection.Create(GetRootPart().Bodies[10]).AddToActive()
newMaterial.VolumeGeometries.Set(Selection.GetActive())
```

- Origin or directions selection

```
displaySource = SpeosSim.SourceDisplay.Create()
displayOrigin = Selection.Create(GetRootPart().CoordinateSystems[0])
displaySource.OriginPoint.Set(displayOrigin)

displayXDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[0])
```

```
displaySource.XDirection.Set(displayXDirection)

displayYDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[1])
displaySource.YDirection.Set(displayYDirection)
```

- Oriented faces selection (used in surface sources or FOPs to orientate the normal to the selected faces)

```
orientedFace = SpeosSim.OrientedFace.Create()
emissiveFace = Selection.Create(GetRootPart().Bodies[7].Faces[0])
orientedFace.Face.Set(emissiveFace)
orientedFace.ReverseDirection = True

surfaceSource = SpeosSim.SourceSurface.Create()
surfaceSource.EmissiveFaces.Add(orientedFace)
```

16.5.1.5. ConvertToScriptVersion

Description

This function allows you to convert retrieved geometry objects, from a Speos object selection attribute (i.e. Items, LinkedObjects), from the latest Speos API version to the anterior script version that you want to edit.

Python Definition

```
import SpaceClaim.Api.V20 as scriptNameSpace

def ConvertToScriptVersion(obj):
    doc = Window.ActiveWindow.Document
    res =
scriptNameSpace.Moniker[scriptNameSpace.IDocObject].FromString(obj.Moniker.ToString()).Resolve(doc)

    return res
```

Example

```
docObjInScriptVersion = ConvertToScriptVersion( docObjInSpeosVersion )
print docObjInScriptVersion # => V20 => OK
```

16.5.2. Specific Methods

This section gathers specific methods that are not covered in the API resource file.

16.5.2.1. Light Expert Methods

When Light Expert is enabled in the definition of simulations, the LXP flag can be enabled on particular sensors.

This action is done handling LXPEnabledSensor object:

```
inverseSimulation = SIM.SimulationInverse.Create()
lxpSensor = SpeosSim.LXPEnabledSensor.Create()
```

```
lxpSensor.LXPSensor.Set(radianceSensor)
lxpSensor.IsLXP = True
inverseSimulation.Sensors.Add(lxpSensor)
```

16.5.2.2. Simulation Methods

Simulation methods can be used for every type of simulation available in Speos.

- **Export**exports the simulation: `directSimulation.Export("C:\\exportedSimulation\\")`
- **Isolate** creates an isolated simulation feature: `directSimulation.Isolate()`
- **LinkedExport** creates a linked export simulation: `directSimulation.LinkedExport()`
- **GetResultFilePaths** returns the list of results generated by the simulation:

```
results = directSimulation.GetResultFilePaths()
for resultFile in results:
    print resultFile
```

```
copiedMaterial.Delete()
```

- **GetSimulationSettings** returns a SimulationSettings object that allows you to modify geometry and simulation settings.
- **SetSimulationSettings** applies the changes made with the **GetSimulationSettings** function.

```
simulationSettings = interactiveSimulation.GetSimulationSettings()
simulationSettings.MeshingStepValue = 10
interactiveSimulation.SetSimulationSettings(simulationSettings)
```

- **SelectAll** selects all the related features of a same category (source, sensor or geometry).

```
directSimulation.Sources.SelectAll
directSimulation.Geometries.SelectAll
directSimulation.Sensors.SelectAll
```

16.5.2.3. Simulation Settings

Interactive Simulation Settings

- **GetInteractiveSimulationSettings** returns three Booleans:
  - DrawRays
  - DrawImpacts
  - ReportImpact
- **SetInteractiveSimulationSettings** uses those three Booleans as parameters.

```
simulationSettings = interactiveSimulation.GetInteractiveSimulationSettings(
)
for simulationSetting in simulationSettings:
 print simulationSetting
interactiveSimulation.SetInteractiveSimulationSettings(True, True, False)
```

DRAFT

Direct Simulation Settings

- **GetDirectSimulationSettings** returns an **DirectSimulationSettings** object that allows you to modify simulation settings.
- The **DirectSimulationSettings** object includes four main methods:
  - SetPropagationErrorAnalysis(*bool propagationErrorAnalysis*)
  - SetDispersion(*bool dispersion*)
  - SetFastTransmissionGathering(*bool fastTransmissionGathering*)
  - SetAutomaticSaveFrequency(*int directTemporarySave*)
- **SetDirectSimulationSettings** applies the changes made with the **GetDirectSimulationSettings** function.

```
directSetting = simuDir.GetDirectSimulationSettings()
directSetting.SetPropagationErrorAnalysis(False)
directSetting.SetDispersion(True)
directSetting.SetFastTransmissionGathering(False)
directSetting.SetAutomaticSaveFrequency(1800)
simuDir.SetDirectSimulationSettings(directSetting)
```

Inverse Simulation Settings

- **GetInverseSimulationSettings** returns an **InverseSimulationSettings** object that allows you to modify geometry and simulation settings.
- The **InverseSimulationSettings** object includes five main methods:
  - **SetMonteCarlo** with the parameters

    bool dispersion
    bool splitting
    int nbGatheringRaysPerSource
    int maxGatheringError
    bool fastTransmissionGathering
    int automaticSaveFrequencySeconds
    EnumOptimizedPropagationMode
    optimizedPropagation
    int nbStandardPassesBeforeOptimizedPasses, optional
  - **SetDeterminist** with the parameters:

    EnumPhotonMapMode
    photonMapMode
    int ambientSampling
    int maxNbSurfaceInteractions
    bool antiAliasing
    int specularApproxAngle
  - **SetDeterministPhotonMap** with the parameters:

    int maxNeighbors
    int maxSearchRadius
    bool fastTransmissionGathering
    bool useFinalGathering
    int finalGatheringMaxNeighbors, optional

DRAFT

int finalGatheringSplittingNb, optional

- **SetDeterministPhotonMapBuild** with the parameters:

  int nbPhotonsInDirectPhase
  int nbSurfaceInteractionsInDirectPhase

- **SetPropagationErrorAnalysis(*bool propagationErrorAnalysis*)**

- **SetInverseSimulationSettings** applies the changes made with the **GetInverseSimulationSettings** function.

```
settings = inverseSimulation2.GetInverseSimulationSettings()
settings.SetDeterminist(SIM.InverseSimulationSettings.EnumPhotonMapMode.Load,
 100, 10, False, 0)
settings.SetDeterministPhotonMapBuild(10000,100)
settings.SetDeterministPhotonMap(100, 10000000, True, False)
settings.SetPropagationErrorAnalysis(True)
inverseSimulation2.SetInverseSimulationSettings(settings)
```

16.5.2.4. Command Methods

Command methods give access to the tool functions, like Compute or Preview commands for example.

- **Compute** launches the compute command on the specified objects:
  `SpeosSim.Command.Compute(directSimulation)`
- **ComputeOnActiveSelection** launches the compute command on the selected objects:

```
selection = Selection.Create([interactiveSimulation.Subject,
directSimulation.Subject])
SpeosSim.Command.SetActiveSelection(selection)
SpeosSim.Command.ComputeOnActiveSelection()
```

- **GetInputFolder** returns the path to the **SPEOS input files** directory: `print = Speos.Sim.Comand.GetInputFolder()`
- **GetOutputFolder** returns the path to the **SPEOS output files** directory: `print = Speos.Sim.Comand.GetOutputFolder()`
- **HpcCompute** launches the Speos HPC compute command on the specified objects:
  `SpeosSim.Command.HpcCompute(directSimulation)`
- **HpcComputeOnActiveSelection** launches the Speos HPC compute command on the specified objects:

```
selection = Selection.Create([interactiveSimulation.Subject,
directSimulation.Subject])
SpeosSim.Command.SetActiveSelection(selection)
SpeosSim.Command.HpcComputeOnActiveSelection()
```

- **PreviewCompute** launches the Simulation Preview command on the specified objects:
  `SpeosSim.Command.PreviewCompute(directSimulation)`
- **PreviewComputeOnActiveSelection** launches the Simulation Preview command on the specified objects:

```
selection = Selection.Create([inverseSimulation.Subject,
directSimulation.Subject])
SpeosSim.Command.SetActiveSelection(selection)
SpeosSim.Command.PreviewComputeOnActiveSelection()
```

16.5.2.5. SourceRayFile Methods

This page presents you an example on how to optimize a Ray File Source thanks to the **OptimizeRayFile** method.

```
rayFileSource = SpeosSim.SourceRayFile.Find("Ray-file.1")

# this ray file does not need optimization
rayFileSource.RayFile = ".\\Speos input
files\\rayfile_LT_QH9G_100k_270114_Speos.RAY"
canRayFileBeOptimized = rayFileSource.OptimizeRayFile()

print "Has this ray file been optimized? " + str(canRayFileBeOptimized)

# this ray file does need optimization
rayFileSource.RayFile = ".\\Speos input
files\\rayfile_GW_PSLR31em_yellow_20M_20160909_IES_TM25.TM25RAY"
canRayFileBeOptimized = rayFileSource.OptimizeRayFile()

print "Has this ray file been optimized? " + str(canRayFileBeOptimized)
```

16.5.2.6. LightGuide Methods

This page presents you an example on how to export all the prism parameters of a Light Guide as CSV file thanks to the **ExportAsCSVFile** method.

```
# Find "Light guide.1" Light Guide
LightGuide1 = SpeosDes.LightGuide.Find("Light guide.1")

if not LightGuide1 is None:

# Prism geometries
LightGuide1.StepType = "Control points"

stepConfig = LightGuide1.StepConfigurations
print "Number of configurations: " + str(stepConfig.Count)
config = 0
while config < stepConfig.Count:
   print "Position: " + str(stepConfig[config].Position) + ", Value: " +
str(stepConfig[config].Value)
   config += 1

controlPoint = LightGuide1.StepConfigurations.AddNew(0)
controlPoint.Position = 50
controlPoint.Value = 3

child = LightGuide1.StepConfigurations.AddNew(0)
child.Position = 30
child.Value = 4

LightGuide1.OffsetType = "Constant"
LightGuide1.OffsetValue = 4

LightGuide1.StartAngleType = "Input file"
LightGuide1.EndAngleType = "Automatic"

LightGuide1.CSVFile = ".\\LightGuide_export.csv"
```

```
LightGuide1.Compute()

csvFile = currentPath + "\\" + "LightGuide_csv_export.csv"
LightGuide1.ExportAsCSVFile(csvFile)
```

16.5.2.7. ControlPointConfiguration Methods

This page presents you an example on how to define Control Points to create parameter variation along the guide curve of a Light Guide.

```
# Light Guide
LightGuide1 = SpeosDes.LightGuide.Create()

# Guide curve
Curve_LightGuide = GetRootPart().Curves[4]
LightGuide1.GuideCurve.Set(Curve_LightGuide)

# Body
LightGuide1.BodyProfileDiameter = 5

# Prisms orientation
X_Axis =  GetRootPart().Curves[1]
LightGuide1.OpticalAxis.Set(X_Axis)

LightGuide1.PrismsOperationType =
SpeosDes.LightGuide.EnumPrismsOperationType.Hybrid
LightGuide1.ReverseOpticalAxisDirection = False

# Distances
LightGuide1.DistancesType = SpeosDes.LightGuide.EnumDistancesType.Curvilinear
LightGuide1.DistanceStart = 2
LightGuide1.DistanceEnd = 2

# Prism geometries
LightGuide1.StepType = "Control points"

stepConfig = LightGuide1.StepConfigurations
print "Number of configurations: " + str(stepConfig.Count)

for config in LightGuide1.StepConfigurations:
    print "Position: " + str(config.Position) + ", Value: " + str(config.Value)


controlPoint = LightGuide1.StepConfigurations.AddNew(0)
controlPoint.Position = 50
controlPoint.Value = 3

print "After adding new configuration: "
for config in LightGuide1.StepConfigurations:
    print "Position: " + str(config.Position) + ", Value: " + str(config.Value)


# Add after the index
child = LightGuide1.StepConfigurations.AddNew(0)
child.Position = 30
child.Value = 4
```

```
LightGuide1.OffsetType = "Constant"
LightGuide1.OffsetValue = 4

LightGuide1.StartAngleType = "Input file"
LightGuide1.EndAngleType = "Automatic"

LightGuide1.CSVFile = ".\\LightGuide_export.csv"

LightGuide1.Compute()
```

16.5.2.8. ProjectionLens Methods

This page presents you an example on how to get/set the aspheric parameters of a Projection Lens.

```
pl = SpeosDes.ProjectionLens.Find("TIR Lens.1")
pl.BackFaceAspherics[2] = 2.3
print pl.BackFaceAspherics[2]
```

16.5.2.9. ControlPlane Methods

This page presents you an example on how to define a Control Plane that allows you to drive the overall beam spread for a Poly Ellipsoidal Reflector.

```
# Poly Ellipsoidal Reflector
perSurface = SpeosDes.PER.Create()

perSurface.SourcePoint.Set(GetRootPart().Curves[0])
perSurface.ImagePoint.Set(GetRootPart().Curves[6])
perSurface.OrientationAxis.Set(GetRootPart().Curves[3])
perSurface.Symmetry = SpeosDes.PER.EnumSymmetry.SymmetryTo0Plane

angularSectionConfig = perSurface.AngularSections
ParsePER(angularSectionConfig)

# Add angular section
perAngularSection = perSurface.AngularSections.AddNew(0)
perAngularSection.Angle = 55

print "After adding an angular section: "

angularSectionConfig = perSurface.AngularSections
ParsePER(angularSectionConfig)

# Add control plane
controlPlane = perAngularSection.ControlPlanes.AddNew()
controlPlane.Defocus = 0
controlPlane.Position = 15

print "After adding a control plane: "

angularSectionConfig = perSurface.AngularSections
ParsePER(angularSectionConfig)

perSurface.Compute()
```

16.5.2.10. PERAngularSection Methods

This page presents you an example on how to use the **FittingControlPlane** method with a Poly Ellipsoidal Reflector.

```
# Poly Ellipsoidal Reflector

# Find "Poly Ellipsoidal Reflector.1" Light Guide
perSurface = SpeosDes.PER.Find("Poly Ellipsoidal Reflector.1")

if not perSurface is None:
perSurface.Symmetry = SpeosDes.PER.EnumSymmetry.SymmetryTo0Plane

perSurface.Compute()

perSurface2 = perSurface.Clone()
perSurface2.Compute()

# Add angular section
perAngularSection = perSurface2.AngularSections.AddNew(0)
perAngularSection.Angle = 55

# Add control plane
controlPlane = perAngularSection.ControlPlanes.AddNew()
controlPlane.Defocus = 0
controlPlane.Position = 15

# Check position and defocus values
controlPlanes = perAngularSection.ControlPlanes
for controlPlane in controlPlanes:
   print "Control plane position: " + str(controlPlane.Position) + ", defocus:
 " + str(controlPlane.Defocus)

fittingWorked = perAngularSection.FittingControlPlane()
print "After fitting control planes"
print "Fitting did work? " + str(fittingWorked)

for controlPlane in controlPlanes:
   print "Control plane position: " + str(controlPlane.Position) + ", defocus:
 " + str(controlPlane.Defocus)

perSurface2.Compute()

perAngularSection = perSurface2.AngularSections[0]
fittingWorked = perAngularSection.FittingControlPlane()
print "Fitting did work? " + str(fittingWorked)
```

16.5.2.11. EyeboxConfiguration Methods

This page presents you an example on how to define a Head-Up Display Optical Analysis with a multi-eyebox configuration using the **TiltAngle** method.

```
multiEyeBoxMirrors = hoaSimulation.Mirrors.GetMultiEyeBoxMirrorPossibleValues()


for multiEyeBoxMirror in multiEyeBoxMirrors:
    if multiEyeBoxMirror.Contains(GetRootPart().Bodies[1].GetName()):
       hoaSimulation.Mirrors.MultiEyeBoxMirror = multiEyeBoxMirror
```

```
       break

hoaSimulation.Mirrors.TiltRotationAxis.Set(GetRootPart().Curves[1])


ebConfigurations = hoaSimulation.Mirrors.EBMirrorConfigurations

for ebConfig in ebConfigurations:
    if ebConfig.EBConfigName.Contains("Lower"):
       ebConfig.TiltAngle = -2
    elif ebConfig.EBConfigName.Contains("Upper"):
       ebConfig.TiltAngle = 2
```

16.5.2.12. HUDOD Advanced Parameters Methods

This page presents you an example on how to get/set HUD Optical Design Advanced Parameters to optimize, correct or adjust the HUD system.

AdvancedParameters allows you to get/set the following dynamics parameters: PGU Usage, Curvature Balance, Mirror Size Ratio, Stopping Criterion.

```
hod = SpeosDes.HUDOD.Find("HUD Optical Design.1")
hod.AdvancedParameters['Mirror size ratio'] = 1.4
print hod.AdvancedParameters['Curvature Balance']
for key in hod.AdvancedParameters:
    print key
```

16.5.2.13. CADUpdate Methods

This page presents you an example on how to import and update a geometry from another CAD software into Speos.

Importing the Geometry

```
from System.IO import Path

currentFilePath = GetRootPart().Document.Path
currentPath = Path.GetDirectoryName(currentFilePath)

speFile1 = currentPath + "\\" + "lguide.prt"
speFile2 = currentPath + "\\" + "led.prt"

test1 = SpeosAsm.CADUpdate.Import(speFile1)
test2 = SpeosAsm.CADUpdate.Import(speFile2)
```

Updating the Geometry

```
def checkImportedPart(importedPart):
   print "File path: " +
str(SpeosAsm.CADUpdate.GetLastImportedFilePath(importedPart))
   print "Last update: " +
str(SpeosAsm.CADUpdate.GetLastImportedFileDateTime(importedPart))

def updateImportedPart(importedPart):
```

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

```
   lastUpdate = SpeosAsm.CADUpdate.GetLastImportedFileDateTime(importedPart)
   bUpdate = SpeosAsm.CADUpdate.Update(importedPart, True, True)
   print "Update did work (unmodified parts skipped)? " + str(bUpdate)

   bUpdate = SpeosAsm.CADUpdate.Update(importedPart, True, False)
   print "Update did work (unmodified parts not skipped)? " + str(bUpdate)

   # Update
   previousUpdate = lastUpdate
   lastUpdate = SpeosAsm.CADUpdate.GetLastImportedFileDateTime(importedPart)
   if lastUpdate != previousUpdate:
      print "Last update: " +
str(SpeosAsm.CADUpdate.GetLastImportedFileDateTime(importedPart))

# Update all imported files that have been imported
currentPart = GetRootPart()
importedParts = SpeosAsm.CADUpdate.GetImportedPartsUnder()

bUpdate = SpeosAsm.CADUpdate.UpdateAll(currentPart, True, True)
print "Update all parts did work (unmodified parts skipped)? " + str(bUpdate)


bUpdate = SpeosAsm.CADUpdate.UpdateAll(currentPart, True, False)
print "Update all parts did work (unmodified parts not skipped)? " +
str(bUpdate)

for importedPart in importedParts:
   checkImportedPart(importedPart)
   updateImportedPart(importedPart)
```

16.5.3. Speos Core Methods

This section gathers the Speos Core methods that are not covered in the API resource file.

16.5.3.1. OpenFile

Description

This function allows you to open a *.sv5 or *.speos file.

Syntax

object.OpenFile(BSTR strFileName) As Int

- *object*: SPEOSCore
- *BSTR strFileName*: This variable is composed of the path, the filename and the extension
- Int return: returns 0 if succeeded

Example

```
from System import Type, Activator

#Creates SPEOSCore COM server
```

DRAFT

```
type = Type.GetTypeFromProgID("SV5.document")
SPEOSCore = Activator.CreateInstance(type)

#Opens sv5 file
fileName = mainPath + "C:\\NewSimulation.sv5"
commandline = ""
retval = SPEOSCore.OpenFile(fileName)
```

16.5.3.2. RunSimulation

Description

This function allows you to run a simulation.

Syntax

object.RunSimulation(Int nSimulationIndex, BSTR strCommandLine) As Int

- *object*: SPEOSCore
- *Int nSimulationIndex*: Simulation index, 0 by default
- *BSTR strCommandLine*: This variable corresponds to the command line
- Int return: returns 0 if succeeded

Example

```
from System import Type, Activator

#Creates SPEOSCore COM server
type = Type.GetTypeFromProgID("SV5.document")
SPEOSCore = Activator.CreateInstance(type)

#Opens sv5 file
fileName = mainPath + "C:\\NewSimulation.sv5"
commandline = ""
retval = SPEOSCore.OpenFile(fileName)

#Runs simulation
retval = SPEOSCore.RunSimulation(0, commandline)
```

16.5.3.3. ShowWindow

Description

This function allows you to display the Speos Core window.

Syntax

object.ShowWindow(Int nShowWindow) As Int

- *object*: SPEOSCore

DRAFT

- *Int nShowWindow*: 1 to show the window, 0 to hide it
- Int return: returns 0 if succeeded

Example

```
from System import Type, Activator

#Creates SPEOSCore COM server
type = Type.GetTypeFromProgID("SV5.document")
SPEOSCore = Activator.CreateInstance(type)

#Shows SPEOSCore interface
retval = SPEOSCore.ShowWindow(1)
```

16.5.3.4. Speos Core Command Lines

Description

The Speos Core command lines for Speos allow you to create scripts to automate multiple simulation launches without using the Speos Core interface.

Command Lines

Command Line	Description
FILENAME	Speos Core system file to open
-C	Command line mode (no GUI)
-G	Launch Speos GPU Solver
-S (SSS)	Launch simulation number SSS on FILENAME
-t (ttt)	Specified simulation thread number
-p (ppp)	Specify process priority (ppp) 0 - 5 • 0: idle • 2: normal • 5: realtime
-r (rrr)	Specifies simulation ray or pass number (rrr)
-D (DDD)	Specifies simulation duration in minutes (DDD)
-J (JJJ)	Specifies HPC job name (JJJ)
-N (NNN, default = 1)	Specifies HPC job node number (NNN)
-W (WWW)	Specifies HPC job wall clock in hours (WWW)
-?	show the help

DRAFT

17: Block Recording Tool

The Block Recording tool enables you to record operations you perform in Speos, and play back these recorded operations while keeping Speos feature links to the imported external CAD geometries.

> **Note:** The Block Recording tool is in BETA mode for the current release.

17.1. Understanding the Block Recording Tool

The following page helps you understand how to use the Block Recording Tool in Speos.

Description

The goal of recording operations (SpaceClaim Design operations and/or Speos features) on imported external CAD geometries is to be able to play again these operations after you updated or modified the geometries in the external CAD software. This saves you from applying again manually the different operations in the Speos environment.

For more information on how to use the Block Recording tool, refer to the **SpaceClaim Recording** documentation.

Generic Workflow

> **Important:** The Block Recording tool can be opened for only one document in a session. It does not work for multiple *.scdocx documents opened in the same session.

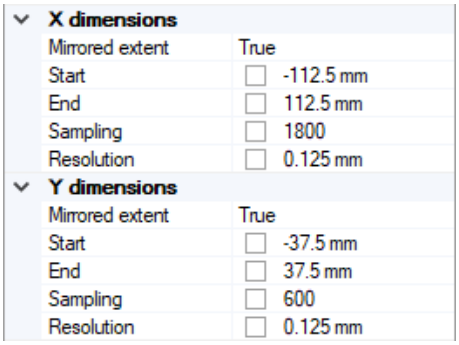
> **Warning:** Once the Block Recording tool is activated, do not deactivate it or you will loose all the added blocks.

1. In Speos, import external CAD geometries.
2. Activate the Block Recording tool
   
3. If you want, you can apply SpaceClaim Design Modeling operations.
4. Create Speos feature (apply material, create a source, a sensor, a simulation, an Optical Part Design feature).

   To add a parameter to the Block Recording you have two possibilities:

   - Modify the parameter's value in the feature definition.
   - Check the check box that appears in front of the parameters that can be exported into the Block Recording or Workbench.

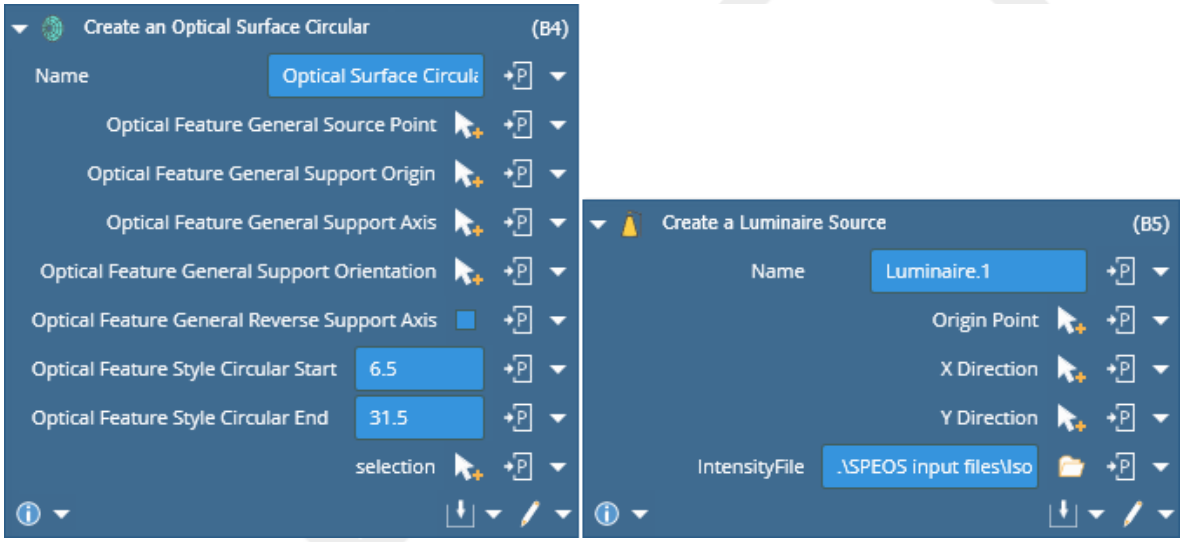
     The parameter will be exported to the block recording with its current value, or with its default value if you have not modified the value.



For more information on non-compatible Speos feature, refer to Non-Compatible Speos Features and Actions with the Recording Tool on page 836.

5. Exit the definition of the feature when you are done with it to create the feature block in the Block Recording tree: press **[Esc]** or uncheck **Edit** in the Design tab.

   The feature appears with the different parameters modified in one same block.



6. Run a Speos simulation.

> **Note:** When you run a simulation, only **CPU compute** and **GPU compute** are recorded. **HPC compute** is not recorded.

7. Save the project.

   You created a list of recorded blocks containing operations that can be played back.

8. In the External CAD Software, modify the geometries according to your needs.
9. Save the project.
10. In Speos, in the Block Recording tool, you can:

    - click **Continue**

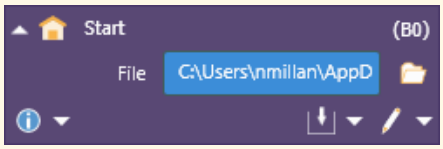
      to play up to the end of the recorded blocks.

- click **Next Feature**

  to play the recorded blocks step by step.
- right-click a block and select **Move to** to play up to this specific recorded block.

Note: If you modified the filename of the CAD part, make sure to modify it in the **Start Block**.



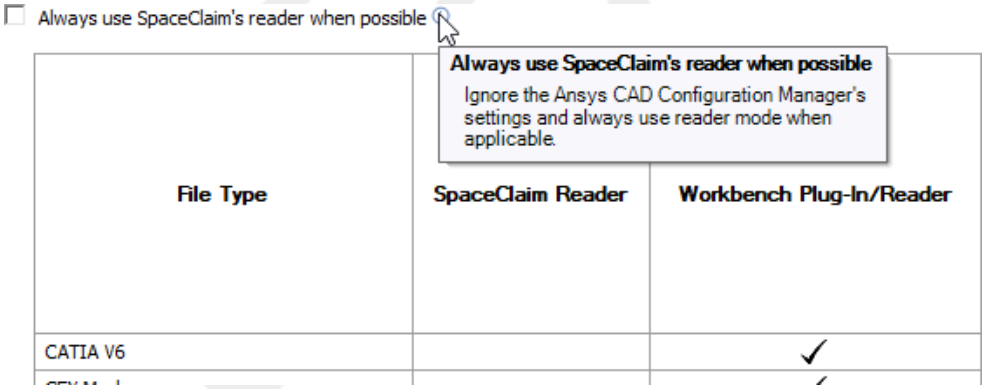
Operations are played back on the modified geometries, and you can see that Speos links are kept.

17.2. Configuring the Environment for the Block Recording Tool

The following procedure helps you configure the SpaceClaim environment to use the Block Recording Tool with the Speos features.

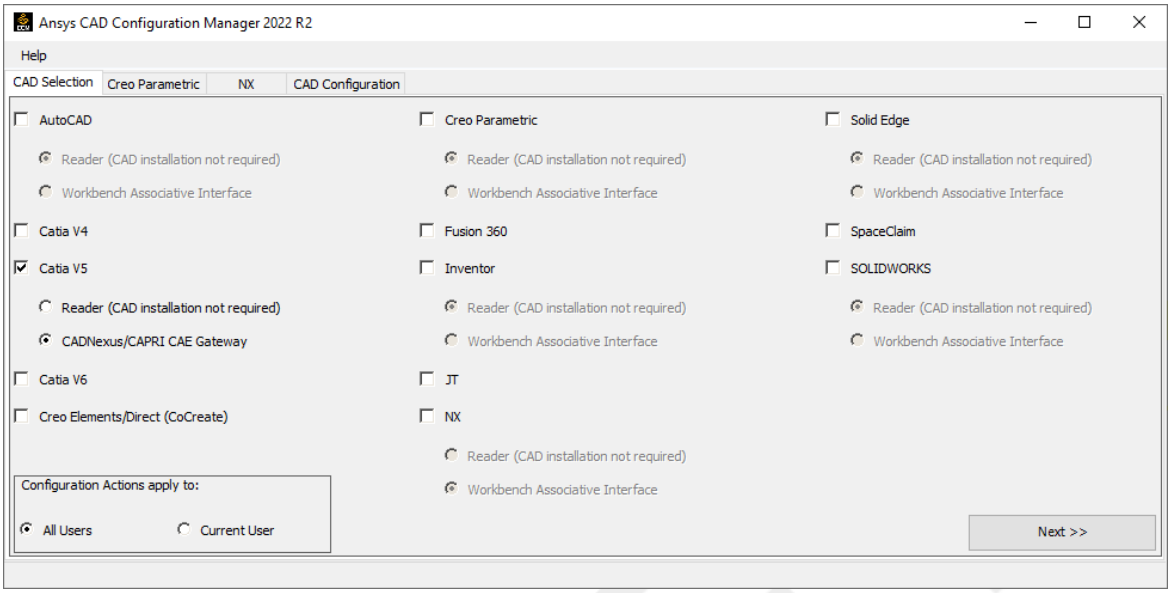
To configure the environment:

1. Make sure you meet the prerequisite for the **CAD Associative Geometry Interface** you want to use described in the following **chapter**.
2. In Speos, click **File** > **Speos Options**
3. Expand the **File Options** tab, and select **Workbench**.
4. Deactivate the option **Always use SpaceClaim's reader when possible** if not already done to activate the **Workbench Plug-In/Reader**.

   The option **Always use SpaceClaim's reader when possible** is activated by default.
5. Make sure you have installed the **Ansys Geometry Interfaces**.
   - If you have not installed it:
     a. Refer to the following **page** and follow the procedure from **Step 1** to **Step 5**.
     b. At **Step 5**, according to the CADs you want, select the Associative Geometry Interface. For instance: **Catia, Version 5 Geometry Interface**.
     c. Continue the procedure and finish the installation.
   - If you have installed the Ansys Geometry Interfaces without configuring the CAD Geometry interfaces:

a. From **Start**, open **Ansys 20XX RX** > **CAD Configuration Manager 20XX RX**



b. In the **CAD Selection** tab, check the CADs you want and select **Workbench Associate Interface** (in case of Catia V5, select **CADNexus/CAPRI CAE Gateway**).

c. Click **Next**.

d. In the **CAD Configuration** tab, click **Configure Selected CAD Interfaces**.

e. Click **Exit**.

The Block Recording tool is ready to be used with Speos and imported CAD files.

17.3. Non-Compatible Speos Features and Actions with the Recording Tool

The following page provides you with the Speos features and actions that cannot be recorded in the Block Recording tool.

> **Important:** The following list may not be exhaustive. Some features or actions might be recorded, however they are not considered as officially supported/compatible.

You cannot record:

- the Sub-folder creation in the Speos tree.
- the object move between folders in the Speos tree.
- the Generic selection when defining a Speos feature.

  Instead, use the specific selection to select elements such as Source, Sensor, Geometry.
- the Simulation Options.
- the following actions from Contextual Menus:
  - Light Guide: Export As CSV File
  - Micro Optical Stripes: Export As CSV File
  - Micro Optical Stripes: Extract Tooling Path
  - LiDAR Simulation: Export

- LiDAR Simulation: Linked Export
- LiDAR Simulation: Isolate
- Direct Simulation: Export
- Direct Simulation: Linked Export
- Direct Simulation: Isolate
- Inverse Simulation: Export
- Inverse Simulation: Linked Export
- Inverse Simulation: Isolate
- Interactive Simulation: Export Rays as Geometry
- Light Box Export, when right-clicking on Password: Generate Password
- OPTProjectedGrid Result: Export projected grid as geometries
- Ray File Source: Optimize ray file for Speos simulation
- UV Mapping: Insert a new UV map below
- Irradiance Sensor: Show Grid

> **Note:** Note that you can record the **Show Grid** action if you use the **Options** from the sensor definition.

17.4. Troubleshooting: Smart Variable with Wrong Selection

Known Issue

In case of a selection (Smart Variable), sometimes SpaceClaim cannot find which geometry to select, and may select the wrong one.

Solution

In the script editor, add a custom block that defines the correct selection by using a filter function provided in the SpaceClaim API scripting.

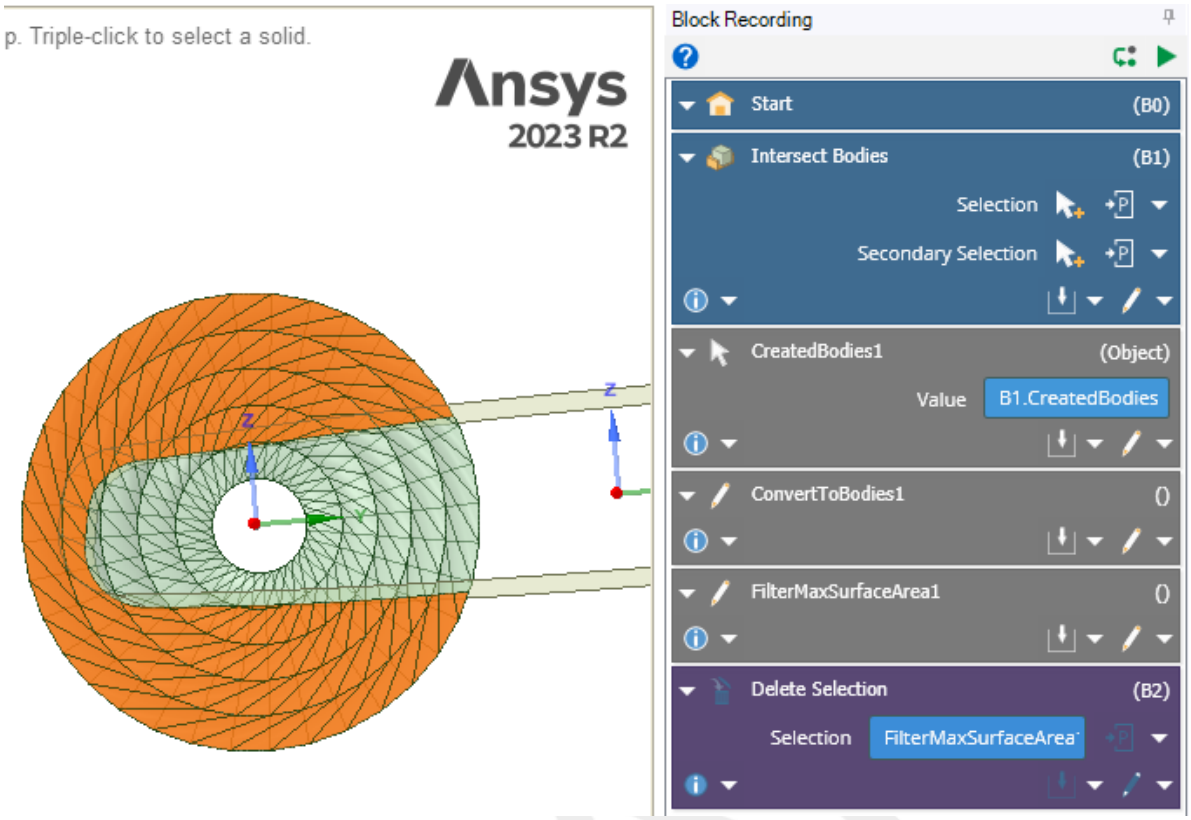
Example

The green part needs to be kept, and the orange part needs to be removed.

The code to write would be:

```
Selection = B1.CreatedBodies.ConvertToBodies().FilterMaxSurfaceArea()
Delete.Execute(Selection)
```

DRAFT


p. Triple-click to select a solid.
Ansys
2023 R2
Block Recording
Start (B0)
Intersect Bodies (B1)
Selection
Secondary Selection
CreatedBodies1 (Object)
Value B1.CreatedBodies
ConvertToBodies1 ()
FilterMaxSurfaceArea1 ()
Delete Selection (B2)
Selection FilterMaxSurfaceArea

DRAFT

18: Speos Sensor System Exporter

The Speos Sensor System Exporter is a standalone tool to post-process Exposure Maps calculated by Speos.

> **Note:** The Speos Sensor System Exporter is in BETA mode for the current release.

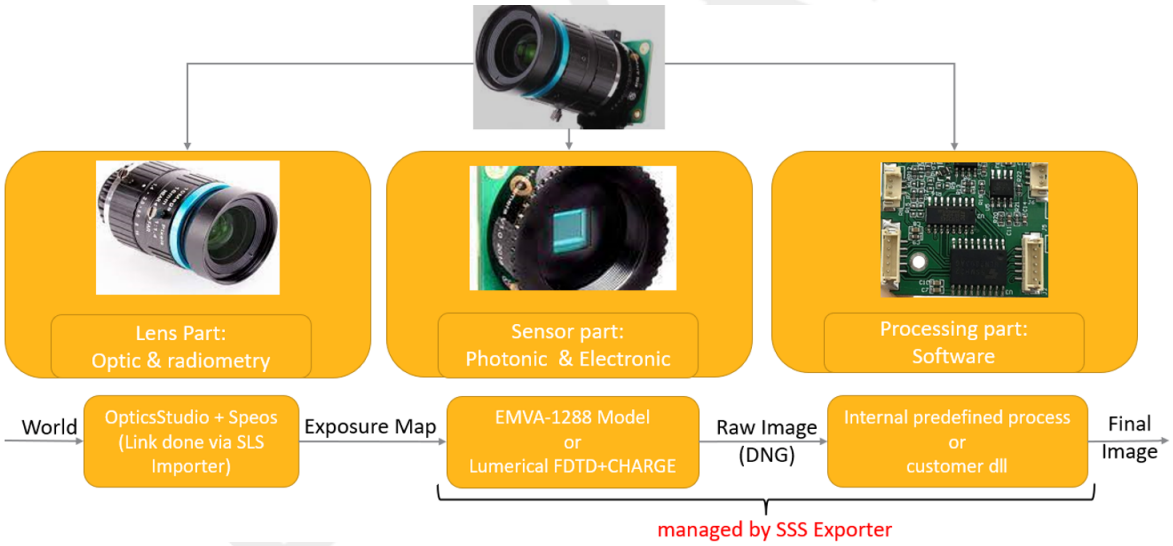
18.1. Speos Sensor System Exporter Overview

The following page gives you an overview of the Speos Sensor System Exporter and how to use it.

What is Speos Sensor System Exporter

Speos Sensor System Exporter is a standalone tool to post-process Exposure Maps calculated by Speos. Using a Camera Sensor, Speos can generate Exposure Map in front of a CCD (Charge-Coupled Device) or CMOS (Complementary Metal Oxide Semiconductor) sensor. Speos Sensor System Exporter converts this Exposure Map into a Raw image and developed image or data.

Raw image calculation is based on a **Reduce Order Model** of the sensor. The model is based on **EMVA 1288 standard (release 3.1)**.



How It Works

Speos Sensor System Exporter comes with a minimal GUI that provides feedback about the calculation performed.

Inputs and main parameters are saved in a file coded with YAML standard. This file is easily editable with a TXT editor. It contains instructions that Speos Sensor System Exporter can execute.

Another YAML code file contains sensor properties.

Speos Sensor System Exporter can be started using a *.bat file with the following command lines inside:

```
"Path to Speos Sensor System Exporter exe" "Path to Input filename"
Pause
```

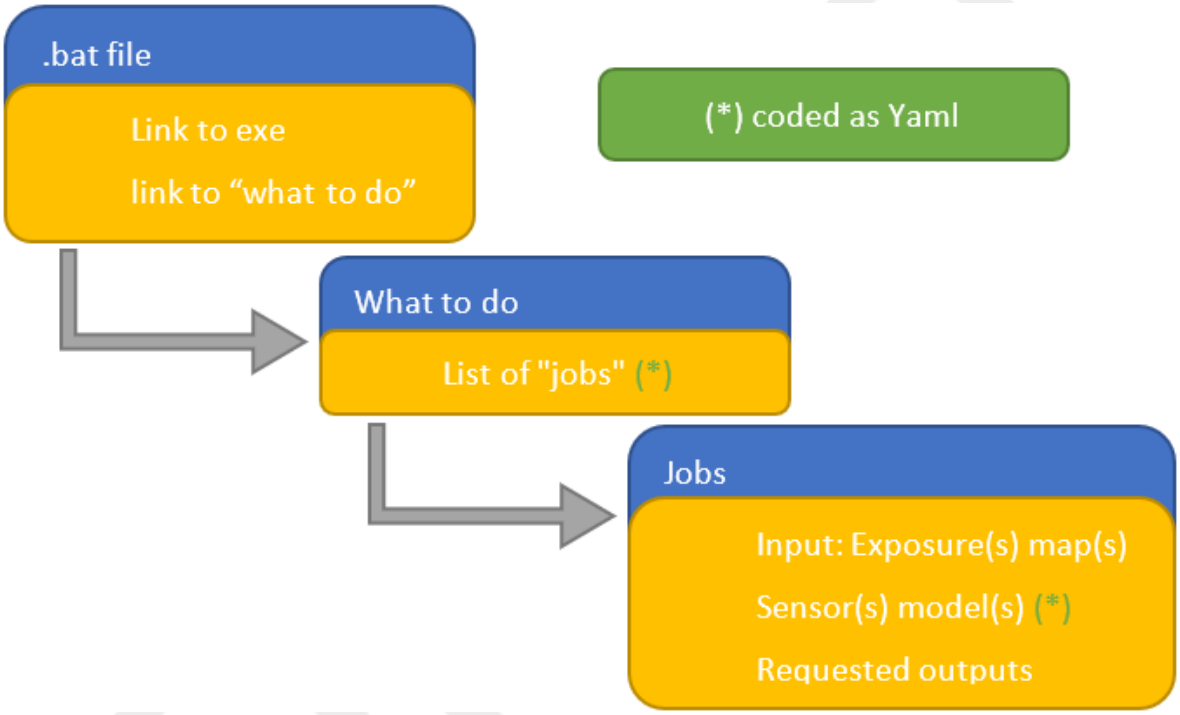
The following chapters presents you how to define The YAML inputs file and YAML sensor properties file.

How to Start

From version 2023 R2, Speos Sensor System Exporter is integrated into the Speos installation. The *.exe file is in Speos viewer folder.

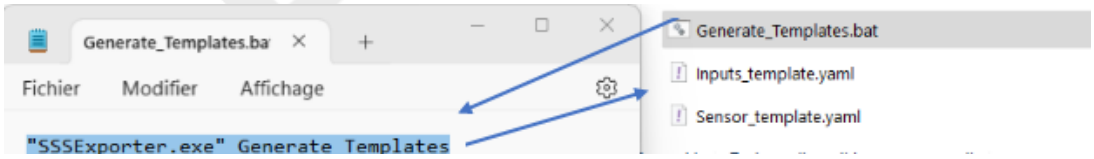
To download the last updated version of the Speos Sensor System Exporter, please click the following link.

Figure 115. Speos Sensor System Exporter Basic Usage



You can generate a template version of both YAML input files directly from the Speos Sensor System Exporter, using the following command:

```
"SSSExporter.exe" Generate Templates
```



18.2. YAML Files Description

The following chapter presents the two types of YAML file needed as input for the Speos Sensor System Exporter.

18.2.1. YAML File Code Introduction

The following page quickly introduces you with the YAML file standard to better understand the YAML files to set in the context of the Speos Sensor System Exporter.

If you want to go directly to Speos Sensor System Exporter YAML Files description, refer to YAML Input Parameters File and YAML Sensor Properties File.

Description

YAML is a human-readable data serialization standard that can be used in conjunction with all programming languages and is often used to write configuration files.

> **Note:** For more information, you can refer to the YAML documentation on Fileformat or Circleci websites.

Basically a YAML file contains keys:

- Each key can be associated to value(s). A value can be a number, a string, a list of something, or a sub-key.
- ":" is used as relation between the key and the value.
- Comments can be added using #.
- Tabulations are used to define the structure.

Example

The following example is based on a YAML file corresponding to the inputs parameters to give to the Speos Sensor System Exporter using the **Given files** mode.

Given files mode along with **All in folder** modes correspond to the two modes used to define the input parameters. They are explained in the Working Modes chapter.

```
Logging Level:      # first key
  File: DEBUG       # first sub-key and associated value
  Console: INFO     # second sub-key and associated value
Mode: Given files   # second key and associated value
Given files:
  Set 0:
    Exposure maps: /Inputs/Exposure 3000K.xmp
    Sensor: /Inputs/Sensor 3000K.yaml
    Output folder: /Outputs
    Raw export: ['dng']
    Processed export: ['png']
  Set 1:
    Exposure maps: /Inputs/Exposure 5000K.xmp
    Sensor: /Inputs/Sensor 5000K.yaml
    Output folder: /Outputs
    Raw export: ['dng']
    Processed export: ['png']
  Set 2:
    Exposure maps: /Inputs/Exposure 7000K.xmp
    Sensor: /Inputs/Sensor 7000K.yaml
    Output folder: /Outputs
    Raw export: ['dng']
    Processed export: ['png']
```

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

18.2.2. YAML Input Parameters File

The following section presents you the two ways of writing the YAML file that takes as inputs the Exposure Map and Sensors to process, as well as the output folder and export types.

18.2.2.1. YAML Input Parameters File Template

The following page provides you with the full YAML Input Parameters file template to fill.

The YAML Input Parameters File template can be automatically generated thanks to the Speos Sensor System Exporter:

In the Windows cmd prompt, enter the following command line: `"SSSExporter.exe" Generate_Templates`

```
Logging Level:
  File: 'DEBUG'
  Console: 'INFO'
Default Working Folder: /Inputs    # '' if working folder is the same as this
 file. Else set path to folder that will be use by default to find inputs.
Mode:                              # 'All in folder' or 'Given files'
All in folder:                     # needs to be filled if 'All in folder'
mode is chosen
  Input folder:                  # path to input folder  (format: 'c:\folder')

  Output folder:                   # path to output folder
  Xmp backup folder:               # path to folder where processed xmp can
be backup. if not present or '' ==> no backup
  Sensor:                          # sensor filename  (file format: yaml)
  Raw export: []                   # tuple from : 'xmp', 'dng'   (format:
['xmp', 'dng'])
  Electron export: []              # tuple from : 'xmp'
  Noise export: []                 # tuple from : 'xmp' (expressed in number
of e-)
  Processed export: []             # tuple from 'jpg', 'png', 'tiff', 'bmp'
Given files:                       # needs to be filled if 'Given files' mode
 is chosen
  Set 0:                           # first jog
    Exposure maps:                 # 'filename ' (for ldr mode) or tuple of
'filename ' (for hdr mode) (format: 'c:/temp/exposure1.xmp' or
['c:/temp/exposure1.xmp', 'c:/temp/exposure2.xmp', 'c:/temp/exposure3.xmp'])
    Sensor:                        # sensor filename  (file format: yaml) or
tuple of filename for hdr mode)
    Output folder:                 # path to output folder
    Rename:                       # optional string to change result filename

    Raw export:                    # tuple from : 'xmp', 'dng'   (format:
['xmp', 'dng'])
    Electron export: []            # tuple from : 'xmp'
    Noise export: []               # tuple from : 'xmp' (expressed in number
of e-)
    Processed export: []           # tuple from  from 'jpg', 'png', 'tiff',
'bmp' and 'hdr' in case of multi exposures (note: tiff as 16bits)
    Save previous: False           # True or False
    HDR:                           #inputs used only in case of HDR mode.
      Min:                         # set min value  [0, 1]
      Multiply Factor:             # set ratio (positive value. 1 ==> no
modifications)
      Gamma:                       # set exponent (power)  (positive value. 1
```

```
 ==> no modifications)
    Add text:                        # possibility to add text on images
      Text 0:                        # first text to add
        Position:                    # position in pixel from top left [i, j]
        Font:                       #choose in SIMPLEX, PLAIN, DUPLEX, COMPLEX,
 TRIPLEX, COMPLEX_SMALL, SCRIPT_SIMPLEX, SCRIPT_COMPLEX, FONT_ITALICSIMPLEX
        Scale:                       # size of text (1 by default)
        Color:                       # text color  ([255, 255, 0] by default)
        Thickness:                   # thickness of text (1 by default)
        Line type:                   # ? (2 by default)
        Text:                        # text to add (string)
      Text 1:                        # second text to add
                                     # idem as first text
  Set 1:                             # second jog
                                     # idem Set 0
  Set 2:                             # third jog
                                     # idem Set 0
Video:
  Filename:                          # video filename (without extension)
  Frame rate:                        # 25, 30, 60 ext.
  Add text:                          # possibility to add text on the complete
video
  Text 0:                            # first text to add
    Position:                        # position in pixel from top left [i, j]
    Font:                           #choose in SIMPLEX, PLAIN, DUPLEX, COMPLEX,
 TRIPLEX, COMPLEX_SMALL, SCRIPT_SIMPLEX, SCRIPT_COMPLEX, FONT_ITALICSIMPLEX
    Scale:                           # size of text (1 by default)
    Color:                           # text color  ([255, 255, 0] by default)
    Thickness:                       # thickness of text (1 by default)
    Line type:                       # ? (2 by default)
    Text:                            # text to add (string)
  Text 1:                            # second text to add
                                     # idem as first text
```

18.2.2.2. Debug Options

The following section presents the different debug options to help you understand the possible issues in your results.

Two logs are generated while Speos Sensor System Exporter runs:

- One log is displayed in the DOS command window (named ‘Console’ in this document)
- One log is a file named **log file.log** and written in same folder as the *.bat file.

For both logs, two levels of detail can be specified:

- **Info**: only main information is displayed/written.
- **Debug**: more detailed information is added.

> **Note:** By default, **Console** is set to **Info** level whereas **File** is set to **Debug**.

It’s possible to change both options with the main first key of the Input file **Logging Level**:

```
Logging Level:
   File: 'DEBUG'
   Console: 'INFO'
```

Figure 116. Example of Console Content during Processing

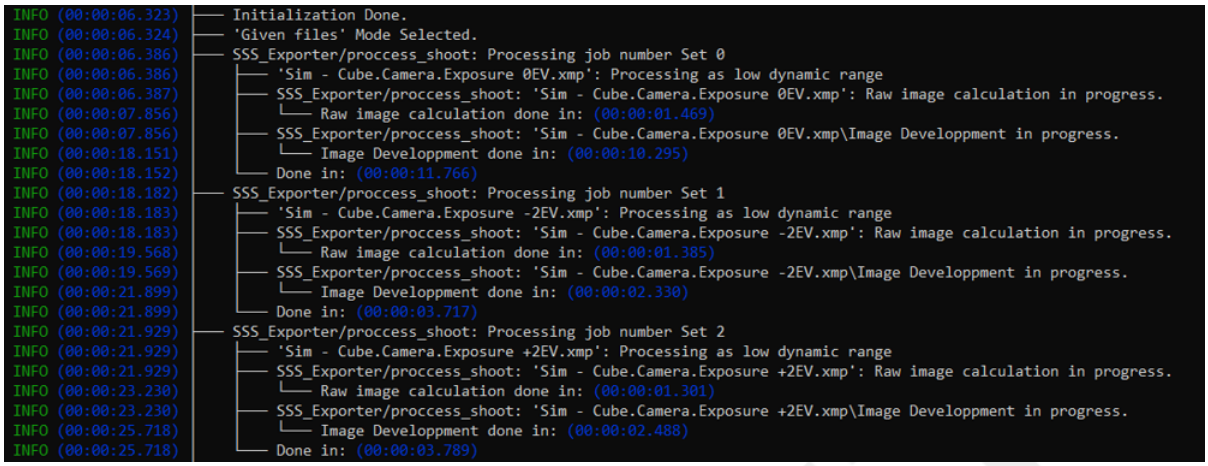
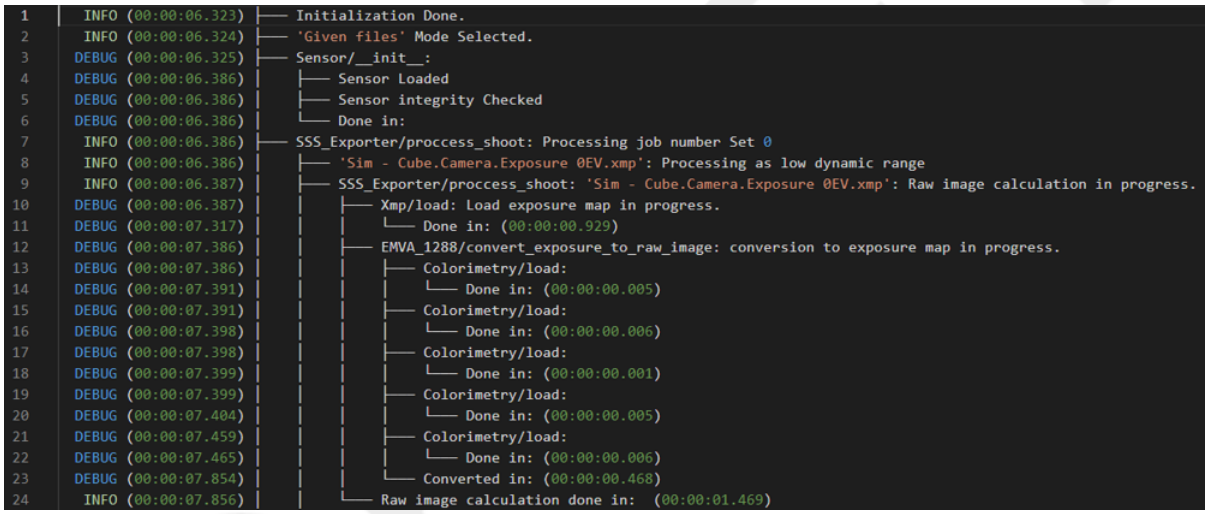


Figure 117. Example of Log File Content



18.2.2.3. Working Modes

The following section presents how to set the two possible working modes to define the input parameters.

18.2.2.3.1. Mode: All in Folder

Principle

Speos Sensor System Exporter does a recurrent scan of a given folder. Exposure Maps are processed as soon as they are stored in the folder. All maps are processed with the same sensor and conditions.

To run the Speos Sensor System Exporter using this mode, refer to the section below that lists the keys that must be filled (even if some are optional).

Keys

Input folder

The **Input folder** represents the scanned folder. Process starts as soon as an Exposure Map is detected in this folder. If several maps are detected at the same time, those will be processed one by one.

```
Input folder:    # path to input folder (format: 'c:\folder')
```

Output folder

The **Output folder** corresponds to the folder where results are saved.

```
Output folder:    # path to output folder
```

Xmp backup folder

Xmp backup folder: by default, and for the **All in folder** mode, Exposure Maps are removed from the Input folder after the process is completed. This is done to avoid processing several times the same map.

- Fill in the key to move the map into a specified folder to avoid removal.
- Do not fill in the key (or if the key is not present) to remove the map and so avoid the backup.

```
Xmp backup folder:    # path to folder where processed xmp can be backup
```

Sensor

The **Sensor** corresponds to the path to the YAML Sensor Properties file.

```
Sensor:    # sensor filename
```

Export

5 different types of export are available. For each export, a tuple can be set to indicate the output file format required:

- Electron export

> **Note:** **Electron export** corresponds to the **Noise export** and the **Signal export** combined.

- Noise export
- Raw export
- Processed export
- Signal export

```
Raw export: []          # tuple from: 'xmp','bin' 'dng' (format: ['xmp', 'dng'])
Electron export: []       # tuple from: 'xmp','bin'
Noise export: []          # tuple from: 'xmp','bin' (expressed in number of e-)
Processed export: []      # tuple from:'jpg', 'png', 'tiff', 'bmp'
Signal export: []         # tuple from: 'xmp','bin'
```

xmp format corresponds to files that can be opened using Virtual Photometric Lab.

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

bin format corresponds to binary files that can be loaded using a Python script:

- using 'fromfile' function from the 'NumPy' library
- data type is **np.uint16** for all outputs except Noise export which is **np.int16**

> **Important:** **Signal export** and **bin** format are available only from the updated version of Speos Sensor System Exporter that you can download from this link.

Example: You want to get **Raw export** as dng and xmp, plus **Processed export** as png, and neither **Electron export** nor **Noise export**

```
Raw export: ['xmp', 'dng']
Electron export: []
Noise export: []
Processed export: ['png']
Signal export: []
```

> **Note:** Export keys with no tuple [] can be omitted.

exe Termination

To end the exe, a filename **terminate** (without extension) must be added in the scanned folder.

18.2.2.3.2. Mode: Given Files

Principle

With the **Given files** mode:

- Exposure maps to process are explicitly given to the Speos Sensor System Exporter.
- Each process is referenced with the key **Set** and a **number**.

```
   Set 0:
   Set 1:
...
   Set n:
```

- Each **Set** can be processed with a specific sensor, conditions and export options.
- A **Set** key contains a list of sub keys.
- The first **Set** index must be 0, then next Set are continuously incremented.

Example of a Set Key

```
   Set 0:
      Exposure maps: Inputs/Sim - Cube.Camera.Exposure 0EV.xmp
      Sensor: Inputs/Sensor 4ms.yaml
      Output folder: 'Outputs/LDRI'
      Rename: LDRI 0EV
      Raw export: ['xmp', 'dng']
      Processed export: ['png']
```

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

Key for Set

Exposure maps

The **Exposure maps** key is a link to one Speos Exposure Map or Tuple of links to Exposures Maps. Tuples are used in case of HDRI processing (several Exposure Maps used to generate a single high dynamic range image).

Single map example

```
Exposure maps: c:/exposure_map.xmp
```

Tuple of maps example

```
Exposure maps: ['c:/exposure_map_1.xmp',.., 'c:/exposure_map_3.xmp']
```

Sensor

The **Sensor** key is the link to one sensor file or Tuple of links to sensor. Tuples are used in case of HDRI processing (in this case, you must specify a same number of sensors as number of maps).

Single sensor example

```
Sensor: 'c:/temp/sensor.yaml'
```

Tuple of sensors example

```
Sensor: ['c:/temp/sensor_1.yaml', ..., 'c:/temp/sensor_3.yaml']
```

Output folder

The **Output folder** corresponds to the folder where results are saved.

```
Output folder: # path to output folder
```

Rename

Rename: by default, results have a same name basis as the input exposure map. Then a suffix is added:

- _electron_ for Electronic export
- _noise_ for noise export
- _raw_ for raw export
- _developed_ for developed export

If specified as string, the name basis can be replaced. This is particularly sseful if several Sets are based on the same exposure map.

Export

5 different types of export are available. For each export, a tuple can be set to indicate the output file format required:

- Electron export

> **Note:** **Electron export** corresponds to the **Noise export** and the **Signal export** combined.

DRAFT

- Noise export
- Raw export
- Processed export
- Signal export

```
Raw export: []          # tuple from: 'xmp', 'bin', 'dng' (format: ['xmp',
'dng'])
Electron export: []     # tuple from: 'xmp', 'bin'
Noise export: []       # tuple from: 'xmp', 'bin' (expressed in number of e-)
Signal export: []       # tuple from: 'xmp', 'bin'
Processed export: []    # tuple from 'jpg', 'png', 'tiff', 'bmp' 'hdr'
```

xmp format corresponds to files that can be opened using Virtual Photometric Lab.

bin format corresponds to binary files that can be loaded using a Python script:

- using 'fromfile' function from the 'NumPy' library
- data type is **np.uint16** for all outputs except Noise export which is **np.int16**

> **Important: Signal export** and **bin** format are available only from the updated version of Speos Sensor System Exporter that you can download from this link.

Example: You want to get **Raw export** as dng and xmp, plus **Processed export** as hdr, and neither **Electron export** nor **Noise export**

```
Raw export: ['xmp', 'dng']
Electron export: []
Noise export: []
Signal export: []
Processed export: ['hdr']
```

> **Note:** Export keys with no tuple [ ] can be omitted.

Save previous

If **Save previous** is set to **True** and if a result with a same filename already exists, then the previous result is renamed with additional suffix that contains the original date/time of file creation.

If **Save previous** is set to **False** or if not set at all, the previous result is replaced by a new one.

HDR

The **HDR** key is used only in case of HDRI processing. The sub keys define the tone mapping to compress HDR data into standard image:

- Min: positive number between 0 and 1. 0 has no effect.
- Ratio: positive value. 1 has no effect.
- Gamma: positive value. 1 has no effect.

The corresponding code is:

```
# hdr_image coded as float in [0, 1] interval
Compressed_image = ((np.clip(hdr_image - offset, 0, 1))*ratio)**(1/gamma)
```

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.
Published: 2023-10-23T14:08:14.04-04:00

18.2.2.4. General Remarks

The following page lists remarks or information to take into account when you define your YAML file.

- All keys are case sensitives.
- For keys that refer to a file, path can be:
  - absolute (from x:/)
  - relative using '../' to jump to parent folder
  - from working folder using '/'. Working folder is the folder where the *.bat file is.
- Referenced folder (as **Output folder** for example) must exists before starting Speos Sensor System Exporter.
- Electron map, Noise map and Raw map are in DN (digital number)
- *.dng and *.tiff exports are coded on 16-bit.
- *.hdr export is coded as float.
- *.bmp, *.jpg and *.png are coded on 8-bit.
- In case of reference to a RGB sensor, input exposure map must have an even horizontal and vertical resolution.

18.2.3. YAML Sensor Properties File

The following section shows you how to write the YAML input file describing the sensor properties used to generate the raw image, final image, or possible data.

18.2.3.1. YAML Sensor Properties File Overview

Exposure Map post-processing can be split into two main steps:

- Exposure Map to raw image

  Calculations performed in this first step are based on the **EMVA 1288 standard** with the possibility to improve the model using data generated from Lumerical model (quantum efficiency with angular dependency).

- Raw image to developed image

  Calculations performed in this second step is based on a generic basic model.

Inputs (sensor properties, options and used conditions) used for both calculations are saved into the YAML Sensor Properties file. The YAML file is divided into 8 main keys:

- EMVA Standard version
- References
- Operating Conditions
- Properties
- Pre-Processing
- Lumerical data
- EMVA data
- Development

18.2.3.2. YAML Sensor Properties File Template

The YAML Sensor Properties File template can be automatically generated thanks to the Speos Sensor System Exporter:

DRAFT

In the Windows cmd prompt, enter the following command line: "SSSExporter.exe" Generate_Templates

```
EMVA Standard version: 3.1     # 3.1, 4.0-Linear or 4.0-General (only 3.1
supported in 2022-R2)
References:
  Vendor:                      # Name of the manufacturer (Optional)
  Camera:                      # Name of the camera (Optional)
  Sensor:                      # Name of the sensor (Optional)
  type:                        # Sensor type (CMOS, CDD, other) (Optional)
Operating Conditions:
  Gain:
    Value:
    Unit:                      # in Db or RN (RN for Real Number)
  Offset:
    Value:
    Unit:                      # in DN only  (DN for Digital Number)
  Exposure Time:
    Value:
    Unit:                      # s, ms, us (do not use 'µ')
  Temperature:
    Value:
    Unit:                      # in C or K
Properties:
  Resolution:                  # Number of pixels (i, j)
    H:                         # value or Auto (if auto, then calculated from
 speos map resolution)
    V:                         # value or Auto (if auto, then calculated from
 speos map resolution)
  Pixel Size:                  # in µm only
    H:                         # value or Auto (if auto, then calculated from
 speos map resolution and size)
    V:                         # value or Auto (if auto, then calculated from
 speos map resolution and size)
  Bits depth:                 # ADC capacity (possible values: 8, 10, 12, 14,
 16)
Pre-processing:
  Speckle removing:            # False or threshold ratio
  Colorimetric Filtering:      # True or False
Lumerical data:                # None or link to *.json file (if present 'IR
& UV Filter', 'Quantum Efficiency' and 'Bayer Matrix' fields are ignored and
must be set to 'From Lumerical data')
EMVA data:
  IR & UV Filter:
    Filename:                  # None or link to a *.spectrum file or From
Lumerical data
    Unit:                      # 'Percentage' or '0 to 1'
  Quantum Efficiency:          #(nu)  or  From Lumerical data
    Filename:                  # link to a *.spectrum file
    Unit:                      # 'Percentage' or '0 to 1'
  Bayer Matrix:
    Type:                     # None or i*j (with i and j for bayer structure
 size) or From Lumerical data
    #(example for a 2x2 bayer structure size)
    00 spectrum:               # link to *.spectrum file
    01 spectrum:               # link to *.spectrum file
    10 spectrum:               # link to *.spectrum file
    11 spectrum:               # link to *.spectrum file
    Unit:                      # 'Percentage' or '0 to 1'
  System Gain:                 # Overall system Gain (K)
    Value:
    Unit:                      # 'DN/electron'
```

```
  Temporal Dark Noise:          # (sigma_d or sigma_y.dark)
    Value:
    Unit:                       # 'electron' or 'bits' or 'dB'
  AST:                          # Absolute sensitivity threshold
    Value:
    Unit:                       # 'electron' or 'photon'
    Wavelength:                 # in nm (used only in case of given unit in
photon)
  DR:                           # Dynamic Range
    Value:
    Unit:                       # 'DN' or 'bits' or 'dB'
  Dark Current:
    Mean:
      Value:
      Unit:                     # 'electron/s'
    Standard Variation:
      Value:
      Unit:                     # 'electron/s'
    Td:                         # Doubling Temperature Interval
      Value:
      Unit:                     # 'K' or 'C'
    Tref:                       # reference temperature
      Value:
      Unit:                     # 'K' or 'C'
  Spatial Non Uniformity:
    DSNU:
      Value:
      Unit:                     # 'DN'
    PRNU:
      Value:
      Unit:                     # 'Percentage'
Development:
  Demosaizing Method:           # bilinear, Malvar2004, Menon2007, DDFAPD
  Linearization:
    Type:                       # None, 'Table' or 'Polynome'
    Data:                       # [] if 'None', lookup table from 0 to max DN
value if 'Table', DN_out = sum(Pi*(DN_in)**i) if 'Polynome'
  Rescaling:
    Black Level:                # None, Auto or Digital Number coded on 16bits
  (only for DNG output)
    White Level:                # None, Auto or Digital Number coded on 16bits
  (only for DNG output)
  Orientation:                  # 1: normal, 3: 180° rotation. See Tiff-Ep 6
specifications for other referenced values
  Color Saturation Correction: # True or False
  Colorimetry:
    Shoot Illuminant:
      Type:                     # File or Predefined
      Data:                     # link to *.spectrum file or Predefined
Illuminant: A, B, C, D50, D55, D65, D75
      Unit:                     # Percentage or 0 to 1 or NA (for Predefined)
    Color Space:              #development Color Space: sRGB, Adobe RGB (1998),
 Apple RGB, Best RGB, Beta RGB, Bruce RGB, CIE RGB, ColorMatch RGB, Don RGB
4, ECI RGB v2, Ekta Space PS5, NTSC RGB, PAL/SECAM RGB, ProPhoto RGB, SMPTE-C
 RGB, Wide Gamut RGB
    # Adaptation Method:        # XYZ Scaling, Bradford, Von Kries
    # Configuration x:          # this section is automatically filled
    #   Illuminant:
    #     Data:                 # Shoot Illuminant data
    #       Wavelengths:
    #       Values:
```

```
    #       XYZ:
    #   Color Space:           # sRGB, Adobe RGB (1998), Apple RGB, Best RGB,
 Beta RGB, Bruce RGB, CIE RGB, ColorMatch RGB, Don RGB 4, ECI RGB v2, Ekta
Space PS5, NTSC RGB, PAL/SECAM RGB, ProPhoto RGB, SMPTE-C RGB, Wide Gamut RGB

    #   Adaptation Method:
    #   Camera Neutral:        # [x, x, x]
    #   Color Calibration:     # [[x, x, x],[x, x, x],[x, x, x]]
    Configuration x:           # please leave this line as it is.
```

18.2.3.3. EMVA Standard Version

The **EMVA Standard version** key informs about the referenced EMVA 1288 released version.

Currently, only the 3.1 release is supported.

```
EMVA Standard version: 3.1
```

18.2.3.4. References

The **References** key lists sub-keys that are only optional information, and not used by Speos Sensor System Exporter.

> **Note:** You can add sub-keys to the **References** list.

```
References:
   Vendor:    # Name of the manufacturer (Optional)
   Camera:    # Name of the camera (Optional)
   Sensor:    # Name of the sensor (Optional)
   type:      # Sensor type (CMOS, CDD, other) (Optional)
```

18.2.3.5. Operating Conditions

The **Operating Conditions** key gives information about conditions of usage for the sensor. There are 4 sub-keys to fill:

- Gain
- Offset
- Exposure Time
- Temperature

Gain and Offset

Gain and **Offset** sub-keys are camera setting that you can adjust.

- **Offset** adds or subtracts a constant value to your signal.

  Offset unit is DN (Digital Number).

- **Gain** multiplies your signal by a constant factor.

  Gain unit is Db or RN (Real Number).

Both sub-keys are applied just before generating raw image.

Default values are 0 and 1.

> **CAUTION:** Do not confuse **Gain** and **System Gain**.

```
   Gain:
      Value: 1
      Unit: RN    # in Db or RN (RN for Real Number)
   Offset:
      Value: 0
      Unit: DN    # in DN only (DN for Digital Number)
```

Exposure Time

The **Exposure Time** key gives the duration on which the sensor has collected photons. The value must be the same as the one used for the Speos simulation used to generate the Exposure map.

Exposure Time unit is us (not "µ", but "u"), ms or s.

```
   Exposure Time:
      Value: 10
      Unit: ms    # s, ms, us (do not use 'µ')
```

Temperature

The **Temperature** key gives the camera temperature when used.

Temperature unit is C (not "°C") or K.

```
   Temperature:
      Value: 25
      Unit: C    # in C or K
```

18.2.3.6. Properties

The **Properties** key gives information about sensor general properties. There are 3 sub-keys to fill:

- Resolution
- Pixel Size
- Bits depth

Resolution

The **Resolution** key gives the number of lines and rows of the sensor pixels.

```
   Resolution:    # Number of pixels (i, j)
      H: Auto     # value or Auto (if Auto, then calculated from Speos map
resolution)
      V: Auto     # value or Auto (if Auto, then calculated from Speos map
resolution)
```

DRAFT

Note: By default, **Auto** value is used. In this case information are calculated from the Exposure Map.

Pixel Size

The **Pixel Size** key gives the width and height of one pixel.

```
   Pixel Size:    # in µm only
      H: Auto     # value or Auto (if Auto, then calculated from Speos map
resolution and size)
      V: Auto     # value or Auto (if Auto, then calculated from Speos map
resolution and size)
```

Note: By default, **Auto** value is used. In this case information are calculated from the Exposure Map.

Bits depth

The **Bits depth** key defines how many bits of tonal or color data are associated with each pixel or channel.

```
   Bits depth: 12    # ADC capacity (possible values: 8, 10, 12, 14, 16)
```

18.2.3.7. Pre-Processing

Speckle removing

The **Speckle removing** key is a convolution filter applied on Exposure Map to remove high peak due to Speos simulation.

If **Speckle removing** is set to **False**, nothing is done. Otherwise, function is automatically activated if a you define a threshold ratio as value.

```
   Speckle removing: 100
```

Colorimetric Filtering

The **Colorimetric Filtering** key is a filter applied on Exposure Map to remove colorimetric noise due to Speos simulation.

If **Colorimetric Filtering** is set to **False**, nothing is done. If set to **True**, filtering calculation is done.

```
   Colorimetric Filtering: True
```

DRAFT

18.2.3.8. Lumerical Data

From version 2023 R1, you can input Lumerical calculation result regarding quantum efficiency. Data from Lumerical are stored into a *.json file. The link to this file is set using the Lumerical data key:

```
Lumerical data:
   Mode: V2
   Filename: RGB-IR basic EQE.json
```

From version 2023 R2, there are two Lumerical models and formats for *.json file:

- Legacy version (V1) that takes into account only Chief ray angle
- New version (V2) that considers Chief ray angle and Marginal ray angles.

The **Mode** key indicates which type of input is provided.

In this case, **IR & UV Filter**, **Quantum Efficiency**, and **Bayer Matrix** sub-keys in the **EMVA data** key are ignored.

If you want to activate Lumerical input, and the sub-keys **IR & UV Filter**, **Quantum Efficiency**, and **Bayer Matrix**, set the sub-keys to **From Lumerical data** as follows:

```
   EMVA data:
      IR & UV Filter:
         Filename: From Lumerical data
         Unit:
      Quantum Efficiency:
         Filename: From Lumerical
         Unit:
      Bayer Matrix:
         Type: From Lumerical data
         00 spectrum:
         01 spectrum:
         10 spectrum:
         11 spectrum:
         Unit:
```

If you want to use the EMVA 1288 model standard and not the Lumerical data, do not fill in the sub-keys.

18.2.3.9. EMVA Data

The **EMVA data** key gives information about the sensor performance in relation with the EMVA 1288 model standard. There are 9 sub-keys to fill:

- IR & UV Filter
- Quantum Efficiency
- Bayer Matrix
- System Gain
- Temporal Dark Noise
- AST
- DR
- Dark Current
- Spatial Non Uniformity

DRAFT

IR & UV Filter

A UV and/IR cut can be added on the sensor cover window. If present, the **IR & UV Filter** key allows you to take this filter into account.

The value taken is a link to a *.spectrum file (Speos native format).

> **Note:** Setting the value to **None** define the filter as not present (or already specified into the Speos Camera sensor definition).

By default, *.spectrum files are expressed in percentage. In this case Unit key must be Percentage.

> **Note:** You can also define the **Unit** sub-key in [0;1] if the data in the *.spectrum file correspond to that.

```
   IR & UV Filter:
      Filename: c:/uv_ir.spectrum  # None or link to a *.spectrum file
      Unit:  # 'Percentage' or 'O to 1'
```

Quantum Efficiency

The **Quantum Efficiency** key value is a link to a *.spectrum file that represents the sensor quantum efficiency.

Usually, data given by sensor manufacturers are the combination of the sensor efficiency and the RGB Bayer filter. In this case, the value should be a *.spectrum file with 100% constant value.

```
   Quantum Efficiency: #(nu)
      Filename: c:/QE.spectrum  # link to a *.spectrum file
      Unit:  # 'Percentage' or 'O to 1'
```

Bayer Matrix

Monochrome Sensor

In case of a monochrome sensor, The **Bayer Matrix** key is not used and the **Type** sub-key value must be set to **None**.

```
Bayer Matrix:
Type: None
```

RGB Sensor

In case of a RGB sensor:

- the **Type** sub-key value must be a combination of 4 letters chosen in ‘R’, ‘G’ and ‘B’.
- The following four sub-keys **00 spectrum**, **01 spectrum**, **10 spectrum**, **11 spectrum** must be set with a link to a *.spectrum file.

```
   Bayer Matrix:
      Type: RGGB
      00 spectrum: c:/Red.spectrum   # link to *.spectrum file
      01 spectrum: c:/Green1.spectrum   # link to *.spectrum file
```

Release 2024 R1 - © Ansys, Inc. All rights reserved.
Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Published: 2023-10-23T14:08:14.04-04:00

```
    10 spectrum: c:/Green2.spectrum   # link to *.spectrum file
    11 spectrum: c:/Blue.spectrum    # link to *.spectrum file
```

System Gain

In the sensor, the charge units accumulated by the photo irradiance is converted into a voltage, amplified, and finally converted into a digital signal by an analog-to-digital converter (ADC). The whole process is assumed to be linear and can be described by a single quantity, the **System Gain** key.

The **System Gain** units is DN/e- (digits per electrons).

```
System Gain:
   Value: 0.5
   Unit: DN/electron
```

Temporal Dark Noise

Dark Noise results from the fact that even if there is no light at all hitting a pixel, the photodiode "faucet" still has a small flow of "leakage" electrons that are generated thermally.

The **Temporal Dark Noise** value can be expressed in electrons, bits or dB.

```
Temporal Dark Noise:
   Value: 3
   Unit: Delectron
```

AST

Absolute sensitivity threshold (AST) is the number of photons needed to get a signal equivalent to the noise observed by the sensor.

The **Value** sub-key must be a positive number in electrons or in photons Unit.

In case of photon-based value, you must define the **Wavelength** sub-key corresponding to the wavelength of the photon (to be able to convert into electron using Gain system). Wavelength unit is nm.

```
AST: # Absolute sensitivity threshold
   Value: 3
   Unit: photon   # 'electron' or 'photon'
   Wavelength: 545  # in nm (used only in case of given unit in photon)
```

DR

Dynamic range (DR) is defined as the ratio of the signal saturation to the Absolute Sensitivity Threshold (AST).

The **Value** sub-key is a positive number in DN, bits or dB Unit.

```
DR: # Dynamic Range
   Value: 3500
   Unit: DN
```

DRAFT

Dark Current

The dark signal is mainly caused by thermally induced electrons. Therefore, the dark signal has an offset (value at zero exposure time) and increases linearly with the exposure time. Because of the thermal generation of charge units, the dark current increases roughly exponentially with the temperature.

The **Dark Current** key is based on four sub-keys:

- the **Mean** sub-key corresponds to the average value of e-/s for the Tref temperature.
- the **Standard Variation** sub-key corresponds to the variation around the mean value in e-/s.
- the **Tref** sub-key corresponds to the value for Reference Temperature.
- the **Td** sub-key corresponds to the temperature interval that doubles the dark current.

```
Dark Current:
   Mean:
      Value: 8
      Unit: electron/s    # 'electron/s'
   Standard Variation:
      Value: 4
      Unit: electron/s    # 'electron/s'
   Td:                    # Doubling Temperature Interval
      Value:  10
      Unit: K             # 'K' or 'C'
   Tref:                  # reference temperature
      Value: 20
      Unit: C             # 'K' or 'C'
```

Spatial Non Uniformity

The model discussed so far considered only a single or average pixel. All parameters of an array of pixels will however vary from pixel to pixel.

For a linear sensor, there are only two basic non-uniformities. The characteristic curve can have a different offset and different slope for each pixel.

- The dark signal varying from pixel to pixel is called dark signal non-uniformity, abbreviated to **DSNU**.

  DSNU value is defined in DN.

- The variation of the sensitivity is called photo response nonuniformity, abbreviated to **PRNU**.

  PRNU value is defined in percentage.

```
Spatial Non Uniformity:
   DSNU:
      Value: 3
      Unit: DN    # 'DN'
   PRNU:
      Value: 5
      Unit: percentage
```

18.2.3.10. Development

Development corresponds to the second main step in Exposure Map post-processing.

The "Development" term is an echo to analog/film photography. Regarding digital photography, it is a software process coded into the camera that converts raw image (most of the time not given to user) and final/output image.

Two development types are available:

- Generic basic development directly implemented in the Speos Sensor System Exporter as described in the Generic Development section.
- Customizable development done from an external python script, as described in the External Development section.

Generic Development

Parameters used in the development calculations are defined into the main **Development** key divided into 5 sub-keys:

- Demosaizing Method
- Linearization
- Orientation
- Color Saturation Correction
- Colorimetry

Demosaizing Method

If the **Bayer Matrix** Type sub-key of the EMVA data key is set to a value different from **None**, a Demosaizing step is performed to get raw R, G and B channels.

4 methods are available:

- bilinear
- Malvar2004
- Menon2007
- DDFAPD

For more information on the methods, refer to the Bayer CFA Demosaicing and Mosaicing API Reference.

The **Demosaizing Method** key must specify one of those methods.

```
   Demosaizing Method: bilinear
```

Linearization

The **Linearization** key allows you to correct sensor non-linearity. If sensor is assumed to be linear, the **Type** sub-key must be set to **None**.

```
   Linearization:
      Type: None
```

If you want to correct the non-linearity, you have two possibilities:

- 1D lookup table (0 to max DN value)
- polynomial correction (DN_out = sum(Pi*(DN_in)**i))

Orientation

The **Orientation** key allows you to change the image orientation and/or apply a symmetry:

- 1 = The 0th row represents the **visual top** of the image, and the 0th column represents the **visual left-hand side**.
- 2 = The 0th row represents the **visual top** of the image, and the 0th column represents the **visual right-hand side**.
- 3 = The 0th row represents the **visual bottom** of the image, and the 0th column represents the **visual right-hand side**.
- 4 = The 0th row represents the **visual bottom** of the image, and the 0th column represents the **visual left-hand side**.
- 5 = The 0th row represents the **visual left-hand side** of the image, and the 0th column represents the **visual top**.

DRAFT

- 6 = The 0th row represents the **visual right-hand side** of the image, and the 0th column represents the **visual top**.
- 7 = The 0th row represents the **visual right-hand side** of the image, and the 0th column represents the **visual bottom**.
- 8 = The 0th row represents the **visual left-hand side** of the image, and the 0th column represents the **visual bottom**.

> **Note: 1** and **3** values are mainly used:
>
> - 1 keeps the original image.
> - 3 balances an optical 180° rotation.

```
Orientation: 3
```

Color Saturation Correction

The **Color Saturation Correction** sub-key can be set to **True** or **False**:

- If **True**: an additional process corrects the color shifting due to pixel saturation.
- If **False**: Color Saturation Correction is not applied.

```
Color Saturation Correction: True
```

Colorimetry

The final image colorimetry depends on 3 inputs to be set into sub-keys:

- Shoot Illuminant
- Target Color Space
- Adaptation method

From these inputs, 2 intermediate data are automatically calculated and added into the sensor file:

- Camera Neutral vector
- Color Calibration Matrix

> **Note:** For more information on matrix, you can refer to the following websites Strollwithmydog and Brucelindbloom.

Therefore, the **Colorimetry** key is divided into 4 sub-keys:

- **Shoot Illuminant**
- **Color Space**
- **Adaptation Method**
- **Configuration x**

If colorimetric data have been previously calculated, an additional **Configuration i** (do not confuse it with **Configuration x**) is added to each combination of [Shoot Illuminant, Color Space, Adaptation method].

Shoot Illuminant

The **Shoot Illuminant** key gives the main illuminant used for the scene calculation with Speos.

- The **Type** sub-key defines if the illuminant is predefined or comes from a Speos *.spectrum file.
- The **Data** sub-key sets the predefined illuminants (A, B, C D50, D55, D65, or D75) or the *.spectrum file to use.

DRAFT

Example: Predefined Illuminant

```
   Shoot Illuminant:
      Type: Predefined
      Data: D50
      Unit: NA
```

In case of a **Predefined** illuminant, **Unit** sub-key is always set to **NA**.

Example: From File

```
   Shoot Illuminant:
      Type: File
      Data: c:/shoot.spectrum
      Unit: Percentage  #Percentage or '0 to 1'
```

Color Space

The **Color Space** key gives the color space in which the image will be targeted.

The available **Color Space** values are: sRGB, Adobe RGB (1998), Apple RGB, Best RGB, Beta RGB, Bruce RGB, CIE RGB, ColorMatch RGB, Don RGB 4, ECI RGB v2, Ekta Space PS5, NTSC RGB, PAL/SECAM RGB, ProPhoto RGB, SMPTE-C RGB, Wide Gamut RGB.

```
   Color Space: Best RGB
```

Adaptation Method

The **Adaptation Method** key defines the adaptation method used to switch from one to another color space.

Available methods are:

- XYZ Scaling
- Bradford
- Von Kries

```
   Adaptation Method: Bradford
```

Configuration i

If colorimetric data have been previously calculated, an additional **Configuration i** (do not confuse it with **Configuration x**) is added to each combination of [Shoot Illuminant, Color Space, Adaptation method].

The **Configuration i** key contains data calculated during a previous post-process. As this calculation takes time, results are added to the file to allow you to reuse them.

Configuration x

The **Configuration x** key line should be present at the last line of the sensor file and kept as it is.

```
   Configuration x:    # please leave this line as it is.
```

External Development

How to call an external development

Instead of predefined development model, you can plug specific development algorithms called via a Python script.

DRAFT

To do that, the two following keys must be set in **Development** section:

```
Development:
   Type: External Script  #'Internal Generic' or 'External Script'
   External Script: Development.py
```

Script requirements

Script must be written in Python version superior to 3.9.

Script must contain a function named **Development** with:

- two arguments:
  - Rawimages: the set of raw images generated by the Speos Sensor System Exporter
  - Sensors: the set of sensor data
- one output: image data coded as a 3D numpy array (color channel, i, j)

DRAFT

19: Troubleshooting

This section describes known non-operational behaviors, errors or limitations found in Speos.

A workaround is given if available.

Inclusion in this document does not imply the issues or limitations are applicable to future releases.

Additional known issues and limitations relevant to the 2024 R1 release may be found in the **Known Issues and Limitations** document, and in the Ansys Customer Portal in the Class 3 Error Reports.

19.1. Known Issues

This section describes known non-operational behaviors or errors found in Speos.

A workaround is given if available.

19.1.1. Materials

Known Issue	Workaround or Solution
When hiding a geometry contained in a sub-component of a project, its associated texture preview (if existing) is not hidden and still displayed in the 3D view. (TFS 704282)	
The file RL_BlackLetter_PlasticPolicePlate.anisotropicbsdf from the ROAD LIBRARY FOR SENSORS SIMULATIONS library does not work correctly.	

19.1.2. Sources

Known Issue	Workaround or Solution
Thermic source applied on surface with texture is not supported in simulation.	

19.1.3. Sensors

Known Issue	Workaround or Solution
Results of Sequence Detection (*.OptHash *.OptSequence) are not removed if you run again the simulation with the sensor **Layer** parameter set to another value than **Sequence**. (TFS 674798)	

DRAFT

Known Issue	Workaround or Solution
In a sensor definition, when applying a XMP template with a certain type (photometric, radiometric, etc.), and defining the same type in the general section, an error occurs. (TFS 450549)	Do not consider the error as it does not prevent the simulation to run and the results to be correct.

19.1.4. Components

Known Issue	Workaround or Solution
When creating a 3D Texture, the projection of some 3D Texture elements on the edges of the surface may appear as floating along the vertical axis and not positioned on the surface. (TFS 777597)	Try to: • manually modify the mapping file to correct the points that are in the wrong vertical axis. • change the CAD model to reduce the edge tolerance as much as possible and re-compute the 3D texture.
If you modify a Speos Light Box that is used in a Speos Pattern feature, the Speos Pattern does not update with the modified Speos Light Box. (TFS 706919)	Clear the **Pattern File** field in the Speos Pattern feature, then reimport the Speos Light Box to get the modifications.
Speos Light Box: if you select a sub-feature of a Speos Light Box to add to a simulation, the link in the simulation to the sub-feature might be lost after the Speos Light Box Import re-compute. (TFS 639072)	Select the Speos Light Box Import instead of the sub-feature of the Speos Light Box.
No OPT3D Mapping file is generated after creating a 3D Texture. (TFS 425524 SF 32620)	Create a new 3D Texture to generate the OPT3DMapping file.
Placing the origin point of the 3D texture tangent to the support geometry cause construction issues. (TFS 202323)	Slightly shift the point from the support surface so that it is no longer tangent with the support and regenerate the 3D texture.
Components cannot be imported as pattern files when creating a 3D texture. (TFS 256663)	Use a **body** instead of a component.

19.1.5. Simulation

Known Issue	Workaround or Solution
Simulation is not compatible with a named selection (group of geometries) containing one or several faces of a surface body. (TFS 888268)	You must select the surface body to use the named selection in the simulation.
As a result of an Inverse Simulation using a Camera Sensor or an Irradiance Sensor when the Light Expert is activated, rays generated using the Gathering algorithm are not integrated in the *.lpf result. Therefore few rays may appear in the *.lpf result. However all rays are correctly integrated in the XMP result. (TFS 646080)	
Local Meshing: the Meshing can be wrong when the **Step** value applied is smaller than the face size. (TFS 552272 551166 551355)	Check the meshing quality using **Preview Meshing** and adjust if needed.

DRAFT

Known Issue	Workaround or Solution
When the **Automatic Compute** is activated for an Interactive Simulation, the **Preview Meshing** on geometries is not available. (TFS 279412)	Deactivate the **Automatic Compute**.
Canceling a Monte-Carlo Inverse Simulation does not create intermediate result in the Simulation tree. (TFS 451371)	
Using special characters in feature names prevent simulations from running correctly. (TFS 378374 SF 31224)	
Inverse simulations can take more time than expected when several sensors are involved. For a simulation with Y sensors and X time stopping criteria, the simulation may take X * Y time to run. (TFS 338429)	
Trying to open a new session of Speos while computing a simulation does not work. (TFS 202650 - SF 25120)	
Selecting geometry groups containing a lot of items might cause the software to freeze. (TFS 210096)	

19.1.6. Optical Part Design

Known Issue	Workaround or Solution
Freeform Lens: In some case, you may have a collimating lens providing inaccurate results. This may be due to different factors: the lens' surface may be too close to the source and the size of the lens aperture may be incorrectly defined. (TFS 817681)	Try modifying the distance between the source and the lens' surface, and the lens' aperture.
When the Progress bar seems to be blocked, it may correspond to a sub-operation on the modeler that cannot be precisely monitored. (TFS 702752)	
Opening an Optical Part Design project in two different Speos versions may present different results. (TFS 202587)	
Settings defined from the SpaceClaim **Display** tab are not kept after the recompute of an OPD feature. (TFS 569412)	
Optical Lens: When using script (only), the assignment of ID on faces is reverted between freestyle and non-freestyle Optical Lens, causing different selections. (TFS 599871)	
Optical Surface: When using script (only), the assignment of ID on faces is reverted between freestyle and non-freestyle Optical Surface, causing different selections. (TFS 599871)	
Link between OPD features and other Speos features are not necessarily preserved for some elements' faces that have specific statuses (example: sewing faces of the Optical Surface, back face of the Optical Lens, etc.).	Recreate the links when necessary.

DRAFT

Known Issue	Workaround or Solution
There is a current known problem with the curvature orientation (concave/convex) where stripes curvature can be inverted.(358255)	
Some performance issues might slow down the light guide generation. (202139)	
When prisms have the exact same width as the light guide's width, some prisms might fail to build. (TFS 238577 - SF 26584)	Decrease the prisms width (even of 0.001mm) and rebuild the feature.
The **Style Override** mode (**Transparent** or **Opaque**) is not kept when modifying and updating the feature. (TFS 202120)	

19.1.7. Head-Up Display

Known Issue	Workaround or Solution
The Rotation angle output of a HUD freeform mirror is not available.	
The unit used for the Ghost value is not consistent in every context. Different units can be displayed in the tree, in Workbench and/or in the simulation report. (TFS 336224)	
Trying to export a HOD freeform mirror into a CATIA part or product might break the mirror's borders into several splines. (TFS 240276 - SF 26661)	Use CATIA "Extract" command to merge the segments of each edge. - Or - Extrapolate the mirror and cut it to obtain your initial surface.

19.1.8. Results

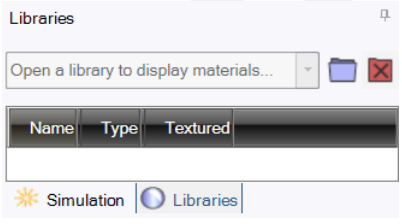
Known Issue	Workaround or Solution
When using an *.ies source file with a very peaky intensity distribution in an Interactive Simulation or a LiDAR Simulation, if you generate a Projected Grid out of the simulation, then the Projected Grid may not be visible in the 3D view due to the peaky distribution. (TFS 652669 SF 38651)	
Results of Sequence Detection (*.OptHash *.OptSequence) are not removed if you run again the simulation with the sensor **Layer** parameter set to another value than **Sequence**. (TFS 674798)	

DRAFT

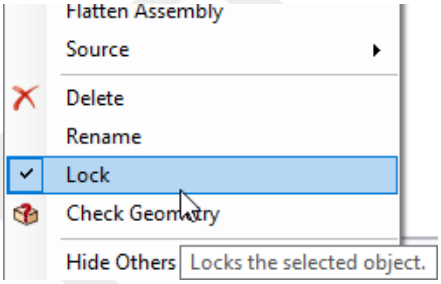
19.1.9. Automation

Known Issue	Workaround or Solution
Block Recording: Sometimes you cannot play the recorded blocks due to one or several hidden blocks in error. A message warns you of hidden blocks in error. (TFS 820381)	1. Right-click in the Block Recording panel and deactivate Hide Non-Model Changing Blocks to show hidden blocks. 2. Delete the hidden blocks in error.
Speos Light Box password: some properties have been removed fro the API Automation Interface which may have an impact on your scripting projects: • Light box export field: Password, and Active Password. Contextual Action: Generate Password. • Lightbox import fields: Password • Speos pattern fields: Password (TFS 683823)	
Optical Lens: When using script (only), the assignment of ID on faces is reverted between freestyle and non-freestyle Optical Lens, causing different selections. (TFS 599871)	
Optical Surface: When using script (only), the assignment of ID on faces is reverted between freestyle and non-freestyle Optical Surface, causing different selections. (TFS 599871)	
Running an inverse simulation in debug mode sometimes never ends (the progression bar is stopped during the first simulation).	Run the script using the **Run** mode.

19.1.10. Miscellaneous

Known Issue	Workaround or Solution
When the **Libraries** panel is located in the same group as the **Structure** panel, if you click **Material Libraries**  to hide the Libraries panel, the panel hides only after you stop hovering over the Material Libraries button. (TFS 601790) 	Move the Libraries panel into a group that does not contain the Structure panel.
Optical Library: ANSYS_OPTICAL_LIBRARIES_202XRX_WINX64.zip may fail to be unzipped with the Windows standard zip function due to some filepath being too long. (TFS 883546 SF 63321)	Use 7-zip.

DRAFT

Known Issue	Workaround or Solution
Speos Templates (*.scdot) are not supported. You can neither save as nor open a *.scdot file. If you try to save as *.scdot, the file will be automatically saved as *.scdocx instead. (SF 61626)	
When using a SpaceClaim document that references an external document, if SpaceClaim is not opened the process to copy the external document into the "otherDocs" folder is not done. (810170)	Make sure to open Spaceclaim (via Workbench) before saving the Workbench project. Then you can save the Workbench project.
Locked document are not supported. (TFS 812291 SF 56983)	Unlock document to modify your Speos project. Beware that unlocking must be performed in the same version as the version in which it has been locked. 
General: Speos applications only support ASCII / ANSI / UTF-8 file formats for all input text files (*.spectrum, *.ies, *.bsdf, etc.). (TFS 784679)	
Licensing Management: opening a version 2022 R2 then opening a version 2023 R1 does not work. (TFS 730734)	First open the version 2023 R1, then open the version 2022 R2.
Axes are not exported when saving a project to another format than scdocx. (TFS 656985 - SF 38823)	
When a project is added to a geometry in Workbench, the Output Files are not copied. (TFS 202762)	
The following characters are not supported in Speos features' name: **" < > | : * ? \ /** (TFS 263147)	
After opening a component (Open Component) from the Structure tree, Speos objects added this component are not displayed in the tree. (TFS 449252)	Refresh the tree.
After a Speos tree refresh, the UV Mapping and Local Meshing sections may not appear at the same place in the tree. (TFS 449445)	
Speos session closes when a Speos instance is opened from workbench.(TFS 372481)	
Unwanted 3D view rendering artifacts can be seen with certain geometries when the distance from origin of the scene to the point of view is very far away. (TFS 435735 SF 32841)	
If you import an external CAD part using the **Open file** command, no CAD Update link is created; then you cannot use the Geometry Update tool to update the part. (TFS 408007 SF 32027)	

DRAFT

Known Issue	Workaround or Solution
In some cases, a version upgrade might prevent Speos block from being correctly loaded in Ansys Workbench.	Delete or adjust the version of the following environment variables: **SPEOSSC_BIN_PATH**, **SPACECLAIM_PATH**. Then, reload the Speos block in Ansys Workbench.
When migrating a project created before 2021 R1 version, some features may appear in error "Invalid link context [...]" and may have lost their link to sub features.(TFS 326138)	
When working with heavy projects, Speos might show a performance loss for Speos objects edition. (TFS 252829)	
Importing models containing heavily faceted parts (parts containing a lot of small faces) may take a large amount of time and a size difference between the original model and the one imported can be observed.(TFS 259625)	
When using a comma as decimal separator, trying to modify values from the definition panel causes all decimals to be erased. (TFS 202092)	From your regional settings (Control Panel>Clock and Region>Region>Additional Settings) change the decimal symbol from "," (comma) to "." (dot).
In some cases, when defining sources, surfaces can only be selected from the 3D view and not from the structure tree. (TFS 202224)	
In some case, importing Complex/heavy data in Speos might be lengthy. (TFS 202463 - SF 24742)	
Moving a body through a copy/paste in the structure tree might break its relationship to Speos materials. (TFS 202463)	
When importing a HUD feature from Speos for CATIA V5, the HUD elements fail to be imported with the geometry. (TFS 214218)	

19.2. Error Messages

19.2.1. Not enough Speos HPC Licenses

Problem

The following message is displayed:

"Licensing Error

Not enough Speos HPC licenses"

DRAFT

Cause

The parameter **Number of threads** takes into account the cores of the machine first, then the threads, leading to the error if your license has a lower number of core than your computer.

Example: Your computer has 6 cores (12 threads) and your license 4 cores. If you define **Number of threads** to more than 4, let's say 8, the parameter first checks the 6 cores of the machine to assign them to the license, then 2 threads. But you only have a 4-core license, which leads to the error.

Solution

From the Speos options, specify the number of threads that are available in your license:

1. Go to **File** > **Speos Options**.
2. In the **Light Simulation** section, adjust the **Number of threads** so that they match the number of threads that are available in your license.

   > **Note:** While needing to define a number of threads to be used for simulation, it is actually cores that are going to be taken from the license.

3. Compute the simulation again.

19.2.2. Proportional to Body size STEP and SAG parameters are not respected

Problem

The following message is displayed:

"Proportional to Body size" STEP and SAG parameters are not respected.

After an external part import, some microscopic or empty bodies may have been created. When using a "Proportional to Body size" meshing on these bodies, ratios "size of body"/STEP and "size of body"/SAG are set to a minimum of 10nm, leading to larger meshing on those bodies.

Please review:

- Value of SAG
- Value of STEP
- Bodies that may be too small or empty

Cause

After an external part import, some microscopic or empty bodies may have been created. When using a "Proportional to Body size" meshing on these bodies, ratios "size of body"/STEP and "size of body"/SAG are set to a minimum of 10nm, leading to larger meshing on those bodies.

Solution

If the solutions provided in the error message do not work, try the following solution:

In the **Repair** tab, click **Small Faces**.

DRAFT

Note: To find the body which size is not proportional, in the **Measure** tab, you can use the **Measure** tool.

Note: The rounded value done does not impact the simulation as the light behavior on a body with a bounding box lower than 10nm is not managed as the body is lower than the wavelength (which otherwise would create artifacts).

19.2.3. Surface Extrapolated

Problem

The following message is displayed:

"Surface Extrapolated"

Cause

The technique used to calculate the image position requires to launch some rays around the image. When the constraints are tensed in the HUD Optical Design (HOD), the image projection may be very close to the mirror sides. That means some vignetting may happen: some of the additional rays launched for HUD Optical Analysis (HOA) calculation may miss the border of the mirror.

Solution

1. Modify the mirror size and/or the pupil diameter.
2. Analyze the system with HOA.
3. Iterate until you get a correct system.

19.2.4. Invalid Support: Offset Support Is Not Possible

Problem

The defined Freestyle Lens cannot be generated on the given support.

Cause

Solution

1.
2.
3.

DRAFT

20.1.1. Copyright and Trademark Information

© 2024 ANSYS, Inc. Unauthorized use, distribution or duplication is prohibited.

ANSYS, ANSYS Workbench, AUTODYN, CFX, FLUENT and any and all ANSYS, Inc. brand, product, service and feature names, logos and slogans are registered trademarks or trademarks of ANSYS, Inc. or its subsidiaries located in the United States or other countries. ICEM CFD is a trademark used by ANSYS, Inc. under license. CFX is a trademark of Sony Corporation in Japan. All other brand, product, service and feature names or trademarks are the property of their respective owners. FLEXlm and FLEXnet are trademarks of Flexera Software LLC.

Disclaimer Notice

THIS ANSYS SOFTWARE PRODUCT AND PROGRAM DOCUMENTATION INCLUDE TRADE SECRETS AND ARE CONFIDENTIAL AND PROPRIETARY PRODUCTS OF ANSYS, INC., ITS SUBSIDIARIES, OR LICENSORS. The software products and documentation are furnished by ANSYS, Inc., its subsidiaries, or affiliates under a software license agreement that contains provisions concerning non-disclosure, copying, length and nature of use, compliance with exporting laws, warranties, disclaimers, limitations of liability, and remedies, and other provisions. The software products and documentation may be used, disclosed, transferred, or copied only in accordance with the terms and conditions of that software license agreement

ANSYS, Inc. and ANSYS Europe, Ltd. are UL registered ISO 9001: 2015

U.S. Government Rights

For U.S. Government users, except as specifically granted by the Ansys, Inc. software license agreement, the use, duplication, or disclosure by the United States Government is subject to restrictions stated in the Ansys, Inc. software license agreement and FAR 12.212 (for non-DOD licenses).

Third-Party Software

See the **legal information** in the product help files for the complete Legal Notice for ANSYS proprietary software and third-party software. If you are unable to access the Legal Notice, contact Ansys, Inc.

Published in the U.S.A.

Protected by US Patents 7,639,267, 7,733,340, 7,830,377, 7,969,435, 8,207,990, 8,244,508, 8,253,726, 8,330,775, 10,650,172, 10,706,623, 10,769,850, D916,099, D916,100, 11,269,478, 11,475,184, and 2023/0004695.

Copyright © 2003-2024 ANSYS, Inc. All Rights Reserved. SpaceClaim is a registered trademark of ANSYS, Inc.

Portions of this software Copyright © 2010 Acresso Software Inc. FlexLM and FLEXNET are trademarks of Acresso Software Inc.

Portions of this software Copyright © 2008 Adobe Systems Incorporated. All Rights Reserved. Adobe and Acrobat are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries

Ansys Workbench and GAMBIT and all other ANSYS, Inc. product names are trademarks or registered trademarks of ANSYS, Inc. or its subsidiaries in the United States or other countries.

Contains BCLS (Bound-Constrained Least Squares) Copyright (C) 2006 Michael P. Friedlander, Department of Computer Science, University of British Columbia, Canada, provided under a LGPL 3 license which is included in the SpaceClaim installation directory (lgpl-3.0.txt). Derivative BCLS source code available upon request.

Contains SharpZipLib Copyright © 2009 C#Code

Anti-Grain Geometry Version 2.4 Copyright © 2002-2005 Maxim Shemanarev (McSeem).

Some SpaceClaim products may contain Autodesk® RealDWG by Autodesk, Inc., Copyright © 1998-2010 Autodesk, Inc. All rights reserved. Autodesk, AutoCAD, and Autodesk Inventor are registered trademarks and RealDWG is a trademark of Autodesk, Inc.

CATIA is a registered trademark of Dassault Systèmes.

Portions of this software Copyright © 2010 Google. SketchUp is a trademark of Google.

Portions of this software Copyright © 1999-2006 Intel Corporation. Licensed under the Apache License, Version 2.0. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0.

Contains DotNetBar licensed from devcomponents.com.

KeyShot is a trademark of Luxion ApS.

MatWeb is a trademark of Automation Creations, Inc.

2008 Microsoft ® Office System User Interface is licensed from Microsoft Corporation. Direct3D, DirectX, Microsoft PowerPoint, Excel, Windows, Windows Vista and the Windows Vista Start button are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Portions of this software Copyright © 2005 Novell, Inc. (http://www.novell.com)

Creo Parametric and PTC are registered trademarks of Parametric Technology Corporation.

Persistence of Vision Raytracer and POV-Ray are trademarks of Persistence of Vision Raytracer Pty. Ltd.

Portions of this software Copyright © 1993-2009 Robert McNeel & Associates. All Rights Reserved. openNURBS is a trademark of Robert McNeel & Associates. Rhinoceros is a registered trademark of Robert McNeel & Associates.

Portions of this software Copyright © 2005-2007, Sergey Bochkanov (ALGLIB project). *

Portions of this software are owned by Siemens PLM © 1986-2011. All Rights Reserved. Parasolid and Unigraphics are registered trademarks and JT is a trademark of Siemens Product Lifecycle Management Software, Inc.

This work contains the following software owned by Siemens Industry Software Limited: D-CubedTM 2D DCM © 2021. Siemens. All Rights Reserved.

SOLIDWORKS is a registered trademark of SOLIDWORKS Corporation.

Portions of this software are owned by Spatial Corp. © 1986-2011. All Rights Reserved. ACIS and SAT are registered trademarks of Spatial Corp.

Contains Teigha for .dwg files licensed from the Open Design Alliance. Teigha is a trademark of the Open Design Alliance.

Development tools and related technology provided under license from 3Dconnexion. © 1992 – 2008 3Dconnexion. All rights reserved.

TraceParts is owned by TraceParts S.A. TraceParts is a registered trademark of TraceParts S.A.

Contains a modified version of source available from Unicode, Inc., copyright © 1991-2008 Unicode, Inc. All rights reserved. Distributed under the Terms of Use in http://www.unicode.org/copyright.html.

Portions of this software Copyright © 1992-2008 The University of Tennessee. All rights reserved. [1]

Portions of this software Copyright © XHEO INC. All Rights Reserved. DeployLX is a trademark of XHEO INC.

This software incorporates information provided by American Institute of Steel Construction (AISC) for shape data available at http://www.aisc.org/shapesdatabase.

This software incorporates information provided by ArcelorMittal® for shape data available at http://www.sections.arcelormittal.com/products-services/products-ranges.html.

All other trademarks, trade names or company names referenced in SpaceClaim software, documentation and promotional materials are used for identification only and are the property of their respective owners.

DRAFT

*Additional notice for LAPACK and ALGLIB Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:-Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.-Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.-Neither the name of the copyright holders nor the names of its contributors may be used to endorse promote products derived from this software without specific prior written permission.

BCLS is licensed under the GNU Lesser General Public License (GPL) Version 3, Copyright (C) 2006 Michael P. Friedlander, Department of Computer Science, University of British Columbia, Canada. A copy of the LGPL license is included in the installation directory (lgpl-3.0.txt).

Please contact open.source@ansys.com for a copy of the source code for BCLS.

Eigen is licensed under the Mozilla Public License (MPL) Version 2.0, the text of which can be found at: https://www.mozilla.org/media/MPL/2.0/index.815ca599c9df.txt. Please contact open.source@ansys.com for a copy of the Eigen source code.

HDF5 (Hierarchical Data Format 5) Software Library and Utilities

Copyright (c) 2006, The HDF Group.

NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities

Copyright (c) 1998-2006, The Board of Trustees of the University of Illinois.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted for any purpose (including commercial purposes) provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or materials provided with the distribution.

3. In addition, redistributions of modified forms of the source or binary code must carry prominent notices stating that the original code was changed and the date of the change.

4. All publications or advertising materials mentioning features or use of this software are asked, but not required, to acknowledge that it was developed by The HDF Group and by the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign and credit the contributors.

5. Neither the name of The HDF Group, the name of the University, nor the name of any Contributor may be used to endorse or promote products derived

from this software without specific prior written permission from The HDF Group, the University, or the Contributor, respectively.

DISCLAIMER:

THIS SOFTWARE IS PROVIDED BY THE HDF GROUP AND THE CONTRIBUTORS "AS IS" WITH NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED. In no

event shall The HDF Group or the Contributors be liable for any damages suffered by the users arising out of the use of this software, even if advised of the possibility of such damage. Anti-Grain Geometry - Version 2.4 Copyright (C) 2002-2004 Maxim Shemanarev (McSeem)

Permission to copy, use, modify, sell and distribute this software is granted provided this copyright notice appears in all copies. This software is provided "as is" without express or implied warranty, and with no claim as to its suitability for any purpose.

Some ANSYS-SpaceClaim products may contain Autodesk® RealDWG by Autodesk, Inc., Copyright © 1998-2010 Autodesk, Inc. All rights reserved. Autodesk, AutoCAD, and Autodesk Inventor are registered trademarks and RealDWG is a trademark of Autodesk, Inc.

CATIA is a registered trademark of Dassault Systèmes.

Portions of this software Copyright © 2013 Trimble. SketchUp is a trademark of Trimble Navigation Limited.

This software is based in part on the work of the Independent JPEG Group.

Portions of this software Copyright © 1999-2006 Intel Corporation. Licensed under the Apache License, Version 2.0. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0

Contains DotNetBar licensed from devcomponents.com.

Portions of this software Copyright © 1988-1997 Sam Leffler and Copyright (c) 1991-1997 Silicon Graphics, Inc.

KeyShot is a trademark of Luxion ApS.

MatWeb is a trademark of Automation Creations, Inc.

2010 Microsoft ® Office System User Interface is licensed from Microsoft Corporation. Direct3D, DirectX, Microsoft PowerPoint, Excel, Windows/Vista/Windows 7/Windows 8/Windows 10 and their respective Start Button designs are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Portions of this software Copyright © 2005 Novell, Inc. (Licensed at http://stuff.mit.edu/afs/athena/software/mono_v3.0/arch/i386_linux26/mono/mcs/class/Managed.Windows.Forms/System.Windows.Forms.RTF/)

Pro/ENGINEER and PTC are registered trademarks of Parametric Technology Corporation.

POV-Ray is available without charge from http://www.pov-ray.org. No charge is being made for a grant of the license to POV-Ray.

POV-Ray License Agreement

DISTRIBUTOR'S LICENCE AGREEMENT

Persistence of Vision Raytracer(tm) (POV-Ray(tm))

13 August 2004

Licensed Versions: Versions 3.5 and 3.6

Please read through the terms and conditions of this license carefully. This is a binding legal agreement between you, the "Distributor" and Persistence of Vision Raytracer Pty. Ltd. ACN 105 891 870 ("POV"), a company incorporated in the state of Victoria, Australia, for the product known as the "Persistence of Vision Raytracer(tm)", also referred to herein as "POV-Ray(tm)". The terms of this agreement are set out at http://www.povray.org/distribution-license.html ("Official Terms"). The Official Terms take precedence over this document to the extent of any inconsistency.

1. INTRODUCTION

1.1. In this agreement, except to the extent the context requires otherwise, the following capitalized terms have the following meanings:

(a) Distribution means:

(i) a single item of a distribution medium, including a CD Rom or DVD Rom, containing software programs and/or data;

(ii) a set of such items;

(iii) a data file in a generally accepted data format from which such an item can be created using generally available standard tools;

(iv) a number of such data files from which a set of such items can be created; or

(v) a data file in a generally accepted data storage format which is an archive of software programs and/or data;

(b) Derived Code means all software which is derived from or is an adaptation of any part of the Software other than a scene file;

(c) Intellectual Rights means:

(i) all copyright, patent, trade mark, trade secret, design, and circuit layout rights;

(ii) all rights to the registration of such rights; and

(iii) all rights of a similar nature which exist anywhere in the world;

(d) Licensed Version means the version set out at the top of this agreement against the heading "Licensed Version" and all minor releases of this version (ie releases of the form x.y.z);

(e) POV Associate means any person associated directly or indirectly with POV whether as a director, officer, employee, subcontractor, agent, representative, consultant, licensee or otherwise;

(f) Modification Terms means the most recent version from time to time of the document of that name made available from the Site (g) Revocation List means the list of that name linked to from the Official Terms;

(h) Site means www.povray.org;

(i) Software means the Licensed Version of the Persistence of Vision Raytracer(tm) (also known as POV-Ray(tm)) (including all POV-Ray program source files, executable (binary) files, scene files, documentation files, help files, bitmaps and other POV-Ray files associated with the Licensed Version) in a form made available by

POV on the Site;

(j) User Licence means the most recent version from time to time of the document of that name made available from the Site.

2. OPEN SOURCE DISTRIBUTIONS

2.1. In return for the Distributor agreeing to be bound by the terms of this agreement, POV grants the Distributor permission to make a copy of the Software by including the Software in a generally recognised Distribution of a recognised operating system where the kernel of that operating system is made available under licensing terms:

(a) which are approved by the Open Source Initiative (www.opensource.org) as complying with the "Open Source Definition" put forward by the Open Source Initiative; or

(b) which comply with the "free software definition" of the Free Software Foundation (www.fsf.org). 2.2. As at June 2004, and without limiting the generality of the term, each of the following is a "generally recognised Distribution" for the purposes of clause 2.1: Debian, Red Hat (Enterprise and Fedora), SuSE, Mandrake, Xandros, Gentoo and Knoppix Linux distributions, and officially authorized distributions of the FreeBSD, OpenBSD, and NetBSD projects.

2.3. Clause 2.1 also applies to the Software being included in the above distributions 'package' and 'ports' systems, where such exist;

2.4. Where the Distributor reproduces the Software in accordance with clause 2.1:

(a) the Distributor may rename, reorganise or repackage (without omission) the files comprising the Software where such renaming, reorganisation or repackaging is necessary to conform to the naming or organisation scheme of the target operating environment of the Distribution or of an established package management system of the target operating environment of the Distribution; and (b) the Distributor must not otherwise rename, reorganise or repackage the Software.

3. DISTRIBUTION LICENCE

3.1. Subject to the terms and conditions of this agreement, and in return for Distributor agreeing to be bound by the terms of this agreement, POV grants the Distributor permission to make a copy of the Software in any of the following circumstances:(a) in the course of providing a mirror of the POV-Ray Site (or part of it), which is made available generally over the internet to each person without requiring that person to identify themselves and without any other restriction other than restrictions designed to manage traffic flows;

DRAFT

(b) by placing it on a local area network accessible only by persons authorized by the Distributor whilst on the Distributor's premises;

(c) where that copy is provided to a staff member or student enrolled at a recognised educational institution;

(d) by including the Software as part of a Distribution where:

(i) neither the primary nor a substantial purpose of the distribution of the Distribution is the distribution of the Software. That is, the distribution of the Software

is merely incidental to the distribution of the Distribution; and

(ii) if the Software was not included in the Distribution, the remaining software and data included within the Distribution would continue to function effectively and

according to its advertised or intended purpose;

(e) by including the Software as part of a Distribution where:

(i) there is no data, program or other files apart from the Software on the Distribution;

(ii) the Distribution is distributed by a person to another person known to that person; or

(iii) the Distributor has obtained explicit written authority from POV to perform the distribution, citing this clause number, prior to the reproduction being

made.

3.2. In each case where the Distributor makes a copy of the Software in accordance with clause 3.1, the Distributor must, unless no payment or other consideration of any type is received by Distributor in relation to the Distribution:

(a) ensure that each person who receives a copy of the Software from the Distributor is aware prior to acquiring that copy:

(i) of the full name and contact details of the Distributor, including the Distributor's web site, street address, mail address, and working email address;

(ii) that the Software is available without charge from the Site;

(iii) that no charge is being made for the granting of a licence over the Software.

(b) include a copy of the User Licence and this Distribution License with the copy of the Software. These licences must be stored in the same subdirectory on the distribution medium as the Software and named in such a way as to prominently identify their purpose;

3.3. The Distributor must not rename, reorganise or repackage any of the files comprising the Software without the prior written authority of POV.

3.4. Except as explicitly set out in this agreement, nothing in this agreement permits Distributor to make any modification to any part of the Software.

4. RESTRICTIONS ON DISTRIBUTION

4.1. Nothing in this agreement gives the Distributor: (a) any ability to grant any licence in respect of the use of the Software or any part of it to any person;

(b) any rights or permissions in respect of, including rights or permissions to distribute or permit the use of, any Derived Code;

(c) any right to bundle a copy of the Software (or part thereof), whether or not as part of a Distribution, with any other items, including books and magazines. POV may, in response to a request, by notice in writing and in its absolute discretion, permit such bundling on a case by case basis. This clause 4.1(c) does not apply to Distributions permitted under clause 2;

(d) any right, permission or authorisation to infringe any Intellectual Right held by any third party.

4.2. Distributor may charge a fee for the making or the provision of a copy of the Software.

DRAFT

4.3. Where the making, or the provision, of a copy of the Software is authorised under the terms of clause 3 but not under those of clause 2 of this agreement, the total of all fees charged in relation to such making or provision and including all fees (including shipping and handling fees) which are charged in respect

of any software, hardware or other material provided in conjunction with or in any manner which is reasonably connected with the making, or the provision, of a copy of the Software must not exceed the reasonable costs incurred by the Distributor in making the reproduction, or in the provision, of that copy for which the fee

is charged.

4.4. Notwithstanding anything else in this agreement, nothing in this agreement permits the reproduction of any part of the Software by, or on behalf of:

(a) Any person currently listed on the Revocation List from time to time;

(b) Any related body corporate (as that term is defined in section 50 of the Corporations Law 2001 (Cth)) of any person referred to in clause 4.4(a);

(c) Any person in the course of preparing any publication in any format (including books, magazines, CD Roms or on the internet) for any of the persons identified in paragraph (a);

(d) Any person who is, or has been, in breach of this Agreement and that breach has not been waived in writing signed by POV; or

(e) Any person to whom POV has sent a notice in writing or by email stating that that person may not distribute the Software.

4.5. From the day two years after a version of the Software more recent than the Licensed Version is made available by POV on the Site clause 3 only permits reproduction of the Software where the Distributor ensures that each recipient of such a reproduction is aware, prior to obtaining that reproduction, that that reproduction of the Software is an old version of the Software and that a more recent version of the Software is available from the Site.

5. COPYRIGHT AND NO LITIGATION

5.1. Copyright subsists in the Software and is protected by Australian and international copyright laws.

5.2. Nothing in this agreement gives Distributor any rights in respect of any Intellectual Rights in respect of the Software or which are held by or on behalf of POV. Distributor acknowledges that it does not acquire any rights in respect of such Intellectual Rights.

5.3. Distributor acknowledges that if it performs out any act in respect of the Software without the permission of POV it will be liable to POV for all damages POV may suffer (and which Distributor acknowledges it may suffer) as well as statutory damages to the maximum extent permitted by law and that it may also be liable to

criminal prosecution.

5.4. Distributor must not commence any action against any person alleging that the Software or the use or distribution of the Software infringes any rights, including Intellectual Rights of the Distributor or of any other person. If Distributor provides one or more copies of the Software to any other person in accordance with the agreement, Distributor waives all rights it has, or may have in the future, to bring any action, directly or indirectly, against any person to the extent that such an action relates to an infringement of any rights, including Intellectual Rights of any person in any way arising from, or in relation to, the use, or distribution, (including through the authorisation of such use or distribution) of:(a) the Software;

(b) any earlier or later version of the Software; or

(c) any other software to the extent it incorporates elements of the software referred to in paragraphs (a) or (b) of this clause

5.4.

6. DISCLAIMER OF WARRANTY

6.1. To the extent permitted by law, all implied terms and conditions are excluded from this agreement. Where a term or condition is implied into this agreement and that term cannot be legally excluded, that term has effect as a term or condition of this agreement. However, to the extent permitted by law, the liability

of POV for a breach of such an implied term or condition is limited to the fullest extent permitted by law.

6.2. To the extent permitted by law, this Software is provided on an "AS IS" basis, without warranty of any kind, express or implied, including without limitation, any implied warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property of any third party. The Software has inherent limitations including design faults and programming bugs.

6.3. The entire risk as to the quality and performance of the Software is borne by Distributor, and it is Distributor's responsibility to ensure that the Software fulfils Distributor's requirements prior to using it in any manner (other than testing it for the purposes of this paragraph in a non-critical and non-production environment), and prior to distributing it in any fashion.

6.4. This clause 6 is an essential and material term of, and cannot be severed from, this agreement. If Distributor does not or cannot agree to be bound by this clause, or if it is unenforceable, then Distributor must not, at any time, make any reproductions of the Software under this agreement and this agreement gives the

Distributor no rights to make any reproductions of any part of the Software.

7. NO LIABILITY

7.1. When you distribute or use the Software you acknowledge and accept that you do so at your sole risk. Distributor agrees that under no circumstances will it have any claim against POV or any POV Associate for any loss, damages, harm, injury, expense, work stoppage, loss of business information, business interruption,

computer failure or malfunction which may be suffered by you or by any third party from any cause whatsoever, howsoever arising, in connection with your use or distribution of the Software even where POV was aware, or ought to have been aware, of the potential of such loss.

7.2. Neither POV nor any POV Associate has any liability to Distributor for any indirect, general, special, incidental, punitive and/or consequential damages arising as a result of a breach of this agreement by POV or which arises in any way related to the Software or the exercise of a licence granted to Distributor under this

agreement.

7.3. POV's total aggregate liability to the Distributor for all loss or damage arising in any way related to this agreement is limited to the lesser of: (a) AU$100, and (b) the amount received by POV from Distributor as payment for the grant of a licence under this agreement.

7.4. Distributor must bring any action against POV in any way related to this agreement or the Software within 3 months of the cause of action first arising. Distributor waives any right it has to bring any action against POV and releases POV from all liability in respect of a cause of action if initiating process in relation to that action is not served on POV within 3 months of the cause of action arising. Where a particular set of facts give rise to more than one cause of action this clause 7.4 applies as if all such causes of action arise at the time the first such cause of action arises.

7.5. This clause 7 is an essential and material term of, and cannot be severed from, this agreement. If Distributor does not or cannot agree to be bound by this clause, or if it is unenforceable, then Distributor must not, at any time, make any reproductions of the Software under this agreement and this agreement gives the Distributor no rights to make any reproductions of any part of the Software.

8. INDEMNITY

8.1. Distributor indemnifies POV and each POV Associate and holds each of them harmless against all claims which arise from any loss, damages, harm, injury, expense, work stoppage, loss of business information, business interruption, computer failure or malfunction, which may be suffered by Distributor or any other

party whatsoever as a consequence of:

(a) any act or omission of POV and/or any POV Associate, whether negligent or not;

(b) Distributor's use and/or distribution of the Software; or

(c) any other cause whatsoever, howsoever arising, in connection with the Software. This clause 8 is binding on Distributor's estate, heirs, executors, legal successors, administrators, parents and/or guardians.

8.2. Distributor indemnifies POV, each POV Associate and each of the authors of any part of the Software against all loss and damage and for every other consequence flowing from any breach by Distributor of any Intellectual Right held by POV.

8.3. This clause 8 constitutes an essential and material term of, and cannot be severed from, this agreement. If Distributor does not or cannot agree to be bound by this clause, or if it is unenforceable, then Distributor must not, at any time, make any reproductions of the Software under this agreement and this agreement gives the Distributor no rights to make any reproductions of any part of the Software.

9. HIGH RISK ACTIVITIES

9.1. This Software and the output produced by this Software is not fault-tolerant and is not designed, manufactured or intended for use as on-line control equipment in hazardous environments requiring fail-safe performance, in which the failure of the Software could lead or directly or indirectly to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). POV specifically disclaims all express or implied warranty of fitness for High Risk Activities and, notwithstanding any other term of this agreement, explicitly prohibits the use or distribution of the Software for such purposes.

10. ENDORSEMENT PROHIBITION

10.1. Distributor must not, without explicit written permission from POV, claim or imply in any way that:

(a) POV or any POV Associate officially endorses or supports the Distributor or any product (such as CD, book, or magazine) associated with the Distributor or any reproduction of the Software made in accordance with this agreement; or(b) POV derives any benefit from any reproduction made in accordance with this agreement.

11. TRADEMARKS

11.1. "POV-Ray(tm)", "Persistence of Vision Raytracer(tm)" and "POV-Team(tm)" are trademarks of Persistence of Vision Raytracer Pty. Ltd. Any other trademarks referred to in this agreement are the property of their respective holders. Distributor must not use, apply for, or register anywhere in the world, any word, name

(including domain names), trade mark or device which is substantially identical or deceptively or confusingly similar to any of Persistence of Vision Raytracer Pty. Ltd's trade marks.

12. MISCELLANEOUS

12.1. The Official Terms, including those documents incorporated by reference into the Official Terms, and the Modification Terms constitute the entire agreement between the parties relating to the distribution of the Software and, except where stated to the contrary in writing signed by POV, supersedes all previous

negotiations and correspondence in relation to it.

12.2. POV may modify this agreement at any time by making a revised licence available from the Site at http://www.povray.org/distribution-license.html.

This agreement is modified by replacing the terms in this agreement with those of the revised licence from the time that the revised licence is so made available. It is your responsibility to ensure that you have read and agreed to the current version of this agreement prior to distributing the Software.

12.3. Except where explicitly stated otherwise herein, if any provision of this Agreement is found to be invalid or unenforceable, the invalidity or unenforceability of such provision shall not affect the other provisions of this agreement, and all provisions not affected by such invalidity or unenforceability shall remain in

full force and effect. In such cases Distributor agrees to attempt to substitute for each invalid or unenforceable provision a valid or enforceable provision which achieves to the greatest extent possible, the objectives and intention of the invalid or unenforceable provision.

12.4. A waiver of a right under this agreement is not effective unless given in writing signed by the party granting that waiver. Unless otherwise stipulated in the waiver, a waiver is only effective in respect of the circumstances in which it is given and is not a waiver in respect of any other rights or a waiver in respect of

future rights or actions.

12.5. The validity and interpretation of this agreement is governed by the laws in force in the State of Victoria, Australia. Distributor submits to the exclusive jurisdiction of the courts of that State and courts located within that State exercising federal jurisdiction.

12.6. References in this agreement to "written" and "writing" mean on paper or by fax and expressly exclude email and other forms of electronic communication.

13. CONTACT INFORMATION

13.1. This clause 13 does not form part of the agreement. License inquiries can be made via email; please use the following address (but see 13.2 below prior to emailing) : team-coord-[three-letter month]-[four-digit year]@povray org. for example, team-coord-jun-2004@povray.org should be used if at the time you send the email it is the month of June 2004. The changing email addresses are necessary to combat spam. Old email addresses may be deleted at POV's discretion.

13.2. Note that the address referred to in 13.1 may change for reasons other than those referred to in that clause; please check the current version of this document at http://www.povray.org/distribution-license.html. for the current address. Your inability or failure to contact us is no excuse for violating the licence.

13.3. Do NOT send any email attachments of any sort other than by prior arrangement. Do not send email in HTML format. EMAIL MESSAGES INCLUDING ATTACHMENTS WILL BE DELETED UNREAD.

13.4. The following postal address is only for official license business. Please note that it is preferred that initial queries about licensing be made via email; postal mail should only be used when email is not possible, or when written documents are being exchanged by prior arrangement. While it is unlikely this address will change in the short term it would be advisable to check http://www.povray.org/distribution-license.html for the current one prior to sending postal mail.

Persistence of Vision Raytracer Pty. Ltd.

PO Box 407

Williamstown,

Victoria 3016

Australia

POV-Ray Licence Agreement

GENERAL LICENSE AGREEMENT

FOR PERSONAL USE

Persistence of Vision Ray Tracer (POV-Ray)

Version 3.6 License and Terms & Conditions of Use

version of 1 February 2005

(also known as POVLEGAL.DOC)

Please read through the terms and conditions of this license carefully. This license is a binding legal agreement between you, the 'User' (an individual or single entity) and Persistence of Vision Raytracer Pty. Ltd. ACN 105 891 870 (herein also referred to as the "Company"), a company incorporated in the state of Victoria, Australia, for the product known as the "Persistence of Vision Ray Tracer", also referred to herein as 'POV-Ray'.

YOUR ATTENTION IS PARTICULARLY DRAWN TO THE DISCLAIMER OF WARRANTY AND NO LIABILITY AND INDEMNITY PROVISIONS. TO USE THE PERSISTENCE OF VISION RAY TRACER ("POV-RAY") YOU MUST AGREE TO BE BOUND BY THE TERMS AND CONDITIONS SET OUT IN THIS DOCUMENT. IF YOU DO NOT AGREE TO ALL THE TERMS AND CONDITIONS OF USE OF POV-RAY SET OUT IN THIS LICENSE AGREEMENT, OR IF SUCH TERMS AND CONDITIONS ARE NOT BINDING ON YOU IN YOUR JURISDICTION, THEN YOU MAY NOT USE POV-RAY IN ANY MANNER. THIS GENERAL

DRAFT

LICENSE AGREEMENT MUST ACCOMPANY ALL POV-RAY FILES WHETHER IN THEIR OFFICIAL OR CUSTOM VERSION FORM. IT MAY NOT BE REMOVED OR MODIFIED. THIS GENERAL LICENSE AGREEMENT GOVERNS THE USE OF

POV-RAY WORLDWIDE. THIS DOCUMENT SUPERSEDES AND REPLACES ALL PREVIOUS GENERAL LICENSES.

INTRODUCTION

This document pertains to the use of the Persistence of Vision Ray Tracer (also known as POV-Ray). It applies to all POV-Ray program source files, executable (binary) files, scene files, documentation files, help files, bitmaps and other POV-Ray files contained in official Company archives, whether in full or any part thereof, and are herein referred to as the "Software". The Company reserves the right to revise these rules in future versions and to make additional rules to address new circumstances at any time. Such rules, when made, will be posted in a revised license file, the latest version of which is available from the Company website at

http://www.povray.org/povlegal.html.

USAGE PROVISIONS

Subject to the terms and conditions of this agreement, permission is granted to the User to use the Software and its associated files to create and render images. The creator of a scene file retains all rights to any scene files they create, and any images generated by the Software from them. Subject to the other terms of this license, the User is permitted to use the Software in a profit-making enterprise, provided such profit arises primarily from use of the Software and not from distribution of the Software or a work including the Software in whole or part.

Please refer to http://www.povray.org/povlegal.html for licenses covering distribution of the Software and works including the Software. The User is also granted the right to use the scene files, fonts, bitmaps, and include files distributed in the INCLUDE and SCENES\INCDEMO sub-directories of the Software in their own scenes. Such permission does not extend to any other files in the SCENES directory or its sub-directories. The SCENES files are for the User's enjoyment and education but may not be the basis of any derivative works unless the file in question explicitly grants permission to do such.

This licence does not grant any right of re-distribution or use in any manner other than the above. The Company has separate license documents that apply to other uses (such as re-distribution via the internet or on CD) ; please visit http://www.povray.org/povlegal.html for links to these. In particular you are advised that the sale, lease, or rental of the Software in any form without written authority from the Company is explicitly prohibited. Notwithstanding anything in the balance of this licence agreement, nothing in this licence agreement permits the installation or use of the Software in conjunction with any product (including software) produced or distributed by any party who is, or has been, in violation of this licence agreement or of the distribution licence (http://www.povray.org/distribution-license.html)

(or any earlier or later versions of those documents) unless:

a. the Company has explicitly released that party in writing from the consequences of their non compliance; or

b. both of the following are true:

i. the installation or use of the Software is without the User being aware of the abovementioned violation; and

ii. the installation or use of the Software is not a result (whether direct or indirect) of any request or action of the abovementioned party (or any of its products), any agent of that party (or any of their products), or any person(s) involved in supplying any such product to the User.

COPYRIGHT

Copyright © 1991-2003, Persistence of Vision Team.

Copyright © 2003-2004, Persistence of Vision Raytracer Pty. Ltd.

Windows version Copyright © 1996-2003, Christopher Cason.

Copyright subsists in this Software which is protected by Australian and international copyright laws. The Software is NOT PUBLIC DOMAIN. Nothing in this agreement shall give you any rights in respect of the intellectual property of the Company and you acknowledge that you do not acquire any rights in respect of such intellectual property rights. You acknowledge that the Software is the valuable intellectual property of the Company and that if you use,

modify or distribute the Software for unauthorized purposes or in an unauthorized manner (or cause or allow the forgoing to occur), you will be liable to the Company for any damages it may suffer (and which you acknowledge it may suffer) as well as statutory damages to the maximum extent permitted by law and also that you may be liable to

criminal prosecution. You indemnify the Company and the authors of the Software for every single consequence flowing from the aforementioned events.

DISCLAIMER OF WARRANTY

express or implied, including without limitation, any implied warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property of any third party. This Software has inherent limitations including design faults and programming bugs. The entire risk as to the quality and performance of the Software is borne by you, and it is your responsibility to ensure that it does what you require it to do prior to using it for any purpose (other than testing it), and prior to distributing it in any fashion. Should the Software prove defective, you agree that you alone assume the entire cost resulting in any way from such defect.

This disclaimer of warranty constitutes an essential and material term of this agreement. If you do not or cannot accept this, or if it is unenforceable in your jurisdiction, then you may not use the Software in any manner.

NO LIABILITY

When you use the Software you acknowledge and accept that you do so at your sole risk. You agree that under no circumstances shall you have any claim against the Company or anyone associated directly or indirectly with the Company whether as employee, subcontractor, agent, representative, consultant, licensee or otherwise ("Company Associates") for any loss, damages, harm, injury, expense, work stoppage, loss of business information, business interruption, computer failure or malfunction which may be suffered by you or by any third party from any cause whatsoever, howsoever arising, in connection with your use or distribution of the Software even where the Company were aware, or ought to have been aware, of the potential of such loss. Damages referred to above shall include direct, indirect, general, special, incidental, punitive and/or consequential. This disclaimer of liability constitutes an essential and material term of this agreement. If you do not or cannot accept this, or if it is unenforceable in your jurisdiction, then you may not use the Software.

INDEMNITY

You indemnify the Company and Company Associates and hold them harmless against any claims which may arise from any loss, damages, harm, injury, expense, work stoppage, loss of business information, business interruption, computer failure or malfunction, which may be suffered by you or any other party whatsoever as a consequence of any act or omission of the Company and/or Company Associates, whether negligent or not, arising out of your use and/or distribution of the Software, or from any other cause whatsoever, howsoever arising, in connection with the Software. These provisions are binding on your estate, heirs, executors, legal successors, administrators, parents and/or guardians.

This indemnification constitutes an essential and material term of this agreement. If you do not or cannot accept this, or if it is unenforceable in your jurisdiction, then you may not use the Software.

HIGH RISK ACTIVITIES

This Software and the output produced by this Software is not fault-tolerant and is not designed, manufactured or intended for use as on-line control equipment in hazardous environments requiring fail-safe performance, in which the failure of the Software could lead or directly or indirectly to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). The Company specifically disclaims any express or implied warranty of fitness for High Risk Activities and explicitly prohibits the use of the Software for such purposes.

CRYPTOGRAPHIC SIGNING OF DOCUMENTS

Changes to this Agreement and documents issued under its authority may be cryptographically signed by the POV-Ray Team Co-ordinator's private PGP key.

In the absence of evidence to the contrary, such documents shall be considered, under the terms of this Agreement, to be authentic provided the signature is

valid. The master copy of this Agreement at http://www.povray.org/povlegal.html will also be signed by the current version of the team-coordinator's key.

The public key for the POV-Ray Team-coordinator can be retrieved from the location https://secure.povray.org/keys/. The current fingerprint for it is

B4DD 932A C080 C3A3 6EA2 9952 DB04 4A74 9901 4518.

MISCELLANEOUS

This Agreement constitutes the complete agreement concerning this license. Any changes to this agreement must be in writing and may take the form of

notifications by the Company to you, or through posting notifications on the Company website. THE USE OF THIS SOFTWARE BY ANY PERSON OR ENTITY IS

EXPRESSLY MADE CONDITIONAL ON THEIR ACCEPTANCE OF THE TERMS SET FORTH HEREIN. Except where explicitly stated otherwise herein, if any provision of this

Agreement is found to be invalid or unenforceable, the invalidity or unenforceability of such provision shall not affect the other provisions of this agreement, and all provisions not affected by such invalidity or unenforceability shall remain in full force and effect. In such cases you agree to attempt to substitute for each invalid or unenforceable provision a valid or enforceable provision which achieves to the greatest extent possible, the objectives and intention of the invalid or unenforceable

provision. The validity and interpretation of this agreement will be governed by the laws of Australia in the state of Victoria (except for conflict of law provisions).

CONTACT INFORMATION

License inquiries can be made via email; please use the following address (but see below prior to emailing) : team-coord-[three-letter month]-[four-digit year]@povray.org for example, team-coord-jun-2004@povray.org should be used if at the time you send the email it is the month of June 2004. The changing email addresses are necessary to combat spam and email viruses. Old email addresses may be deleted at our discretion.

Note that the above address may change for reasons other than that given above; please check the version of this document at http://www.povray.org/povlegal.html for the current address. Note that your inability or failure to contact us for any reason is not an excuse for violating this licence.

Do NOT send any attachments of any sort other than by prior arrangement.

EMAIL MESSAGES INCLUDING ATTACHMENTS WILL BE DELETED UNREAD.

The following postal address is only for official license business. Please note that it is preferred that initial queries about licensing be made via email ; postal mail should only be used when email is not possible, or when written documents are being exchanged by prior arrangement.

Persistence of Vision Raytracer Pty. Ltd.

PO Box 407

Williamstown,

Victoria 3016

Australia

Portions of this software are owned by Siemens PLM © 1986-2013. All Rights Reserved. Parasolid, Unigraphics, and SolidEdge are registered trademarks and JT is a trademark of Siemens Product Lifecycle Management Software, Inc.

SolidWorks is a registered trademark of SolidWorks Corporation.

Portions of this software are owned by Spatial Corp. © 1986-2013. All Rights Reserved. ACIS, SAT and SAB are registered trademarks of Spatial Corp.

Contains Teigha for .dwg files licensed from the Open Design Alliance. Teigha is a trademark of the Open Design Alliance.

Development tools and related technology provided under license from 3Dconnexion. © 1992 – 2008 3Dconnexion. All rights reserved.

•TraceParts is owned by TraceParts S.A. TraceParts is a registered trademark of TraceParts S.A.

Copyright © 1991-2017 Unicode, Inc. All rights reserved.

Distributed under the Terms of Use in http://www.unicode.org/copyright.html. Permission is hereby granted, free of charge, to any person obtaining a copy of the Unicode data files and any associated documentation (the "Data Files") or Unicode software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that either (a) this copyright and permission notice appear with all copies of the Data Files or Software, or

(b) this copyright and permission notice appear in associated Documentation.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE

WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.

Portions of this software Copyright © 1992-2008 The University of Tennessee. All rights reserved.

This product includes software developed by XHEO INC (http://xheo.com).

Portions of this software are owned by Tech Soft 3D, Inc. Copyright © 1996-2013. All rights reserved. HOOPS is a registered trademark of Tech Soft 3D, Inc.

Portions of this software are owned by MachineWorks Limited. Copyright ©2013. All rights reserved. Polygonica is a registered trademark of MachineWorks Limited.

Apache License

Version 2.0, January 2004 http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition,

"Control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition,

"Submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License.

You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution

intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely

responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

DRAFT