



Contents

Legals	1
Preface	3
About this User Guide	3
Note on this User Guide	3
Looking for Help?	3
What's new?	5
Installation	9
Minimum System Requirements	9
Windows and Linux	9
Mac OS X	10
Installing Clarisse	10
Installation on Windows	10
Requirements	10
Installing Microsoft Visual C++ 2010 x64 Redistributable	10
Installing Python	10
Installation on Linux	11
Installation on Mac OS X	11
Python in Clarisse	11
Specifying Python Location	11
Licensing	13
Obtaining a License	13
ILISE License Server	15
Installing Licenses	16
Using LICINFO	16
Setting ILISE location to Clarisse	17

Command Line Option	17
Using LICMAN	17
Configuration	18
Licensing Type	18
Node Locked Licensing	18
Installing a Node Locked License	20
Floating Licensing	20
Ilise Port	21
Ilise Location (Local)	21
Starting Ilise (Local)	21
Restarting Ilise	21
Shutting Down Ilise (Local)	22
Testing the Connection	22
Getting License Information	22
Saving Connection Settings	23
Licensing	23
Requesting Licenses	24
Installing Floating Licenses	24
Diagnostic	24
Tutorial: Installing a floating license locally	25
Getting a license file	25
Installing the license	26
Starting ILISE locally	28
Using Clarisse	29
Command-Line Options	29
Configuration and Environment	30
Understanding the Workflow	30
Clarisse User Interface Basics	32
Clarisse Main Window	33
Menu Bar	33

Selection Toolbar	34
Selection Field	35
Multiselection	35
Navigation Buttons	36
Selection History	36
Understanding Viewports	37
Working with Widgets	37
Splitting Viewports	39
Widget Custom Selection	40
Shelves	42
Adding a new script	42
Editing an existing script	43
Removing a script	43
Customizing Display	43
Resetting Configuration	43
Slots	44
Drag and Drop	44
Hiding Main Shelf	45
Status Bar	45
Evaluation Status	45
Customizing the Interface	46
Using Workshops	46
Creating new main windows	46
Creating Widgets in sub-windows	46
Changing User Interface Color Scheme	47
Changing User Interface Color Display	48
File Browser	50
Overview	50
Favorites	50
Working with Objects	51
Understanding Objects	51

Objects Classes	51
Class Inheritance	51
Role while connecting Objects	52
Browsing the Class Hierarchy	53
Creating Objects	54
Selecting Objects	55
Selecting Dependencies	55
Input Dependencies	55
Output Dependencies	56
Selection Tools	56
Renaming Objects	57
Limitations for Naming Objects	57
Deleting Objects	57
Moving Objects	58
Editing Objects	58
Attributes	58
Name	58
Type	59
Flags	59
Protection	59
Modifiers	60
Category	60
Path	60
Duplicating Objects	61
Copies	61
Instanting Objects	62
Localizing Attributes	62
Unlocalizing Attributes	63
Memory Manager	63
Understanding Contexts	63
Contexts as Folders	64
Context and Object Path	65

Context and Object Visibility	65
Built-in Contexts	66
Default Context	66
Creating Contexts	66
Deleting Contexts	67
Moving Contexts	67
Copying Contexts	67
Renaming Contexts	67
Selecting Contexts	67
Instancing Contexts	67
Understanding Assets	68
Importing Geometries	69
Assigning Materials	69
Updating Geometry	70
Importing Images	71
Importing Texture Maps	71
Importing Scenes	72
Importing Projects	72
Using the Browser	73
Overview	73
Tree View	73
Item View	74
Per-Class Item Filtering	75
Icon View	75
Modifying Icon Size	75
Using the Search Engine	76
Using the Attribute Editor	76
Overview	77
Working with Attributes	77
Attribute Types	77
Numeric	78
Boolean	78

Filename	78
Color	79
Gradient	80
Insert Key	80
Set Key Color	80
Select Key	80
Delete Key	80
Start Range	81
End Range	81
Using the Color Picker	81
Overview	81
Color Gradients	82
Color Field	82
Color Slider	82
Reverting to Original Color	82
Color Numeric Mode	82
Using Color Swatches	82
Validating Color	82
Curve	83
Preset	83
Reference	83
Reference List	84
Adding Items	85
Selecting Items	85
Removing Items	85
Object List	86
Adding Items	86
Selecting Items	86
Deactivating Items	87
Removing Items	87
Reordering Items	88
Attribute Modifiers	88

Texture	89
Animation	89
Working with Instances	90
Localizing	91
Localize In	91
Unlocalizing	92
Working with Images	93
A Word on Evaluation	93
Active Evaluation Engine	93
Stopping an Evaluation	94
Using the Image View	94
Overview	94
The Image Area	95
2D Navigation Basics	95
Panning	95
Zooming	95
Fitting	96
3D Navigation Basics	96
Orbiting Camera	96
Moving Camera	96
Fitting	96
Assigning Material	96
Adding Scene Objects	96
The Image Toolbar	96
Selection	97
Progress Bar	97
Master Image	97
Limited Region	98
Rendering in Progress	98
Overlay	98
Color Correction	98

Alpha Checkerboard	98
Saving the Image	99
Sampling Quality	99
Image History	103
Layers and Channels Display	103
Working with Layers	104
Color Layer	104
File Layer	104
3D Layer	105
Setting the Viewpoint	105
Setting the Renderer	106
Managing Visibility	106
Lighting	107
Material Override	108
Shading Layer	108
Render Channels	108
Clipping	108
AOVs	109
Image Layer	109
Reference Layer	109
Blending Modes	110
Applying Filters	110
Pixel Filters	111
Kernel Filters	111
Using the Layer Editor	111
Overview	111
Creating Layer	112
Removing Layer	112
Reordering Layers	112
Setting Layer Opacity	113
Setting Layer Blending Mode	113

Working with Scene Objects	115
Geometries	115
Implicits	115
Box	115
Cylinder	116
Sphere	117
Particles	117
Point Array	118
Point Cloud	119
Density Mode	120
Geometry Mode	120
Random Distribution	122
Blue Noise Distribution	123
Point File	124
Point Volume	124
Fur	125
Generator	127
Interpolate	129
How it works?	130
Density	132
Frizziness	133
Knot Count	134
Clumping	135
File	136
Polymesh	136
Normals	137
Subdivision Surfaces	138
Raytrace Mode	139
Raytrace Tessellation Mode	140
Polyfile	141
Polygrid	142

Polysphere	143
Displacement Mapping	144
Displacement Tessellation Mode	145
Displacement	146
Value	146
Bound	147
Working with Shading Groups	147
Using the Material Linker	148
Overview	148
Edit Menu	148
Assigning Material	149
Assigning Clip Maps	149
Assigning Displacement	150
Shading Group Visibility	151
Localizing Shading Groups	152
Using the Material Editor	152
Overview	152
Navigation Basics	152
Panning	152
Zooming	153
Fitting	153
Selecting Nodes	153
Adding new Nodes	153
Connecting Nodes	154
Inserting Nodes	154
Disconnecting Nodes	155
Removing Nodes	156
Auto Layout	157
Combiners	158
Combining Scene Objects	159
Mode	159
Scatterers	159

Geometry Support	161
Geometry	161
Use Support Normals	162
Scatter Position	164
Scatter Rotation	166
Scatter Scale	168
Uniform Scale Variance	169
Lighting and Rendering	171
What's a Renderer?	171
Raytracer	171
Anti Aliasing Samples	173
Anti Aliasing Filtering	173
Filter	174
Filter Size	174
Anti Aliasing Oversampling	174
Subsample Quality	175
Shading Oversampling	175
Understanding Transparency	178
Alpha Threshold	179
Alpha Depth	180
Reflection Depth	182
Refraction Depth	183
Motion Blur	184
Motion Blur Settings	184
Motion Blur Sampling Mode	185
Shutter Curve	187
Lock Sampling Noise	189
Depth of Field	189
Tone Mapping	190
Previz Mode	192
Creating a Custom Previz Light Set	193

Disable Shading	194
Light Quality Depth Multiplier	194
Max Shadow Depth	197
Max Global Illumination Depth	198
Export AOVs	200
Lighting in Clarisse	200
Common Attributes	200
Light Linking	200
Lights and Transparency	201
Light Sampling	202
Directional Lights	203
Light Occlusion	203
Light Attenuation	207
Falloff	208
Attenuation	210
Area	212
Texturing	213
Distant	214
Texturing	215
Linear	217
Texturing	217
Point	218
Texturing	219
Spot	220
Texturing	221
Non Directional Lights	222
Ambient	222
Ambient Occlusion	224
Dome	227
Monte Carlo Global Illumination	227
Shading and Texturing	231
Standard Material	231

Luminosity	231
Specular Reflection	232
Backlighting	232
Bump Mapping	233
Normal Map	234
Reflection	235
Energy Mode	236
BRDF and Fresnel	238
Transparency and Refraction	239
Transparency Mode	241
Anisotropy	243
Subsurface Scattering (SSS)	245
Single and Diffuse Scattering	247
SSS Quality	250
SSS Light Oversampling	257
Texturing	261
Spatial Projections	261
Projection Planar	262
Projection Cylindrical	263
Projection Spherical	264
Projection Cubic	265
Camera	266
Parametric	266
UV	267
Image Maps	268
Map and Map File	268
Color Space and Pre Multiplication	268
Filtering Quality	269
Role in Sampling	271
Vertex Color Maps	274
Fur Support Color	275
Instance Color	276

Utility	284
Point Cloud	284
Reorder	285
Rendering Attributes	286
Enable Motion Blur	286
Unseen by Camera	287
Unseen by Rays	288
Cast Shadows	288
Receive Shadows	289
Is Emitter	290
Light Linking	290
Override Material	291
Working with Mattes	292
Rendering Image Sequences	293
Using the Render Manager	293
Disk Output Attributes	294
Render To Disk	294
Gamma Correction	294
Setting Frame Range	294
Filename	295
Format	295
Check Folder	295
Rendering	295
Working with Shading Layers	296
What's a Shading Layer?	296
How it works?	296
Creating a Shading Layer	297
Assigning Shading Layers	297
The Rule Concept	297
Item Name Syntax	299
Relative Path	299
Absolute Path	300

Specifying Shading Groups	300
Wildcards	300
Accessing to item's sub-elements	301
Evaluation Order	302
Shading Layer Hierarchy	302
Using the Shading Layer Editor	303
Overview	303
Adding New Rules	303
Inserting Rules	304
Deleting Rules	304
Shifting Rules Up	304
Shifting Rules Down	304
Clear All Rules	304
Replace Text	304
Tree View	305
Display Mode	305
Collapse All	305
Expand All	305
Parent	305
Rule View	305
Enabling/Disabling Rules	306
Editing Item Paths	306
Setting a Material	306
Toggling Shading Group Visibility	306
Using AOVs	306
General Workflow	306
AOV Store	308
Channels and Layers	308
Using the Channel Layer Editor	308
Overview	308
Adding New Layer	309
Removing a Layer	309

Clearing Unused Channels	309
Disabling AOVs	309
Extracting AOVs	310
Exporting AOVs to Disk	310
Animating in Clarisse	311
Working with Scene Items	311
Moving Items	311
Rotating Items	311
Scaling Items	312
Pivot Points	312
Parenting Items	313
Using the Hierarchy View	313
Using Constraints	313
Adding Constraints	313
Removing Constraints	313
Orient	314
Point	314
Scale	314
Target	315
Parent	316
MOT Player	316
ABC Player	316
Set Motion Key	316
Delete Motion Key	316
Using Deformers	316
Adding Deformers	317
Removing Deformers	317
Texture Displacement	317
Displacement Axis	318
X Axis	319
Y Axis	319

Z Axis	320
XYZ Axis	320
MDD Player	320
ABC Player	322
Using the Timeline	322
Overview	322
Playback Controls	323
Time Slider	323
Time Fields	323
Time Units	323
Using the Graph Editor	324
Overview	324
Attribute Tree	324
FCurve Editor	325
Navigation Basics	325
Panning	325
Zooming	325
Fitting	326
Selecting	326
Curves	326
Keys	326
Inserting Keys	326
Editing Keys	326
Removing	326
Keys	326
Curves	326
Key Editor	326
Using the 3D View	329
Overview	329
Navigation Basics	330
Orbiting	330

Moving	330
Fitting	330
Autofit	330
Look Through Item	330
Display Selection Mode	330
Rendering Mode	331
Display Quality	332
Sampling Quality	332
Scene Item Visibility	333
Hiding/Unhiding	333
Display Mode	333
Shading Layer	334
Using Tools	337
Select	337
Translate Item	337
Rotate Item	338
Scale Item	339
Pick Fit	340
Clone Stamp 3D	341
Picker	343
Tools and Transparency	343
Using the Resource View	345
Overview	345
Memory Statistics	346
Empty Resources	346
File Resources	347
Clearing Resources	348
Resource List	349
Using the Path Manager	351

Overview	351
Path List	351
Changing Selected Path	352
Checking Paths	352
Search and Replace	353
Object Attribute List	354
Working with Variables	357
Basic Usage	357
Variable Naming Limitation	357
Variable Types	357
System	358
Built-in	358
Using Time Variables	358
Padding	359
Custom	359
Variables and Evaluation	359
Using the Variable Editor	359
Overview	360
Loading	360
Saving	360
Adding	360
Setting	361
Deleting	361
Clearing	361
Understanding Clarisse Projects	363
Serialization Basics	363
Entries	363
Values	363
Groups	363
Comments	364

Directives	364
Path Resolution	364
Preprocessor	365
#include	365
Usage	365
Command	366
Context Sensitivity	366
Execution Order	366
Argument Types	366
import_abc	367
Usage	367
set_value	367
Usage	368
Copying and Pasting	368
Using CNode/CRender	371
Differences between CNode and CRender	371
Exporting a Render Archive	371
Setup on a render farm	372
Installation Folder	372
Configuration File	372
CNode Usage	373
CRender Usage	373
Arguments	373
Optional Arguments	373
Info	373
Reference	374
Interactive*	375
Script*	375
Search Path	376
Print Expanded*	376
Frame	376

Threads	377
Configuration File	377
Module Path	377
License Server	377
Stats	377
Verbose	378
Override Arguments	378
Image	378
Output	379
Format	379
Start Frame	380
End Frame	380
Frame Step	380
Gamma Correction	380
FPS	380
Motion Blur Sample	381
Motion Blur Direction	381
Motion Blur Length	381
Texture Cache	381
Examples	381
Basic	381
Specifying an Image	382
Rendering Multiple Images	382
Using the Interactive Mode	383
The Prompt	383
Built-in Commands	383
cd	383
ls	384
Useful Python Commands	385
Exiting	385
Stopping a Running Script	386

Tutorials	387
Clarisse Basics	387
User interface Basics	387
Combiner Basics	393
Using The Browser	393
Using The Attribute Editor	393
Group Basics	395
Using the Browser (Quickest Way)	395
Using the Attribute Editor	395
Drag and Drop into a group attribute	397
Scatterer Basics Part 1	400
Creating a Point Volume	401
Creating a Scatterer	401
Scatterer Basics Part 2	402
Displacements Basics	407
Setting Displacements	407
Setting Bounds	409
Setting the Texture	410
Material Assignment Basics	413
Drag and Dropping in the Image View	414
Using the Material Linker	415
Material Overriding	416
Light Linking Basics	417
3D Layer	418
Scene Object	419
Lights	420
Layering Basics	421
Setting Background Visibility	422
Setting Foreground Visibility	423
Cutting out Foreground Alpha	423
Global Illumination and Ambient Occlusion Basics	425

HDRI Lighting Basics	429
Sampling Quality Basics	434
Camera Mapping Basics	439
UDIM Texture Basics	444
Clarisse 101	445
Discovering the User Interface	445
Editing your first object	446
The Layer Editor	447
Importing geometry	448
Assigning materials	451
Setting an environment	453
Organizing items	454
The Material Editor	456
Working on materials	458
Building our ground	459
The ground material	465
Setting the layout	466
Splitting into Layers	467
Rendering our work	469
Setup Image Output	469
Render Images	470
Clarisse 103: Advanced Shadows and Reflections	470
Creating group	470
Setting your groups	472
Clarisse 104: Layers and Mattes	474
Create your first layer	474
Shadows Layer	477
Setup Mattes	478
Clarisse 105: Using Scatterers	479
What is a scatterer?	480
Scatterer in action	480

Creating a Point Cloud	480
Referencing the Point Cloud	481
How to create a forest	482
The Island	490
Creating a plant library	491
Preparing plants for scattering	492
Using contexts to re-use your work	495
Shading the plants	499
Creating the first scatterer	506
Preparing your scatterer for texturing	512
Texturing your scatterer	517
Varying your shading using the instance texture	521
Texturing the ground	523
Creating the grass clumps	527
Scattering the grass	530
Adding props and palmtrees	534
More plants and trees	536
Lighting and compositing	538
Technical Specifications	541
Minimum System Requirements	541
General Features	541
File Formats	542
3D File Formats	542
2D File Formats	542
User Interface	542
Editors and Viewers	542
Geometry and Scene Object	543
Animation	543
Compositing	543
Image	543

Blendings	544
Filters	544
Rendering	544
Lighting	545
Shading and Texturing	545
Licensing	545
Third Party Licenses	547
FLTK	547
OpenImageIO	547
Alembic	548
Boost C++ Libraries	549
MurmurHash3	550
GLEW	551
Appendix A	555
Disclaimer	555
End User License Agreement	555
Index	563

Legals

Clarisse™ User Guide. Copyright © 2013 Isotropix SAS. All Rights Reserved. Use of this User Started Guide and the Clarisse software is subject to an End User License Agreement (the "EULA"), the terms of which are incorporated herein by reference. This User Guide and the Clarisse software may be used or copied only in accordance with the terms of the EULA. This User Guide, the Clarisse software and all intellectual property rights relating thereto are and shall remain the sole property of Isotropix SAS. ("Isotropix") and/or Isotropix's licensors. The EULA can be read in this User Guide, Appendix A.

Isotropix assumes no responsibility or liability for any errors or inaccuracies that may appear in this User Guide and this User Guide is subject to change without notice. The content of this User Guide is furnished for informational use only.

Except as permitted by the EULA, no part of this User Guide may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, recording or otherwise, without the prior written permission of Isotropix. To the extent that the EULA authorizes the making of copies of this User Guide, such copies shall be reproduced with all copyright, trademark and other proprietary rights notices included herein. The EULA expressly prohibits any action that could adversely affect the property rights of Isotropix and/or Isotropix licensors, including, but not limited to, the removal of the following (or any other copyright, trademark or other proprietary rights notice included herein):

Clarisse™ software © 2013 Isotropix SAS. All Rights Reserved.
Clarisse™ is a trademark of Isotropix SAS.

In addition to those names set forth on this page, the names of other actual companies and products mentioned in this User Guide (including, but not limited to, those set forth below) may be the trademarks or service marks, or registered trademarks or service marks, of their respective owners in the United States and/or other countries. No association with any company or product is intended or inferred by the mention of its name in this User Guide.

Linux ® is a registered trademark of Linus Torvalds.

Windows ® is the registered trademark of Microsoft Corporation.

Mac, Mac OS, Snow Leopard are trademarks of Apple, Inc., registered in the U.S. and other countries.

Isotropix

Cap Omega

Rond Point Benjamin Franklin

CS 39521

34960 Montpellier Cedex 2

FRANCE

Preface

About this User Guide

Clarisse is a new breed of high-end 2D/3D animation software which is the result of years of teamwork between R&D engineers and high-end CG artists. It has been designed to simplify the daily work of professional CG artists, allowing them to work directly on final images while alleviating the complexity of 3D creation. Clarisse is literally the fusion of a compositing software, a 3D rendering engine and finally, an animation software. Its workflow has been designed from scratch to be “image-centric” so that artists can work constantly while visualizing their final image with full effects on.

Note on this User Guide

Clarisse is a professional application and this user guide is intended for professionals. We intentionally try to keep this guide as simple as possible but we assume the reader already a solid background in computer graphics.

For a more in depth technical documentation please use the HTML based one provided by Clarisse. You can browse the HTML based documentation by hitting **F1** function key while running Clarisse or directly from Clarisse's menu bar **Help > Clarisse Help...**

Looking for Help?

If you are having any issues concerning the installation or this guide, please contact our Support Department (support@isotropix.com) by email.

What's new?

New in Clarisse 1.6

Please note the following list is not exhaustive. For the extensive feature list please read the release notes.

Personal Learning Edition (PLE) Mode

You can now choose to start it in PLE by typing `clarisse -ple`. Note: Clarisse can be started in PLE if not valid license is found.

New Node Locked Licensing

Clarisse now provides node locked licensing scheme. Node locked licenses are bound to the machine and don't need llise to run.

New CRender

CRender is a new application designed only to render images. Unlike CNode, it doesn't provide any scripting ability, has no interactive shell mode and can't read directly Clarisse project files. Instead, it reads as input a special render archive format that needs to be exported from Clarisse.

New in Clarisse 1.5

New Shelf Widget

You can now arrange your scripts in shelves.

New Shading Layer

You can now use a new tool called the Shading Layer. It allows you to assign material without explicitly assigning them to geometries. Shading Layers can be assigned in 3D Layers and in 3D View.

New AOV Support

You can now declare arbitrary output variables and export custom buffers.

Combiner Group List Mode

You can now reference groups in a combiner using Group List Mode.

Basic Maths Operation Support

Numeric fields support for basic math operations following this syntax. += , -=, *=, /=

New Application Variables

Custom variables are now creatable using the new Variable Editor.

Custom Previz light set

You can create and set your custom Previz light set.

New Depth of Field

You can now activate control secondary ray sampling using *Light Quality Depth Multiplier*, *Max Shadow Depth* and *Max Global Illumination Depth*.

New Tone Mapping

User driven curve to remap sample color prior to pixel integration.

New Texture Filtering

New *Sub sample Quality* attribute in raytrace to control the quality of texture filtering during anti-aliasing.

New Energy Mode

You can now control how reflection and diffuse blend in the standard material.

New Microfacet reflectance Model

You can now control and texture the anisotropy direction of standard material. Quality of glossy reflections improved greatly.

New Transparency Mode

You can now control the transparency mode of materials when shadows are evaluated.

New Scattering Controls

You can now texture most of Scatterer attributes. Scatterers can now align instances to underlying geometry normals.

New Light Attenuation and Falloff Controls

You can now control lights falloff and attenuation using a custom curve.

New Light Occlusion in Global Illumination

Now area lights are properly occluded during global illumination gathering.

New Path Manager

You can now manage external file dependencies using the new Path Manager.

New Tools Opacity Threshold

Selection tools have now an Opacity Threshold helping the selection of transparent items.

New Browser

You can now sort items by columns and display context contents as icons.

New Search Engine

The search engine in Clarisse has been improved greatly.

New Resource View

You can now monitor and fine tune project memory usage using the Resource Manager.

New Sampling Controls

You can now control secondary ray sampling using *Light Quality Depth Multiplier*, *Max Shadow Depth* and *Max Global Illumination Depth*.

Improved Web Documentation Search Engine

The search engine now covers both the user manual and the module class reference section.

Group in Groups

You can now reference groups in a group.

Motion blur Improvements

Motion Blur Length and *Motion Blur Direction* are now saved in the current project file. Support for deformation, combiners and scatterers. New sampling modes are available. Per renderer shutter curve control.

Installation

Clarisse is a cross platform application and this chapter reviews all the different installation procedures related to each platform. Feel free to skip to your favored platform but make sure your system meets Clarisse system requirements.

Minimum System Requirements

Clarisse has a modern multithreaded architecture and uses all the available system cores. The more cores your CPU has the faster the evaluation gets. At the time of this writing, Clarisse doesn't perform any evaluation on GPUs, so it doesn't require a high-end graphics card.

Windows and Linux

- Intel or AMD based x86-64 CPU supporting SSE2
- 2 GB RAM
- 500 MB disk space available
- 64-bit Operating System
- Windows Vista Business SP2 or Linux Red Hat/Centos 6
- OpenGL 2.0 compliant graphics card with 32 MB of video memory
- Network card
- Display supporting 1280 x 1024 pixel resolution in 24 bit color
- Three-button mouse

To avoid visual glitches and stability issues, please keep your display drivers up-to-date. You can find the latest drivers from the websites of the graphics card manufacturers. For example: www.nvidia.com or www.amd.com

Note for Linux users

We certified Clarisse on RHL/Centos 6.x x64 distributions. However, it has been reported Clarisse runs perfectly fine on other Linux distributions.

Mac OS X

- Intel Core 2 Duo Processor
- 2 GB RAM
- 500 MB disk space available
- Mac OS X 10.6 (Snow Leopard)
- OpenGL 2.0 compliant graphics card with 32 MB of video memory
- Network card
- Display supporting 1280 x 1024 pixel resolution in 24 bit color
- Three-button mouse

Note

Clarisse has been successfully tested on Mac OS X 10.6 (Snow Leopard), Mac OS X 10.7 (Lion) and Mac OS X 10.8 (Mountain Lion). Clarisse isn't a Retina application. Rendering performance degradation may occur depending on the screen resolution.

Installing Clarisse

We don't provide an installer for Clarisse on all platforms. It comes either as an installer on Windows, a DMG iso on Mac OS X and as an archive file on Linux.

Installation on Windows

Launch the installer and follow the instructions.

Requirements

- Microsoft Visual C++ 2010 Redistributable Package x64
- Python 2.7 X86-64 (optional to enable python scripting)

Both third parties are included in the installer.

Installing Microsoft Visual C++ 2010 x64 Redistributable

If you're unsure which package is installed or if it's missing, you can freely download it from here :

<http://www.microsoft.com/en-us/download/details.aspx?id=13523>

To install this package, please follow the installation instructions provided with the package.

Installing Python

Python can be freely downloaded from here :

<http://www.python.org/ftp/python/2.7.3/python-2.7.3.amd64.msi>

To install this package, please follow the installation instructions provided with the package.

Important

Clarisse is a 64-bit application and requires a 64-bit version of Python. Make sure you download the 64-bit installer (called amd64 or X86-64)

Installation on Linux

Extract directly the archive content to any destination folder.

Installation on Mac OS X

To install Clarisse, open the disk image and drag the Isotropix icon into your Applications folder.

Python in Clarisse

Unlike many other applications, Clarisse doesn't provide its own version of Python. Instead, Clarisse uses the one installed in the system. This way, there are no limitation and you can use any available Python extensions.

Python is installed by default on Linux and Mac OS X but not on Windows. To install Python on Windows please refer to Installing Python section.

Important

Clarisse only supports Python 2.X branch and not Python 3.X.

Specifying Python Location

By default, Clarisse should find its way to where Python is installed. However, if the installation path has been customized or if Clarisse is running on a not officially supported distribution, you may have to manually specify Python Location.

You can either set Python environment variable PYTHONHOME to where Python is installed or modify the same variable in `clarisse.env` file. `clarisse.env` file can be found in the directory where Clarisse configuration is located. For more information on Clarisse configuration file please refer to Configuration and Environment section.

Follow the exact same procedure if you wish to specify a custom Python build.

Licensing

Clarisse Licensing system can either work with a node locked license or a with a more flexible floating license system based on a client/server model. Our licensing policy lets users run an infinite number of Clarisse application instances on their computer once it had been granted with one valid license. Licenses are valid for all platforms and can be either temporary in duration or permanent.

Obtaining a License

You can request licenses by contacting Isotropix Sales Department (sales@isotropix.com). To generate you a License Key we need to receive a valid Request Key from you. **A Request Key is only valid for a unique computer in the same way a License Key generated from a Request Key is locked to that same computer.**

Request keys have a short life-time. Make sure to generate a fresh request key before sending it to your local reseller or our Sales Department.

Important

Make absolutely sure the Request Key has been generated on the computer you wish to use as a License Server.

Please note ILISE license server can run on the same computer that will be used to run Clarisse.

In the installation folder, inside the licensing sub-folder, you can find an executable named ireqkey. Open a terminal/command prompt and launch ireqkey or run ireqkey-gui. Copy and paste and email the generated Request Key.

A typical Request Key will look like this dummy request code:

```
GHEF:DELU:2AMP:UOI6:KLSD:WOP5:TOER:MNPY:AOCO
```

Please Copy/paste and email the request key to sales@isotropix.com. Once we receive your Request Key, you will receive a License Key.

A typical floating License Key should look like this dummy license key:

```
5A05F426-5230E12A-F3302CF3-EDE8EDB4  
1F80D0D0-64A2848E-439E61F5-F33EE5BA  
1C771BC0-C04FAA9B-101152E0-2C00D9EF  
4C41A7E8-C5603EEA-06BEE482-21C773DC  
965EAA58-F4694CF7-E47E5019-C01AAB70  
CB57A069-C7AC0828-C7F62453-3B446D93  
0D87FA89-51FAF1AA-10F1C404-3F7DDB85  
4177C153-D5688714-CC761BE0-CE667D2C  
515B911C-26E1C18F-B963BFEE-2D23EE7C  
AE848C9D-59588208-744C67D7-83FE02D4  
3B01B092-EED5A084-4EB1D963-2CB4AA55  
B57938AA-ED14B7B6-12F6EBA1-8C6ED9EE
```

Finally, copy the license file, in `~/.isotropix/Ilise` on Linux, `%APPDATA%\Isotropix\Ilise` on Windows and `/Users/CURRENT_USER/Library/Preferences/Isotropix/Ilise` on Mac.

Important

Package license files must be saved separately. There must be a single license file per license per package. For example, one file for Clarisse iFX and another for CNode. Moreover, each time you receive a new license, the license must be saved separately in a new file. For example if you got 1 Clarisse iFX license, and then later on ordered additional licenses, you would then have 2 license files in total.

A typical node locked License Key should look like this dummy license key:

```
4IGFAUR503LTNB4Z4Q4IVI2HPTLWR75T
SADNVGBVGB4YF5MYI6J47KLKK5CNRDE7
HMKWYE3G2QJATLRXJ2UE2LNSZMZJMBUI
E7OQFTUBBCUSGATW6ZQEXADFNIIMS4R6
Z3LVTN36X3V74PGLHZSV3KDI FVYA3PBX
CJFTD3KKL FK2GHS GD3HAZBQXI7VZDZTW
QWLZEGQ67DYKUB3XZQ4YBPJF6VWE06OF
CENRI4XLJLRKNHOMEORZS723ACWJJGQJ
UBHCEUE6RI5QINHKHSUIU2GCYRDHYYTL
LYP5L4LA6MGBXMYTTT5Q0IKPEGXJMWWF
ZMNUMIZ4JBODRZ6NRGXIPML3LRWKY3FO
83UYNOWSXYK22XQLIZNHUPD03E3VATM4
S35EZBCYIBSMMDW54RGNDTRLUJ6LWwBU
YBFF4IRP4SNPUPY4NZJAJIB252BNEKBO
6PPZKMBXBDKJLH56JZ4VVN2VQAKO4HH
M7RGV4GRKNFQI70G644PKPDREC2HZLYO
V3GVSUE52KCT5FZXW2G4DWLDRL25WIS5
R5MUFKBCSCICOQCZH5V06WSU2Y6YXQ6X
ORMRKH2VIAW6HOXPRI2HXNYZ7BCQADMR
XQDID043DRFN4E4NNTUSK3XPSHAX6YCM
```

Finally, use Clarisse License Manager (LICMAN) to install your license.

ILISE License Server

If you are using a node locked license, you can skip this section.

ILISE is our dedicated License Server and provides licenses for all of our products. ILISE can either be installed on a dedicated license server or installed on a computer used to run Clarisse. ILISE uses the primary network interface card mac address as a unique server identifier. If the network interface card changes (switching off WIFI, installing or removing a network interface card) the license can become invalid. Licenses are then bound to the hardware and not to the operating system. The same license should then work on ILISE whether the computer is running it on Windows, Mac OS X or Linux.

ILISE is currently a standalone application which doesn't run as a service or as a daemon. ILISE is located in the installation folder, inside licensing sub-folder. To start ILISE you'll need to open a terminal/command-prompt and launch the `ilise` executable.

Note for Linux users

ILISE tries to create a system lock preventing multiple ILISE instances running at the same time. Make sure you have administrative privileges before running ILISE or you'll get an error message:

```
error: Unable to open file '/var/run/isotropix_license_server.pid':
Permission denied
```

Installing Licenses

By default, ILISE will look for licenses in %APPDATA%/Isotropix/Ilise on Windows, /Users/CURRENT_USER/Library/Preferences/Isotropix/Ilise on Mac OS X and ~/.isotropix/ilise on Linux. If you wish to specify a different folder, you can use the command line option `-license_folder`

For example:

```
ilise -license_folder /apps/isotropix/clarisse/licenses
```

For more information such as how to set port the number, type `ilise help` in a command line.

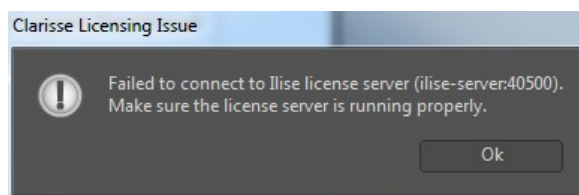
Using LICINFO

If you wish to query license information from any clients, we provide LICINFO a dedicated command line tool. LICINFO allows you to query ILISE to list license information and active connected clients, and finally to restart ILISE. Here are some useful commands. To display all available commands please type `licinfo help` in a command line.

Command	Description	Example
list	Display the list of registered licenses	ilise ILISE-SERVER list
list_clients	Display the list of active clients using licenses	ilise ILISE-SERVER list_clients
restart_server	Restart the license server	ilise ILISE-SERVER restart_server
shutdown_server	Stop the license server. All active connections will be closed	ilise ILISE-SERVER shutdown_server

Setting ILISE location to Clarisse

Each time you launch Clarisse, it tries to connect to ILISE to get a valid license. By default, Clarisse will look for ILISE on the localhost. If Clarisse is unable to connect to ILISE, the following error message box should appear:



You can set the location and the port of ILISE server by launching Clarisse from the command-line or alternatively, using our dedicated tool LICMAN (License Manager).

Command Line Option

To specify the network machine name or the host address hosting ILISE, you can use the command-line argument `-license_server`

For example:

```
clarisse -license_server ILISE-SERVER  
clarisse -license_server 192.168.2.24
```

You can also specify ILISE port in command line:

```
clarisse -license_server ILISE-SERVER:40500  
clarisse -license_server 127.0.0.1:40000
```

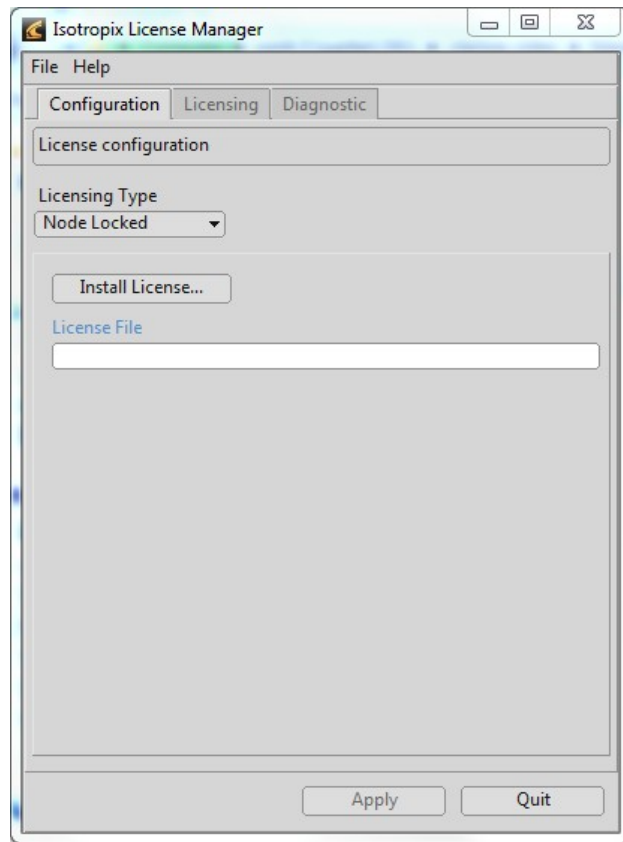
Using LICMAN

LICMAN is our dedicated tool to deal with licensing it has several tabs, each one, dedicated to a specific task.

- *Configuration* allows you to set your licensing configuration to properly license Clarisse, CNode and CRender.
- *Licensing* helps you request new licenses and install floating license files.
- *Diagnostic* is here in case there is an issue with license files.

Configuration

The first tab, selected by default, is called Configuration. This tab allows you to setup the Licensing Type.

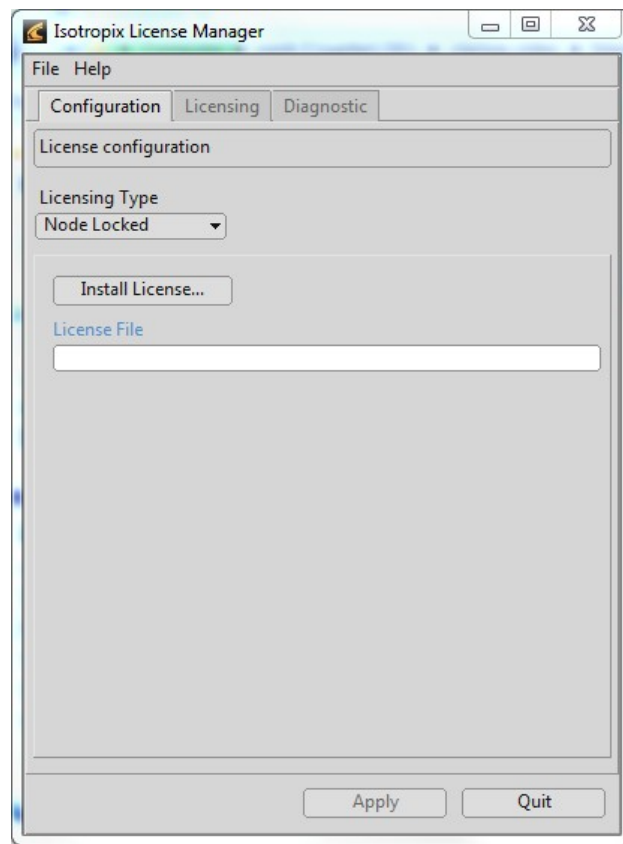


Licensing Type

Using the pull down **Licensing Type**, you can set whether you want to use *Node Locked* or a *Floating* licensing.

Node Locked Licensing

By default, *Licensing Type* is set to *Node Locked*. This allows you to install a node locked license key from a file.



Floating licenses can't be installed as node locked licenses. If you are unsure of the license type, open the license file in a text editor. Here is how a node locked license looks like if opened in a text editor:

```
4IGFAUR503LTNB4Z4Q4IVI2HPTLWR75T
SADNVGBVGB4YF5MYI6J47KLKK5CNRDE7
HMKWYE3G2QJATLRXJ2UE2LNSZMZJMBUI
E7OQFTUBBCUSGATW6ZQEXADFNIIMS4R6
Z3LVTN36X3V74PGLHZSV3KDI FVYA3PBX
CJFTD3KKLFK2GHS GD3HAZBQXI7VZDZTW
QWLZEGQ67DYKUB3XZQ4YBPJF6VWE06OF
CENRI4XLJLRKNHOM EORZS723ACWJJGQJ
UBHCEUE6RI5QINHKHSUIU2GCYRDHYYTL
LYP5L4LA6MGBXMYTTT5Q0IKPEGXJMMWF
ZMNUMIZ4JBODRZ6NRGXIPML3LRWKY3FO
83UYNOWSXYK22XQLIZNHUPD03E3VATM4
S35EZBCYIBSMMDW54RGNDTRLUJ6LWwBU
YBFF4IRP4SNPUPY4NZJAJIB252BNEKBO
6PPZKMBXBDKJKLH56JZ4VVN2VQAKO4HH
M7RGV4GRKNFQI70G644PKPDREC2HZLYO
V3GVSUE52KCT5FZXW2G4DWLDRL25WIS5
R5MUFKBCSCICQ CZH5V06WSU2Y6YXQ6X
ORMRKH2VIAW6HOXPRI2HXNYZ7BCQADMR
XQDIDO43DRFN4E4NNTUSK3XPSHAX6YCM
```

Note

Please note a Node Locked license is bound to a specific machine. A same license can't be used on another machine.

Installing a Node Locked License

To install a node locked license, press **Install License...** and browse for the file license.key you should have received by email. This will copy the license file to the correct default location. If successful, the location of the license file is displayed in *License File* text field.

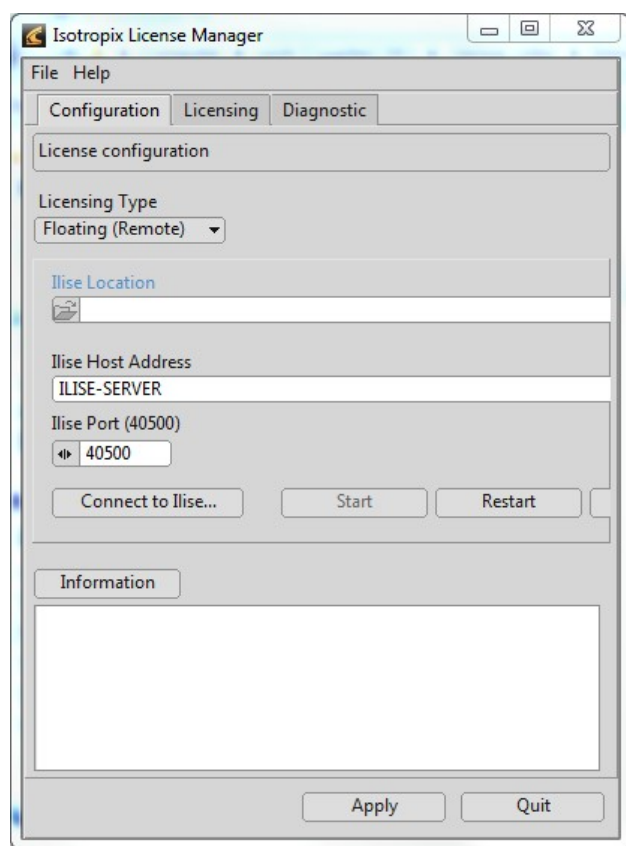
Once the license is installed, press **Apply**. Quit LICMAN, and run Clarisse.

Note

Installing a node locked license requires administrative privileges.

Floating Licensing

Using the pull down **Licensing Type** you can set whether ILISE server will run in *Floating (Local)* or in *Floating (Remote)*.



When set to *Remote*, you need to specify the hostname or the IP address of the machine running ILISE Server in the *Ilise Host Address* text field.

When set to *Local*, ILISE Server is running on the same machine that is running LICMAN. This setting is basically equivalent to set the connection type to *Remote* and set *Ilise Host Address* to LOCALHOST or 127.0.0.1. However, when set to *Local* there are some more available options allowing the control of ILISE.

Ilise Port

By default, ILISE runs on port 40500. However ILISE can be explicitly set to run on a different port. If the port number on the ILISE side is different to the default one (40500), please set the correct port number in the field *Ilise Port*.

Ilise Location (Local)

You shouldn't almost never have to change the location of ILISE binary as it should point to the correct location. However, if you have customized your installation or moved ILISE binary to a custom location, you can set here the path where LICMAN should look for ILISE binary.

Starting Ilise (Local)

You can start ILISE Server by pressing the Start button. This button is only available when Connection Type is set to *Local*. If you get an error message when starting ILISE, please note there are only three reasons why the Start can fail.

- ILISE Server is already running. You can't run multiple instances of ILISE on the same machine.
- ILISE binary isn't found. To solve this issue, please set the correct ILISE location to *Ilise Location* text field.
- ILISE can't run as it requires Administration Privilege. Please Quit LICMAN and run it with Administration Privilege enabled.

Restarting Ilise

You can restart ILISE Server (and not the machine) by pressing the Restart button. This is particularly useful if you have installed new licenses and you want ILISE to load them up. Please be aware all connected clients get disconnected after pressing Restart button. However, as soon as the ILISE Server is back online, clients reconnect automatically. You don't need to be particularly careful when restarting ILISE as, most of the time, it is so quick

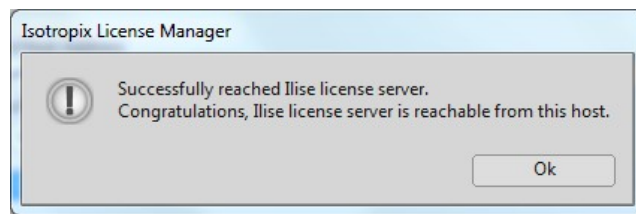
clients don't even notice they've been disconnected.

Shutting Down Ilise (Local)

Shutdown ILISE Server (and not the machine) by pressing the Shutdown button. Use this if you want to shutdown ILISE. Please be aware all connected clients will get disconnected and won't be able to reconnect until ILISE is restarted. However, as soon as the ILISE Server is back online, clients reconnect automatically. You don't need to be particularly careful when restarting ILISE as, most of the time, it is so quick clients don't even notice they've been disconnected.

Testing the Connection

You can test your connection settings by clicking *Connect to Ilise...*



If you get the previous message then congratulations, you're all setup! Otherwise please follow the instructions in the error message dialog.

Note

A successful connection test doesn't mean a valid license is available. If no valid license is found then Clarisse or CNODE pops up an error message.

Getting License Information

If you successfully setup and connected to ILISE, you can display useful license related information. By clicking on the Information button, you can see how many licenses ILISE can serve in total and what are the host currently connected to the server. This is useful in case a user left an unused Clarisse opened and no more license is available.

For example:

```
CNODE RENDERNODE 1.0 MAINTENANCE EDITION PERMANENT LICENSE FEATURE
1500 SEAT(S) LICENSED TO JOHN SMITH CO. 0/1500 OF AVAILABLE LICENSES
AND 0 RUNNING CLIENTS
CLARISSE IFX 1.0 PERMANENT LICENSE FEATURE 120 SEAT(S) LICENSED TO
JOHN SMITH CO. 1/120 OF AVAILABLE LICENSES AND 1 RUNNING CLIENTS

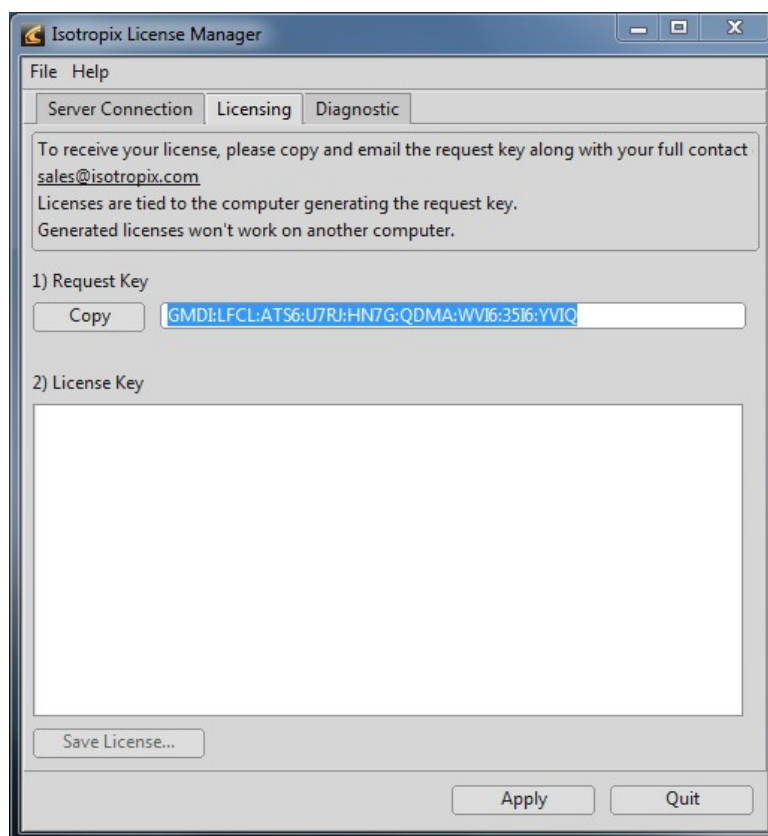
License #1 Isotropix Clarisse iFX 1.0 117/120 REMAINING LICENSE(S)
3 running client(s):
cgstation-152@192.168.1.156
cgstation-028@192.168.3.5
cgstation-002@192.168.2.121
```

Saving Connection Settings

To save the configuration, simply click on the **Apply** button. If you **Quit** without applying your modifications, your current configuration is lost.

Licensing

The Licensing tab allows you to request new licenses using a request key or to installed freshly received floating licenses. This tab is only available if you set **Licensing Type** to *Floating*

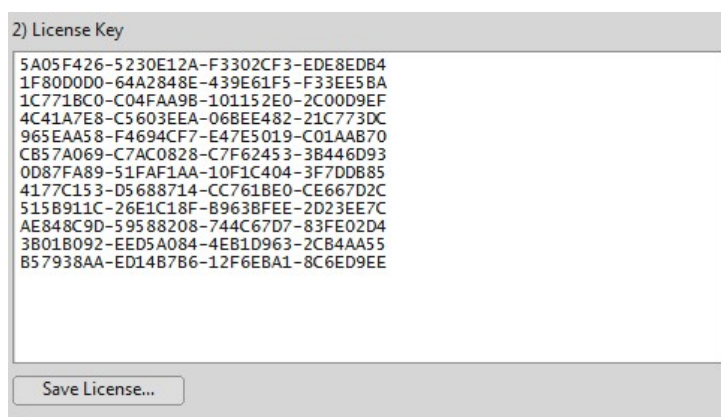


Requesting Licenses

If you wish to purchase new licenses, press the Copy button and paste the content of the clipboard in an email addressed to sales@isotropix.com. Please remember to put your full details and the number and type of license you require. Our Sales team will reply to you as quickly as possible. Alternatively you can use our online store from our website: www.isotropix.com

Installing Floating Licenses

Using LICMAN you can install licenses on the machine running ILISE. Open your license file using a text editor, copy its content into the clipboard and paste it in the designed area:



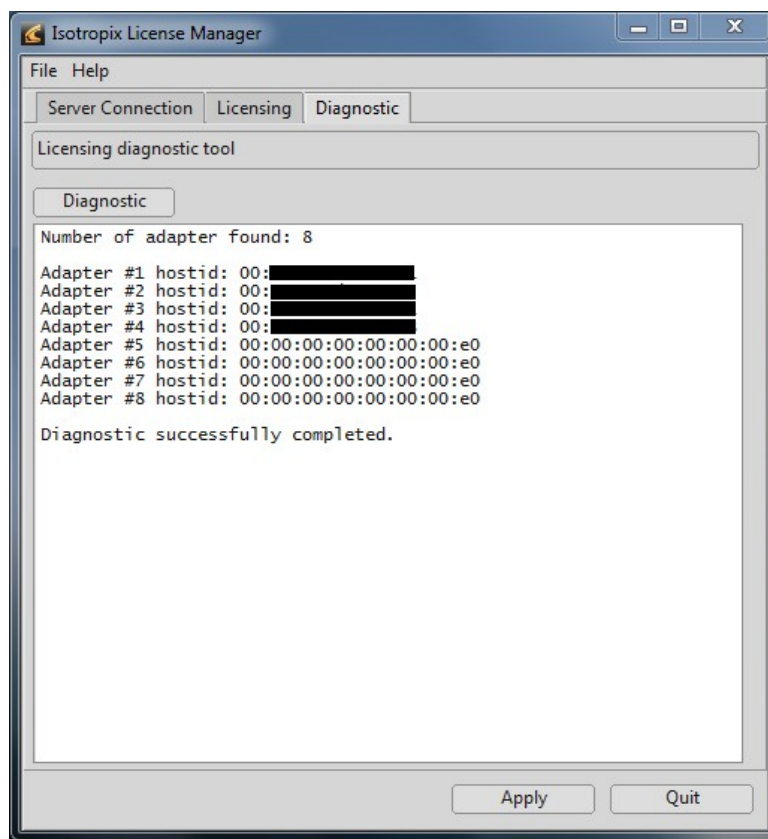
Press **Save License...** and name your license as you like. LICMAN redirects you to the directory where ILISE looks for its license files by default. By default, ILISE will look for licenses in %APPDATA%/Isotropix/Ilise on Windows, /Users/CURRENT_USER/Library/Preferences/Isotropix/Ilise on Mac OS X and ~/.isotropix/Ilise on Linux.

Note

When you install a new set of licenses, please remember to restart ILISE server so it can serve the newly installed licenses. To restart ILISE please refer [here](#)

Diagnostic

The Diagnostic tab is useful in case you have licensing issues and you're in contact with our Support team. By clicking to Diagnostic, LICMAN will output useful information that will help our Support team sorting your issue out. This tab is only available if you set **Licensing Type** to *Floating*



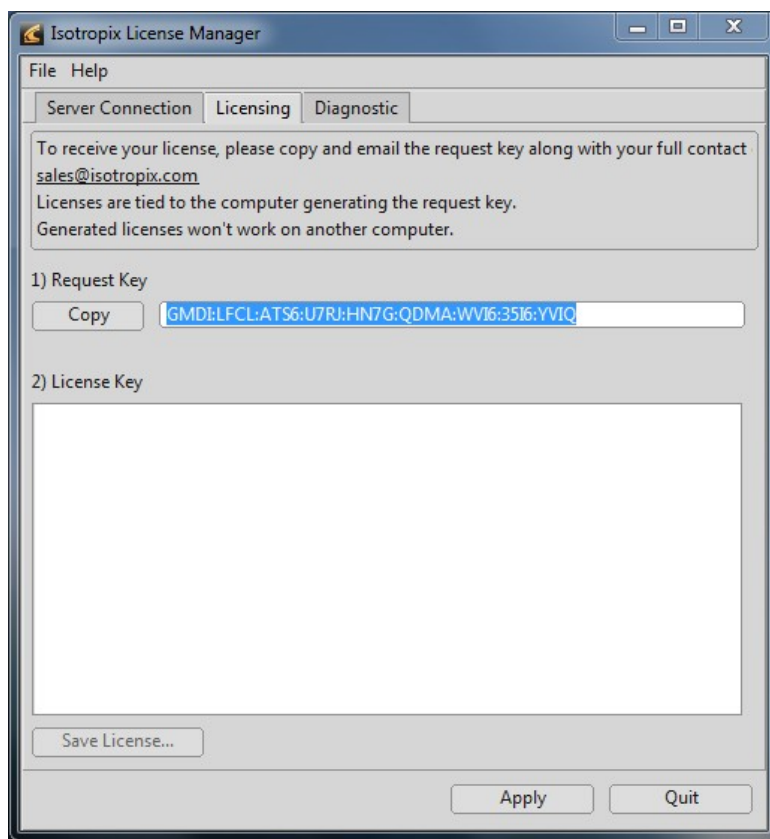
Tutorial: Installing a floating license locally

In this tutorial, we will see the different steps to install a license on a local machine that will act as ILISE server. If you already have your license file please skip the Getting a license file section.

Getting a license file

In order to get your license file, you will need a request key. If you wish to use ILISE and Clarisse on a same machine like you had a node locked license, just launch LICMAN on your machine. Alternatively, you can use another machine to act as ILISE license server. **Please remember licenses are bound to the machine that generates the request key. In other words, a license won't work on another machine.**

Launch LICMAN and select the *Licensing* tab. You should see:



In this example the request key is:

```
GMDI:LFCL:ATS6:U7RJ:HN7G:QDMA:WVI6:35I6:YVIQ
```

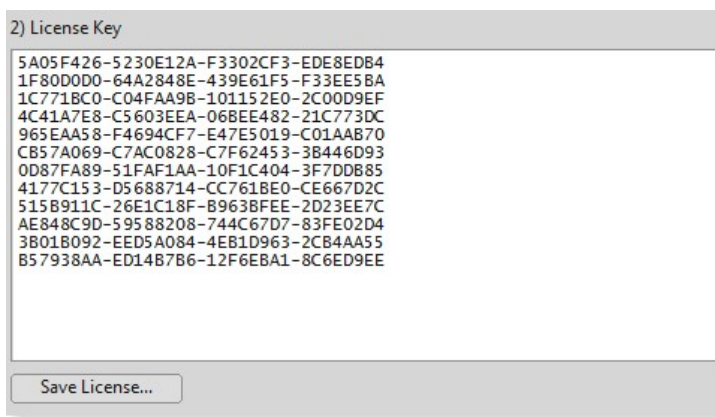
Press the Copy button to copy the request key in your clipboard and paste it in an email to sales@isotropix.com. Remember to put your full contact details so you can interact properly with our Sales team. Once the Sales team agreed to send you a license you will receive your license file in an email.

Installing the license

You should have your license file by now. A license file looks like this dummy one:

```
5A05F426-5230E12A-F3302CF3-EDE8EDB4
1F80D0D0-64A2848E-439E61F5-F33EE5BA
1C771BC0-C04FAA9B-101152E0-2C00D9EF
4C41A7E8-C5603EEA-06BEE482-21C773DC
965EAA58-F4694CF7-E47E5019-C01AAB70
CB57A069-C7AC0828-C7F62453-3B446D93
0D87FA89-51FAF1AA-10F1C404-3F7DDB85
4177C153-D5688714-CC761BE0-CE667D2C
515B911C-26E1C18F-B963BFEE-2D23EE7C
AE848C9D-59588208-744C67D7-83FE02D4
3B01B092-EED5A084-4EB1D963-2CB4AA55
B57938AA-ED14B7B6-12F6EBA1-8C6ED9EE
```

Open the license file using any text editor and Copy its content in the clipboard. Then paste it in LICMAN such as this:



Note for Mac OS X Users

To paste the content of the clipboard you need to press CTRL + V as CMD + V doesn't work currently.

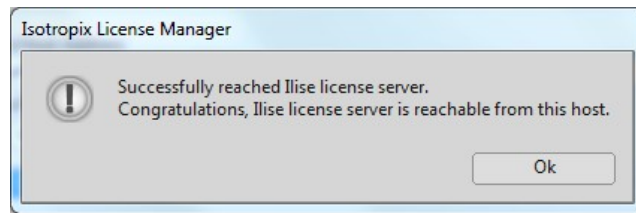
Once the content is pasted. Press **Save License...** A file browser pops up and name your license file as you like. LICMAN redirects you to the directory where ILISE looks for its license files by default. By default, ILISE will look for licenses in %APPDATA%/Isotropix/Ilise on Windows, /Users/CURRENT_USER/Library/Preferences/Isotropix/Ilise on Mac OS X and ~/.isotropix/Ilise on Linux.

Note

Each time you install a new license make sure not to overwrite existing ones: each file describes a set of licenses.

Starting ILISE locally

Once the license has been properly saved, you will need to launch ILISE. Set **Connection Type** to *Local*. Just ignore *Ilise Location* and press **Start** button. That's it! Press **Connect to Ilise...** to test the connection. If you get this message box then ILISE is up and running.



Hit then **Apply** to save the configuration. You can check information on licenses served by ILISE by clicking on **Information**

```
CNODE RENDERNODE 1.0 MAINTENANCE EDITION PERMANENT LICENSE FEATURE
1500 SEAT(S) LICENSED TO JOHN SMITH CO. 0/1500 OF AVAILABLE LICENSES
AND 0 RUNNING CLIENTS
CLARISSE IFX 1.0 PERMANENT LICENSE FEATURE 120 SEAT(S) LICENSED TO
JOHN SMITH CO. 1/120 OF AVAILABLE LICENSES AND 1 RUNNING CLIENTS

License #1 Isotropix Clarisse iFX 1.0 117/120 REMAINING LICENSE(S)
3 running client(s):
cgstation-152@192.168.1.156
cgstation-028@192.168.3.5
cgstation-002@192.168.2.121
```

Congratulations! You are ready to start Clarisse.

Using Clarisse

In this chapter we will go through all the essential elements to help you understand how to use Clarisse. You may be in a hurry to explore the software by yourself or to start Clarisse's tutorials, but we strongly encourage you to do so only after you've got familiar with the next two sections: **Understanding the workflow** and **Clarisse Interface Basics**.

As we mentioned in the first chapter, we will try to keep this guide as simple as possible. At anytime you can access a more in depth reference documentation by hitting **F1** function key while running Clarisse or by clicking directly on Clarisse menu bar **Help > Clarisse Help...**

Command-Line Options

Clarisse supports several command-line options and the most common usage is to launch Clarisse while specifying the project to load.

```
clarisse my_project.project
```

There are also other options available.

Option	Description
<code>-config_file</code> <i>config_file</i>	Specify an alternate configuration file (ex: <code>-config_file clarisse.cfg</code>)
<code>-license_server</code> <i>SERVER:PORT</i>	Specify the license server location (ex: <code>-license_server LOCALHOST:40500</code>)
<code>-script</code> <i>script_file</i>	Specify the path of a script to execute at startup. Note the script is executed after the specified project is loaded.
<code>-search_path</code> <i>path</i> ...	Specify an ordered list of path used when looking for includes for example.
<code>-module_path</code> <i>path</i> ...	Specify the path of Clarisse's module. This argument is really helpful if you wish to specify custom or third party modules. This argument supports an argument list. Path order defines look-up priority.

Configuration and Environment

Clarisse stores many settings in a configuration file called `clarisse.cfg`. Clarisse also uses an environment file `clarisse.env` used to declare environment variables such as python path (`PYTHONHOME`) or license server location (`ILISE_SERVER`). The syntax of this file is really straight forward:

```
VARIABLE_NAME=VALUE
```

For example:

```
PYTHONHOME=c:\python27  
ILISE_SERVER=jupiter:40500
```

Note

Value of environment variables are platform dependent. Those variables are automatically imported to Clarisse as system variables. Please also note that environment variables defined in `clarisse.env` are overridden by variables with the same name if already declared in the shell environment.

For more information on variables please refer to Working with Variables section.

By default, those files are located in the home directory of the current user. The path to the home directory depends on the platform running Clarisse.

You will find those files in `%APPDATA%/Isotropix/Clarisse` on Windows, `/Users/CURRENT_USER/Library/Preferences/Isotropix/Clarisse` on Mac OS X and `~/.isotropix/clarisse` on Linux.

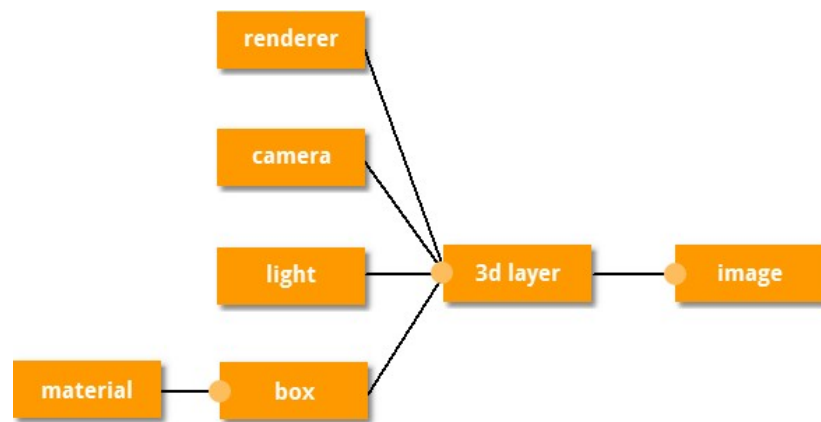
You'll also need to make sure Python libraries are reachable by Clarisse binaries. Typically this can be done by extending the `PATH` on Windows, `LD_LIBRARY_PATH` on linux and `DYLD_LIBRARY_PATH` on Mac OS X.

Understanding the Workflow

Clarisse introduces a procedural object model-based workflow approach. This approach is quite similar to a node-based workflow where you connect nodes to each other. In Clarisse, nodes are more advanced and are called objects. An object is basically an instance of a concept that can represent anything from 3d geometry to an image filter, to user interface elements.

Objects can define attributes of different types that are freely modifiable by users. For example:

- *Scene Item* position is defined by the *Translate* attribute of type distance.
- *Scene Item* parent is defined by the *Parent* attribute of type *Scene Item* reference.



Nodal representation of a simple 3d scene in Clarisse

As you can see in this figure, 3d renders in Clarisse are not just a result: geometries, lights, camera and renderer are referenced (plugged) by a 3d layer that is, itself, referenced by an image. All of these items are project items and an image can either be a typical 3d scene or a fully composited image used as final output. Note also that images can even be used as input textures for materials.

Any object attributes can be modified, animated, textured or driven by other object attributes. For example:

- to animate the rotation of a geometry object, users will animate its *Rotate* attribute.
- the *Diffuse* attribute of a material object will be textured by referencing a *Texture* object of *Fractal Noise* class.

All these objects are organized within a container we call a project meant to be saved to disk. A project doesn't necessarily describe a 3d scene, it can be used to act as a material or geometry library, to describe multiple scenes and even full sequences. A Project is basically saving, in a single file, all the objects and the user interface layout that have been created and organized within a Clarisse session.

In Clarisse, object data is loaded on an as-needed basis during evaluation: if nothing requires an evaluation, user-attribute modification won't trigger any computation nor consume memory. For example:

- if a user modifies the transformation of a parent object without displaying it or its child, no computation will occur in a *3d View*.
- if an image map or a 3d geometry is referenced by project items and no viewer is displaying them, no data will be loaded.

What this means for users is they get instant load on projects that potentially reference gigabytes of data.

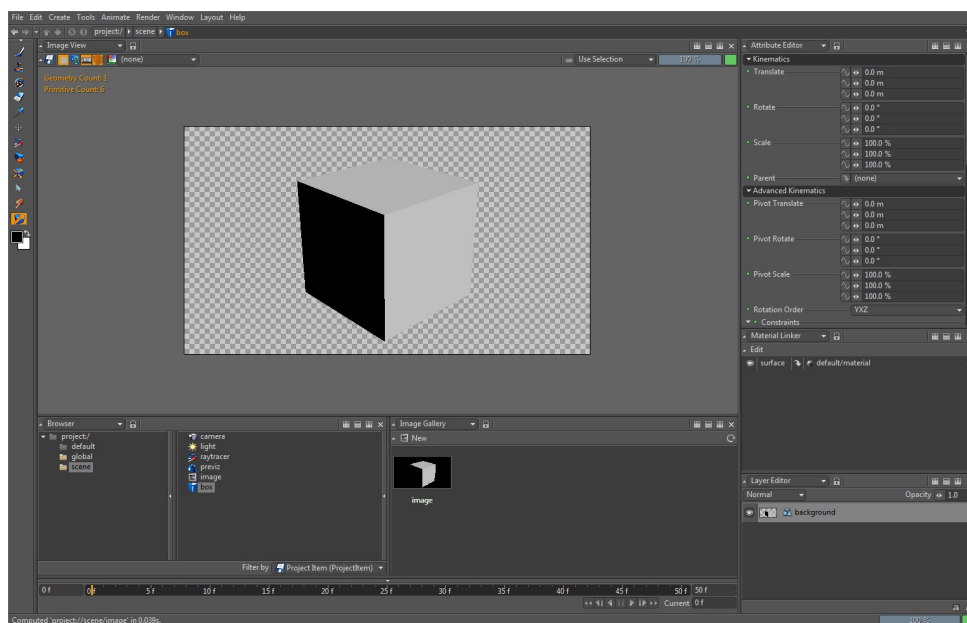
In the end the workflow is quite simple:

- object attributes (or properties) are displayed and modified through the Attribute Editor
- scene Items are displayed through the 3d View
- projects are browsed and organized through either the Browser or the Explorer
- 3d scenes or Images are rendered when displayed through the Image View
- data is loaded and computed on an as-need basis

For a more detailed explanation of the object workflow, please refer to the **Working with Objects** chapter.

Clarisse User Interface Basics

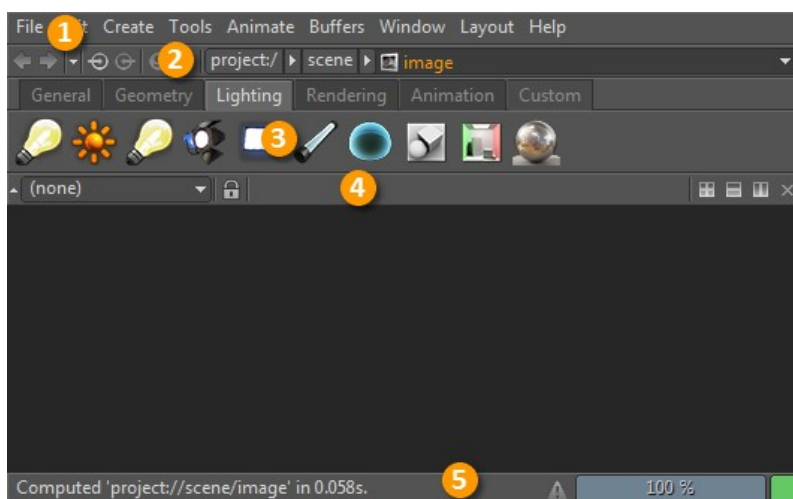
The Clarisse user interface can be fully customized to fit your particular needs. You can change the default color scheme of the application to match the external lighting of your working environment or customize entirely the layout of your workspace to suit your immediate needs. By default, Clarisse uses the *Render Workshop* layout preset and the *Dark* scheme.



Render Workshop in Clarisse

Clarisse Main Window

A typical Clarisse layout is made of several *viewports* that display *widgets*. A *widget* is a graphical element, usually an editor like an *Attribute Editor* or an *Image View* that is displayed in a *viewport*. A Clarisse main window will always display a *menu bar* (1), a *selection toolbar* (2), a *shelf* (3), a *viewport* (4) and a *status bar* (5).



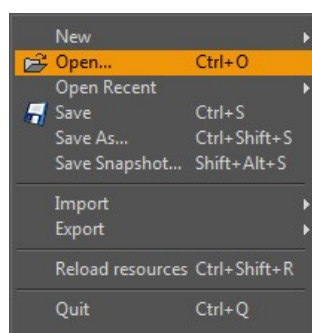
Empty Clarisse main window

Menu Bar

The application menu bar is always located at the top of Clarisse main window.

Each menu item defines an action that is processed when selected. These actions are organized into several categories.

Category	Function
File	Disk operations related to the current project. Loading, saving importing or exporting files.
Edit	Edit current selection, navigate and set preferences.
Create	Project item object creation
Tools	Object tools translate, rotate...
Animate	Keyframe and animation related actions
Render	Rendering related actions
Window	Manage new Clarisse window and sub window list
Layout	Edit and change current main window layout
Help	Access to user documentation

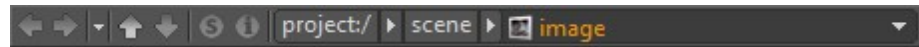


Clarisse file menu.

Note: Each time a key shortcut is bound to an action, the key sequence triggering the action will appear at the right of the action. For example, pressing Ctrl + O, means pressing simultaneously the Control and O keyboard keys to Open a file. The ellipsis (...) found at the right of an action name means it requires user input.

Selection Toolbar

The Selection toolbar is a mini toolbar designed to navigate and quickly manage Clarisse selections. By design the selection toolbar is intended to look and feel like the navigation toolbar found in popular file or web browsers. Please note that mastering navigation in Clarisse is very simple and it's one of the keys to working faster.



Clarisse selection toolbar

The selection toolbar is divided in two parts. On the left you'll find navigation buttons and on the right the selection field displaying the project path to the current selection.

Selection Field

In this figure, the object *image*, displayed in orange in the *selection field*, is selected. In this example, the objects full path in the project is `project://scene/image` where *project* and *scene* are contexts. Please note that contexts are displayed within click-able buttons.

The application selection field always display the current application context. In this example, the current context of the application is `project://scene`. New objects are always created in the current context of the application.

A quick word on Contexts

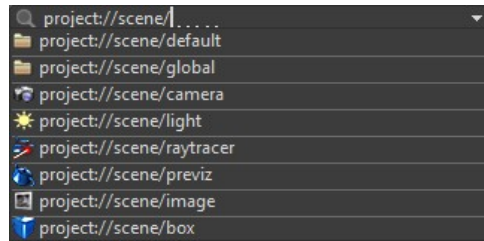
Contexts are quite similar to folders you can find on a computer disk. Contexts are used to organize your projects, handle visibility and much more... For more information please refer to the Understanding Contexts section.

In the same way, you would click on a folder button in a file browser to dive into a folder, clicking on a context button selects the context. Clicking on the drop down arrow pops up a menu displaying all the sub-contexts. If you select one of the sub-contexts the selection field path gets automatically updated. In a few clicks, it's very intuitive and quick to browse to the context you are looking for.

Multiselection

When a selection is composed of multiple objects, the selection field displays the name of the last selected object followed by, between square brackets, the number of selected items. In a multiselection with, for example, two objects *image* and *box*, the selection field displays `box[2]`. While in multiselection, there is also a drop down arrow displayed after the square brackets. If you click on the drop down arrow located after the square brackets, a menu displaying the content of your selection pops up.

Using the selection field, you can directly select objects. To select objects, click on the text field and directly type the path to your object. Notice while typing a selection list with an auto-complete feature pops up. Use the mouse or the Up and Down arrow key to scroll through the project items.



selection field with auto-complete

Navigation Buttons

Icon	Shortcut	Name	Function
	Backspace	Previous Selection	Set current selection to the previous one found in the selection history.
	Shift + Backspace	Next Selection	Set current selection to the next one found in the the selection history.
	None	Selection History	Display the selection history.
	Alt + Page Up	Select Input	Select objects referenced by the objects currently selected.
	Alt + Page Down	Select Output	Select objects referencing the objects currently selected.
	Ctrl + Page Up	Select Source(s)	Select current selection sources.
	Ctrl + Page Down	Select Instance(s)	Select current selection direct instances.

Selection toolbar icons

Most of these actions can be found in the *Edit* menu of the Clarisse menu bar. For more information about object referencing, instancing and sourcing, please refer to the Working with Objects section.

Selection History

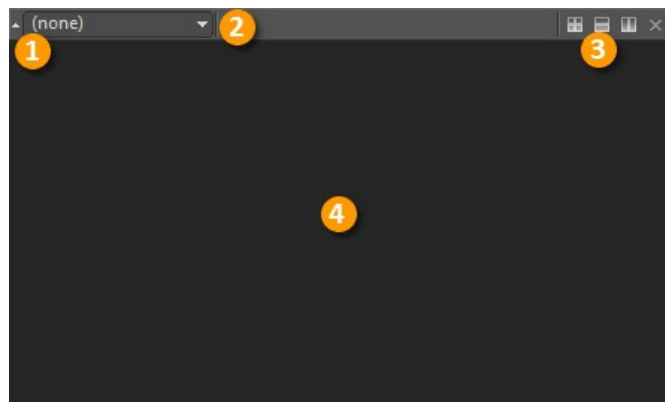
Each time the selection is modified, the previous selection is stored in the selection history. You can limit the size of the selection history using the Preferences Panel. To display the current selection history, click on the drop down arrow located on the left of *Next Selection* button. The selection history menu lets you browse the history menu and the content of selection sets. Using

this menu, you can change the current selection to a former one or you can explicitly change the selection to an individual member of a previous selection set. Selection sets are displayed using the square brackets formalism described in the Multiselection section. However, you can browse the set content by hovering the mouse over a selection set that has multiple items within it.

To clear the selection history, pop up the selection history and choose **Clear Selection History**.

Understanding Viewports

The *viewport* is an area of window space designed to display user interface elements that are called *Widgets* in Clarisse. Most of the time, those widgets are viewers or editors such as the *Image View*, the *Attribute Editor* or the *Layer Editor*. New widgets such as third-party widgets can easily be added to Clarisse using the Clarisse SDK. *Viewports* are the user interface entry point to create a fully customized layout.



An empty viewport

(1) Minimize Toolbar (2) Viewport Selection (3) Viewport Splitters (4) Widget Area

A viewport consists of a toolbar that has a pull down button (2) to select the widget and 4 buttons (3) used to manage the viewport layout. If you wish to save some space, the viewport toolbar can be minimized by clicking on the minimize arrow button (1).

Working with Widgets

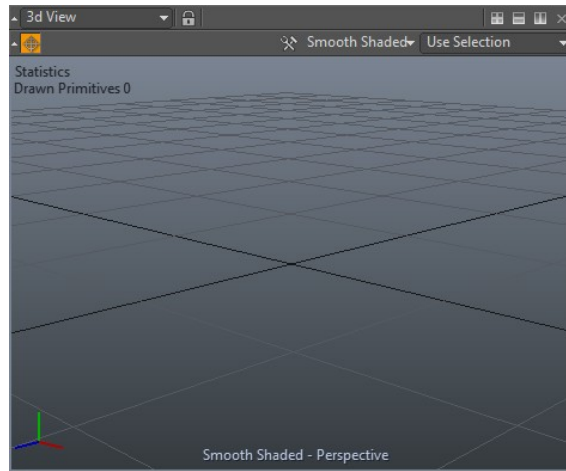
To display a widget in a viewport, you need to click on the pull down button (2) located left of the viewport toolbar. When clicked, a menu displaying the list of

all available widgets pops up. There is no limit to the number of widgets displayed in Clarisse. You can display as many widgets as you like.

Widget	Function
3d View	View to inspect and edit scene items in 3d
Attribute Editor	View and edit attributes of selected objects
Browser	Browse and manage project and project items
Class Explorer	Explore the content of projects sorted by Object Classes
Explorer	Hierarchical project browsing and managing
Graph Editor	View and edit FCurves
History Editor	Manage command (undo) history
Image Gallery	View all project images
Image View	View, edit, modify images
Layer Editor	Edit image layers and layer blending
Log	Application log display
Material Editor	Nodal representation of materials and textures
Material Linker	Assign materials to shading group and toggle shading group visibility
Progress View	Display application progress bars
Resource View	Display all resources used in the project
Timeline	Edit current project time
Tools	Display application tools
Tools Options	Display current tools options
UV View	View and inspect geometry UV maps
Variable Editor	View and edit project and system variables

Example of widgets and their purpose

Click on the menu item to select your widget, it will be displayed in (4). Note the name of the selected widget is displayed in (2). If you wish to change the widget that is displayed click on (2) again and select (none) to remove the widget or choose any other widget.



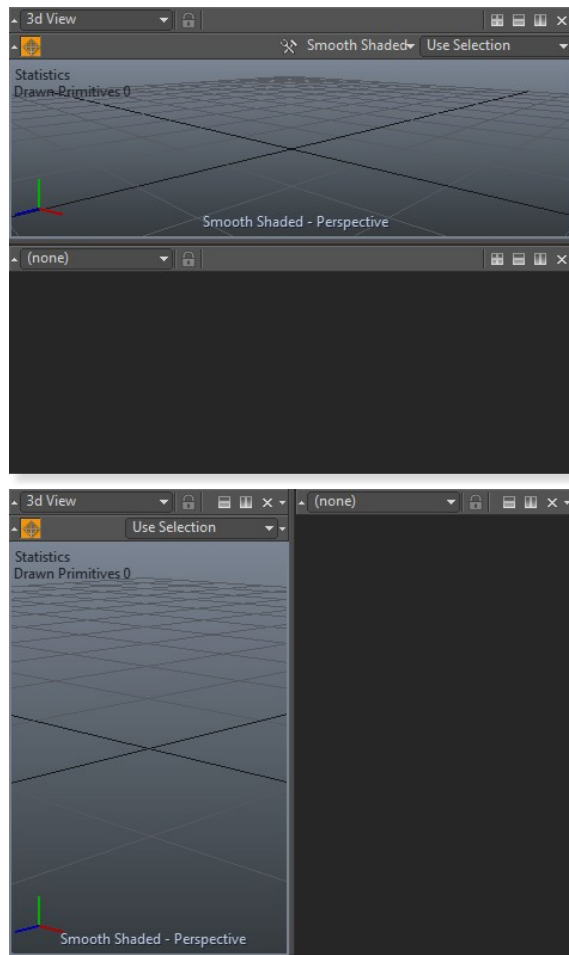
A viewport displaying a 3d View

Splitting Viewports

Using the button displayed in (3) Viewports can be split hierarchically to embed new sub viewports. In the same way viewports can be removed using the cross button.

There are 3 ways to split a viewport:

- **horizontal split**, splits a viewport in half horizontally
- **vertical split**, splits a viewport in half vertically
- **quad split**, splits a viewport in four both horizontally and vertically



A viewport after an horizontal split on the left
and after a vertical split on the right

Once split, you can resize viewports by dragging on either the horizontal or the vertical split bar.

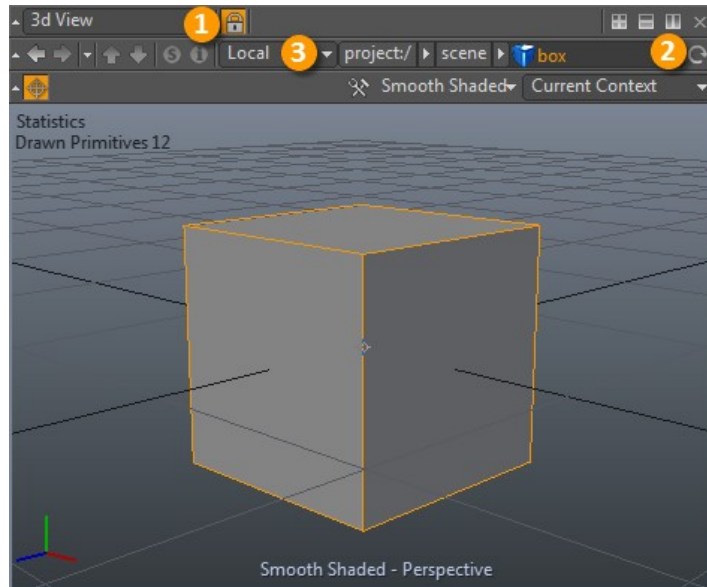
Tip

To clear all viewports in the workspace, you can use from the Clarisse menu bar **Layout > Clear All Viewports**.

Widget Custom Selection

By default, widgets are synchronized to the application selection (aka global selection). Each time the selection changes, widgets synchronize to the new selection and update accordingly their display. For example, if a user choose a box object in a *Browser*, a displayed *Attribute Editor* will display the box object attributes.

However, users often want to keep the selection sticky (to perform drag a drop...) or they want to make editors synchronize themselves on a shared selection slot. Clarisse allows widgets to display their own selection toolbar to let user control, locally, the behavior of their selections.



A 3d view with a custom selection

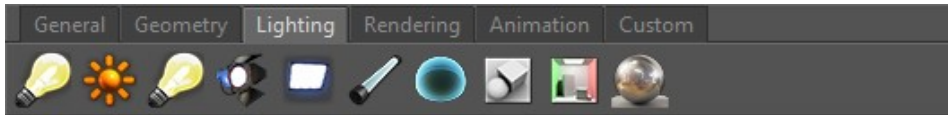
If a displayed widget depends on the selection, a lock button will systematically appears in the *viewport* toolbar. Clicking the lock button (1) enables widget custom selection mode. In this mode a widget manages its own selection (*Local* selection) and gets disconnected from the application global one. To manage its selection, the widget gets its own *Selection Toolbar* that is very similar to Clarisse main window one. In this mode, widgets can still import the current application selection by clicking on the *Refresh* button (2) of their *Selection Toolbar* or by hitting F5.

Widget custom selection offers 3 different modes that can be changed using (3):

- *Local* mode, the widget has its own selection
- *Global* mode, the widget is synchronized to the application selection
- *Slot* mode, widgets sharing the same slot share the same selection. This mode links widget selection with other widgets. There are up to 8 selection slots available.

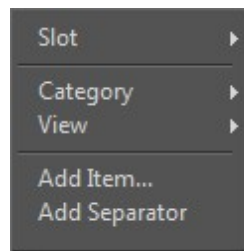
Shelves

Clarisse 1.5 introduced a shelf widget dedicated to scripts sorting. Shelves can be displayed in the application main window or as widgets in a viewport. Clarisse supports up to 8 different shelves that can have independent configurations. Within shelves, scripts can be arranged in categories, have a visual name, a description and display a custom icon.



Default Clarisse shelf.

To display Shelf menu options right click in any empty area.



Adding a new script

To add a new script, bring the shelf menu and select **Add Item...**

The 'Add Item' popup dialog box contains the following fields and controls:

- Title:** A text input field with a value of 'My Dummy Script' (labeled 1).
- Description:** A text area with a value of 'This is an example' (labeled 2).
- Category:** A dropdown menu with 'Custom' selected (labeled 3).
- Category Custom:** A text input field with a value of 'My Category' (labeled 4).
- Script Path:** A text input field with a value of '/data/clarisse/scripts/dummy.py' (labeled 5).
- Icon:** A dropdown menu with 'Custom' selected (labeled 6).
- Icon Custom:** A text input field with a value of '/data/clarisse/scripts/dummy.png' (labeled 7).
- Buttons:** 'Add' and 'Cancel' buttons at the bottom right.

Add Item popup

(1) Script Name* (2) Script Description (3) Category (4) Custom category name
(5) Script file path* (6) Icon Selection (7) Custom icon file path

* Required fields

When specifying a file path for either the script or the icon, the file isn't copied. Each time the script is executed the file is loaded from disk. **If you were to delete those files, Clarisse would be unable to execute the scripts.**

To add the script to the shelf, just press **Add**.

Tip

When using a custom icon, we recommend using 32x32 sized images. However, specified images that don't match the resolution are automatically resized to 32x32.

Editing an existing script

Once a script is added to the shelf, its settings can be edited. To edit an existing shelf item, right click on it. Press **Edit Settings...** to bring the settings panel.

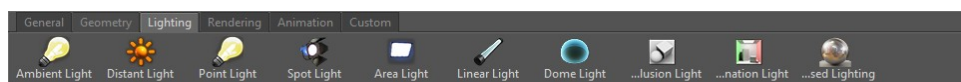
Removing a script

To edit a shelf item, right click on it and press **Remove**. This action is not undoable.

Customizing Display

Shelves display can be customized. You can choose to sort the shelf using item categories to automatically generate tabs (default). You can also choose to toggle script's name display name.

To display script names, bring the shelf menu and choose **View > Icons With Titles**. To disable script name display choose **View > Icons**



Icon names and category tabs

To disable category tabs sorting, bring the shelf menu and choose **Category > All**. To enable category sorting choose **Category > Group By Category**.

Resetting Configuration

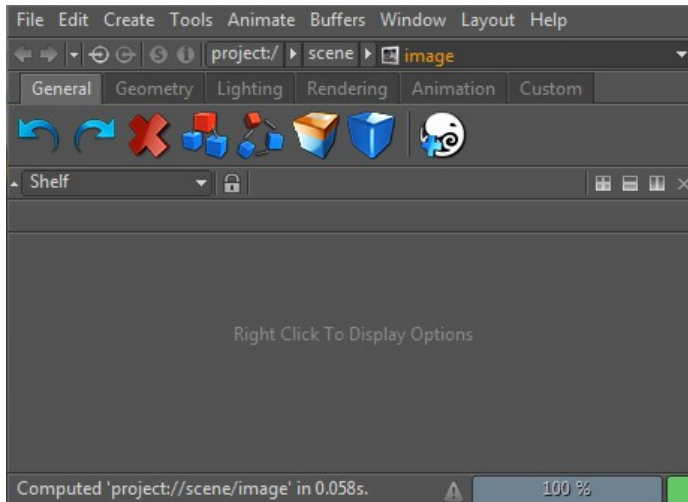
You can revert to the original shelf configuration (factory settings) by selecting **Layout > Shelf Toolbar > Reset to Default** from the application main menu bar.

Note

You will loose all your shelf customization. This action isn't undoable.

Slots

Clarisse supports up to 8 different shelves that have independent configurations. To change to another slot, bring the shelf menu. In the **Slot** sub menu choose any slot you like.



Main shelf displaying slot 1, the other one displaying slot 2 (empty)

Drag and Drop

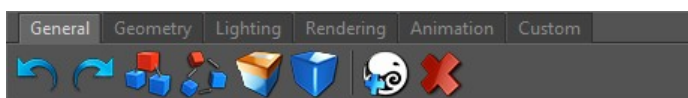
Shelves support drag and drop to reorganize the display order of your scripts.



Drag the item, here we chose Delete. An insert marker appears displaying where it will be dropped.



Then simply release the mouse button.



In the same way, shelves support drag and drop of tabs. Finally, you can drag and drop tabs or items between shelves.

Hiding Main Shelf

You can show/hide the main window shelf by selecting **Shelf Toolbar > Show/Hide**.

Status Bar

The status bar is always located at the bottom of Clarisse main window. It is divided in 3 parts.



Clarisse main window status bar

(1) Application Message Log (2) Evaluation Progress Bar (3) Evaluation State

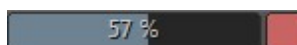
The Application Message Log displays the latest log message while Evaluation Progress Bar and Evaluation State both display informations on running evaluations.

Clicking on the Application Message Log (1) opens a Log window, and clicking on Evaluation Progress Bar (2) opens the Progress View.

Evaluation Status

In Clarisse, evaluations are almost always computed in the background. They can range from an image rendering, to a geometry deformation computation, to a file loading, to mip-map generation...

Users can still use Clarisse and work on scenes while evaluations are running in the background. This is why the status bar always displays the evaluation status to let users know what's happening in the background. When an evaluation is running the Evaluation Progress Bar (2) displays its current percentage of completion while the Evaluation State (3) is red and turns green when the evaluation has ended.



Status during an evaluation

Customizing the Interface

At any moment, even during a running evaluation, the Clarisse user interface can be fully customized. **Please note layouts are saved in project files.**

Using Workshops

To quickly modify the layout, we have provided a set of predefined layouts called workshops. To select a workshop go to **Layout > Presets**.

Workshop	Favored usage
Animation	This layout is optimized for animating
Shading	This layout is optimized for material shading
Render	This is the default layout designed to work on final images

You can also start a layout from scratch. To clear all the viewports in a main window select **Layout > Clear All Viewports**.

Creating new main windows

In Clarisse you can create as many Clarisse main windows as you want. To create a new main window select **Windows > New Clarisse**. It's important to stress the fact that main window layouts in Clarisse are independent. For example, you can set a main window with a *Render workshop* while having, at the same time, another main window set with an *Animation workshop*.

Creating Widgets in sub-windows

Instead of displaying *Widgets* inside viewports, you can display them in sub-windows using the **Window** menu and selecting the widget you want. For example, to display an *Attribute Editor* as a sub-window go to **Windows > Attribute Editor**. Sub-windows are owned by and parented to the Clarisse main window that created them:

- They float directly above the main window owner (*This feature isn't working on Mac OS X*)
- If you close the owner window all owned sub-windows are closed

You can toggle the visibility of sub-windows by pressing the **Tab** key or through **Windows > Hide/Show Subwindows**. You can also view the list of sub-windows through **Windows > Active Windows**.

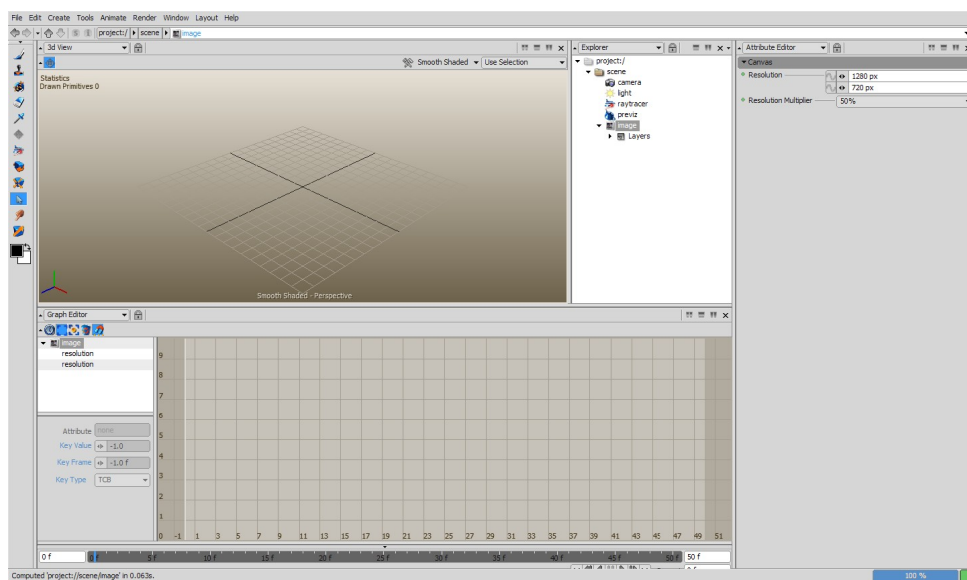
Changing User Interface Color Scheme

By default, Clarisse runs using the Dark color scheme. This color scheme has been fine-tuned to maximize user interface contrast for artists working in studios with low-light environments. Clarisse provide a set of different fine-tuned color schemes to match most lighting environments.

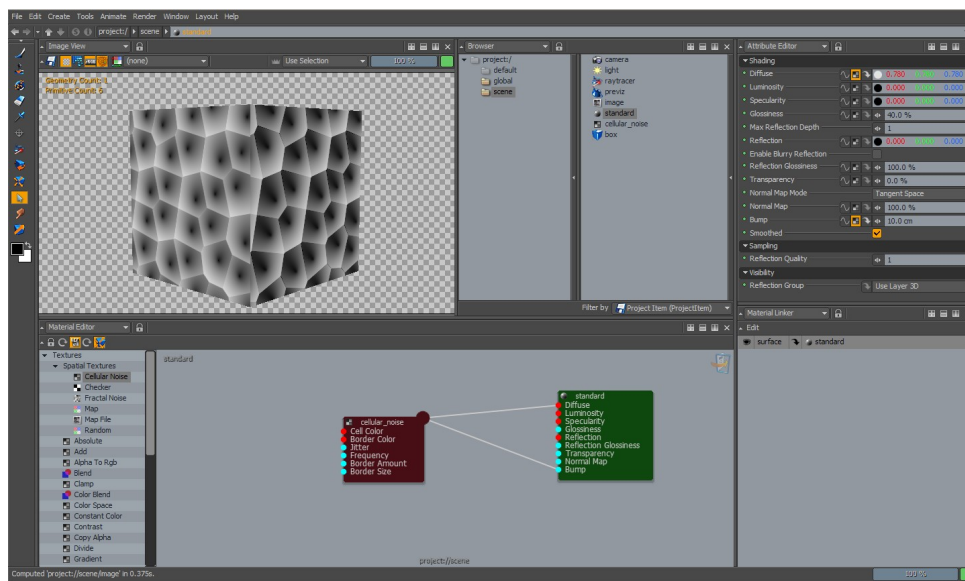
Color Scheme	Command Line Name	Favored usage
Light	light	For bright environments
Dark Alt2	dark_alt2	Dark scheme with bright bluish text field base color
Dark Alt	dark_alt	Dark scheme with bright greyish text field base color
Dark	dark	For low light environments
Darker	darker	For very low light environments. The perfect scheme if you work in a cave filled with vampires...

If you wish to change the current color scheme, open the preference panel using **Edit > Preferences** or **Ctrl + K**. You can change current color scheme in the *User Interface* tab using the *Color Scheme* pull down. The current color scheme is saved in the Clarisse configuration file, so each time Clarisse is launched, it will use the specified color scheme.

You can also set the Clarisse color scheme directly from the command line:
`clarisse -scheme darker`



Animation Workshop with Light color scheme



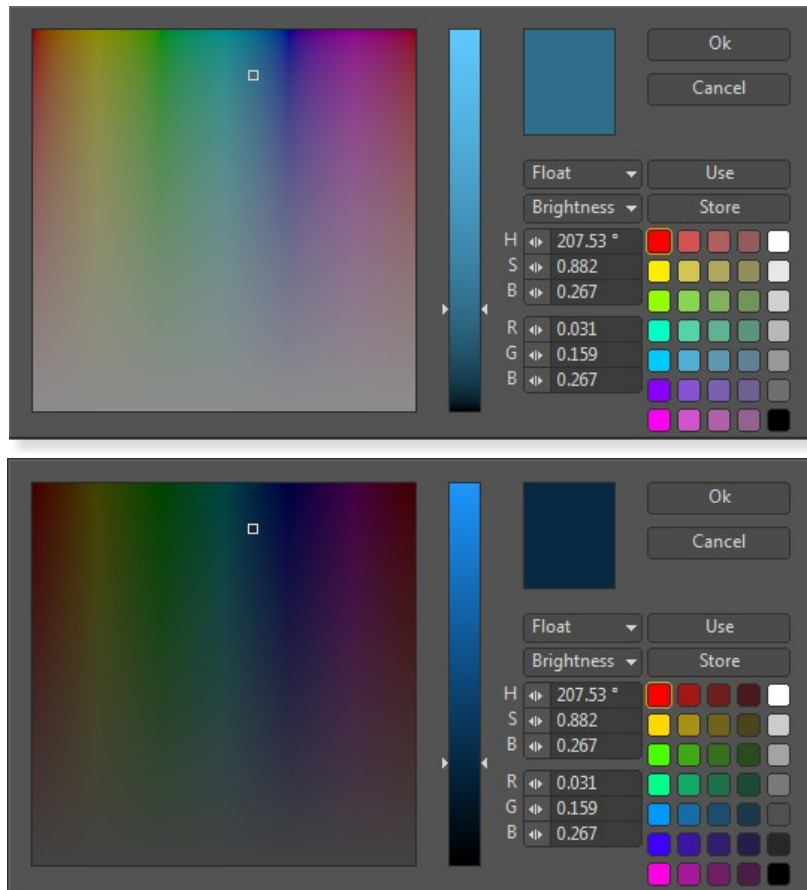
Shading Workshop with Dark Alt 2 color scheme

Changing User Interface Color Display

By default, Clarisse runs using the *sRGB* color display mode, to compensate for how humans perceive light and color. All colors, gradients and images displayed in the user interface are gamma corrected. The color you pick is then the color you get. Please note, this setting is only a display mode. It doesn't affect image internal data as it's kept *Linear*. The color display mode can be changed at anytime even during an evaluation using **Edit > Preferences** or **Ctrl + K**. You can find the *Color Display Mode* setting in the *User Interface* tab. To change the current color display mode use the *Color Display Mode* pull down.



3d view in sRGB (left) in linear (right)



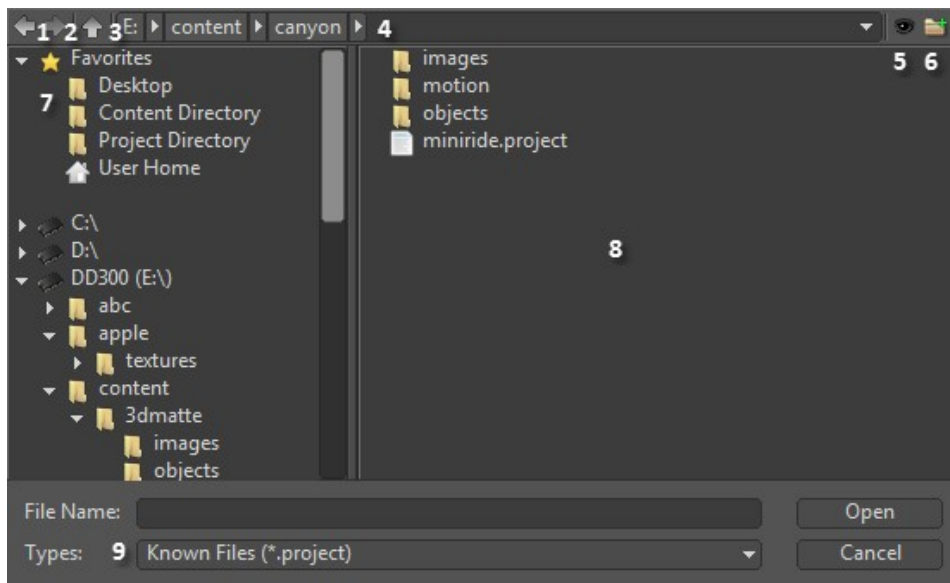
Color picker displaying a same color in sRGB (top) in linear (bottom)

File Browser

By default, Clarisse has its own file browser dedicated to browse, open or save files. You can alternatively decide to use the native one on Windows and Mac OS X platforms only. To use the native File Browser, go the **Edit > Preferences... > User Interface** and change *File Browser* to *Native*.

Overview

The File Browser is divided into several parts.



(1) Previous Folder (2) Next Folder (3) Parent Folder (4) Current Folder (5) Show Hidden Files (6) Create Folder (7) Favorites (8) Current Folder Content (9) File Type Filter

Favorites

Favorites are quick links allowing you to quickly browse folder contents. By default, you can quickly browse to Desktop, Content Directory, Project Directory or Home. You extend default favorites by adding custom ones.

To add a favorite folder, right click on a selected folder in the folder list view (8) and select Add to Favorites in the pop up menu. Please note custom favorites are saved in Clarisse configuration file.

Working with Objects

In this chapter, we will go through the Clarisse object-based workflow. We will explain what *objects* are, what a *class* is and what all the related concepts such as *contexts*, *object-instancing* and *object-localization* are. Most of these concepts are new and key to understanding how Clarisse works. Comprehension of these concepts simplifies project management and boosts user workflow while reducing the learning curve of the software.

Understanding Objects

As introduced in Understanding the Workflow, Clarisse provides a procedural object-model-based workflow. This workflow is quite similar to node-based workflow where you connect nodes to each others. However in Clarisse, nodes are *objects* and *objects* are instances of *classes* describing concepts that can be anything from abstract 3d geometry, to implicit sphere or user interface elements. For people with a programming background, this model is directly inspired from the object-oriented programming (OOP) paradigm. Object-model-based workflow is extremely powerful and easily extendable.

Objects Classes

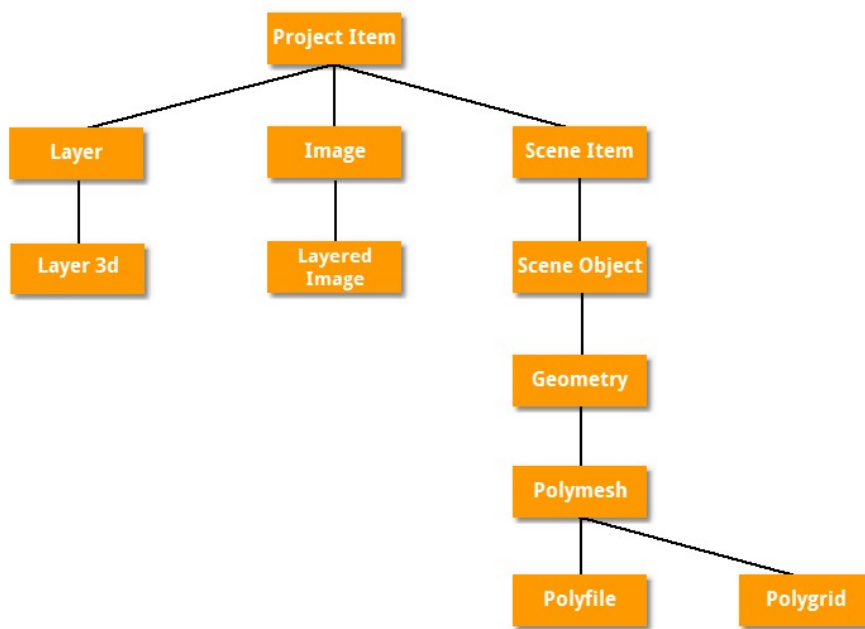
In Clarisse, every single item is based upon *objects* of different *classes*. *Widgets*, *Scene Items*, *Geometries*, *Lights*, *Images*, *Filters*, *Textures*, *Renderers*... are all built-in *classes* Clarisse provides. For example, when a user adds a point light in his scene, an object of *Light Point* class is created in his project. In the very same way, if an *Image View* is displayed in a *Viewport*, there is necessarily an *object* of *Image View* class created in the project. Generally speaking, each class has a name, can have data, can have an engine processing its data and finally *Attributes* that can be of many types.

Class Inheritance

In Clarisse, classes can inherit from other classes. For example, a *Layered Image* (an image composed of layers) inherits from a generic (or abstract) *Image* class. This *Image* class describes what an image is in Clarisse: *Image* class defines two attributes, *Resolution* and *Resolution Multiplier*. It also defines the output image data and an engine that manages this image data. For users, *Attributes* are what really matters, like when they modify the *Resolution* attribute, it modifies the

image data resolution in the image object. In fact, each time an attribute is modified, the class engine will eventually process its data.

For the *Layered Image* class, inheriting from *Image* class means it inherits from all its attributes and features. However, *Layered Image* also adds new attributes and its base *Image* class doesn't have an engine related to the objects of the *Layer* class it handles. In the very same way, the class *PolyFile* (polygonal mesh referencing a file) inherits from *Polymesh* which defines everything related to *Polygons* and *Subdivision Surfaces*.



A class hierarchy example

Without getting too technical, classes in Clarisse are fully dynamic. Extending Clarisse with new concepts and features is very easy. Even if the subject can be somewhat abstract, the user workflow in Clarisse is extremely straight forward: users end up creating objects, connecting them to each other while modifying their attributes.

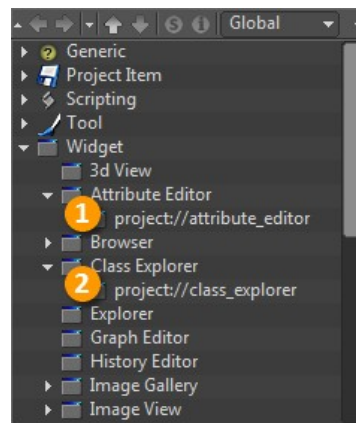
Role while connecting Objects

Inheritance and its resulting class hierarchy is a fundamental concept in an object-model-based workflow. Indeed, connections between objects are only achieved via object-attributes of type *Reference*.

For example, *Scene Item* parenting is achieved via the attribute *Parent* which is of type *Reference*. This attribute only accepts reference objects of class *Scene Item*: if you can't parent a *Scene Item* to an image, you can parent it to a *Camera*, a *Light*, a *Geometry* or anything that inherits directly or indirectly from the *Scene Item* class.

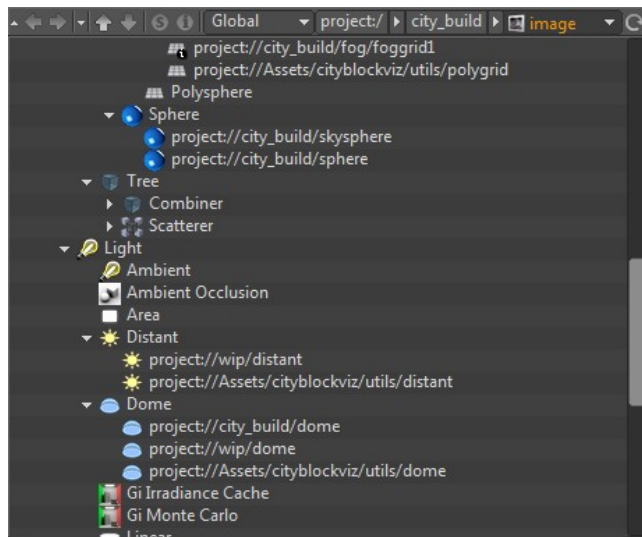
Browsing the Class Hierarchy

Clarisse provides the *Class Explorer*, a hierarchical widget that allows you to see, during a session, the class inheritance hierarchy and all the objects in the project sorted by their classes. To display a *Class Explorer*, select a *Class Explorer* in any *Viewport* or go to **Window > Class Explorer**.



The Class Explorer expanding the Widget Class

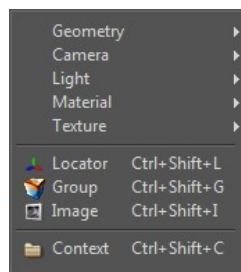
We see clearly here the **Attribute Editor** (1) (`project://attribute_editor`) and **Class Explorer** (2) (`project://class_explorer`) objects in the project. We can also see that both classes, *Attribute Editor* and *Class Explorer* are underneath the *Widget* class.



The class explorer on a typical production scene

Creating Objects

Even if objects can be created in many different ways, they can be created directly from the **Create** menu from any main window **menu bar**.



Create menu

In this menu, objects are sorted by categories. For example, you will find geometries under **Create > Geometry** or you will find materials under **Create > Material**.

Each time a new object is created using the **Create** menu, it will be created within the current application context. If you create, for example, a new material while the current application context is set to `project://scene`, the material will be created in the context `project://scene`.

Note

Newly created objects are always selected.

Selecting Objects

In Clarisse, you can freely mix items within a selection set: *Contexts with Images* or *Materials*, *Geometries*, *Layers*...

There are many different ways to perform item selections. To select the content of the current context, go to **Edit > Select All**. To deselect the current selection go to **Edit > Deselect All**.

You can also select objects using the *Image View* by picking in your final images, using the *3d View*, using the **Selection Toolbar** by typing and searching the object you are looking for... or you can use any of dedicated project management widgets:

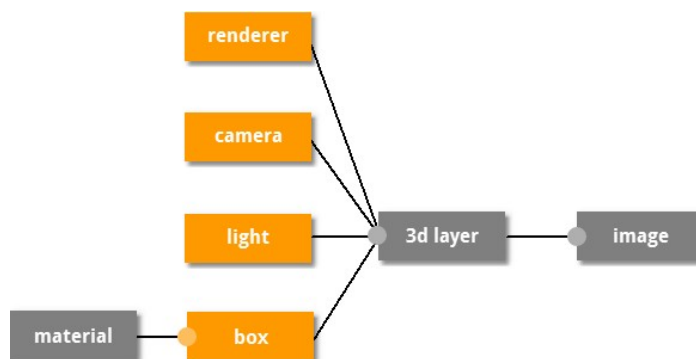
- The *Browser* that provides a file manager-like user interface
- The *Explorer* that allows you to browse your project hierarchically
- The *Class Explorer* that provides a hierarchical view of objects sorted by their class name

Selecting Dependencies

As objects can be connected to others, they can have input and output dependencies. Another type of dependency is the instancing dependency when objects are instances of other objects.

Input Dependencies

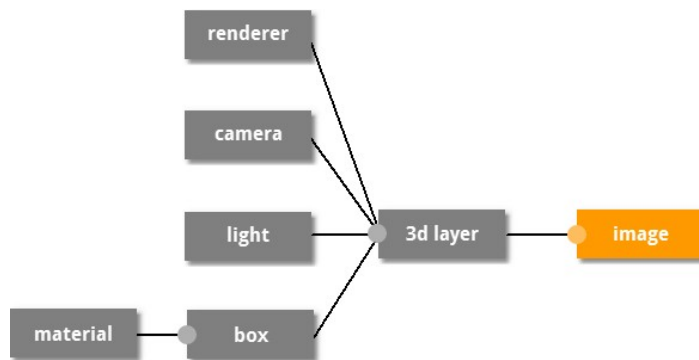
Input dependencies are all the objects that are directly connected to a specific object. In the following example, the 3d layer input dependencies are the renderer, camera, light and box.



3d layer input dependencies in orange

Output Dependencies

Output dependencies are all objects an object is directly connected to. In the following example, the 3d layer output dependency is the image.



3d layer output dependency in orange

Selection Tools

Clarisse provides different tools, all available in the **Edit** menu, to quickly select either sources, instances or input/output dependencies:

Name	Function
Select Instances	Select instances of selected objects
Select Instances Recursively	Select recursively instances of selected objects (instances of instances...)
Select Sources	Select sources of selected objects. If an object of the selection hasn't got a source, it is considered as a source object
Select Sources Recursively	Select recursively sources of selected objects.
Select Outputs	Select objects that are referencing selected objects: if applied on a Layer 3D, it selects the Image referencing the layer.
Select Outputs Recursively	Select the referencing branch
Select Inputs	Select the object referenced by selected objects
Select Inputs Recursively	Select the reference branch
Select All Dependencies	Select all dependencies from the current selection

Renaming Objects

By default, new objects are named after their class name. When you create a new *Standard* material, the newly created material will be named *standard*. However, sometimes it may happen this default name is already taken by another object, which leads to a naming issue. Indeed, whatever their class is, object names are always unique within a context. It's very similar to files in folders: you can't have two files sharing the same filename.

In Clarisse, naming conflicts are automatically resolved by appending two conflicted names a suffix number. For example, if there is already a *standard* material in the current context, the new object gets automatically renamed *standard1*. The next *Standard* material to be created will be renamed to *standard2* and so on...

To rename explicitly a selected object, press the **F2** key or go to **Edit > Rename...** to popup the *Renaming Window*.



Renaming window

If the *Renaming Window* doesn't appear:

- No object or no renamable object is selected
- The user pressed **F2** in a Browser or an Explorer and in this case the rename is performed directly on the item.

Limitations for Naming Objects

You can set the name you want to your objects but they will be automatically renamed to sort out the following limitations:

- names can't start with a digit
- names support only underscores and alphanumeric characters
- names must be unique in a context

Deleting Objects

Deletion is performed on a selection. To delete selected items press **Del** key or go to **Edit > Delete Selection**.

Moving Objects

Moving objects is performed via the use of the Browser or Explorer widget.

Editing Objects

Objects inherit their attributes from their classes and attributes can be modified using the Attribute Editor. Generally speaking, each time an object is modified, there is always at least one of its attributes that has been modified. In Clarisse, it's very simple and editing attribute values is the only way to modify objects:

- To translate a selected *Scene Item*, the *Translate* tool modifies the item's *Translate* attribute.
- When assigning a material to a geometry via a drag and drop, the geometry *Materials* attribute is modified.
- Parenting a light to a geometry is achieved by setting the light *Parent* attribute to the geometry

Attributes

Attributes are modifiable object-properties. They are defined by a name, a type, flags, modifiers and a category:

- the name identifies the attribute
- the type defines its value type (integer, real, reference...)
- flags are related to attribute visibility
- modifiers are Textures or FCurves... modifying input or final attribute value
- the category is a hint used to logically sort attributes of an object

Name

Names identify attributes in an object. This is why there can't be two attributes in a object that share the same name. Attribute names, also share some of the limitations object names have:

- names can't start with a digit
- names support only underscores and alphanumeric characters

To circumvent these limitations, displayed names are automatically converted to a more human readable format. For example, the attribute `enable_subdivision_surface` is displayed as *Enable Subdivision Surface*.

Note

To get the technical name of an attribute, look at the Reference chapter of the web documentation, search for the module class and select the Technical tab to display the technical documentation of the module.

Type

There are many types of attribute values and each of them is displayed differently throughout the application. Attributes can include single or multiple values. When multiple, attributes can either be displayed as lists or successive blocks of values.

Attribute type can also be of type Object. In this case, objects are embedded by the attribute. Embedded objects are standard objects but they are owned by and bundled with the parent object and they can't be directly referenced by anything other than their owners. For more information on attributes, refer to the Attribute Editor chapter.

Flags

Flags define the attribute visibility properties. Attributes can be public (default), private or read-only. Public attributes are freely modifiable by users whereas private attributes are hidden and thus unavailable. Read-only attributes can't be modified by the user. However, the read-only state isn't necessarily permanent: depending on user actions, the read-only flag state can change.

For example on *Polymesh* objects, if *Enable Subdivision Surface* is unchecked, all subdivision surface attributes are flagged as read-only. However, if the attribute is checked, they change to public.

Protection

Attributes can be in a protected state. A protected attribute is an attribute used by a running evaluation. This only happens when a running evaluation depends on the object that has the protected attribute: for example, when the user orbits a camera referenced in a 3d layer used in an image currently being evaluated. As the result of the evaluation needs to be up-to-date, modifying a protected attribute interrupts and restarts the running evaluation.

Note

Attributes can't be protected if no evaluation is running in the background.

Modifiers

To override their values, attributes can be driven by modifiers. There are several kinds of modifiers: textures, FCurves (animation) or source attribute modifiers when the object is an instance and its attribute isn't localized (see Instancing Objects).

Attributes can have multiple modifiers at the same time but only one is actually driving the attribute. The order of driver priority is defined below (highest to lowest):

Priority	Modifier
1	Texture
2	FCurve
3	Attribute Value
4	Attribute Source (can be textured, animated...)

If an attribute is textured, the attribute is only driven by the texture, even if the attribute may have other kinds of modifiers. If an attribute has no texture but is animated, then the FCurve is the driver.

Category

The category logically sorts attributes into groups when displayed. For example, *Resolution* and *Resolution Multiplier* attributes share the same category *Canvas*. They are then displayed in the same group.

The category is also a visual hint helping to improve the understanding of what the attribute does. For example, the attribute *Quality* of a distant light is under the category *Sampling*. This lets the user know the attribute controls the sampling quality of the distant light.

Note

When an attribute has no category defined, it is considered as member of an implicit category named *General*.

Path

Attributes can be only identified in Clarisse projects. The attribute syntax is

pretty straight forward:

```
object_path.attribute_name[value_index]
```

Where:

- `object_path` is the path to the specified object. It can be either relative or absolute.
- `attribute_name` is the real name of the attribute (also called technical name). Please refer to Attributes Name section.
- `value_index` is a number specifying the array component (0 being the index of the first element like in C/C++).

Note

for attributes with single values, you can skip the `value_index` and just specify `object_path.attribute_name`.

For example:

```
project://scene/assets/locator.rotation_order[0]  
project://scene/assets/locator.parent
```

Duplicating Objects

You can duplicate objects with a number of different methods. You can either create copies or instances of objects.

Copies

A copy of a source object creates a new identical copy: all source attributes values and embedded objects are copied. After a copy is created, there is virtually no difference between the source and the copy at the exception of the copied item name.

To duplicate objects using deep copies, you need to select the source object(s) and go to **Edit > Copy** or press **Ctrl + C** to copy items to the clipboard. Once items have been copied to the clipboard you can use **Edit > Paste** or **Ctrl + V** to paste copies as many times as you want.

You can also copy and paste items between two Clarisse applications, or paste the content of a clipboard in a text editor to create custom asset libraries that you would paste later in Clarisse. Here is the content of the clipboard after copying an image into the clipboard.

```
#Isotropix_Clarisse_Clipboard_Serialization 0.9
Image {
  name "image"
  #version 0.9
  copy_from "project://scene/image"
  active_layer 0
  resolution 1280 720
  resolution_multiplier 1
  layers ".background"
  embedded_objects {
    Layer3d {
      name "background"
      #version 0.9
      copy_from "project://scene/image.background"
      active_camera "project://scene/camera"
      renderer "project://scene/raytracer"
    }
  }
}
```

To learn more about Clarisse file formats syntax, refer to **Clarisse File Formats**.

Instancing Objects

In Clarisse, instancing is not a simple render-time duplicate. Instancing is, instead, a very powerful feature that applies for objects from any class and that is meant to be used as a replacement for copies. An instance of an object is an exact copy of its source. However, it's a dynamic copy as all of its attributes are driven by its source object: when you modify one of its attribute-values both the source and instances are modified alike.

Instancing can be used for many things that range from traditional geometry instancing to object-level layer modification. For more information on instancing, localizing and unlocalizing attributes, please refer to the Attribute Editor chapter.

Localizing Attributes

By default, instance attributes are driven by their source. To break this implicit attribute link, you need to localize it. Once localized, attribute values and modifiers are copied and the attribute can be freely modified without being affected by its source.

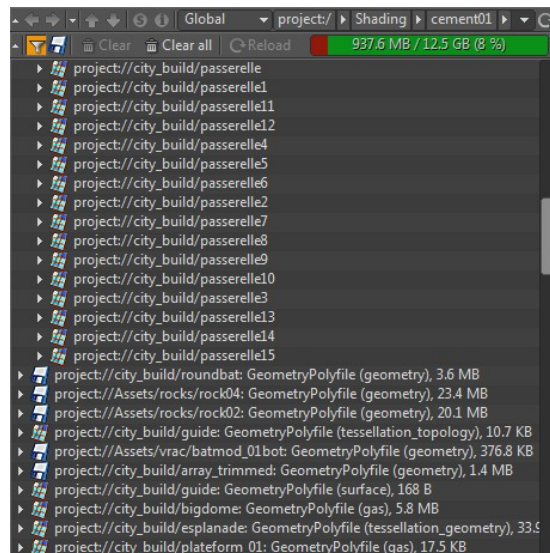
You can localize any attribute you want, and localization can be undone by unlocalizing the attribute. To use instances as a layer of modification you need to localize the attribute you wish to modify to freely adjust it.

Unlocalizing Attributes

Localized attributes can be unlocalized. Unlocalized attributes are driven by source attributes. When you create an instance of an object, all its attributes are unlocalized.

Memory Manager

In Clarisse, you can freely duplicate objects bound to heavy data such as geometries without caring about memory usage. Clarisse is built upon a smart memory manager that detects and eliminates, automatically and on the fly, redundant data. If you create thousands of copies of a geometry with heavy data, data is automatically shared across copies to be stored only once in memory. If you wish to see how much redundant data has been eliminated, you can use the Resource View widget.



Redundant data elimination efficiency of 92% in the Resource View

Understanding Contexts

In traditional animation software, scenes are mainly displayed as a parenting hierarchy and Locators are even hacked to organize production scenes. Moreover, deformers, materials, textures... are typically invisible from the scene hierarchy and a unified view that displays every single item in a scene is very rare.

In an object-model-based workflow, all objects are connected within a unique dependency graph:

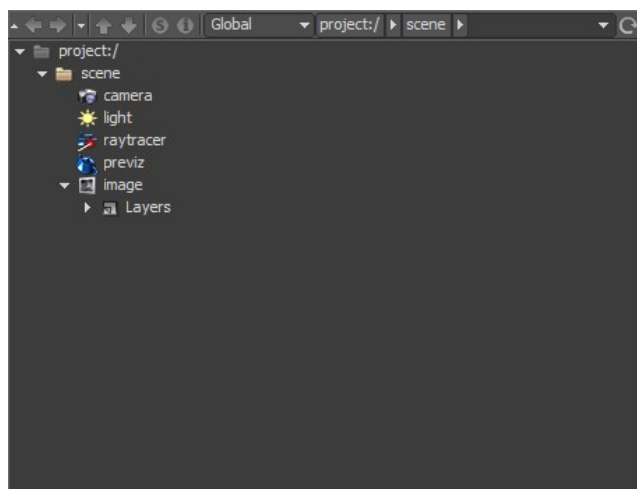
- Whole evaluation from deformation, to shading and texturing, to lighting, to rendering and compositing is fully procedural and completely unified.
- Objects can be shared. For example, a texture referenced in a material can be shared with others to be referenced by deformers.

Usually displayed as a network of nodes, dependency graphs can quickly get complex and unintuitive when working on typical production scenes. This is why we introduced, in Clarisse, a new concept called *Context*, designed to simplify project organization and its management.

Contexts as Folders

Contexts are to objects what folders are to files. In the very same way files are organized in folders and, in Clarisse, sub-folders objects are organized in contexts and sub-contexts.

Now, let's take a look at a Clarisse default scene. To create a new scene, go to **File > New > Scene** or **Ctrl + N**



Explorer view on a default scene

In a default scene, we can see there are 2 contexts: project and scene. The context project is the root of the project, very much alike / on UNIX or C:\ on Windows. Here, in the context scene, we can see several objects: a camera, a light, a raytracer, a previz and an image.

Context and Object Path

In the same way, files and folders are located with either an absolute or a relative path, in Clarisse, objects and contexts are also located with absolute or relative paths. Consequently, in a new scene, the absolute path for the context scene is project://scene. For objects within scene, their absolute path is then:

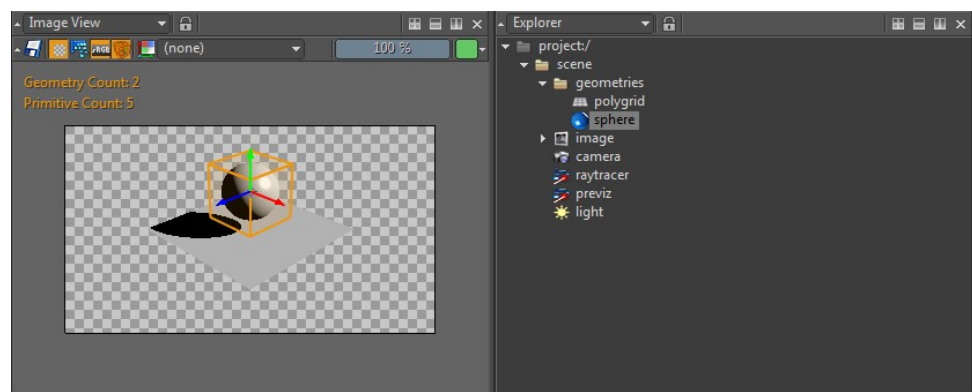
```
project://scene/camera  
project://scene/light  
project://scene/raytracer  
project://scene/previz  
project://scene/image
```

Context and Object Visibility

While Contexts have many similarities with folders, they also have differences. They play a key role in project management and in setting object visibility.

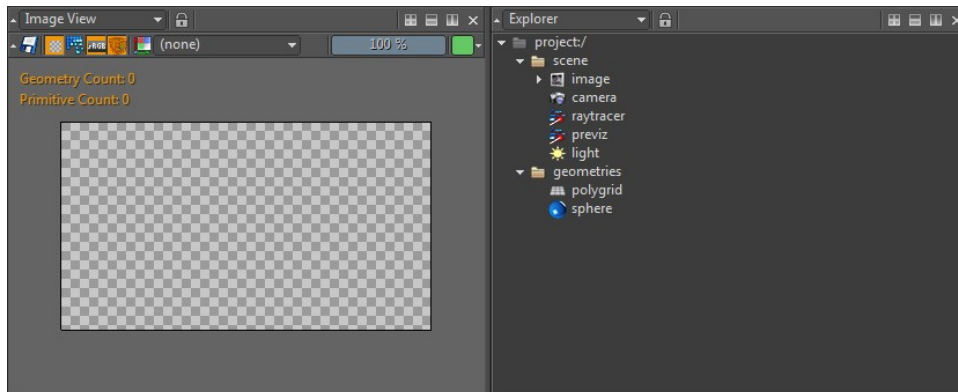
By default, objects in a context can see or reference any objects located in the same or in any sub-context. This rule is recursive, as objects can see or reference items that are under a hierarchy of sub-contexts.

For example, geometries in a context project://scene/geometries will automatically be visible to a layer 3d of the image located in project://scene



The image 3d layer sees geometries within the project://scene/geometries context.

Now if the geometries context is outside the scene context, the 3d layer of the image won't see them anymore.



The image 3d layer doesn't see geometries within the project://geometries context.

Built-in Contexts

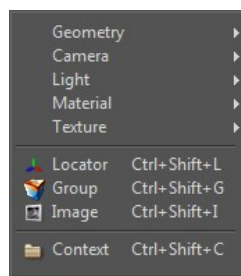
In any Clarisse project, there is one built-in special context that you will always find: default.

Default Context

This context is internally used to automatically create default objects such as the default material. In this context you can't create or remove objects.

Creating Contexts

Even if contexts can be created in many different ways, they can be created directly from the **Create** menu from any main window menu bar.



Create menu

Each time a new context is created using the **Create** menu, it will be created within the current application context. For example, if you create a new context while the current application context is set to project://scene, a new context project://scene/context will be created.

Note

Newly created contexts are always selected.

Deleting Contexts

Deletion is performed on a selection. To delete selected contexts press the **Del** key or go to **Edit > Delete Selection**

Note

Deleting a context deletes recursively the context content (objects and sub-contexts)

Moving Contexts

Moving contexts is performed the same way you move objects. Please refer to Working with Object - Moving Objects

Copying Contexts

Copying contexts is performed the same way you copy objects. However, when you copy contexts, you also copy objects and sub-contexts. For more information on copying, please refer to Working with Objects - Duplicating Objects - Copies

Renaming Contexts

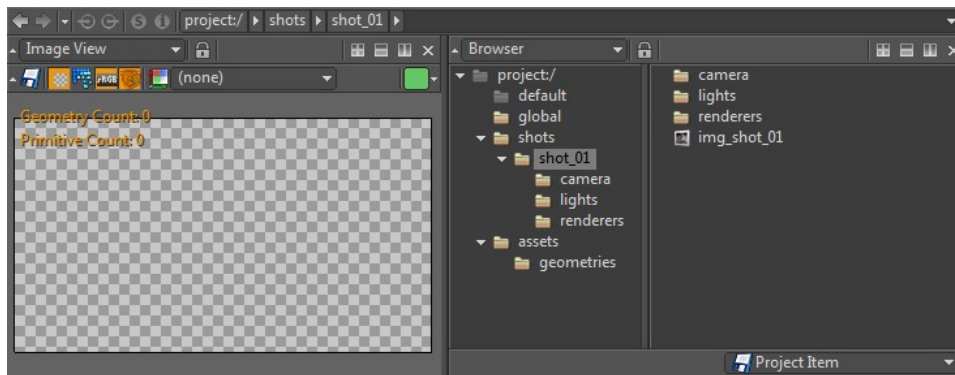
Renaming contexts is performed the same way you rename objects. Please refer to Working with Object - Renaming Objects

Selecting Contexts

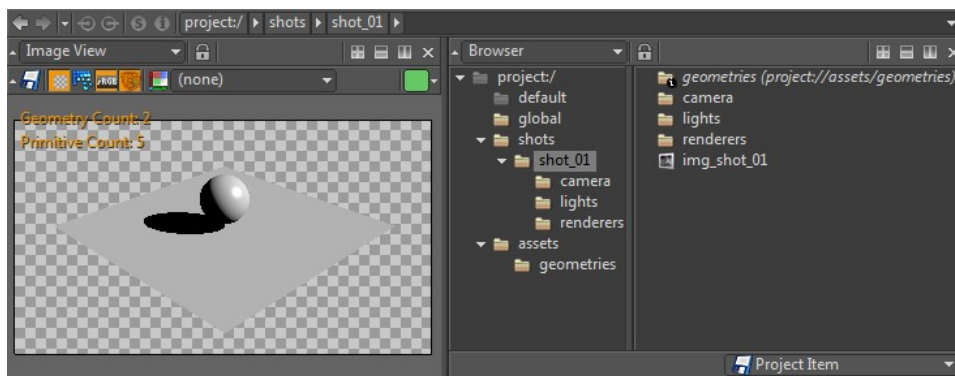
Selecting contexts is performed the same way you select objects. Please refer to Working with Object -Selecting Objects

Instancing Contexts

You can instantiate contexts the very same way you can instantiate objects. However, you can't currently localize an instance of a context. Context instances are used when you want to import the content of a context to a new context hierarchy. This feature is really useful if you freely want to sort your projects. You can logically organize projects by creating contexts to manage your assets while creating others to manage your shots. In this case, context instancing is used to add assets to shots.



project://shots/shot_01 has no geometries



The geometries context including plane and sphere is instantiated in project://shots/shot_01

Note

You can perform the same thing by copying context. However when instancing contexts, updates in instanced contexts are automatically reapplied to contexts that reference them.

Understanding Assets

Clarisse doesn't provide any modeling tools. At the exception of procedural and implicit geometries generated with Clarisse, you'll need to import geometries from other applications. In the same way you'll need to import character animation too. As a matter of fact, a lot of assets you'll end using in your images will come from external applications.

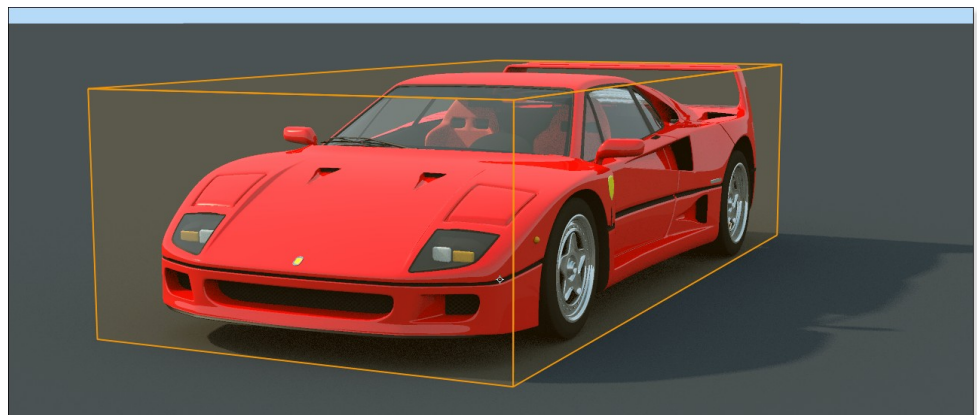
The concept of file or asset is extremely important to Clarisse. In fact when you import an asset, it is dynamically linked to a file as a reference. For example, an imported polygonal mesh is defined by a Polyfile Clarisse object. The polyfile is a polymesh on which geometry is defined by a filename attribute which links to a file. At anytime you can change the path to the file and Clarisse will update everything for you. **You can also specify a path to a file that doesn't exist yet to get linked later on.**

Importing Geometries

To import geometries, go to **File > Import > Geometry...** and select any geometries supported by Clarisse. For more details on supported file formats please refer to the Technical Specifications section. When you import geometries in Clarisse, their topological data isn't saved in the project. In fact, the only thing that gets saved is the object attribute on which one of them links to an external file.

Assigning Materials

In Clarisse, you assign materials to Shading Groups. Shading Groups are a named partition of primitives that are defined in geometries. To learn more about how to assign materials to shading groups, please refer to the Material Linker section.

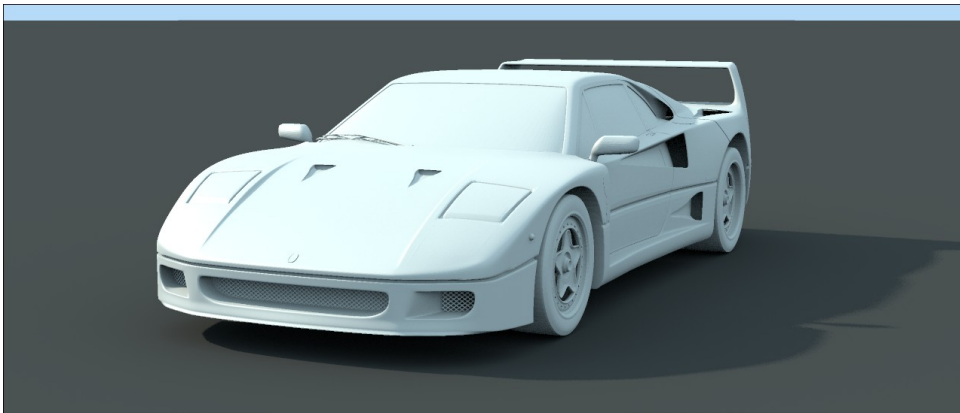


Material assigned to shading groups defined in the geometry

Materials can also directly be assigned to scene objects. To assign a material to a scene object, specify a material to the attribute *Material Override*.



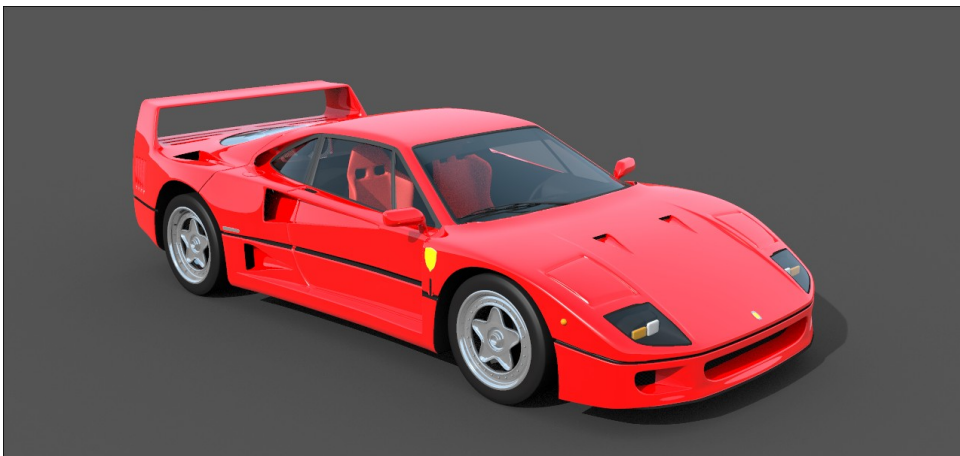
In this case materials assigned to shading groups aren't used. Only the material referenced in *Material Override* is evaluated.



Material Override referencing a default standard material

Updating Geometry

If you select one polyfile and inspect its attribute in the Attribute Editor, you'll find the attribute *Filename* referencing the geometry file path under the *Path* category. In fact, most of the time when you import an object, Clarisse keeps track of the file dependency and at anytime, you're free to modify this file path to another one to link to another geometry.



A Polyfile with its *Filename* attribute set to `$PDIR/geometries/F40.obj`

When you change the link to the geometry by modifying the *Filename* attribute, Clarisse will automatically load the new geometry. If materials were attached to the previous geometry shading groups, Clarisse will automatically do the bindings for you. If the new geometry has matching shading group names, the material that was applied will be automatically be reattached.



Same Polyfile but we modified the Filename attribute to \$PDIR/geometries/golf.lwo

In this example, only few shading groups were matching with the previous geometry (car paint, glasses, tyres). Those shading groups are referencing materials that were defined in the previous geometry. All other shading groups are referencing `project://default/material`

Importing Images

There are several ways to import images (stills or image sequences). Depending on what you want to do, you can import images as a layered image embedding a Layer File, or import them as texture maps.

To import images like a Clarisse image, go to **File > Import > Image...** and browse for any supported images.

Note

You can use Clarisse Images as textures. For example, you can use a render as a texture. To do so, **Create > Texture > Map** and set the image attribute to the image object you want to reference.

For more details on supported file formats please refer to Technical Specifications.

Importing Texture Maps

To import images as texture maps, use either **File > Import > Texture** or **Create > Texture > Map File** and set its *Filename* attribute to the image you want to import.



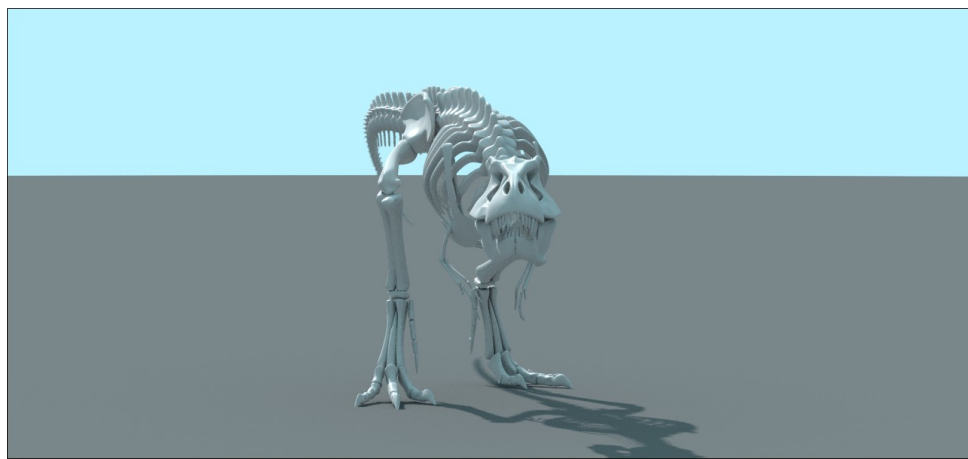
A Map file referencing test.exr image located in the project directory

Again, in the same way Polyfiles are referencing a file, same thing happens for Layer Files and Map Files.

Importing Scenes

You can also directly import animated scenes in Clarisse. Depending the file format, Clarisse can import animated lights, cameras, geometries along with their deformations. For more details on supported file formats please refer to Technical Specifications. To import scenes, go to **File > Import > Scene...** and select any scenes supported by Clarisse.

When you import scenes in Clarisse, items are automatically created inside a new context named after the scene. Depending on the file format, items keep their links with the imported scene file. If the animation items are modified in the imported scene, Clarisse may update its items accordingly. However, if scene modification changes such as adding or removing items happens, updates won't occur in Clarisse.



Imported Alembic scene

Importing Projects

Importing projects can be really useful if you wish to work in libraries. You can, for example, save projects containing a library of materials, geometries, light sets etc... Then import them, for re-use, in an already existing project. To import a Clarisse project in the current one go to **File > Import > Project...** and

select the project(s) you wish to import.

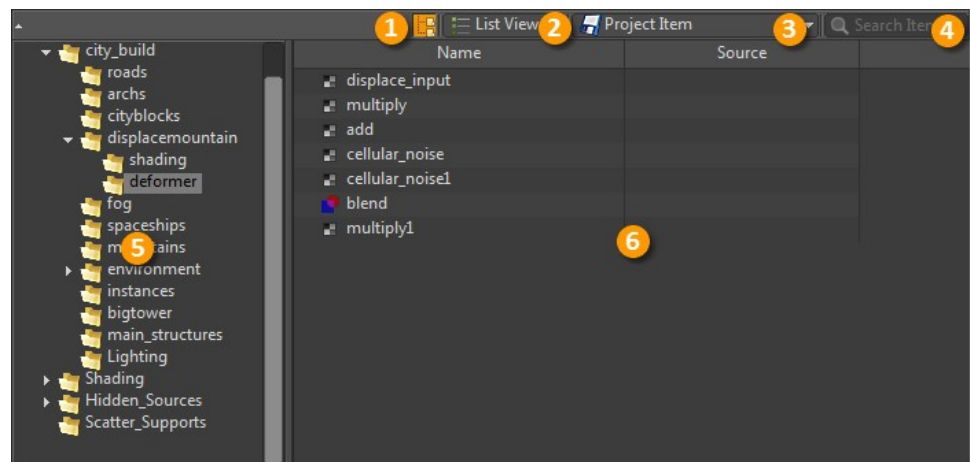
When you import a project, imported items are as if they were actually created in the current project. In other words, it's not a reference and Clarisse doesn't keep the link to the imported project file. If the imported project was to be externally updated, the project importing this specific project wouldn't be updated. External changes are not automatically reflected on import projects.

Using the Browser

The Browser is one of Clarisse central widgets. Quite similar to a file browser, it allows you to browse and manage items in your project.

Overview

Using the Browser, you can select, create and manage project items. It is very close to the file browser you find in your operating system. Not only it can be used to create or sort item but it provides a very useful search feature allowing you to quickly find what you're looking for.



(1) Show/Hide Tree View (2) Item View Display Mode (3) Item View Class Filter
(4) Search Items (5) Tree View (6) Item View

The Browser is divided in 3 sections, its toolbar, the Tree View (5) and the Item View (6).

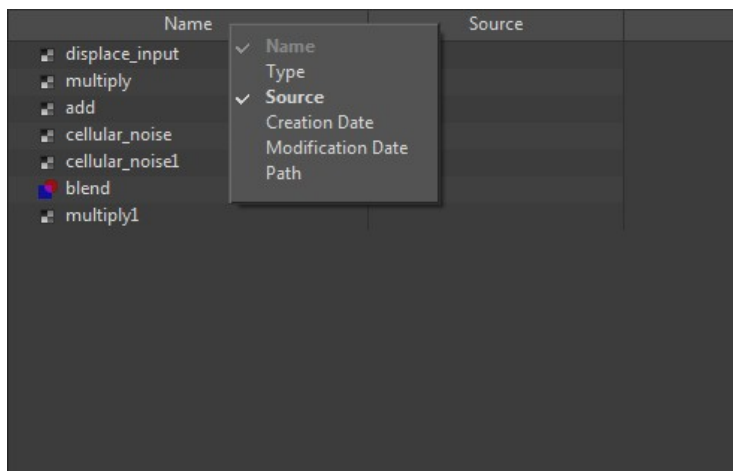
Tree View

The Tree View (5) allows you to explore and select the current application context. It is linked to the Item View (6) so each time you select a context, the

Item View displays its content.

Item View

The Item View (5) allows you to explore and select the content of a context. It provides item sorting abilities in ascending and descending order. By default, the item view displays two columns Name and Source. You can add/remove columns by right clicking on one of the displayed columns.



Right clicking on a columns displays column menu

Column	Description
Name	Display the item name
Type	Display the item Type (Object Class)
Source	Display the source of the instance
Creation Date	Display the creation date of the item
Modification Date	Display the modification date of the item
Path	Display the full item path in the project

By clicking on a column label you can sort items in ascending or descending order. The sorting order is displayed by a little up or down arrow left to the column label. If no arrow is displayed (by default) then the item display order is displayed by creation date.

Name	Type ^	Creation Date
add	TextureAdd	06/27/12 08:20 PM
blend	TextureBlend	06/27/12 08:20 PM
cellular_noise	TextureCellularNoise	06/27/12 08:20 PM
cellular_noise1	TextureCellularNoise	06/27/12 08:20 PM
displace_input	TextureMultiply	06/27/12 08:20 PM
multiply1	TextureMultiply	06/27/12 08:20 PM
multiply	TextureMultiply	06/27/12 08:20 PM

Custom columns sorted by Type

Per-Class Item Filtering

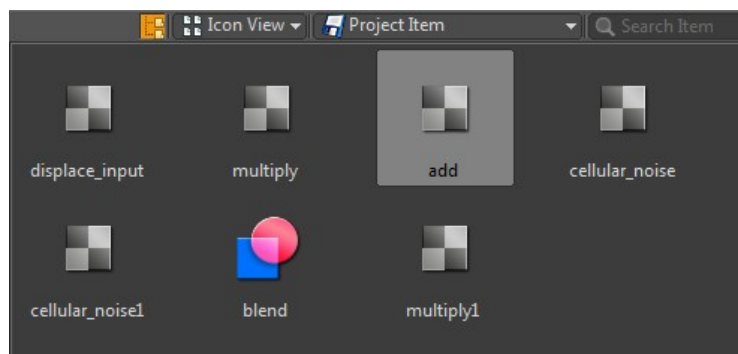
By default, the class filter is set to Project Item so it only displays items that inherits from Project Item. You can filter items displayed by classes in the Item View using (3). For example, you can choose to display only Geometry or Geometry Particle.

Name	Type ^	Creation Date
cellular_noise1	TextureCellularNoise	06/27/12 08:20 PM
cellular_noise	TextureCellularNoise	06/27/12 08:20 PM

Displaying only Texture Spatial

Icon View



By default the Item View displays items as a list. Using the Item View Display Mode button (2), you can switch the display mode to display items as icons.



Item View in Icon Mode

Modifying Icon Size

















You can modify the size of the icons displayed in the Item View by using **CTRL + Mouse Wheel**. This works on both list and icon view.

Name	Type	Creation Date
 add	TextureAdd	06/27/12 08:20 PM
 blend	TextureBlend	06/27/12 08:20 PM

Icon size set to maximum

Using the Search Engine

Clarisse provides a dedicated search engine designed to find items very quickly. The search engine (4) is available in the browser but it can be summoned at anytime by pressing **F3**. To find an item just type its name, class or both and press **Enter**. Results will then be displayed in the Item View.

Name	Source	Type	Path
 city_build/roads/road1	Hidden_Sources/road	GeometryPolyfile	project://city_build/...
 city_build/roads/road2	Hidden_Sources/road	GeometryPolyfile	project://city_build/...
 city_build/roads/road4	Hidden_Sources/road	GeometryPolyfile	project://city_build/...
 city_build/roads/road3	Hidden_Sources/road	GeometryPolyfile	project://city_build/...
 city_build/roads/road5	Hidden_Sources/road	GeometryPolyfile	project://city_build/...
 city_build/roads/road6	Hidden_Sources/road	GeometryPolyfile	project://city_build/...
 city_build/roads/road7	Hidden_Sources/road	GeometryPolyfile	project://city_build/...
 city_build/roads/road8	Hidden_Sources/road	GeometryPolyfile	project://city_build/...
 city_build/archs/archroad1	...ld/main_structures/archroad	GeometryPolyfile	project://city_build/...
 city_build/archs/archroad2	city_build/archs/archroad1	GeometryPolyfile	project://city_build/...
 city_build/archs/archroad3	city_build/archs/archroad1	GeometryPolyfile	project://city_build/...
 city_build/archs/archroad4	city_build/archs/archroad1	GeometryPolyfile	project://city_build/...
 city_build/archs/archroad5	city_build/archs/archroad1	GeometryPolyfile	project://city_build/...
 city_build/archs/archroad6	city_build/archs/archroad1	GeometryPolyfile	project://city_build/...
 city_build/archs/archroad7	city_build/archs/archroad1	GeometryPolyfile	project://city_build/...
 city_build/archs/archroad8	city_build/archs/archroad1	GeometryPolyfile	project://city_build/...

Displaying all results corresponding to geometry.

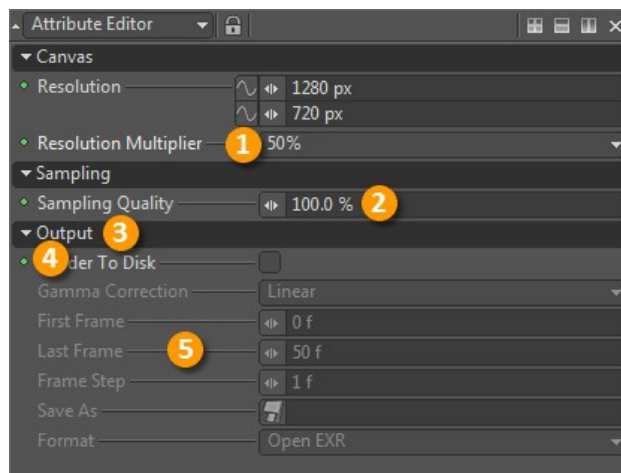
To exit the search mode, just click on the cross icon left to the search input.

Using the Attribute Editor

The Attribute Editor is the central widget allowing you to edit project item attributes. As explained in the Working with Objects section, in Clarisse, all project items are objects and all object-properties are attributes.

Overview

Using the Attribute Editor, you can edit and modify the attributes (properties) of a selected item. The Attribute editor also handles selection of multiple items that can be of different classes. It has been designed to give you the ability to massively edit attributes of an entire set of items.



Attribute editor displaying the attributes of an Image

(1) Attribute Label (2) Attribute field value (3) Attribute Group (4) Evaluation info marker (5) Disabled Attributes

When the evaluation info marker turns red, the attribute is flagged as protected. To learn more about attribute protection, please refer to the Attribute Protection section.

Working with Attributes

Attributes are defined by a label (name) (1) and a field value (2). The field value (2) display depends on the attribute type. In the previous example, the attribute Sampling Quality is of type percentage. Attributes are also logically sorted into horizontal tabs called attribute groups (3). These groups can be collapsed or expanded to set attribute visibility. To expand or collapse an attribute group simply click on the horizontal tab.

Attribute Types

There are different kinds of attributes. Each one is represented by a specific field, for example, an attribute of type Color will be represented by a color field.

Numeric

An attribute of type numeric can define a distance, an angle, a percentage, a resolution in pixels... anything that requires numbers.



An array of 3 numeric attribute values in percentage

To edit the value:

- Simply drag the mini-slider left of the value field.
- Directly type your value in the text field.

Tip

By holding CTRL while dragging the mini-slider, you edit all attribute values at the same time.

Numeric fields support for basic math operations following this syntax. += , -=, *=, /=

Operation	Syntax	Example
+	+=	+= 50 adds 50 to the attribute value
-	-=	-= 50 subtracts 50 to the attribute value
*	*=	*= 50 multiplies current attribute value by 50
/	/=	/= 50 divides current attribute value by 50

Boolean

An attribute of type boolean defines if a value is true or false.



A boolean attribute

To edit the value, simply click on the check-box.

Filename

An attribute of type filename defines a file path.



A filename attribute referencing \$PDIR/objects/chair.lwo

This type of attribute supports the use of environment variables. To manage environment variables please refer to the Environment Variable section.

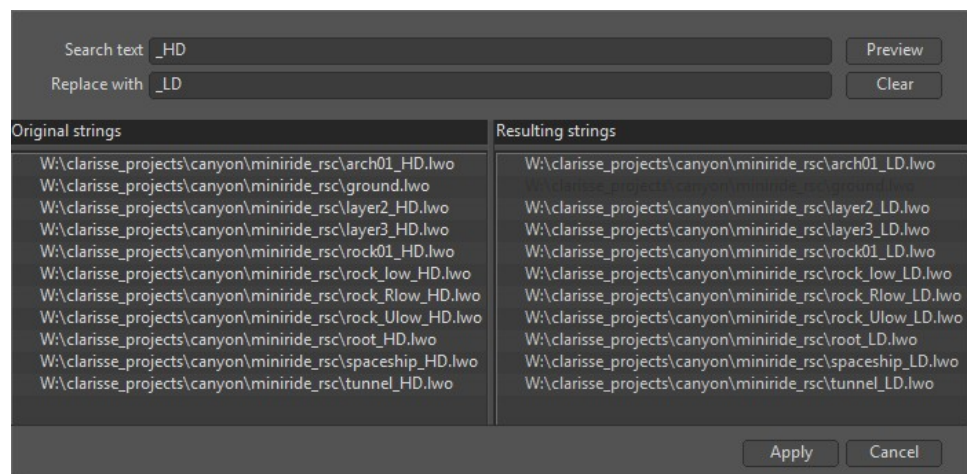
Note

Clarisse automatically tracks for updates on files referenced by filename attributes. If an external modification occurs on any of the referenced files, the user will get notified. To get more information on file updates, please refer to Using the Resource View section.

To edit the filename:

- Click on the folder icon to display the File Browser
- Directly type your filename in the text field
- Select the text field and press CTRL + R to open the batch replace window

The batch replace window is really useful when working with a multi-selection. Similar to most text editors, the batch replace window allows you to replace all filenames according to a replacement pattern.



A replace window applied to a multi-selection of polyfile

Color

An attribute of type color defines either a Luminance (L), Luminance + Alpha (LA), Red + Green + Blue (RGB), Red + Green + Blue + Alpha (RGBA).



A RGB color attribute

To edit the color:

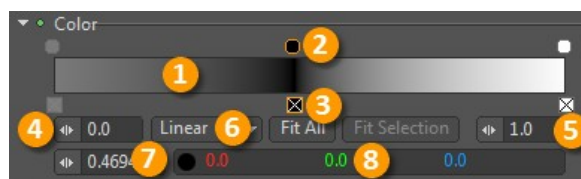
- Click on the color to display the Color Picker. For more information, please refer to the Using the Color Picker section.
- Click on the color channel and drag left or right to decrease or increase the value.
- Middle click on any channels and drag to decrease or increase the value of all channels at the same time.

Tip

Double clicking on any of the color value will allow you to type in your value.

Gradient

An attribute of type gradient defines a gradient that can be either a Luminance (L), or a Luminance + Alpha (LA), Red + Green + Blue (RGB), Red + Green + Blue + Alpha (RGBA).



A RGB Gradient

(1) Gradient (2) Key Color Preview (3) Key Delete (4) Start Range (5) End Range
(6) Key Interpolation (7) Key Value (8) Key Color

Insert Key

To insert a new key click the gradient (1). When inserting a new key, the color is taken from the gradient. By default, newly inserted keys interpolation mode is Linear. You can change Key interpolation by using (6).

Set Key Color

To set key color double click (2) to set the color. You can also change the color by modifying the color attribute.

Select Key

To select a key click on (2). You can select multiple keys by holding Ctrl.

Delete Key

To delete one or multiple keys click on (3).

Start Range

You can set the start gradient range by modifying (4).

Note

This modifies the visual range of the gradient. It has no effect on the evaluation.

End Range

You can set the end gradient range by modifying (5).

Note

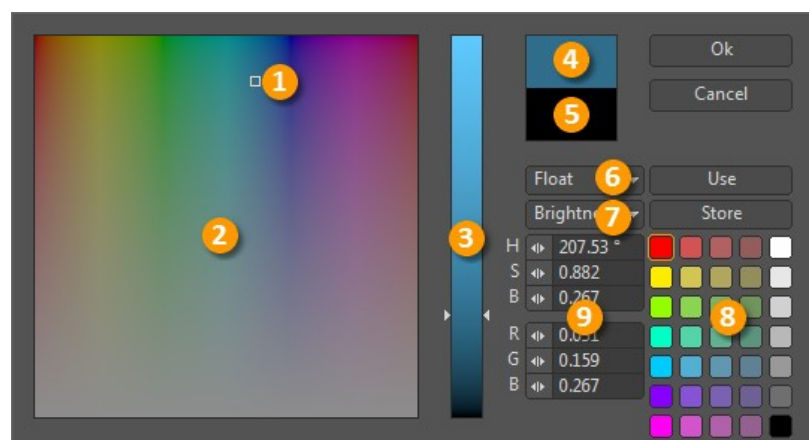
This modifies the visual range of the gradient. It has no effect on the evaluation.

Using the Color Picker

The Color Picker is probably one of the widgets you'll end up using most of the time. It has been designed to be efficient and works along the idiom: What You See Is What You Get. The color picker truthfully displays colors and gradients according to the color display mode (*Linear* or *sRGB*) found in the Preference Panel. The color picker also changes attribute values in realtime which is ideal to interactively preview color tweaks in a render.

Overview

You can choose colors using two color models: HSB and RGB. The color picker also allows to set HDR (high dynamic range) colors. Depending of the color output, the color picker can be used to set the color alpha channel.



Color picker displaying a color in sRGB

(1) Picked Color (2) Color Field (3) Color Slider (4) Selected Color (5) Original Color (6) Color Numeric Mode (7) Color Gradient Mode (8) Color Swatches (9) Color Values

Color Gradients

The color of the color picker gradients is the color field (2) and the color slider (3). When the application color display mode changes to *Linear* or *sRGB*, the gradients update accordingly. Color field and color slider display modes can also be changed by using the gradient mode pull down (7).

Color Field

Pick your color by clicking and dragging the cursor (1) on the color field (2).

Color Slider

Pick your color by clicking and dragging vertically on the color slider (3). For a finer selection, you can also use the mouse wheel to move the cursor vertically.

Reverting to Original Color

When the color picker is opened, it stores in (5) the original color. This color isn't updated until the new color has been explicitly set by clicking on the *Ok* button. You can revert to the original color just by clicking on (5).

Color Numeric Mode

Numeric mode of color input text values can be changed to Float (default) or Integer using (6). In integer mode, saturation, brightness, red, green, blue and alpha values are automatically clamped and range from 0 to 255. In float mode, values go beyond 1.0 (HDR) and can be negative.

Using Color Swatches

The color picker provides user swatches to store and reuse favorite colors and also provides easy color sharing between projects. The color swatches is persistent and is saved in the application configuration file.

Click, to select a color swatch. Double click or press the *Use* button to set the picker color to the swatch color. To record in the selected swatch, the current picker color, click on the *Store* button.

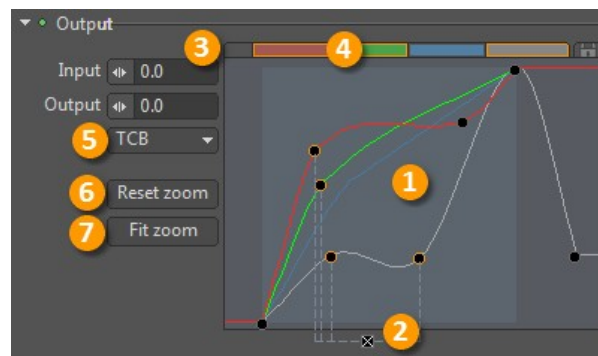
Validating Color

The color picker changes in realtime the color value of the attribute it was summoned from. However, even if the attribute color interactively changes to

the one set in the picker, the new color isn't explicitly set until you pressed the *Ok* button. If you press the *Cancel* button or the **Escape** key, the color of the attribute is automatically reverted to its original value.

Curve

An attribute of type curve defines curves that can be either a Luminance (L), or Luminance + Alpha (LA), Red + Green + Blue (RGB), Red + Green + Blue + Alpha (RGBA).



RGBA Curve attribute

(1) Curve Editor (2) Key deletion (3) Toggle lock on all curves (4) Lock Curve (5) Key interpolation (6) Reset Zoom to 100% (7) Fit Curves

Preset

An attribute of type preset is a predefined list of named values.



A preset attribute

To select one of the predefined values, simply click on the preset button and choose the value in the popup menu.

Reference

An attribute of type reference, references an item of the project. This kind of attribute is used for many things such as parenting, assigning materials, groups...



An attribute of type reference to set the parent

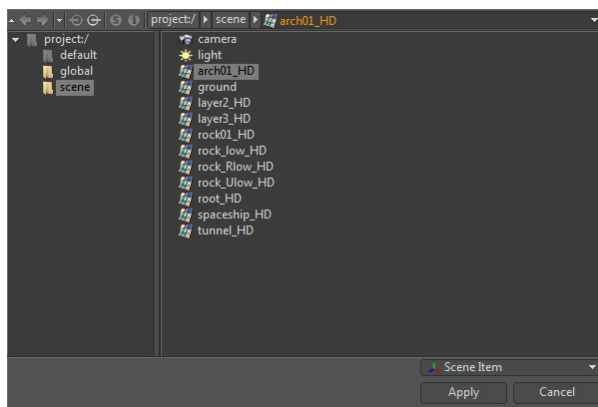
(1) Item Jump in (2) Selected Item

To set the reference item:

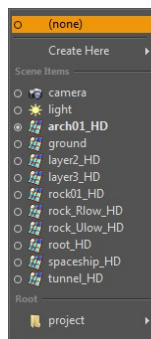
- Drag and drop an item to (2)
- Click on (2) and choose the item.

Note

To select references you can either use the browser dialog (by default) or a pop up menu. To set the object selection mode, go to the Preferences panel. In User Interface tab, select either Use Popup Menu or Use Browser in Object Selection Mode.



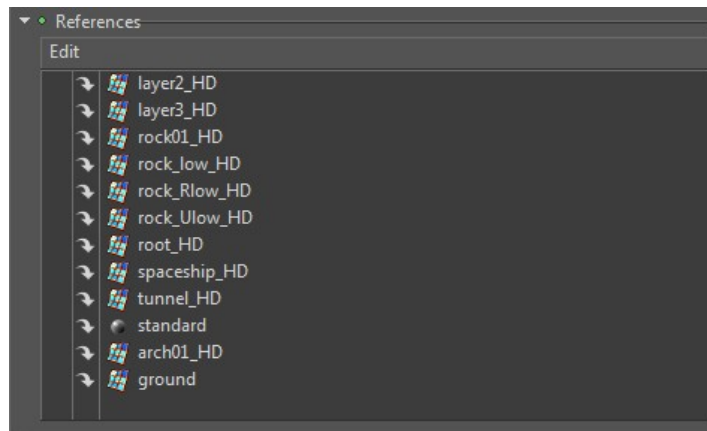
Item selection using the Browser



Item selection using the Popup menu

Reference List

An attribute of type reference list, references a list of items.



An attribute of type reference list

Adding Items

To add items to the reference list just drag and drop them into the list.

Selecting Items

To select a list item just click on it. A Selected item is highlighted.



A selected list item

You can also select multiple items:

- To select a range of list items, click on the item at one end of the range, then hold down the Shift key while clicking on the item at the other end of the range.
- To append a list item in the selection, click on the list item while pressing CTRL. You can also use CTRL to toggle in and out items from the selection.

Note

When you select items in a reference list, items are not selected in the application. To select items in the application selection click on (1).

Removing Items

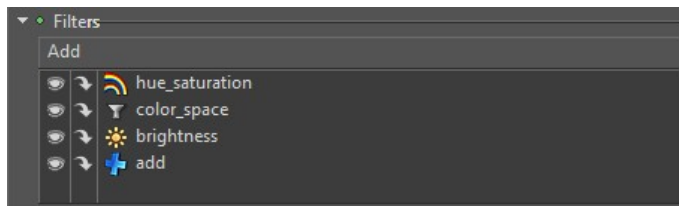
Press Delete to remove selected items from the list.

Note

When you remove items from a reference list, they are not removed from the project.

Object List

An attribute of type object list, is a list of embedded objects. Embedded objects are bundled into the owner object. When you duplicate an object that embeds objects, embedded objects get also duplicated.

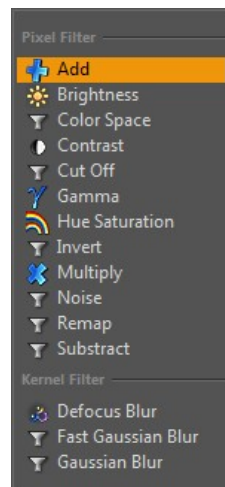


A List of image filters

Object lists are always processed in bottom-up order. In this example add filter is applied first and hue_saturation last.

Adding Items

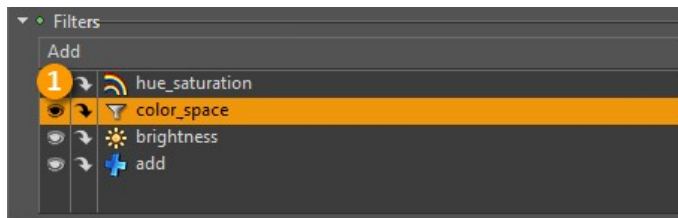
To add items to the list just click on the Add to bring a create menu and click on the item you wish to create.



Add menu displaying image filters

Selecting Items

To select a list item just click on it. A Selected item is highlighted.



A selected list item

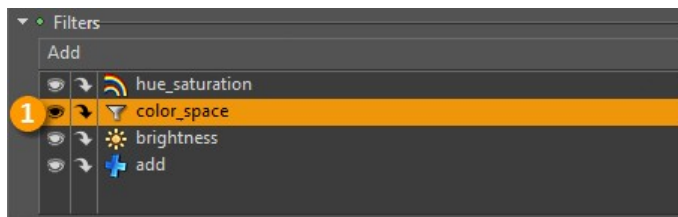
You can also select multiple items:

- To select a range of list items, click on the item at one end of the range, then hold down the Shift key while clicking on the item at the other end of the range.
- To append a list item in the selection, click on the list item while pressing CTRL. You can also use CTRL to toggle in and out items from the selection.

Note

When you select items in a reference list, items are not selected in the application. To select items in the application selection click on (1).

Deactivating Items



To activate or deactivate items click on the eye shaped icon (1).

Note

If you click on the eye shaped icon on a selection, the whole selection is either activated or deactivated.

Removing Items

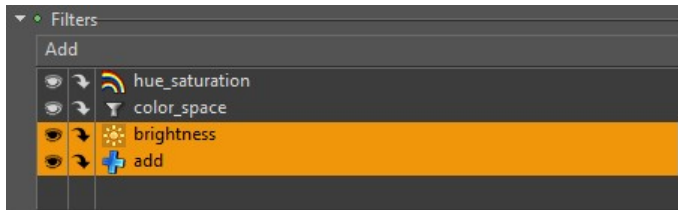
Press Delete to remove selected items from the list.

Note

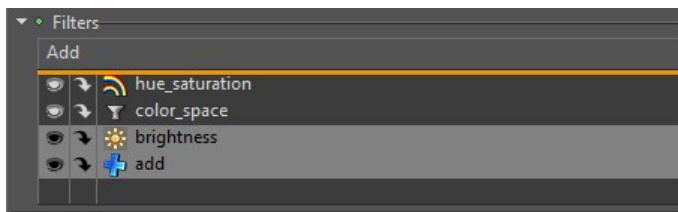
When you remove items from an object list, embedded objects are removed from the project.

Reordering Items

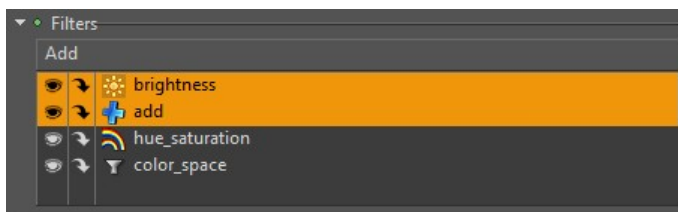
To reorder items select the items you wish to move.



Then Drag to insert the items where you wish to. A marker appears to let you know where the items will be inserted.



Finally drop the items.



Attribute Modifiers

Attributes can have modifiers driving their value. Currently, there are 2 kinds of attribute modifiers available: texture and animation modifiers.



Attribute modifiers

(1) Animation modifier (2) Texture modifier (3) Texture Jump in (4) Attribute field value (here a color field)

At the same time, an attribute can be bound to a value, to an animation as well as a texture. The attribute is evaluated in this order:

- If the attribute is textured, whatever its value or animation, the attribute will get the value of the resulting texture evaluation.
- If the attribute has no texture but is animated, the attribute value will get the value of the animation curve at the current frame.

Texture

Sometimes, attributes can be texturable. When an attribute is texturable, you should see two little icons (1) and (2) appearing on the left side of the attribute value field.



To bind a texture to an attribute:

- Drag and drop a texture to (1)
- Click (1) and choose the texture you wish to create

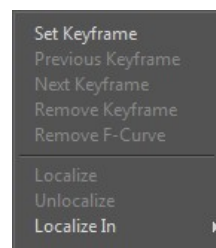
To select a texture that is bound to an attribute, simply click on (2).

Animation

Sometimes, attributes can be animated. When an attribute is animatable, you should see this little icon (2) appearing on the left side of the attribute value field.



To set a keyframe, set the attribute value and Click on (2). You can also set the key frame from the attribute menu by right clicking on the Attribute Label (1).

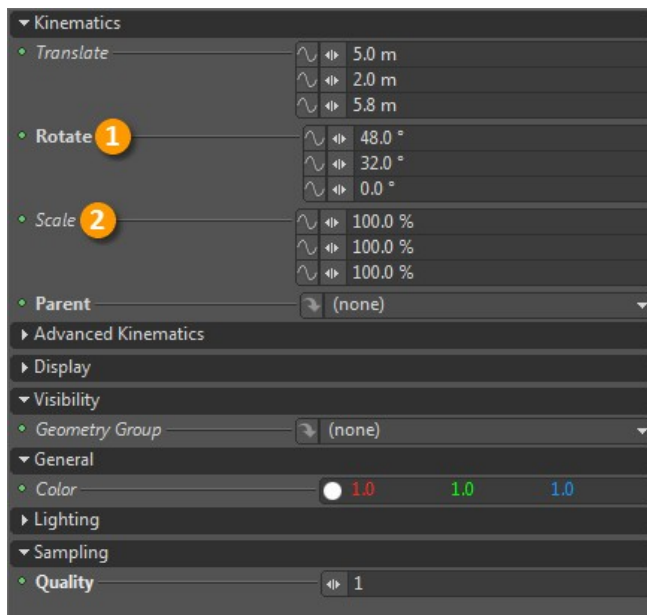


Note

Animated attributes function curves can be displayed with the Graph Editor. Please refer to the Using the Graph Editor section to get more information on how to create and edit function curves.

Working with Instances

Sometimes project items can be instances of other items. Instances are dynamic copies of project items in which attributes are linked and driven by their sources. For more information on instances, please refer to Instancing Objects section.



An instance of a light displayed in the Attribute Editor

The attribute editor has been designed with dedicated visual hints allowing to detect quickly instances when displayed in the Attribute Editor. The rule is really simple:

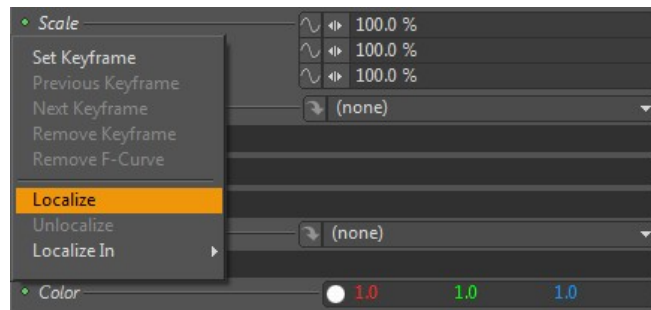
- an attribute of an instance is localized (freed from its source) when the attribute label is displayed in **bold** (1)
- an attribute of an instance is tracking its source when the attribute label is displayed in *italic* (2)

Note for attributes tracking its source

When you modify the value of an attribute tracking its source (displayed in italic), the source attribute is the one getting modified.

Localizing

To localize an attribute, right click the attribute of an instance and select **Localize** in the popup menu. When an attribute is localized, the link with the source attribute is broken. Any modification that occurs to a localized attribute is local to the instance.



Localizing a Scale attribute

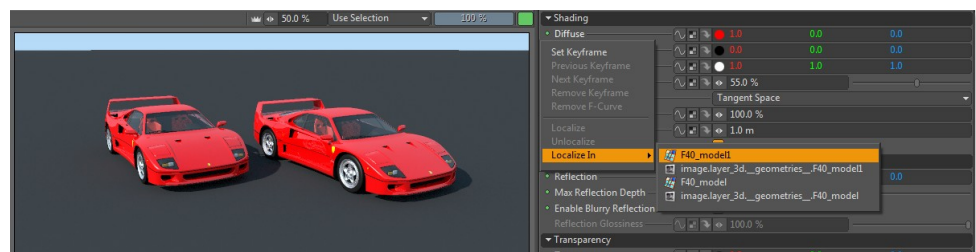
Localized attributes are displayed in **bold**.

Note for animated attributes

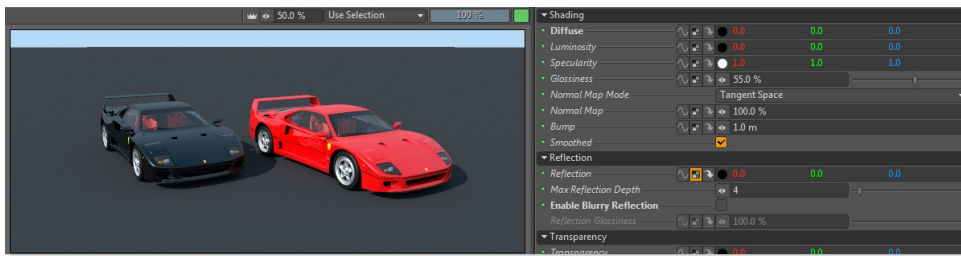
When an animated attribute is localized, the source curve is copied to the localized attribute.

Localize In

Clarisse allows you to modify an attribute value according to its location. This powerful feature is called **Localize In**. It allows you to contextually localize the attribute of any objects to a project path. To contextually localize an attribute, right click the attribute of any object and select **Localize In** in the popup menu. In the sub menu then choose in which object you want to localize the attribute.



Localizing the Diffuse attribute of a shared paint material for the F40_model1

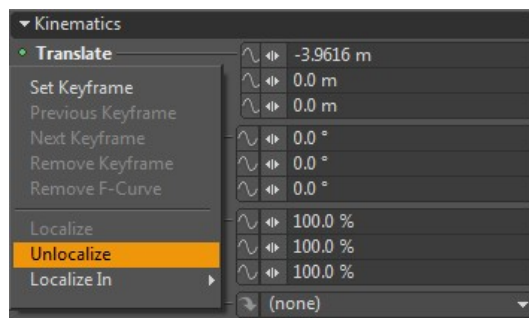


Localized result modifying the diffuse value

When you use the localize in feature, Clarisse creates automatically an embedded instance of the source object. In the previous example, an embedded instance of the paint material has been automatically created into the second car.

Unlocalizing

To unlocalize a localized attribute, right click the attribute of an instance and select Unlocalize in the popup menu. When an attribute is unlocalized, the link with the source attribute is active. Any modification that occurs to the attribute is applied to the source object.



Unlocalizing a Translate attribute

Unlocalized attributes are displayed in *italic*.

Note for animated attributes

When an animated attribute is unlocalized, the curve is driven by the source attribute.

Working with Images

Images in Clarisse are defined by a canvas, a resolution multiplier and a stack of layers. There are many kinds of layers and they all can be mixed together. Like in a compositing software, each layer contains pixels processed by filters. They are then composited within a single image, to produce the final image.

Images can be used as render output and any Clarisse image can be used as texture maps. It's very easy in Clarisse to use renders as textures without the need of saving images to disk.

A Word on Evaluation

Clarisse features an On-demand evaluation system. For example, when an image is displayed in the Image View, the image is queried for evaluation. If it requires evaluation then it gets evaluated. During evaluation, Clarisse will dynamically generate, load and render on demand everything required for the final image. In other words, geometries, texture maps are actually loaded only if needed during a render on a displayed image. If you let the evaluation engine finish its job, you'll get your final image.

Depending on the image complexity, the evaluation can take a while. Hopefully we support full interactive progressive rendering. You are free to interact anytime during evaluation.

Active Evaluation Engine

In Clarisse, there is no need to explicitly query for a rendering. Clarisse evaluation engine is active and always tracks for user modification. If you modify something that affects somehow an image, it's marked as dirty. When displayed, an image gets automatically re-evaluated until it gets final and marked as clean. If you close an Image View displaying a clean image, it won't need to be re-evaluated if displayed back.

On the other hand, if a user modification doesn't affect the image, it keeps clean. Even better, if your modification doesn't affect a running evaluation, it keeps evaluating. For example, if you scrub the timeline and no object in your image is animated:

- if the image is clean, it will be kept that way.
- if the image is being evaluated, it will continue as nothing happened.

Stopping an Evaluation

In the same way, you don't explicitly start evaluation or rendering, you can't explicitly stop a running evaluation. An evaluation will keep going until it's complete. However, evaluation slows down the application only on rare occasions such as loading geometries... If you really want to stop a running evaluation, you can always:

- close the Image View/3d view that is evaluating
- change the selection to something flagged as clean
- delete the item currently evaluated

Using the Image View

The Image View allows you to view, edit and render any Clarisse images. It also performs color correction in real-time, provides interactive progressive rendering and features render region. Even if the Image View displays final images that can be saved to disk, it's not meant to render full image sequences. If you wish to render image sequences, please refer to Using the Render Manager section. This widget relies on Tools for item manipulation. Please refer to Using Tools section.

This widget requires OpenGL 2.0.

Overview

The image view is split in two parts, the image area and the image toolbar.



Image View displaying an image

(1) Image Area (2) Image Toolbar (3) Image Geometry Statistics

The Image Area

The image area (1) is where the image canvas is actually displayed. In this area you can interact with your image while it gets progressively refined into the final one. You can also select directly underlying geometries, materials etc... using tools. Realtime image statistics (3) are also available.

2D Navigation Basics

Panning

To pan press Space and click in the image area using either left or middle mouse button. Drag in any direction to pan the image canvas.

Note

If you wish to center the image in the image area, press Space + C

Zooming

To zoom press Space and right click in the image area. Drag right to zoom in and left to zoom out. The zoom factor is rational resulting in fractional pixel sizes. If you wish to snap to the nearest predefined non fractional pixel size, use Space + Z.

Note

When the Image View evaluates an image, it takes into account the current view zoom factor. If the image is zoomed out, the evaluation ends before it reaches the final quality.

Fitting

To fit the image to the image area. Press Space + F.

3D Navigation Basics

As Images can have both 2D and 3D layers, you can directly navigate in your 3D layers using the Image View.

Orbiting Camera

If a 3D Layer is selected you can directly orbit its camera in the Image View. To orbit the camera, press Alt and Click in the Image View. Drag in any direction to orbit around its center of interest.

Moving Camera

If a 3D Layer is selected you can directly move the camera in the Image View:

- To move the camera, press Alt + Middle click in the Image View. Drag in any direction to move both the camera and its center of interest in the view plane.
- To dolly the camera (move forward or backward), press Alt + Right click in the Image View. Drag right or down to move forward, and left or up to move backward.

Fitting

You can fit the view to the content of a 3D Layer by pressing F. If no item is selected, it fits the view to the bounding box defined by the geometries referenced in the layer. On the other hand, if one or multiple items are selected, it will fit the view to bounding box defined by the selection.

Assigning Material

You can quickly assign materials by dragging and dropping a material to geometry shading groups directly in the image.

Adding Scene Objects

You can quickly add scene objects in your image by dragging and dropping scene objects directly in the image.

The Image Toolbar

The Image Toolbar is used to set many options that drive the Image View behavior. You can, for example, control the interactive progressive rendering, the color correction etc...



Image Toolbar

(1) Save current Image (2) Clear Image History (3) Show Alpha Checkerboard (4) Show Rendering In Progress (5) Color Correction (6) Display Overlay (7) Limited Region (8) Minimum Render Starting Resolution (9) Image History (10) Layer/Channel Display (11) Master Image Switch (12) Sampling Quality (13) Selection Button (14) Evaluation Progress Bar

Selection

By default, the Image View is driven by the current selection. If you select an image in the project, it will be displayed in the Image View.

You can modify this behavior, by locking explicitly the image view to a specific image. To lock the display to an image:

- Drag and drop an image in the image area
- Lock the widget selection (see Selection Toolbar section for more information)
- Click on selection button (13) and choose the image you would like to display.

Progress Bar

When the image progress bar (14) reaches 100%, it means displayed image is marked as clean and fully computed. You will notice there's a colored indicator right to the progress bar. If it's red, it means the displayed image is currently being evaluated. If green, however, the current image isn't.

Note

Don't be surprised if you see progress bar, with a green indicator, that are half way through. Clarisse supports on demand evaluation and depending of what's visible in the image area, the displayed image can simply get partially evaluated.

Master Image

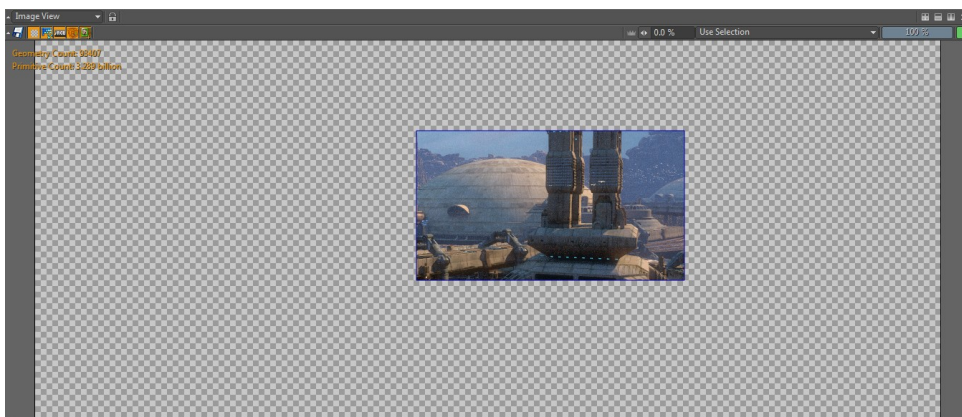
In Clarisse you can work and render multiple images at the same time. When multiple images are evaluated, they all get refined at the same time. Sometimes it's useful to raise the priority of a specific image to get it fully refined before the others. In this case, you can set it as master image by clicking on (11).

Note

Only one image can be set as master image at the same time. However, you can switch to a different master image during evaluations.

Limited Region

Evaluating a whole image can be a slow process. Sometimes it is useful to work on specific areas of an image instead. You can set in your image and layers limited regions to evaluate and render only a specified area of the canvas. To draw a limited region, toggle Limited Region mode by clicking on (7) in the Image Toolbar. Press X and drag the mouse to draw the region.



Rendering in Progress

By default, you see a preview of everything Clarisse is rendering to fully evaluate an image: 3d layers, filters... Sometimes, it's useful to display only a progressive preview snapshot of the fully comped image. To disable Show Rendering in Progress, click on (4).

Overlay

You can toggle on and off statistics and selection bounding box overlay by clicking on the overlay icon (6).

Color Correction

You can toggle on and off sRGB color correction by clicking on the sRGB icon (5). User defined LUT will be supported soon.

Alpha Checkerboard

You can toggle on and off alpha checkerboard display by clicking on the checkerboard icon (3).

Saving the Image

At anytime you can save the currently displayed image by clicking on (1)

Sampling Quality

You can control the sampling quality (12) of all layers found in the displayed image. By default, the quality is set to 100% which means original sampling values are used unmodified. **This value doesn't affect the antialiasing sampling.**



Sampling quality at 100%

Sometimes, it's useful to decrease this value to speed up rendering. In that case original values are used as if they were lowered.



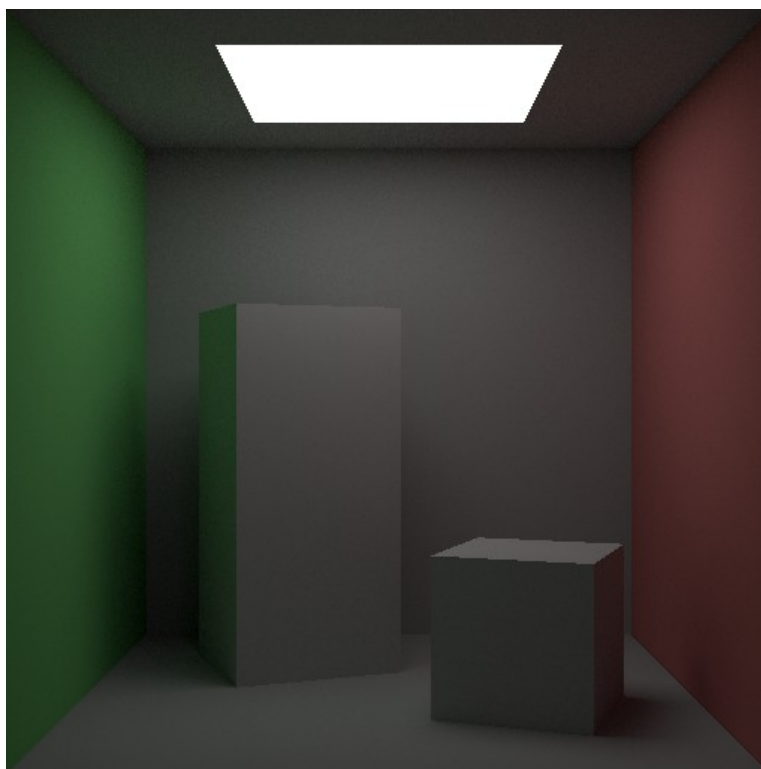
Sampling quality at 50%

This value controls the *Sampling Quality* attribute that is found in the image. Modifying this field or *Sampling Quality* attribute using the attribute editor is completely equivalent.

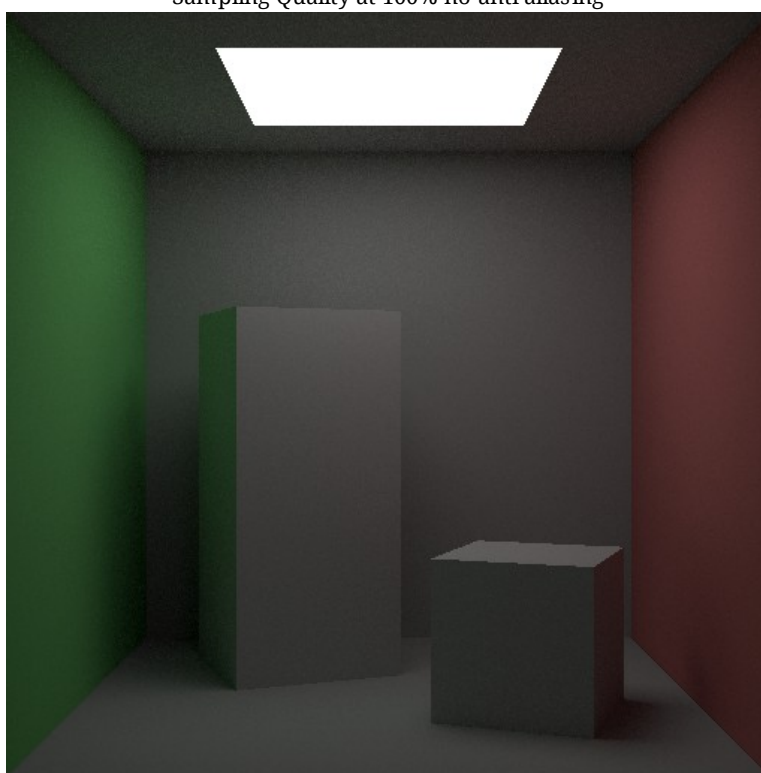
Note

This value can also be increased numerically to go beyond 100%. In that case original sampling values are used as if they were increased.

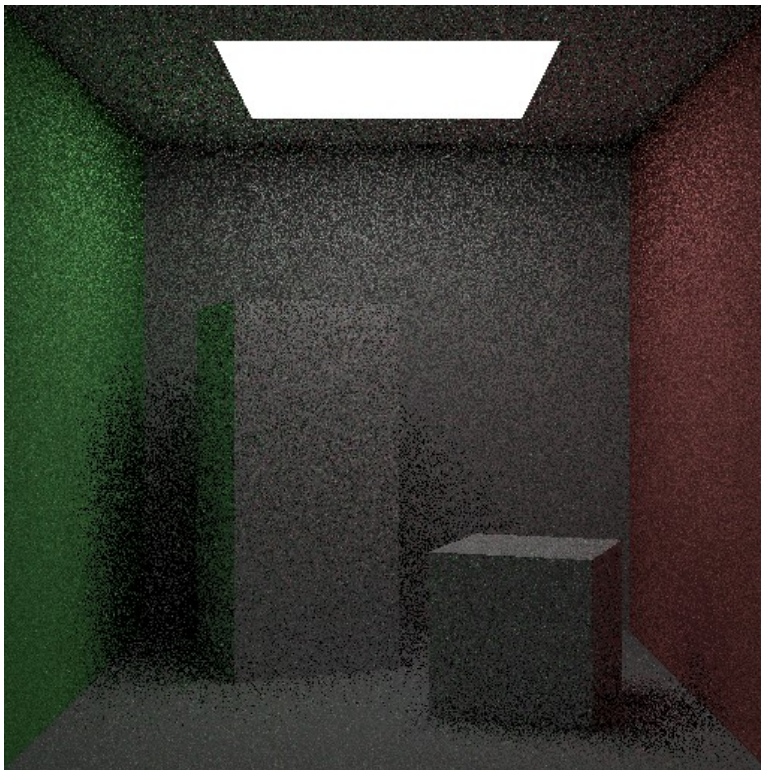
Please note minimum *Sampling Quality* depends on the renderer anti aliasing level. The more anti aliasing you have in your renderer, the more likely low values of *Sampling Quality* won't impact your image: a minimum of 1 ray per sub-sample is required. If your sampling is set to 8 and your anti aliasing set to 10 then, changing *Sampling Quality* won't reduce overall sampling. In this case sampling is maxed out by the anti aliasing value of 10.



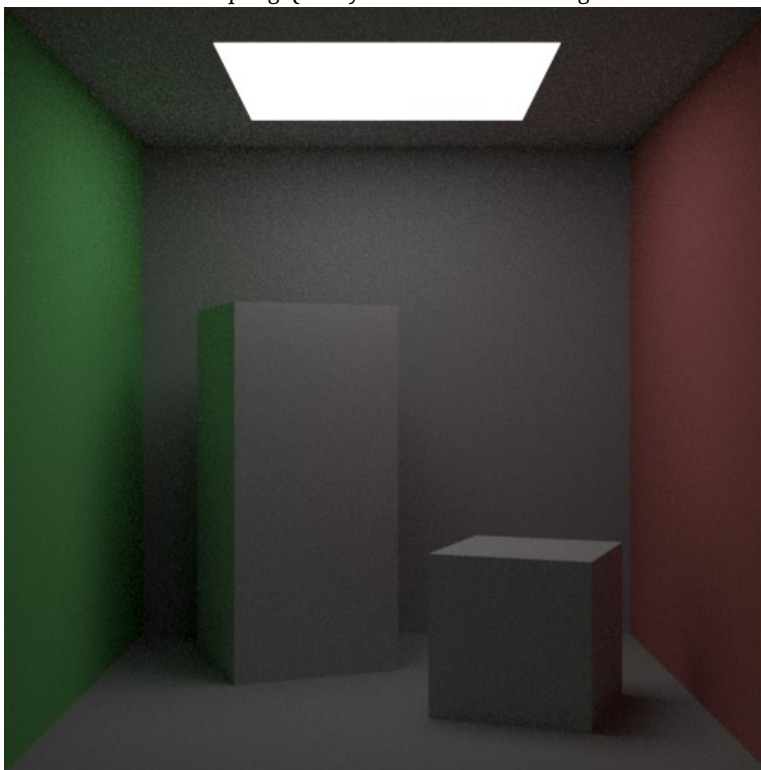
Sampling Quality at 100% no anti aliasing



Sampling Quality at 50% no anti aliasing



Sampling Quality at 0% no anti aliasing



Sampling Quality at 0% and Anti Aliasing at 4.

As you can see, there is a big sampling difference between the two images that have their *Sampling Quality* set to 0%. In the anti aliased version, a minimum of 16 rays are used because the *Anti Aliasing* of renderer is set to 4 (16 rays).

Image History

By default, the Image View stores 10 different image iterations. You can control the image history size in the preferences panel. Go to **Edit > Preferences > Image View** to set *Image History Size*.

By clicking on (2) you can clear the image history. To navigate in the image history click on (9). *Latest* is the latest render. You can use **Page Up** or **Page Down** to set the current image from the history. Use **Shift + Page Up** to select the latest and **Shift + Page Down** for the oldest.

Note

Image history is bound to an Image View instance: each Image View has it's own history. If you close an Image View, you'll lose its image history.

Layers and Channels Display

The Image View can display channel layers and individual channels (AOVs) using (10). By default, the Image View is set to *rgba* and *All*. This means it will display all Red, Green, Blue channels and use alpha to perform alpha blending over the checkerboard. You can also display any other channel layers or display individual channels (which will be displayed in gray shade).



Image View display Red channel



Image View displaying an AOV

Working with Layers

Layers are objects embedded in images. If you duplicate an image having layers you'll copy all embedded layers too. A layer is always an image and can be seen as a sub-image. Layers all shares the following features:

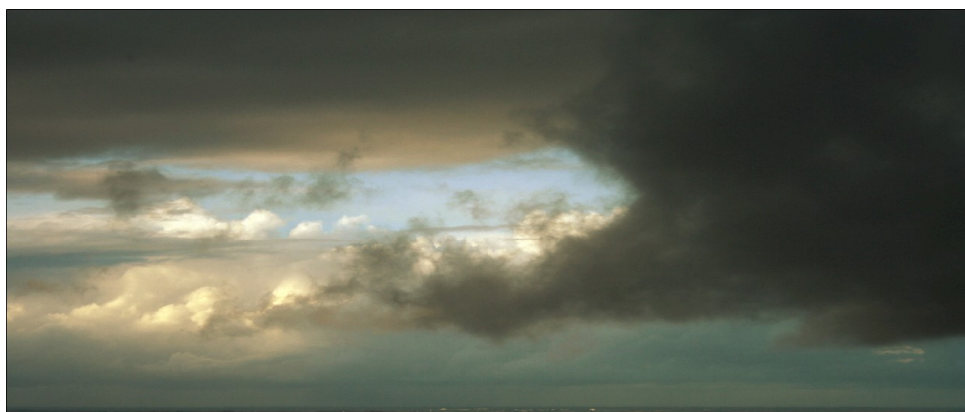
- They have their own resolution (unlimited)
- They can be hidden
- They have their own opacity values and blending modes
- They can be moved within the image canvas
- They can be modified by a stack of 2d filters
- They can be output to disk as stills or image sequences.

Color Layer

The color layer is a basic RGBA color matte layer.

File Layer

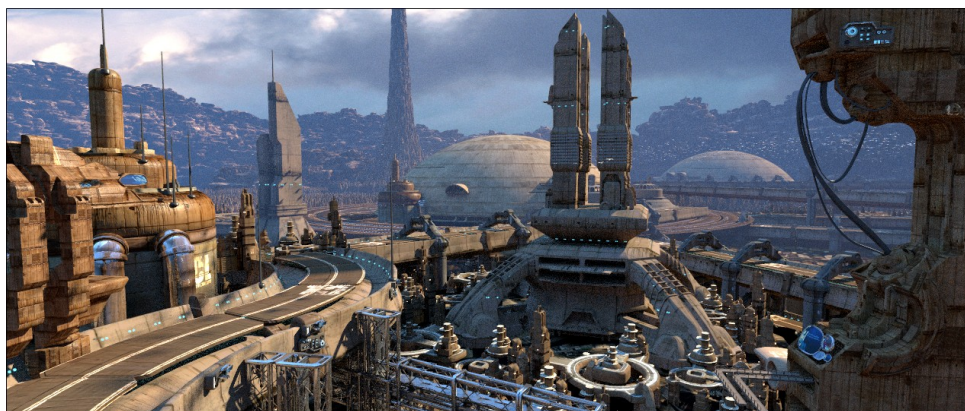
Import image file or sequence as a layer. Please refer to Technical specifications section for detailed information on supported image file formats.



A sky image used as image layer

3D Layer

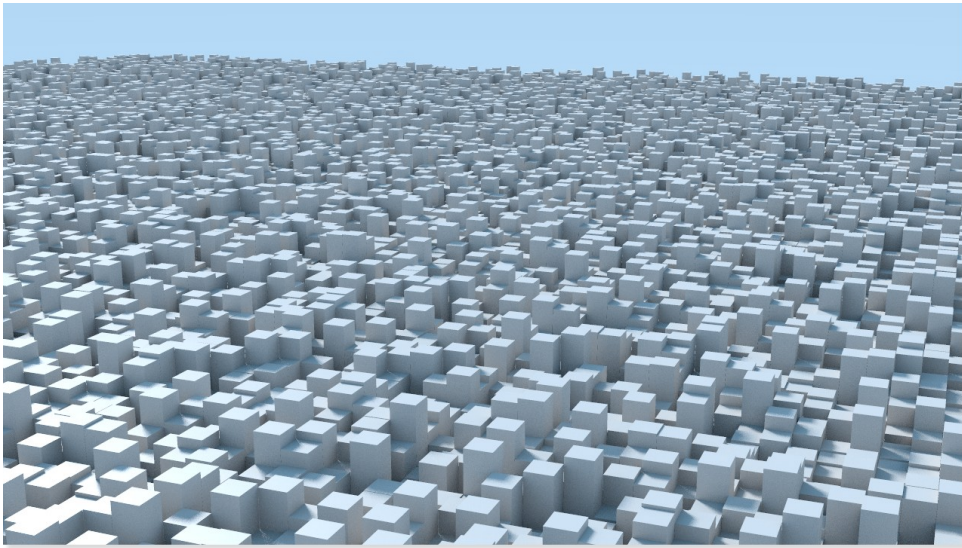
The 3D Layer is basically what stands for a 3D scene in other 3D packages. 3D Layers are defined by a viewpoint (camera), a renderer and a list of lights and scene objects.



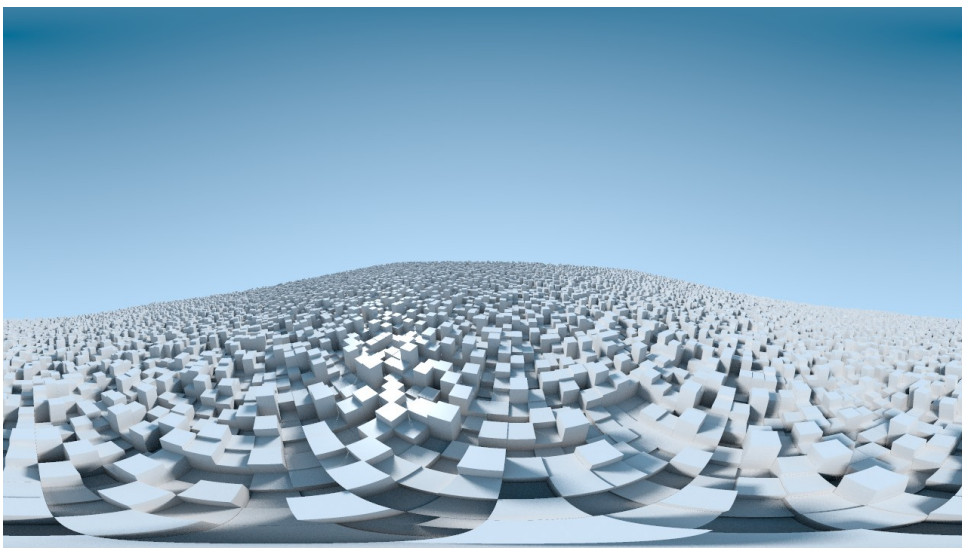
A raw 3d layer

Setting the Viewpoint

To set a viewpoint a 3D layer must reference a camera. In Clarisse, cameras also define projections. To set a viewpoint simply reference a Camera to *Camera* attribute.



Perspective camera



Panoramic camera (360 degrees)

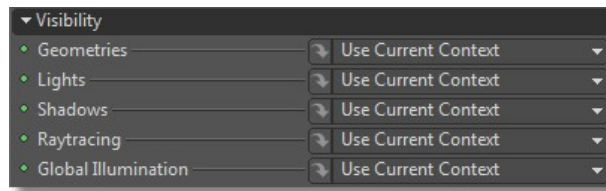
Setting the Renderer

In Clarisse you can setup multiple renderers with different settings in your project. To render a 3d layer, you must reference a renderer. For more information on renderers please refer to [What is a Renderer?](#)

Managing Visibility

By default the 3d layer sees what's in the context of the image embedding the layer. Through the Attribute Editor, you can very easily customize 3d layer visibility. For example, you can explicitly set what are the visible geometries, which lights are going to be used, which geometries are shadow casters, what's seen in reflection/refraction or what's used for global illumination

computations.



Each attribute references a group. You can either drag and drop a group to the attribute field, or directly drag items. When you drag items, a group including the selected items will be automatically created and assigned to the attribute.



Specifying 3d layer visibility

Lighting

By default, scene objects referenced by 3d layers are illuminated by lights that are in image context along with subcontexts. You can override this behavior by explicitly referencing *Lights* attribute to any group of lights.



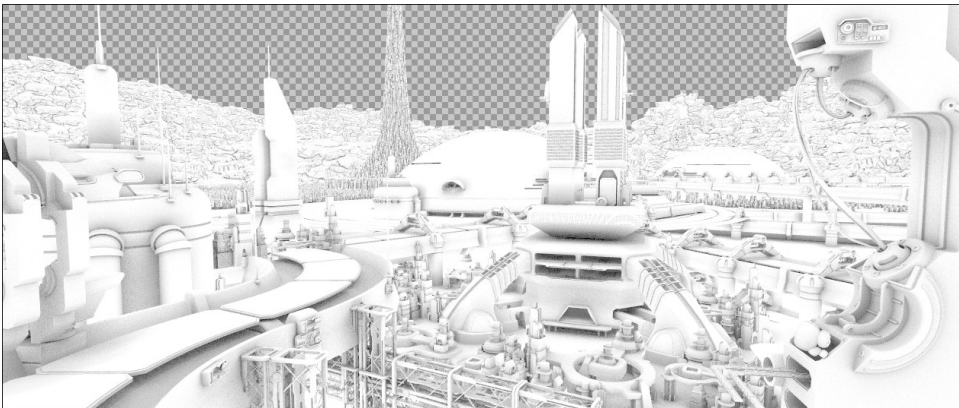
Using 3d layer context lights



A group of lights explicitly illuminating the 3d layer

Material Override

You can override all the materials referenced by your scene objects directly in the 3d layer. This feature is really useful when working on specific render pass. To override all materials, just reference the override material in the Override Material attribute.



Ambient occlusion path using a material override and a ambient occlusion light

Shading Layer

The *Shading Layer* attribute allows you to override materials referenced by your scene using a Shading Layer. For more information on Shading Layers please refer to Working with Shading Layer section.

Render Channels

3D Layers support render channel output that can be activated or deactivated. This can be an useful feature to export render passes.

Clipping

You can clip your 3D layer by setting custom near and far planes. By default,

clipping is disabled.

AOVs

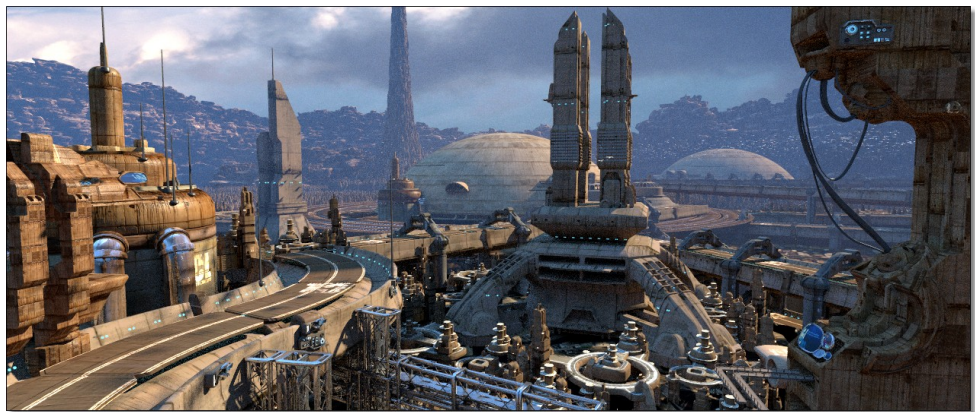
You can export a set of predefined AOVs at the 3D layer level. You'll find them in the *Aovs* category in the Attribute Editor when the layer is selected. For more information on AOVs please refer to Using AOVs section.

Image Layer

Directly references a Clarisse image as a layer.

Reference Layer

References a image layer as a new layer. This is really useful to add glow effect over a 3d layer for example. Create a reference layer referencing any 3d layer. Add a gaussian blur and reduce the layer opacity.



A raw 3D Layer



3D Layer with a blurred reference layer on top

Blending Modes

There are many different kinds of blending modes available:

Mode	Description
Normal	Performs a basic alpha blend
Alpha Replace	Replaces the alpha of the background layer by the foreground one
Alpha Add	Adds the foreground alpha to the background alpha
Alpha Subtract	Subtracts foreground and background alpha
Alpha Multiply	Multiplies both alpha channels
Alpha Divide	Divides both alpha channels
Add	Adds foreground to background
Multiply	Multiplies foreground to background
Screen	Multiplies the inverse of the foreground and background colors
Linear Dodge	Brightens the background color to reflect the foreground color by increasing the brightness.
Overlay	Multiplies or screen color according to the background color
Difference	Subtracts either the foreground color from background or background from foreground
Hue	Creates a result color with the luminance and saturation of the background color and the hue of the foreground color
Saturation	Creates a result color with the luminance and hue of the background color and the saturation of the foreground color
Color	Creates a result color with the luminance of the background color and the hue and saturation of the foreground color
Brightness	Creates a result color with the brightness of the background color

Applying Filters

Layers can be filtered through a stack of image filters. The layer filter stack is available through the Attribute Editor under *Filters* attribute.

Two kinds of image filters are currently available: pixel image filters and kernel image filters. Image Filters are embedded layers. If you duplicate a layer, all its image filters are copied too.

Note

Depending of the Clarisse flavor you are running, filters attribute can be textured. In this case textures are evaluated as if the layer was a flat square facing the camera. This is a very powerful feature you can directly use as texture, textures used in the materials.

Pixel Filters

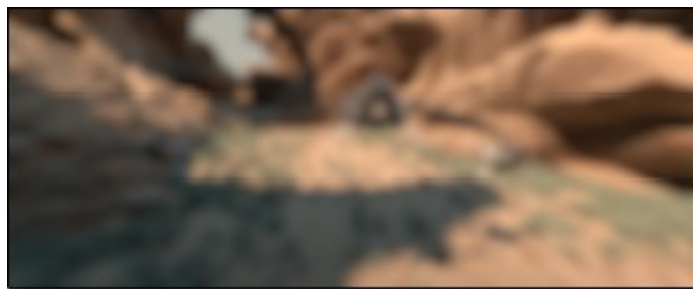
Pixel filters, such as brightness or contrast, operate using only the information contained in one pixel at the time.

Kernel Filters

Kernel filters such as defocus blur or gaussian blur work on a matrix of pixels. The kernel size is independent of the resolution multiplier attribute found in the image. For example, if you set your gaussian blur to 5 pixels, during evaluation, it will perform a 5 pixel blur at 100%, 2.5 at 50% or 10 at 200%.



A 10 pixel wide blur at 100%



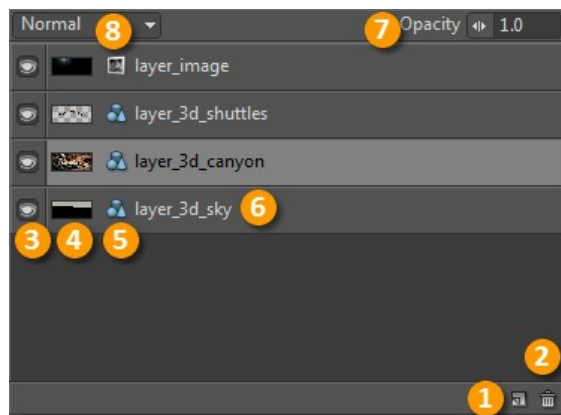
A 10 pixel wide blur at 50%

Using the Layer Editor

The Layer Editor allows you to create, edit and manage Image Layers to perform layered style compositing. For more information on layers, please refer to A Work on Layers section.

Overview

The layer editor manages all kinds of layers and has user interface shortcuts modifying blending mode, opacity and visibility. It displays all image layers embedded in a selected image. The order of compositing is bottom up: the most bottom layer being the background.



The Layer editor

(1) New Layer (2) Delete Selected Layer(s) (3) Visibility (4) Thumbnail (5) Type
(6) Name (7) Opacity (8) Blending Mode

Creating Layer

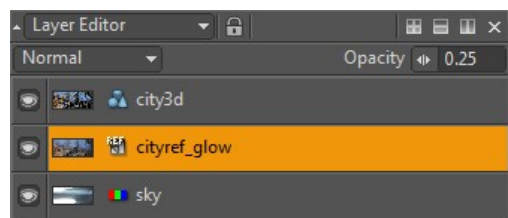
To create a new layer for the current image, click on (1) and choose the type of layer you want to create. By default, new layers are created on top of the selected ones.

Removing Layer

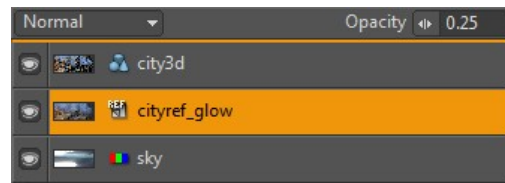
To Remove layers, select one or multiple layers and click on (2)

Reordering Layers

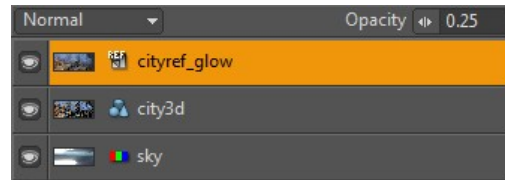
To reorder layers select the layer you wish to move.



Then Drag to insert the layer where you wish to. A marker appears to let you know where the layer is inserted.



Finally drop the layer.



Setting Layer Opacity

Select a layer and change the value in (7). Opacity is a shortcut to the Opacity attribute you can find in the Attribute Editor.

Setting Layer Blending Mode

Select a layer and click on (8). Choose the desired blending in the popup menu. You can set the blending mode by modifying the Blending attribute found in the Attribute Editor. There are many different kinds of blending modes available. For a complete list of blending modes available, please refer to Blending Modes section.

Working with Scene Objects

In Clarisse, Scene Objects are all items that can be rendered. The Scene Object class is an abstract class for items that are enclosed in a bounding box. It inherits from Scene Item so it can be animated. For example, geometries, particles, volumetrics... are all Scene Objects.

Geometries

In Clarisse, there are many different kinds of geometries. A Geometry is a 3d object made of points and primitives. Primitives are necessarily boundable 2d parametric surfaces. For example, for a set of parametric UV, a primitive must be able to return a coherent position in local space. Primitives are necessary mapped to a shading group used to attach materials.

Geometries support also normal map and both multiple vertex color maps and UV maps. Vertex color maps can be accessed using a TextureVertexColorMap in your material shaders. Please refer to Vertex Color Maps section.

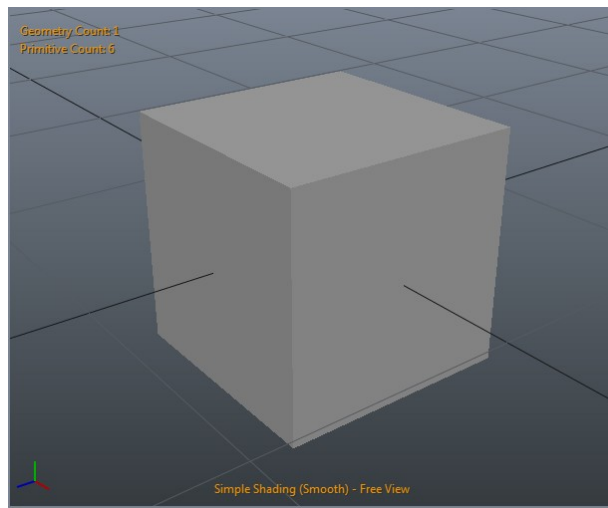
Implicits

Implicits are geometries that usually don't rely on points and therefore can't be deformed. They can be defined as a single primitive.

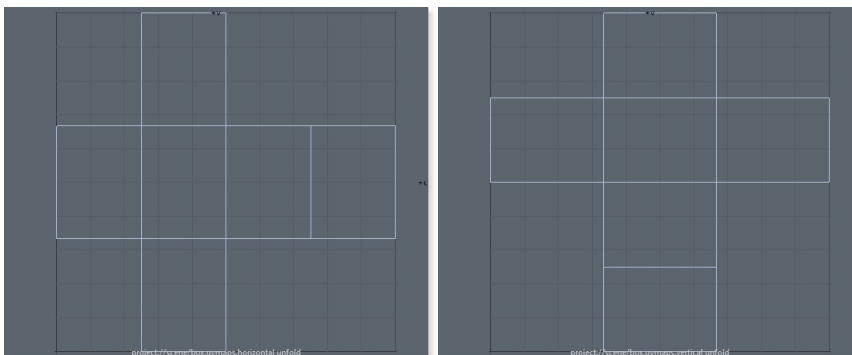
Box

The Box geometry is an implicit box and has 6 primitives.

Attribute	Description
Size	Set the box size.



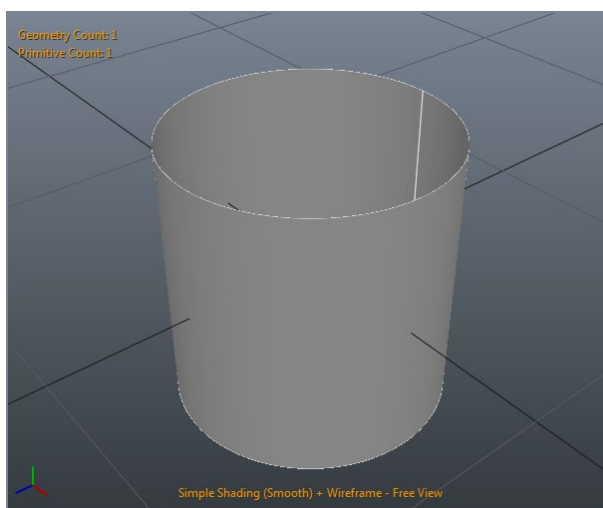
It defines two UV sets:



Cylinder

The Cylinder geometry is an implicit cylinder with no caps and has a single primitive. You can control its size using *Radius* attribute.

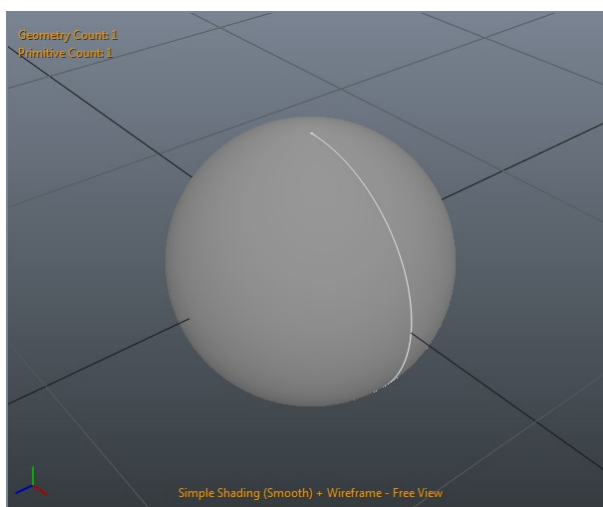
Attribute	Description
Radius	Radius of the cylinder. Cylinder's height is two times the radius value.



Sphere

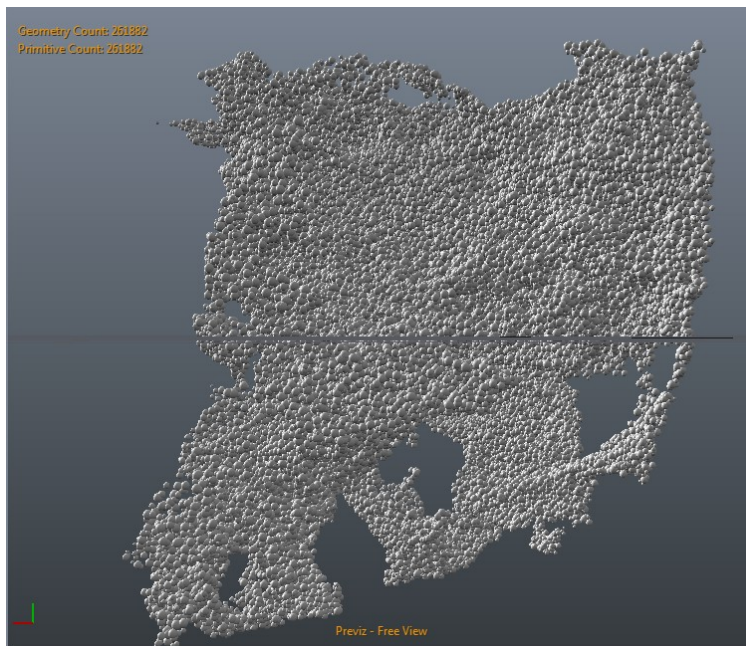
The Sphere geometry is an implicit sphere that has a single primitive.

Attribute	Description
Radius	Set the sphere size.



Particles

Particles are geometries defined by points. They don't define any primitives. They are invisible to renderers. Particles are mainly used as inputs for other classes of objects such as scatterers. You can generate particles in many different ways or simply import them from a file. To visualize your particles you need to display them in the 3d view.

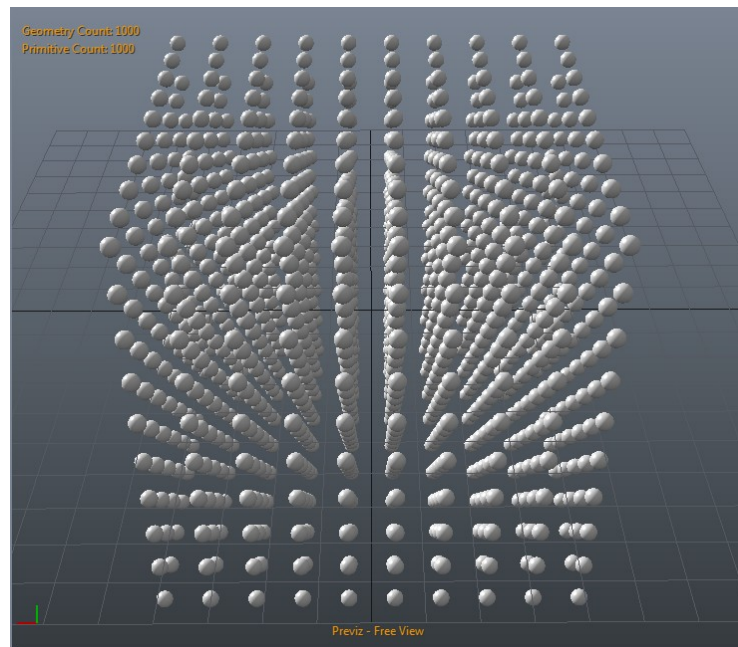


Particles used by a scatterer to scatter spheres

Point Array

The point array is a user-controlled array of points.

Attribute	Description
Size	Size of the array (x, y, z).
Count	Number of point in each axis. To get the total number of points generated simply multiply Count.X * Count.Y * Count.Z
Decimate Value	Decimation threshold. Each point has a chance to be removed. 0% keeps all the points, 50% removes about half of them, 100% removes all of them.
Decimate Texture	Set the texture used to decimate points.
Decimate Seed	Random generator seed used for Decimate Value.

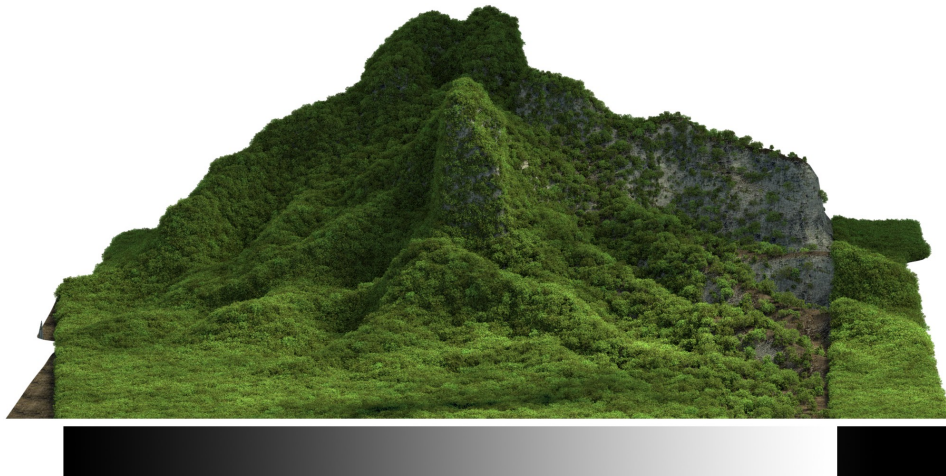


Point array used to scatter spheres

Point Cloud

Point Clouds are point samples generated on arbitrary surfaces. You can control the sampling density and the way the underlying Scene Object is sampled. The point cloud can sample any scene object that defines surfaces including scatterers, combiners and geometries. Please note the samples are sampled on the actual surface. If you sample subdivision or implicit surfaces, the sampling is performed on the limit surface and not on the tessellation.

Attribute	Description
Density	Sampling density in samples per square unit.
Decimate Value	Decimation threshold. Each point has a chance to be removed. 0% keeps all the points, 50% removes about half of them, 100% removes all of them.
Decimate Texture	Set the texture used to decimate points.
Decimate Seed	Random generator seed used for Decimate Value.
Geometry	SceneObject/Geometry to sample
Geometry Mode	Choose on which geometries the sampling is performed. Please refer to Geometry Mode section.



Gradient used as decimate texture on a point cloud

Note

Resulting point cloud keeps underlying geometry information such as normal, uvs, color maps...

Density Mode

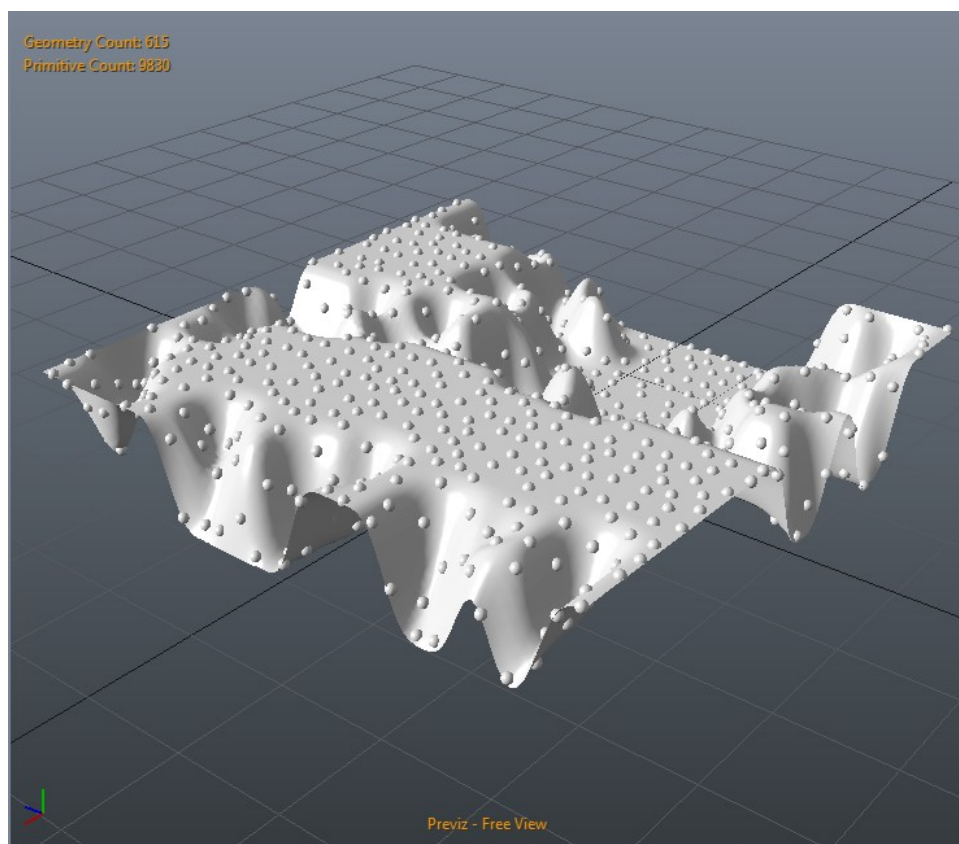
Depending on what you want to achieve, you can either sample underlying geometry using density or point count. By default, the point cloud samples geometries based on input density. To sample using a point count instead, uncheck *Use Density* and set *Point Count* to the number of samples you want.

Note

Please consider Point Count value as a hint. Depending on your sampling mode, typically *Blue Noise*, it may be impossible to reach the number of samples specified.

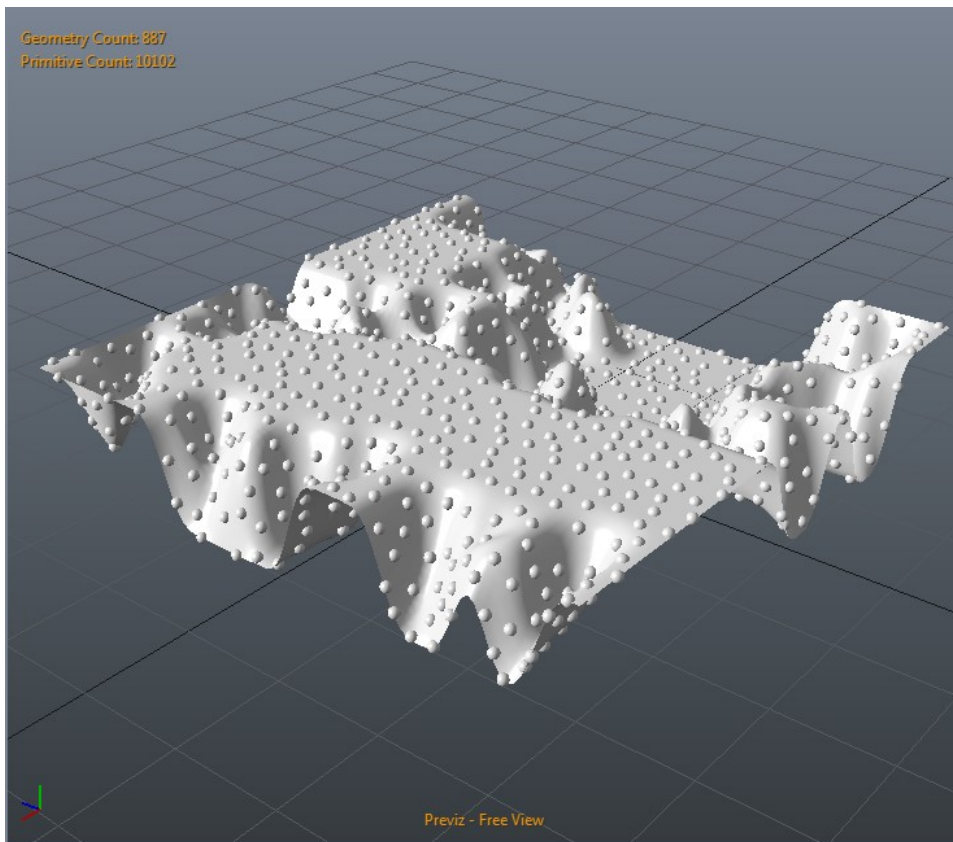
Geometry Mode

By default (Local Base), the sampling is performed on the local (without transformation) base geometry. In this mode, transformations and deformations don't affect the sampling. Samples stick to the deformed surface.



Sampling in Local Base, notice how the spheres stick to the surface.

When in Local Deform, the sampling is performed on the local deformed geometry instead. In this mode, deformations affect the sampling but transformations do not.

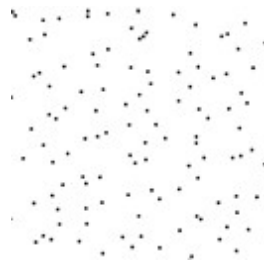


Sampling in Local Deformed, notice uniform distribution.

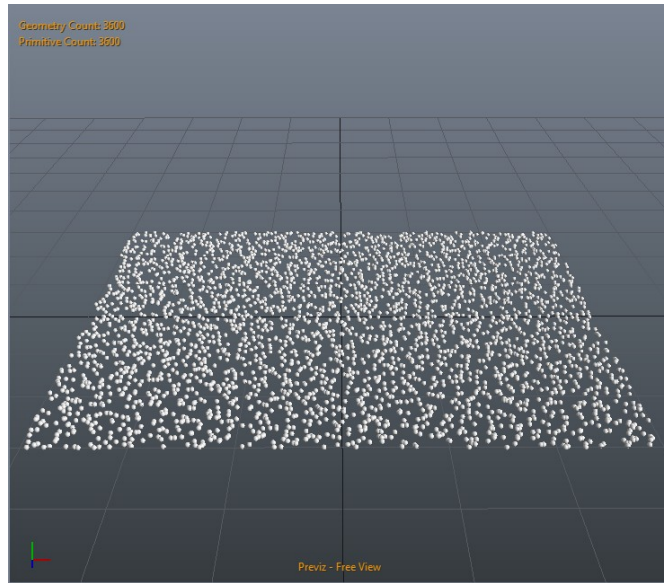
Global modes are working like Local modes however they take into account the transformation of geometries. For example, if you scale up the underlying geometry, it will increase the number of generated samples.

Random Distribution

When the distribution is set to Random, samples are randomly distributed over the underlying geometry. This mode is really interesting if you look for clumps to give a more natural constraint-free distribution.



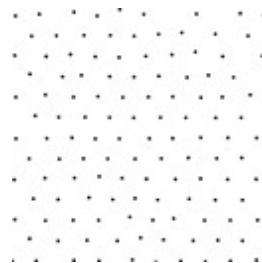
The number of samples will automatically be computed using *Density* attribute. It basically uses the input geometry total surface area and multiply this by the number of samples specified in *Density*. If the area of the input geometry is 10 square meters and you specified a density of 100, you will get $10 * 100 = 1000$ samples.



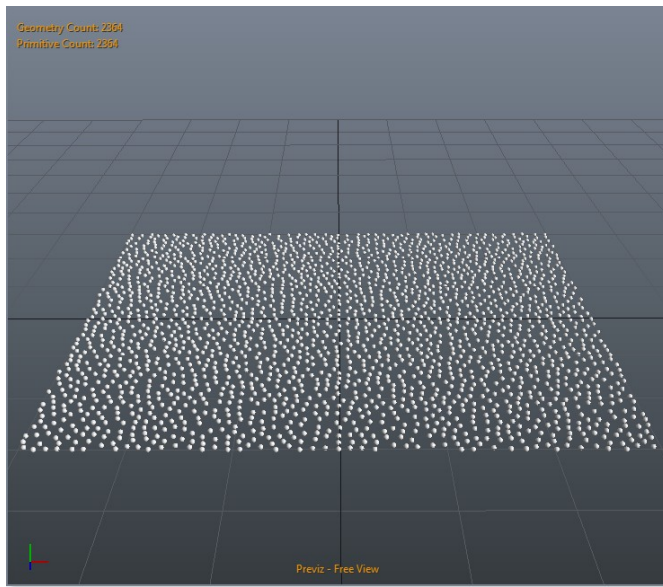
Point cloud using Random distribution

Blue Noise Distribution

When the distribution is set to Blue Noise, samples are distributed randomly over the underlying geometry while taking into account a radius constraint. This mode is really interesting if you want to avoid clumps to have a uniform looking distribution to represent, for example, cells, scales, trees etc. **Please be aware this distribution mode is computationally expensive and currently mono-threaded.**



The number of samples to be reached is determined by using the input geometry total surface area and multiplying this by the number of samples specified in *Density*. However, that count is ideal and can't be reached due to the random origin of the samples. In most cases, you will get about 70% of the ideal sample count.



Point cloud using Blue Noise distribution

Point File

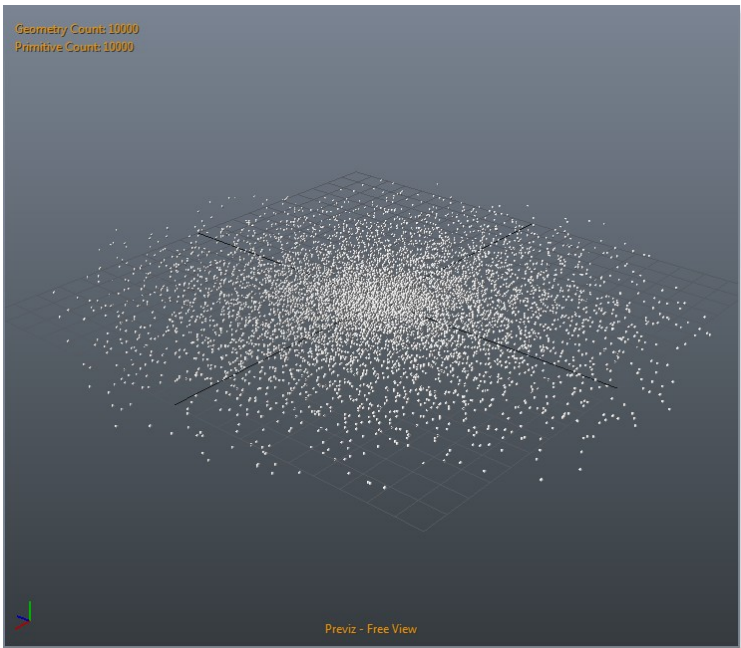
Point File are particles imported from a file.

Point Volume

Point Volume are random samples generated inside a box or an ellipsoid. You can control the point density as well as the sampling falloff.

Attribute	Description
Size	Size of the volume (x, y, z).
Falloff	Sampling falloff starting from the center of the volume.
Geometry	Shape of the volume. (box or ellipsoid)
Count	Number of generated samples.
Sampling Seed	Seed of the random generator used while sampling.
Decimate Value	Decimation threshold. Each points has a chance to be removed. 0% keeps all the points, 50% removes about half of them, 100% removes all of them.
Decimate Texture	Set the texture used to decimate points.

Decimate Seed	Random generator seed used for Decimate Value.
---------------	--

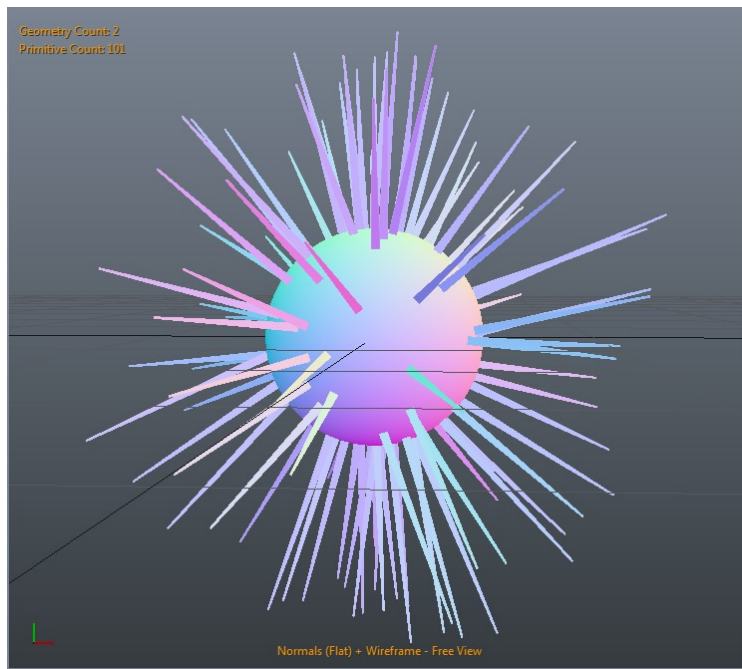


Fur

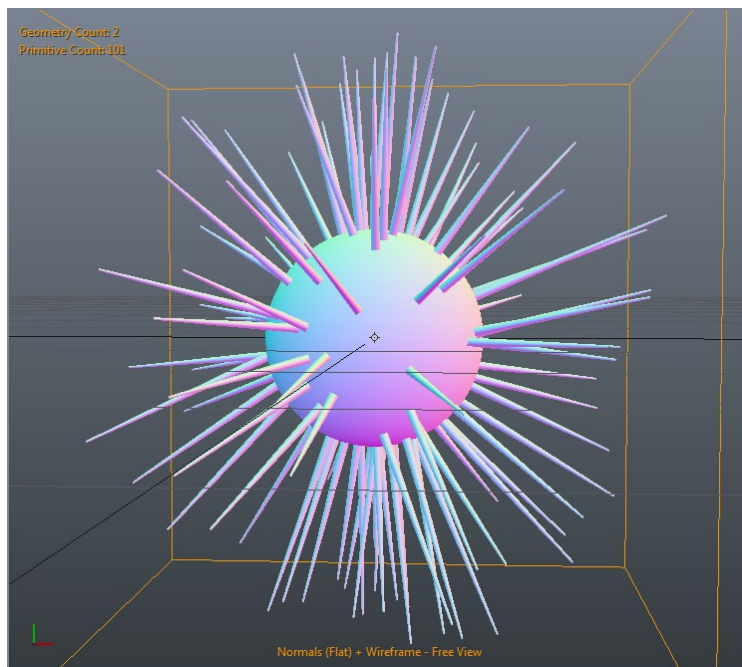
The Fur geometry is an abstract class that defines a mesh of curves used to render hair and fur. Each curve can be tessellated, have a root and tip size. They support flat (ribbon) and smooth (cylinder) rendering.

Attribute	Description
Radius Values	Use internal radius defined in the geometry or use custom ones.
Base Radius	The size of the curve at its root.
Tip Radius	The size of the curve at its tip.
Radius Multiplier	Multiplies all radius.
Segment Count	The number of linear segments for each curve. This attribute basically controls curve tessellation.
Flatness	Control the flatness of the segment. 0% is considered as a cylinder while 100% as a flat ribbon.

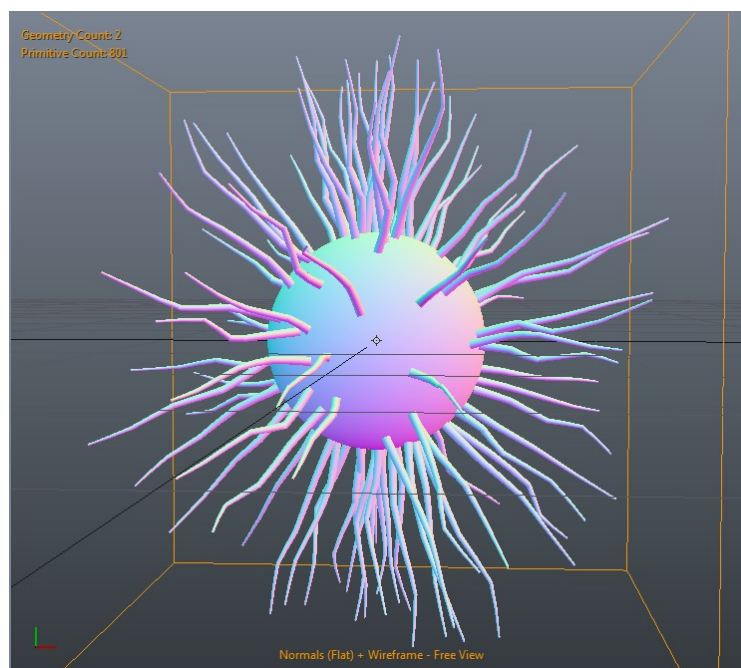
Fur Geometry shared attributes



Flatness at 0%, 1 segment



Flatness at 100%, 1 segment



Flatness at 0%, 8 segments

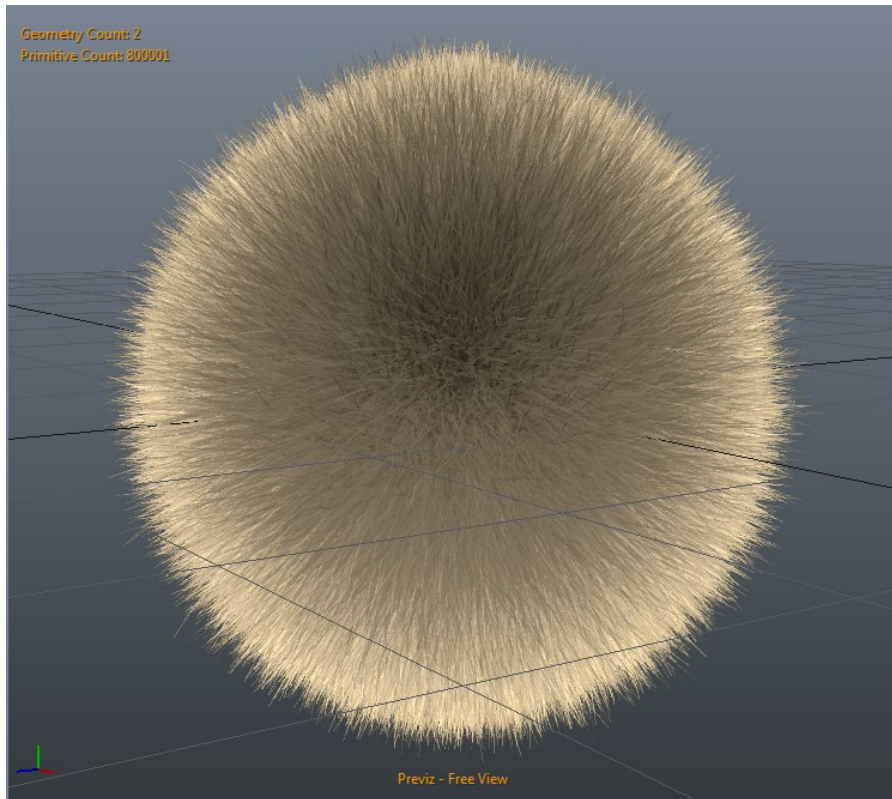
Clarisse provides several kinds of fur generators giving you access, for each fur root, to the underlying geometry UVs. Please refer to Fur Support Color section.

Generator

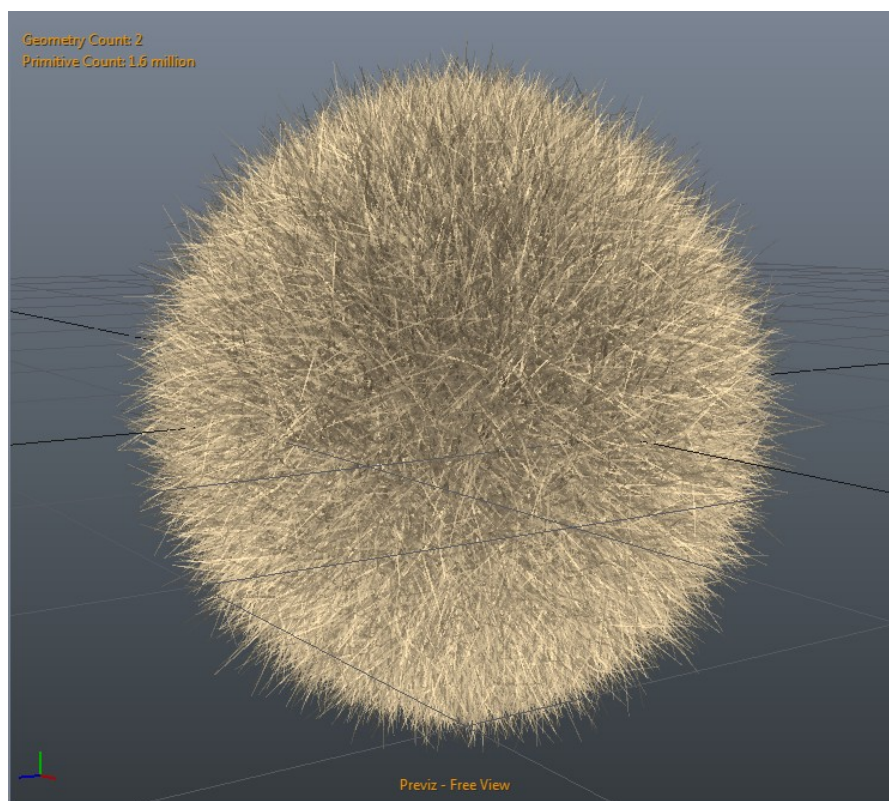
The fur generator is a basic fur generator that allows to grow fur on any kind of geometries. You can control hair length, variation, frizziness etc...

Attribute	Description
Length	Set the length of each curve.
Length Variation	Set the length variation across each curve.
Frizziness	Set how frizzy curves are.
Frizziness Variation	Set the frizziness variation across each curve.
Knot Count	Set the number of knots of each curve. The more knots your curve has the more complex its shape can be.
Gravity	Set the direction of the gravity (3D Vector)
Geometry Support	Set the particles on which each fur will grow on.

The fur generator use input particles (point clouds etc...) geometry to generate fur on. Final fur density is based on the number of input particles. If you have 100 particles in your point cloud, the Generator will end up generating 100 curves. If your geometry has a transformation, you should parent the fur generator to the geometry you are growing fur on.



Fur generated on an implicit sphere



Same fur but with Frizziness set to 100%

Interpolate

Clarisse can interpolate fur using a support geometry and fur guides as input. Guide hair size, motion blur and vertex color maps are also interpolated. Please note the fur interpolate does not perform dynamics.

Fur interpolate can also control the shape of the interpolation and alter the resulting interpolated hair.

Attribute	Description
Density	Sets the number of generated hair.
Geometry Support	Sets the geometry on which the fur will grow on (typically a point cloud).
Guides	Sets the fur used as guides.
Guide Influence Radius	Sets the maximum distance for a guide to have an influence on a sample.
Guide Interpolation	Sets the interpolation function.
Guide Clump	Sets the shape of the clump.
Max Guide Count	Sets the number of guides used for each sample.

Knot Count	Sets the number of control points on the generated curve.
Frizziness	Control the frizziness of the curve.
Frizziness Variation	Control the frizziness variation of the curve.
World Coordinates	Sets if the generation should use world coordinates.



Guide hair on the left, interpolated result on the right



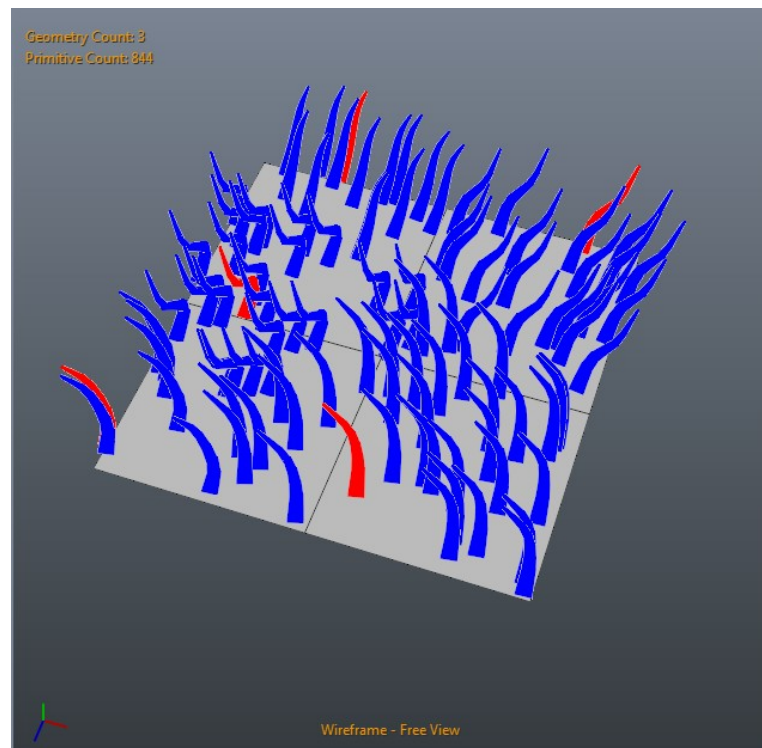
Guide (left) and interpolated (right) hair with motion blur

How it works?

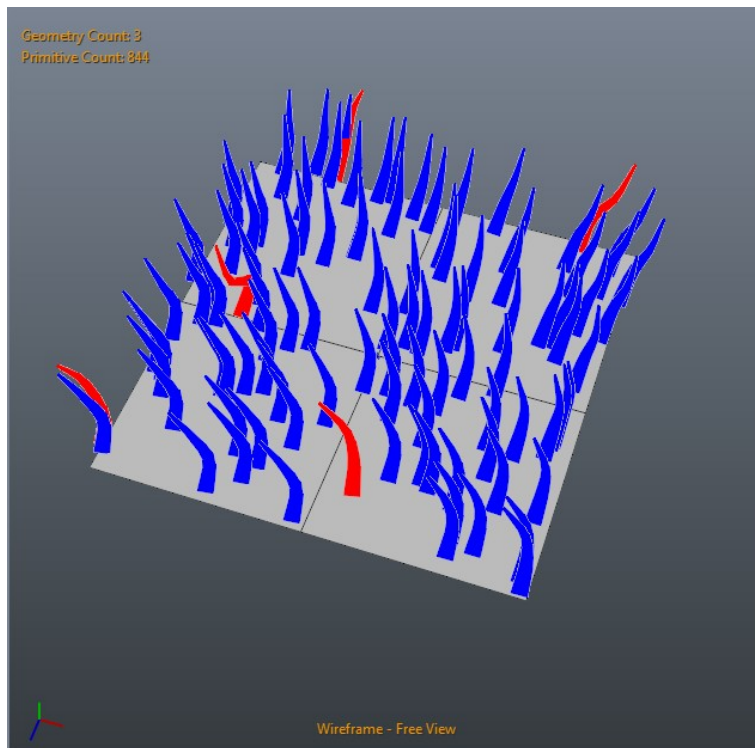
The fur interpolate needs a point cloud or vertices as input. This point cloud is used as roots for the interpolated hair.

In order to generate interpolated curves, the Fur Interpolate also needs input *Guides*. These guides can be any fur object so that nothing prevents you to use a fur interpolate as guides. Using the input guides, the fur interpolate will spatially look up for a certain number of guides. You can control this number using *Max Guide Count* attribute. You can also control the look up radius using *Influence Radius* and *Influence Weight*.

Finally using the *Influence Shape* curve, it blends the shape of the interpolated curve according to the distance of the curve to its guides.



Only one guide used here, each curve is the same as the reference guide

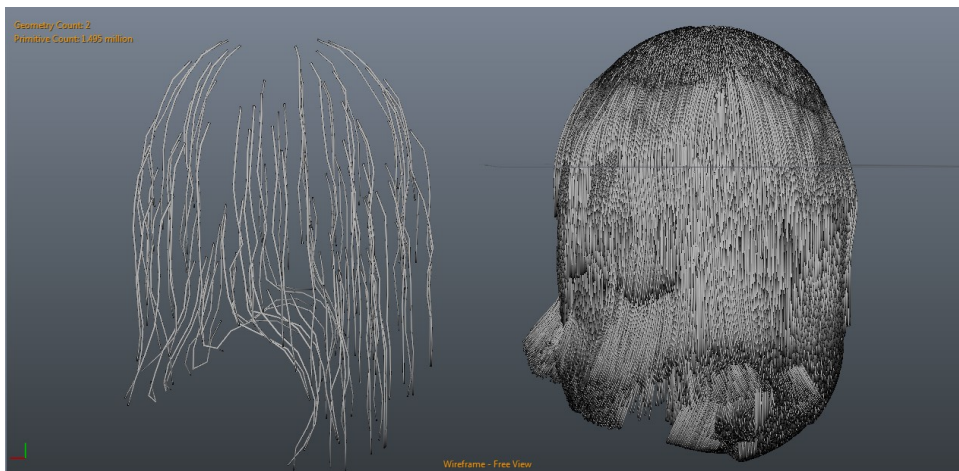


Now 5 guides are used. Note the blending on generated curves.

Please note the fur interpolate works in local space. However, you can toggle it to work in world space using *World Coordinates* attribute.

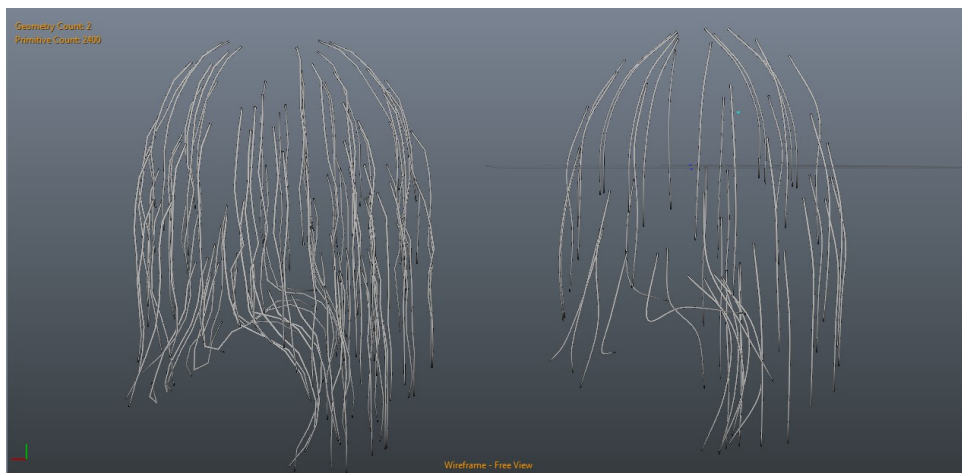
Density

Hair density is controlled by the input point cloud set in *Geometry Support*. Remember, you can texture the density of your point cloud according to a texture that will be used as a mask. Please refer to Point Cloud section for more information.



Guides (left) and interpolated hair using a point cloud density set to 50000 (right)

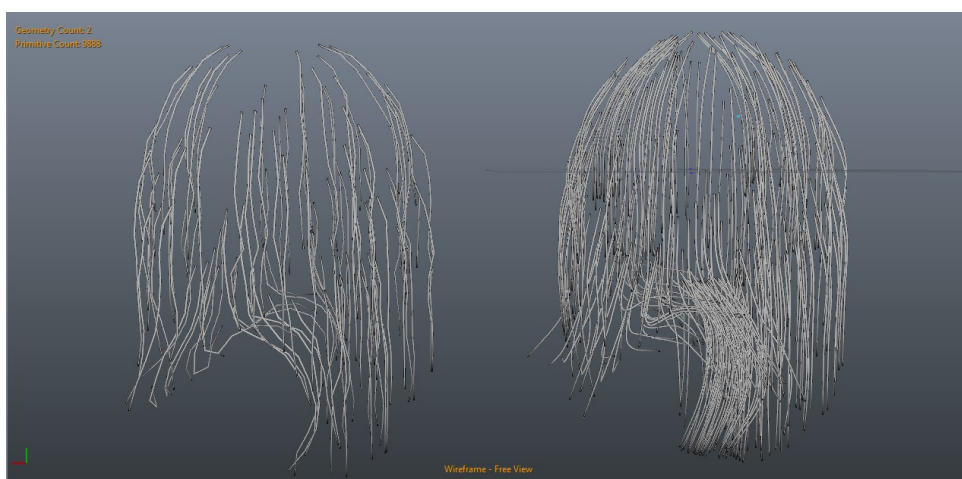
You can use density numbers that are either higher or lower than the input guide hair count.



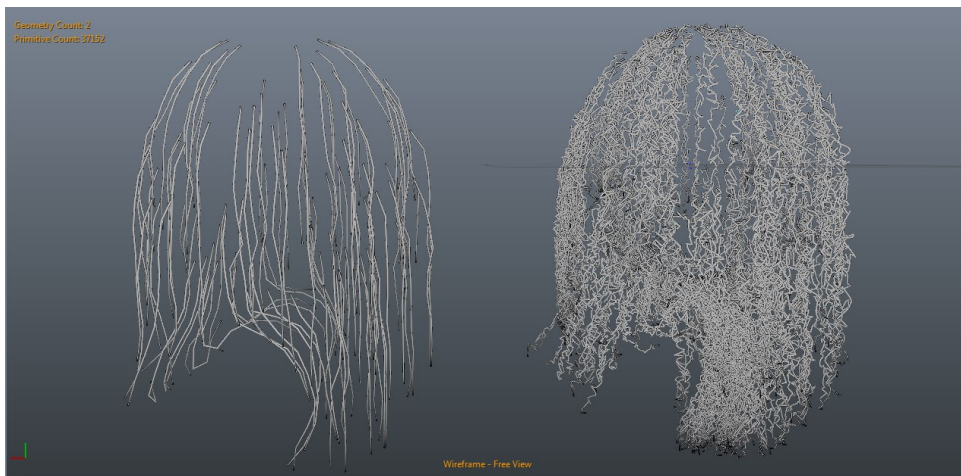
Guides (left) and interpolated hair using a point cloud density set to 50 (right)

Frizziness

Frizziness and *Frizziness Variation* control how frizzy interpolated hair are.



Frizziness set to 0 (default)



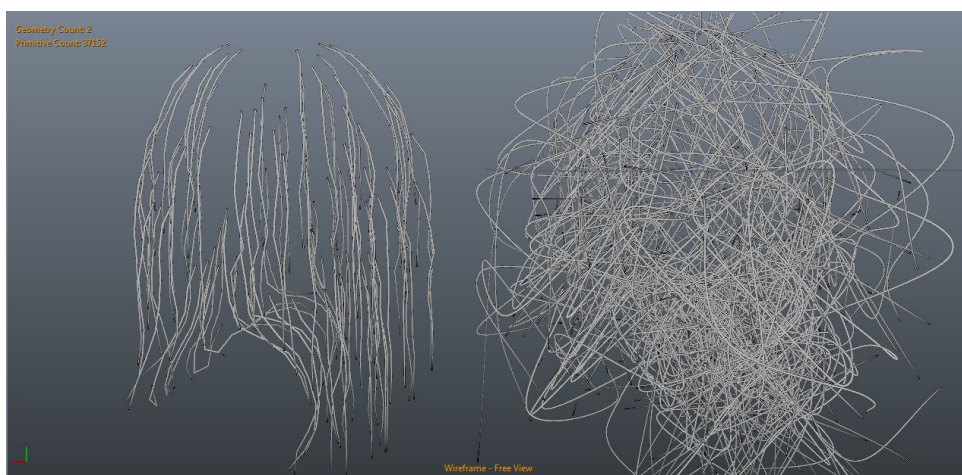
Frizziness set to 100% (64 knots)

Knot Count

Knot Count defines the number of knots for each generated curve. The more knots, the finer you can alter the shape of the curve. Knots are not the number of the segments your curve will be tessellated to. To control the curve tessellation you have to modify the attribute *Segment Count*.



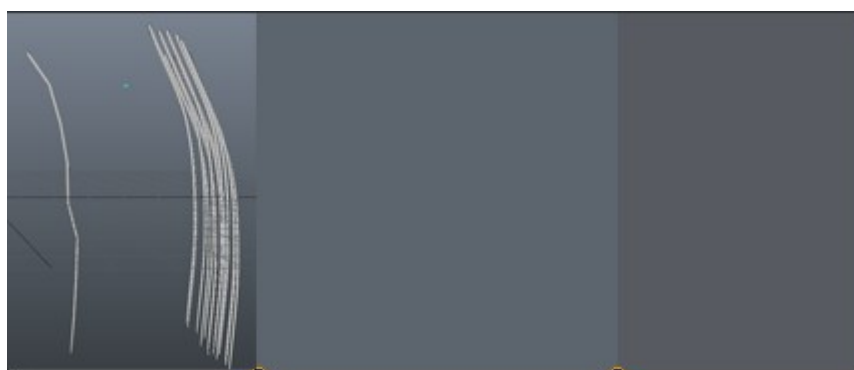
Frizziness set to 100% with 64 knots



Same settings with 4 knots (default)

Clumping

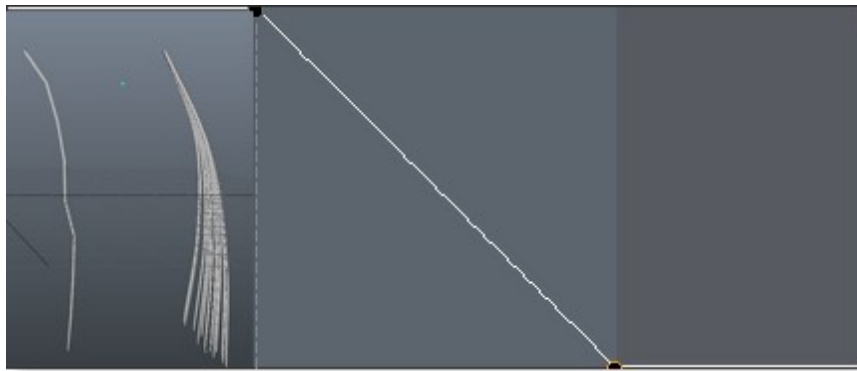
The fur interpolate allows you to clump generated hair on guide strands. Using a custom curve you can control the shape of the clump. The curve controls the stickiness along the guide strand such as values near 1 tend to stick hair on guide strands.



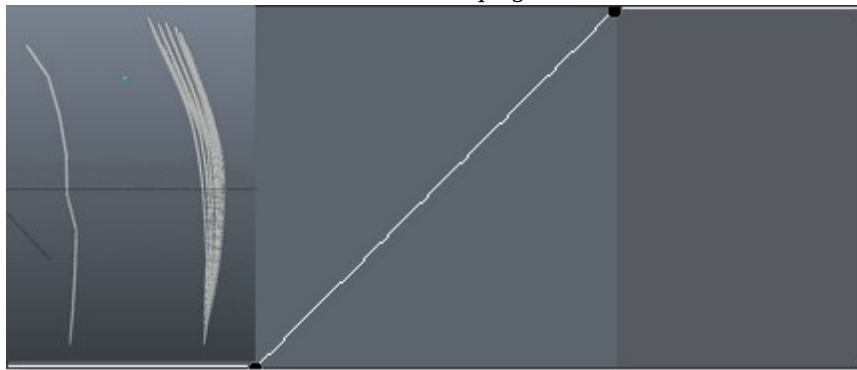
no clump (default)



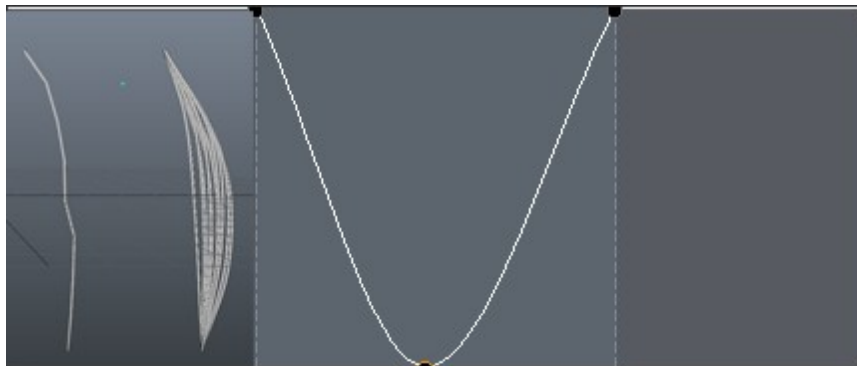
Full clump



Root clumping



Tip clumping



Clumping half way

File

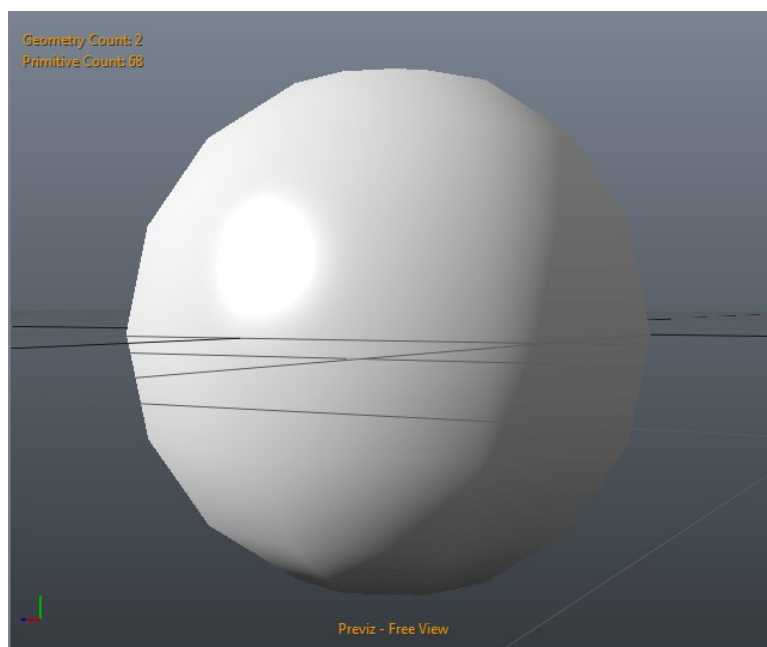
Fur files are curve meshes imported from a file. Please note Clarisse supports curves with different numbers of knots originating from a same file.

Polymesh

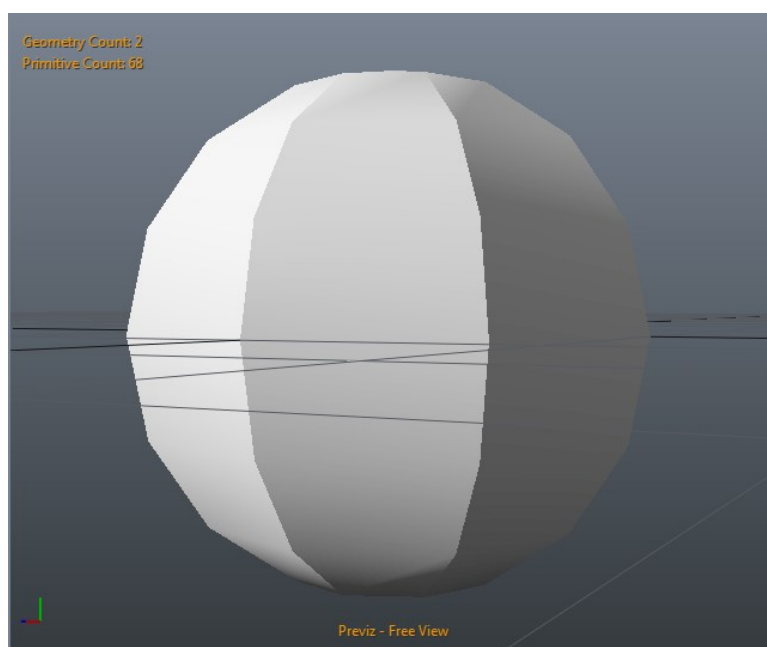
The Polymesh geometry is defined by polygons and vertices. A polygon is a flat plane formed by vertices. In Clarisse, any polymesh can be used as a poly cage for subdivision surfaces.

Normals

By default, smooth normals are rendered using the first available normal map of the polymesh. If no normal map is found, the normals are interpreted as completely smooth. You can change the *Smoothing Mode* and define a custom *Smoothing Angle*. Just change *Smoothing Mode* to *Use Smoothing Angle*.



Using normal maps defined by the geometry

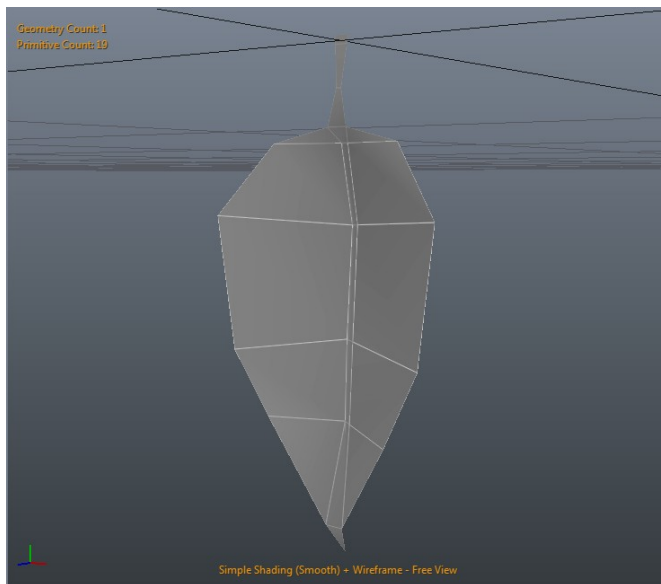


Custom smoothing angle (set to 25)

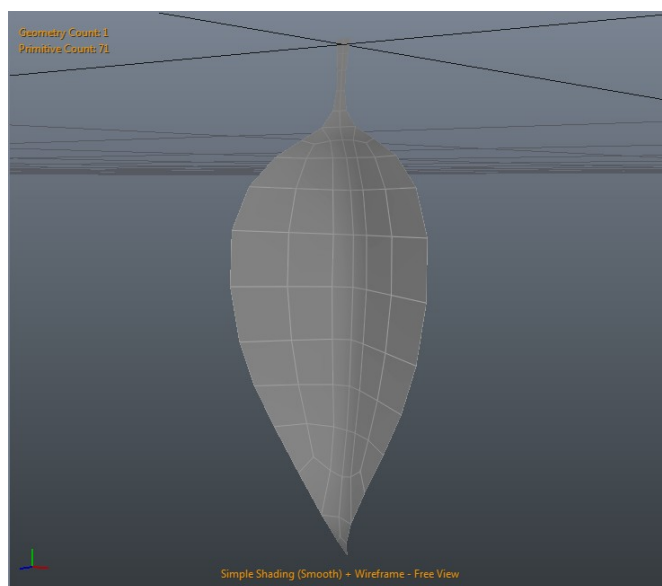
Subdivision Surfaces

Subdivision surfaces (Subd) is a method that creates a smooth surface mesh from a linear polygon mesh. Clarisse subdivision engine creates a mathematical representation of the surface and is able to sample the surface without tessellation. To enable subdivision surfaces, check Enable Subdivision Surface attribute.

Attribute	Description
Enable Subdivision Surface	Converts the polymesh into a subdivision surface mesh
Geometry Interpolation	Specifies the subdivision surface mode.
UV Interpolation	Specifies the subdivision mode for UV maps.
Raytrace Mode	Specifies the subdivision surface rendering engine used for raytracing.
Raytrace Tessellation Level	Specifies the tessellation level used while in <i>Use Tessellation</i> and <i>Tessellate on the Fly</i> modes.



Polymesh Cage

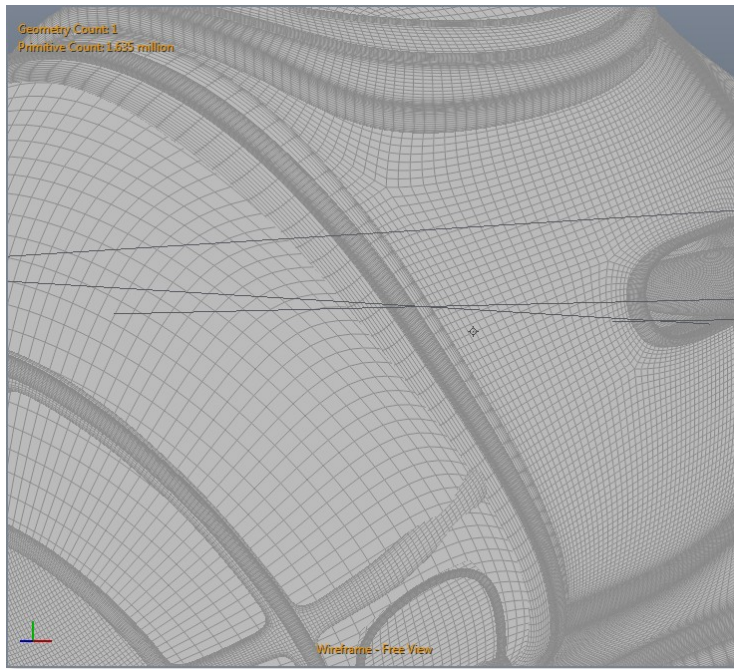


Subdivision surface turned on

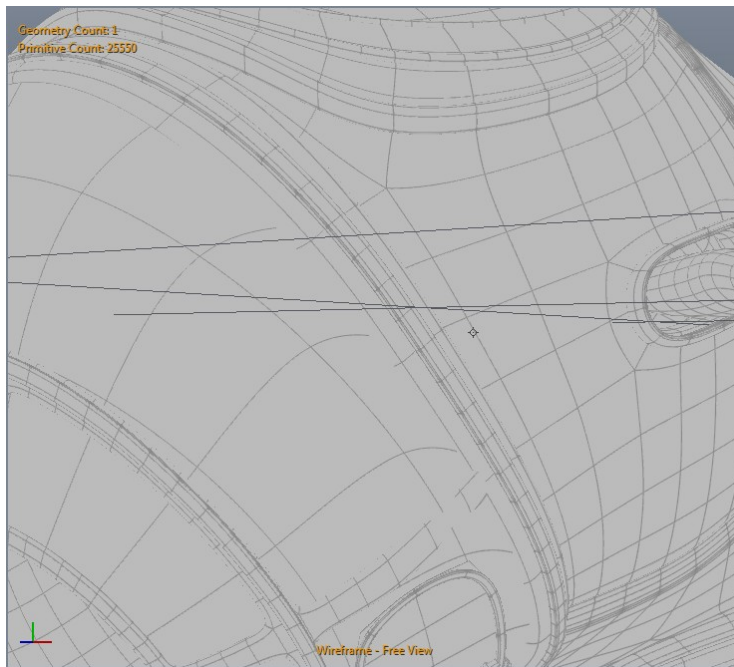
Raytrace Mode

Clarisse provides different methods to handle subdivision surfaces during rendering.

Mode	Description
Fixed Tessellation	Converts the subdivision surfaces into polygons (quads). This is the fastest mode but can require a lot of memory when the Tessellation Level has high settings.
Tessellation of the fly	Performs the tessellation during rendering. This method is slower than Fixed Tessellation but it doesn't require memory.
Evaluate Surface	Evaluates directly the mathematical surface. No tessellation is performed but it is the slowest method.



Fixed Tessellation level 3

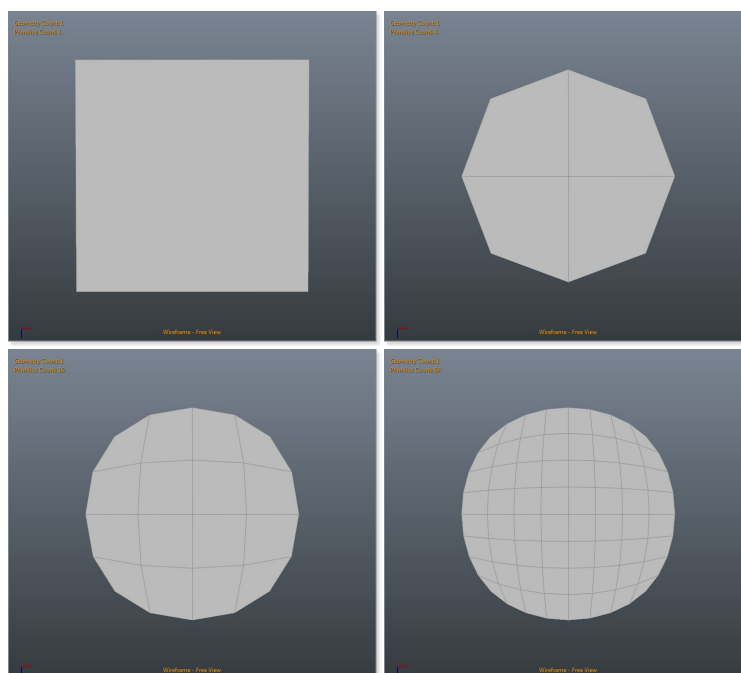


Evaluate surface

Raytrace Tessellation Mode

When set to Fixed Tessellation or Tessellate on the Fly modes, Clarisse subdivision engine relies on this attribute to set the render level of tessellation. A level of 1 means the cage will be subdivided once into quads. Triangles are split into 3 quads, quads into 4 quads... and so on. After the first level of

subdivision the resulting polymesh is a quad mesh: a mesh composed exclusively of quads. At level 2 quads are split into 4 new quads and so on.



A simple quad at level 1, level 2 and level 3

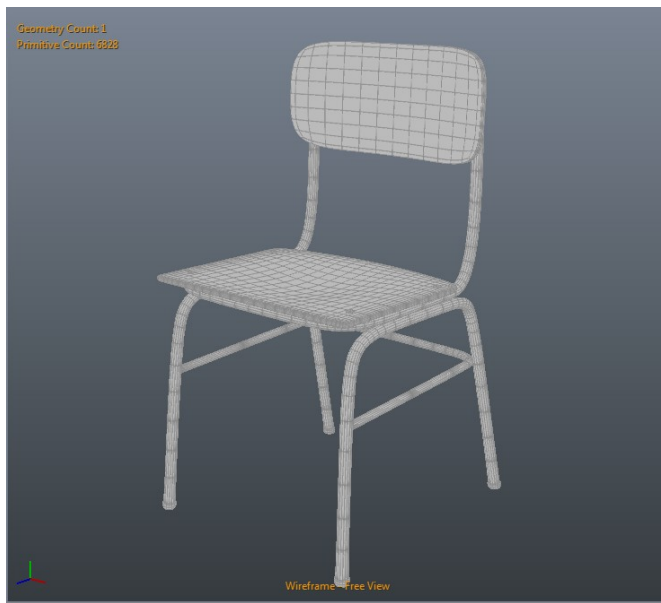
At level 3 each original quad from the cage polymesh are subdivided into 64 quads.

Polyfile

The Polyfile allows you to import an external polygon mesh via file referencing. Importing a geometry via file referencing means that geometry topological data isn't stored in the project. The Polyfile relies instead on the referenced file and each time the project is loaded, the Polyfile re-imports geometry data directly from the referenced file.

Attribute	Description
Filename	Path to the imported geometry file. You can change this path anytime.

You can import geometries using **File > Import > Geometry...** or create a new Polyfile using **Create > Geometry > Polyfile** and then specify its *Filename* attribute to the geometry file path you wish to import. As any other attributes, at anytime you can change the path to import another geometry.



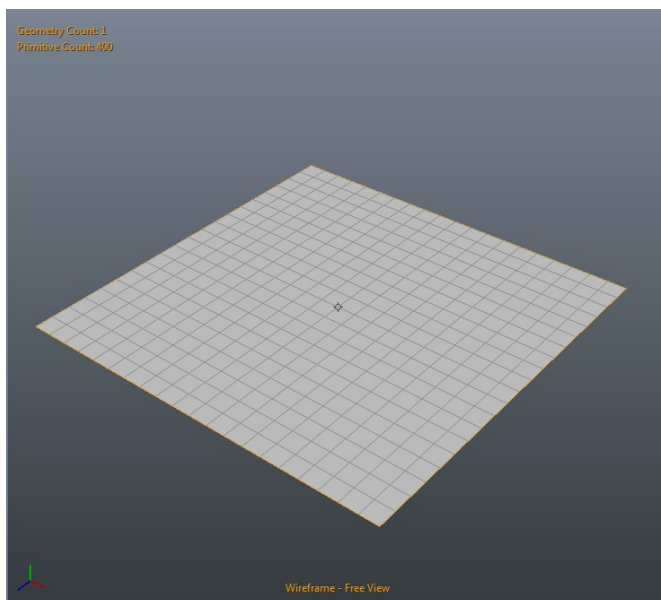
A polymesh imported from a file

To learn more on importing objects please refer to Understanding Assets

Polygrid

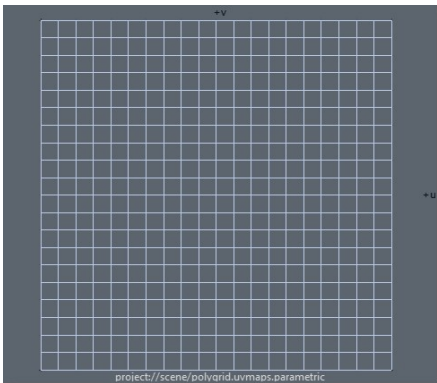
The Polygrid is a simple flat polygon grid. You can control the number of spans in *u* and *v* using *Spans* attribute.

Attribute	Description
Spans	Set the number of spans in <i>u</i> and <i>v</i> .



Polygrid with 20x20 spans

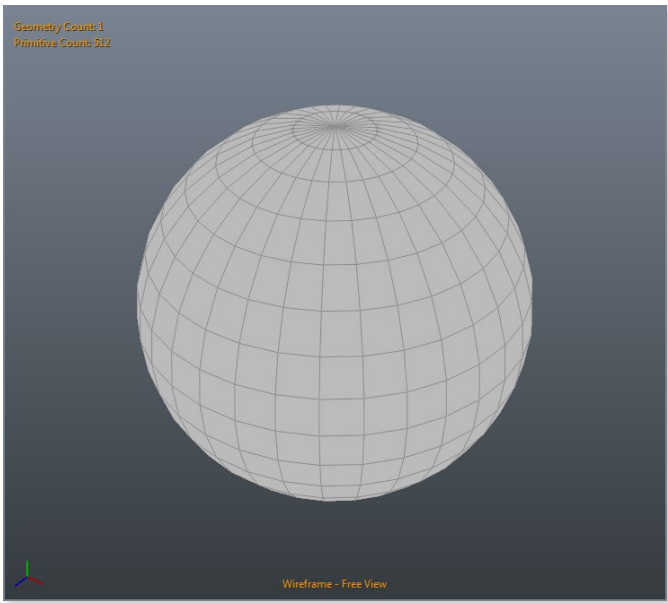
The Polygrid defines a single UV map:



Polysphere

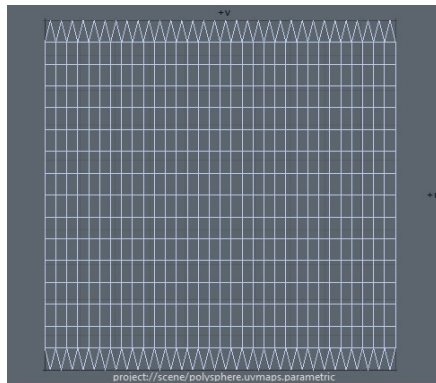
The Polysphere is a simple polygon sphere. You can control the number of spans in *u* and *v* using *Spans* attribute and its size using *Size* attribute.

Attribute	Description
Spans	Set the number of spans in <i>u</i> and <i>v</i> .
Size	Set the sphere size in <i>X</i> , <i>Y</i> , <i>Z</i>



Polysphere with 32x16 spans

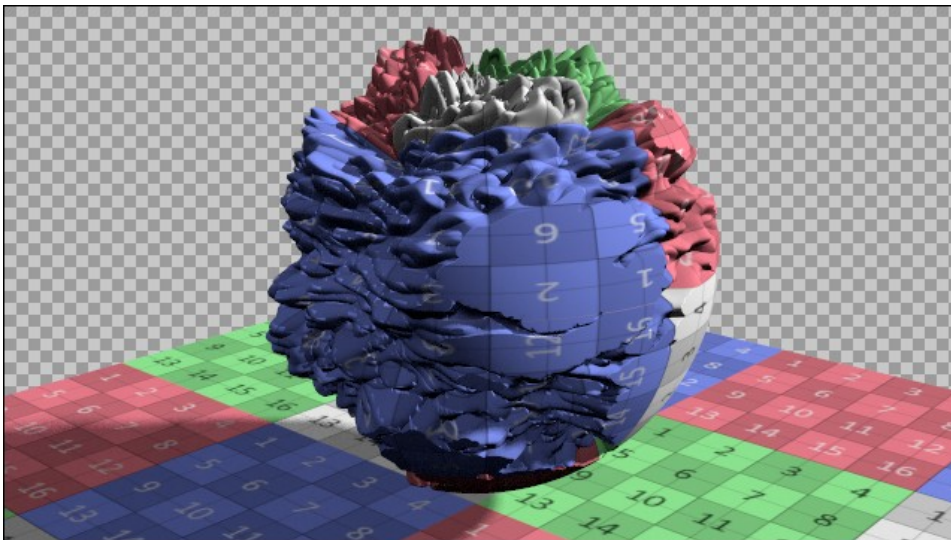
The Polysphere defines a single UV map:



Displacement Mapping

Displacement mapping displaces geometry primitives along surface normals from a texture used as input. To perform displacement mapping, Clarisse needs to tessellate underlying geometries to create a displacement mesh. The displacement map texture is then evaluated on each vertex of this new geometry. Please note that, displacements work directly on triangles and quads. However, if a polygon has more than 4 vertices, the polygon will be triangulated before being tessellated and displaced.

To enable displacement mapping you need to assign a displacement shader to a geometry shading group, using the Material Linker.



A displaced implicit sphere using Adaptive (base) heuristic

Note

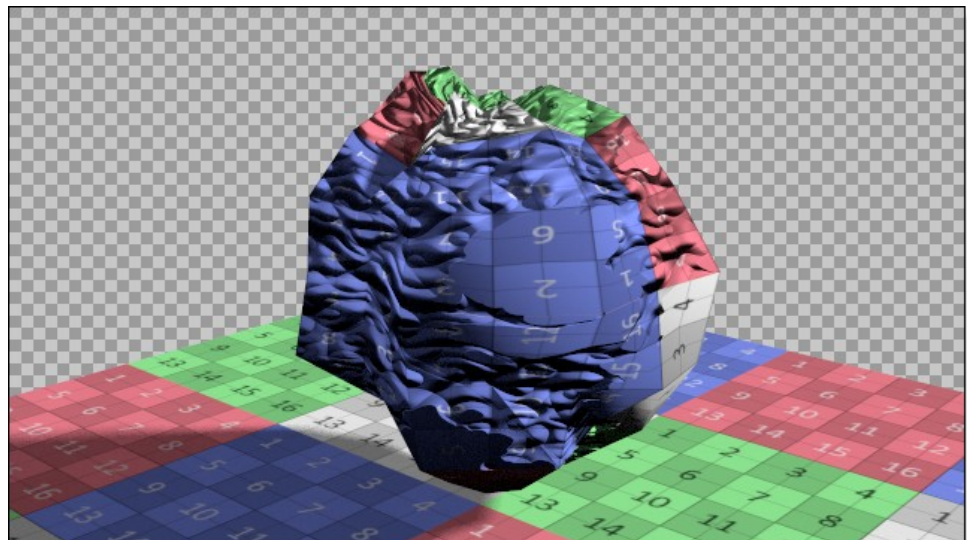
Even if displacements are extremely optimized in Clarisse, they both have a non negligible rendering and memory cost.

Displacement Tessellation Mode

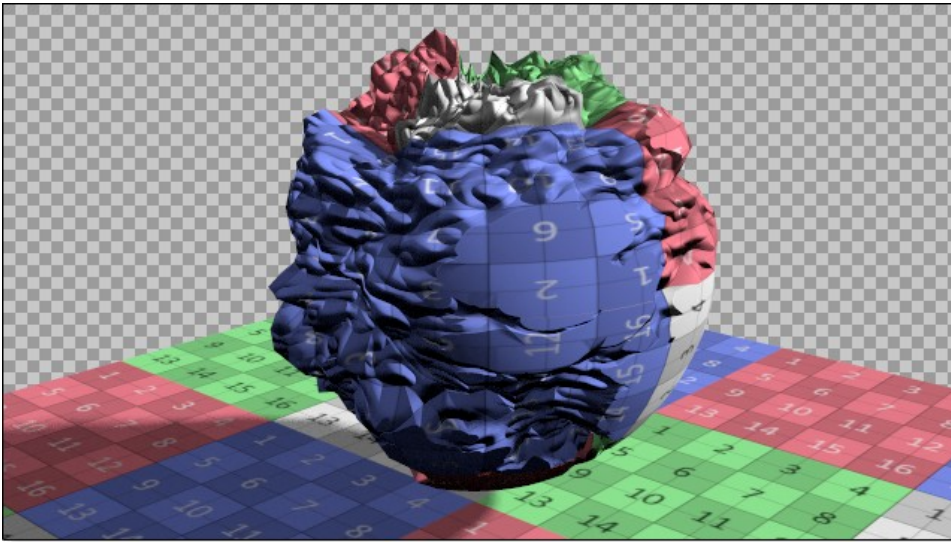
Clarisse provides several tessellation modes to drive tessellation schemes and thus level of detail of displacement. These attributes are defined in geometries.

Mode	Description
Adaptive (base)	This special heuristic performs adaptive tessellation based on primitive areas of the base geometry (without deformation). The level of tessellation is defined by <i>Displacement Adaptive Span Count</i> attribute.
Adaptive (deformed)	This special heuristic performs adaptive tessellation based on primitive areas of the deformed geometries. The level of tessellation is defined by <i>Displacement Adaptive Span Count</i> attribute.
Uniform	Perform uniform non-adaptive tessellation. Each quad of the displacement mesh is subdivided by <i>Displacement Uniform Span Count</i> . For example a level of 2 subdivides a quad into 4 new quads.

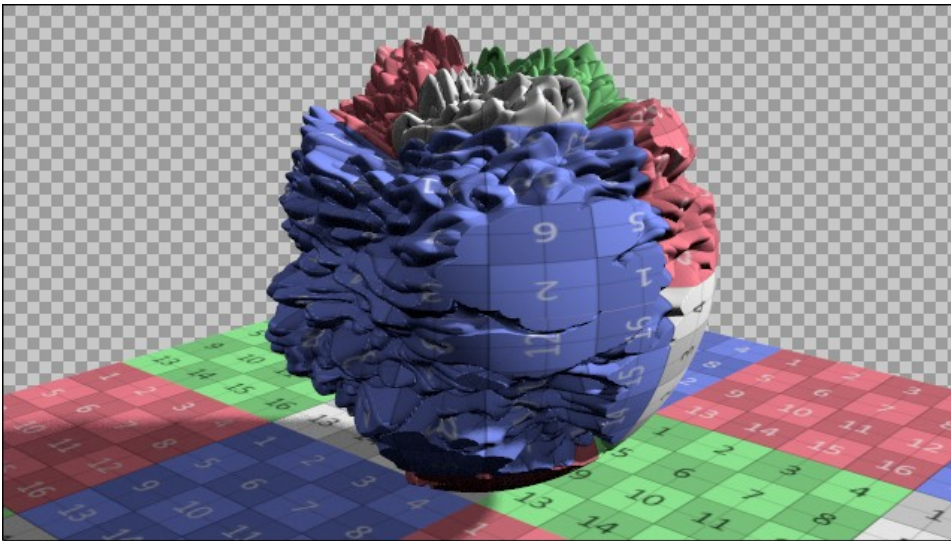
When using adaptive modes, the level of tessellation is defined by *Displacement Adaptive Span Count*. This value is relative to the max extent of the bounding box of the geometry that is displaced. The best way to understand how this heuristic works is to imagine you are subdividing the bounding box of the underlying geometry instead. In fact when you set a value of 200 spans, this defines the size of the micropolygon according to the bounding box size. The geometry is subdivided adaptively using this micropolygon size. The mode doesn't add micropolygon to primitives which area is smaller than the micropolygon size.



Displacement Adaptive Span Count set to 5 on an implicit sphere



Displacement Adaptive Span Count set to 20



Displacement Adaptive Span Count set to 200

Displacement

Displacement is performed using *Displacement* shaders. Similarly to materials, you need to use the Material Linker to assign displacements geometry's shading groups.

Value

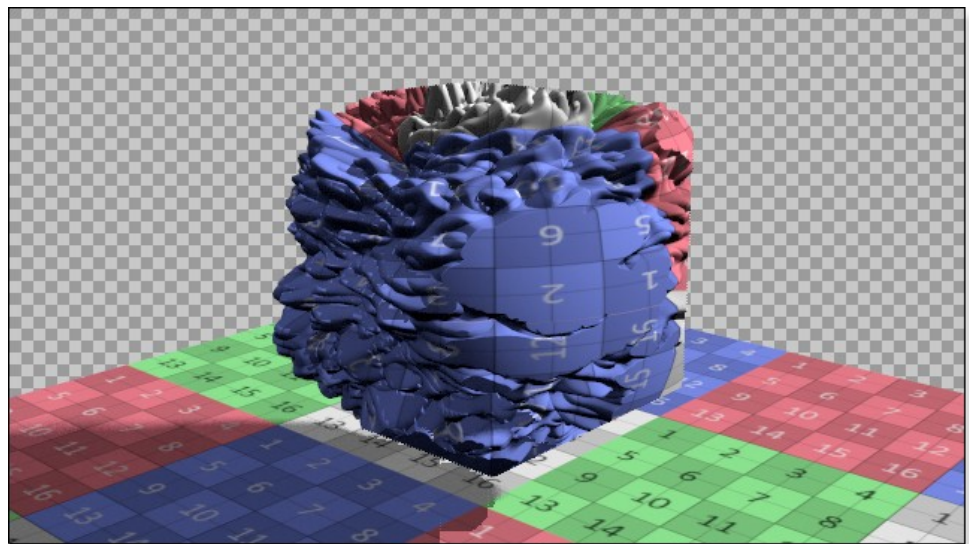
Displacement *Value* is multiplied to the value of the displacement texture. Vertices are then displaced along their normals using texture values.

Please note, displacement *Value* should be ideally relatively small when compared to the geometry size. It has an overhead on both rendering and memory and the bigger the displacement value the higher its overhead is.

Bound

Displacement *Bound* controls the bound of the displacement map. The bound should be carefully set to maximum extension of the displacement texture.

A too low value will introduce clipping whereas a too high value may slowdown considerably render times.



A wrong bound value that is too small, notice the clipping.

Working with Shading Groups

A Shading group is a named partition of primitives defined by geometries. In other packages, they are sometimes called clusters, surfaces or geometry parts.

In Clarisse, material and displacement to geometry link is achieved by assigning materials and optionally displacements to shading groups.

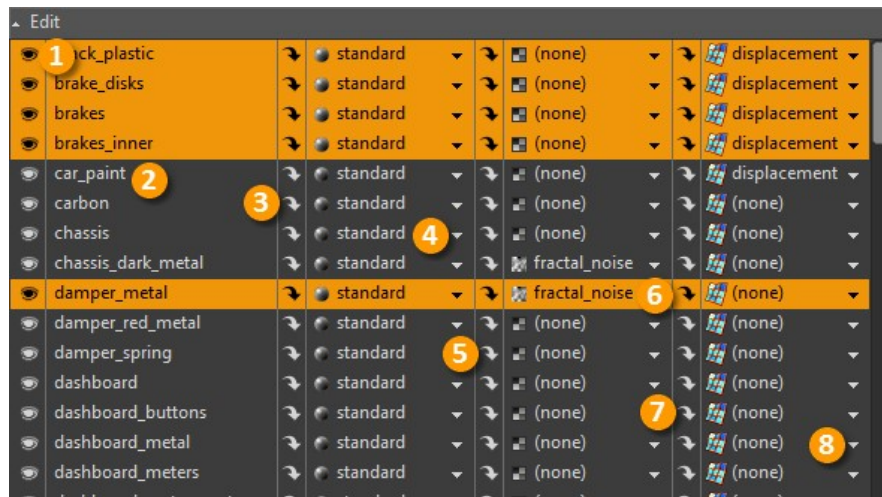
Shading groups are defined in geometries. When you import geometries, shading group definition is also imported. A shading group is a named partition of primitives. There can be many shading groups defined in a single geometry however, a primitive can only be part of a unique shading group. By default, all shading groups are referencing the static material `project://default/material`

Using the Material Linker

The Material Linker is a dedicated widget that manages geometry shading group visibility and material/displacement assignment.

Overview

You can assign materials and set visibility on multiple shading groups at the same time. The Material Linker has also been designed to work on multiple geometries at the same time. When you have multiple geometries in your selection, all their shading groups are displayed and collapsed in one if the shading group name is shared. In a few clicks, you can manage material assignment and shading group visibility of all geometries of your scenes.

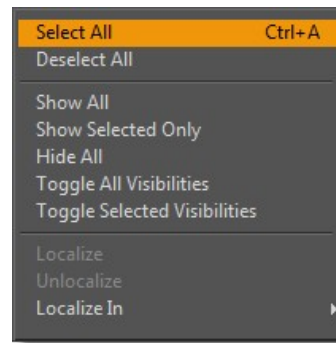


Material Linker displaying shading groups

(1) Shading Group Visibility (2) Shading Group Name (3) Select referenced material (4) Referenced material (5) Select referenced clip map texture (6) Referenced clip map texture (7) Select referenced displacement (8) Referenced displacement

Edit Menu

Shading groups are hidden attributes that can't be seen in the Attribute Editor. You need to use the Material Linker to edit Shading Group association. But as you can't use the Attribute Editor, Localize, Unlocalize and Localize In features are found in the Edit menu of the Material Linker.



Material Linker menu

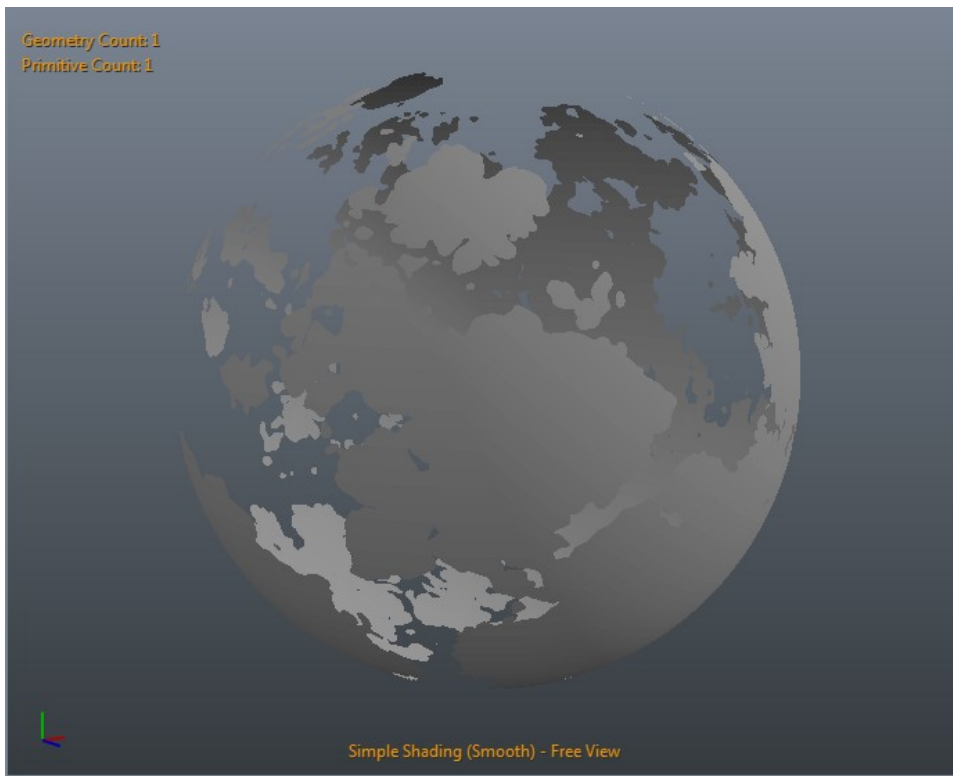
The Material Linker manages its own selection. Helpers are provided here to select all shading group, toggle visibility etc...

Assigning Material

Materials are assigned to shading groups. To assign a material, click (4) and select or create a material. You can also drag and drop a material to a shading group in the Material Linker. In the same way, you assign a material to a single shading group. Assignments can also be done on a multi selection of shading groups.

Assigning Clip Maps

A Clip map texture is a texture that clips the geometry. It behaves exactly as if a binary (on/off) transparency map was attached to its material except a Clip map renders way faster than transparency.



An implicit sphere with fractal noise texture used as clip map

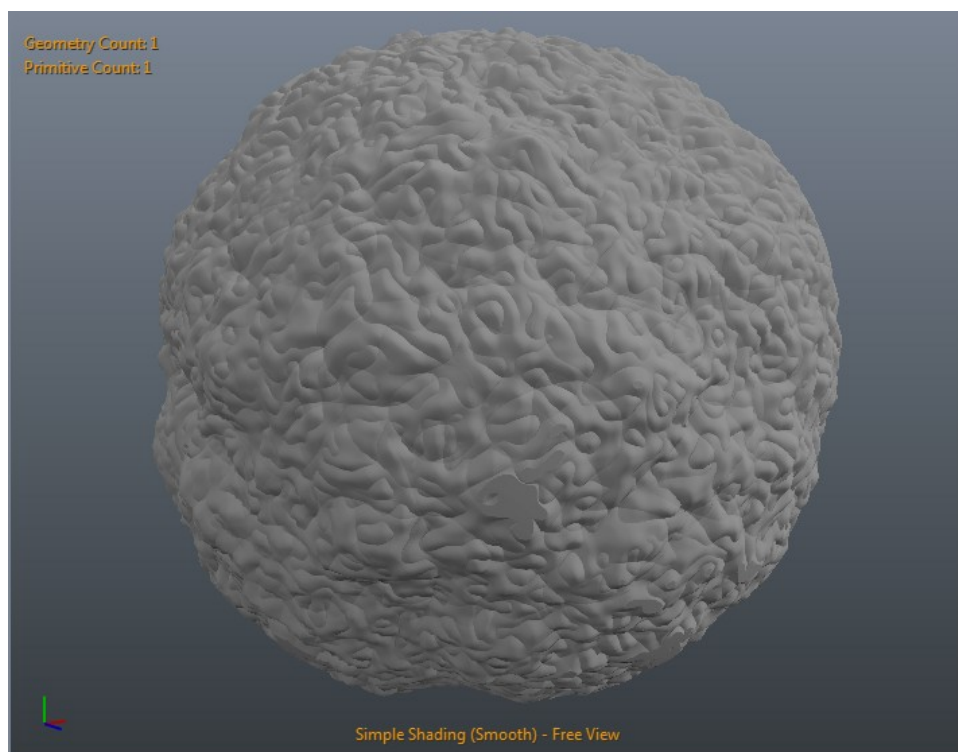
Clip maps are assigned to shading groups. To assign a clip map, click (5) and select or create a new texture. You can also drag and drop a texture to a shading group in the Material Linker. In the same way, you assign a clip map texture to a single shading group. Assignments can also be done on a multi selection of shading groups.

Note

When using clip maps you don't have to check *Enable Transparent Shadows* on lights to see proper shadows.

Assigning Displacement

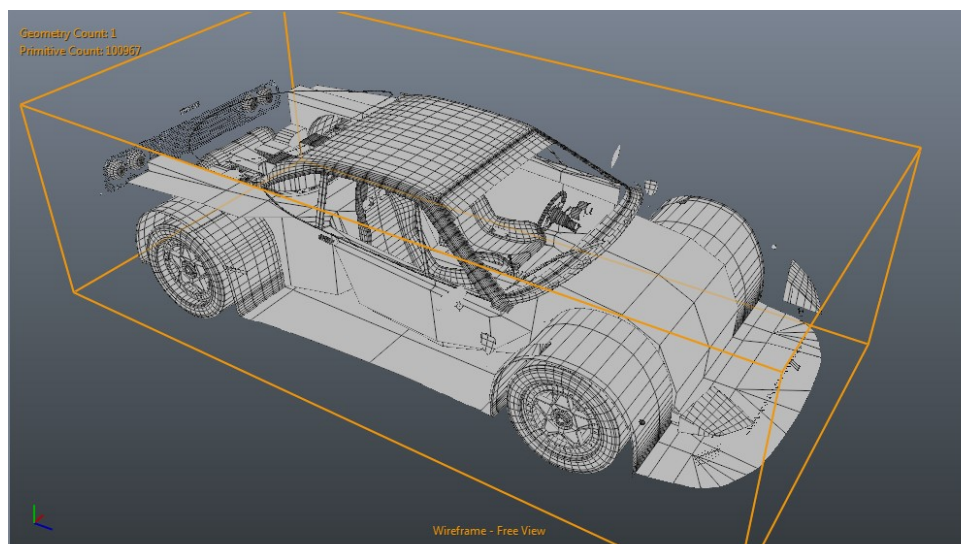
Displacements are assigned to shading groups. To assign a displacement, click (7) and select or create a displacement. You can also drag and drop a displacement to a shading group in the Material Linker. In the same way, you assign a displacement to a single shading group. Assignments can also be done on a multi selection of shading groups.



An implicit sphere with a fractal noise as displacement

Shading Group Visibility

You can toggle on and off shading group visibility. This is very useful if you wish to hide geometry parts. To toggle visibility on selected shading groups just click (1).



Hiding shading group visibility

Note

Hiding shading groups isn't most efficient for rendering speed.

Localizing Shading Groups

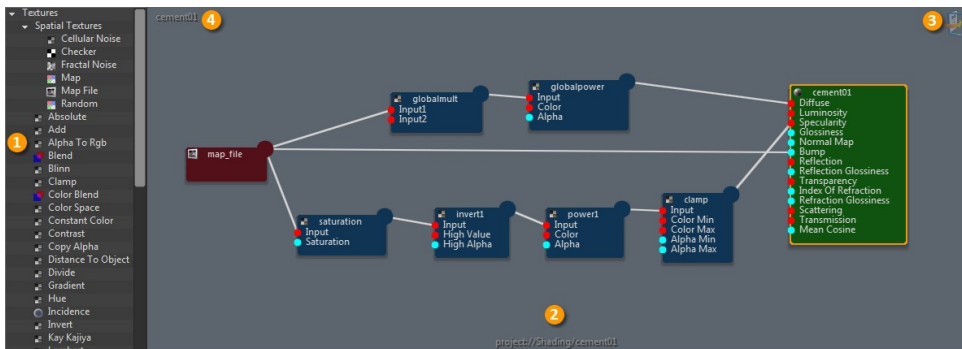
As for any attribute, you can localize shading groups. However, shading groups can't individually be localized. When you localize shading groups of a geometry instance, they all get localized. Localized or unlocalized shading groups are displayed following the Attribute editor convention: **bold** for localized attributes and *italic* for unlocalized ones.

Using the Material Editor

The Material Editor lets you connect textures into a network of nodes to create visually rich material and displacement shaders. You can display multiple materials and displacements at the same time and share textures between materials and displacements. Texture nodes have single output that can be connected to multiple inputs.

Overview

The Material Editor is divided in two areas. The texture tree (1) where you choose new texture nodes to create and the nodal view where you connect nodes (2).



(1) Texture Tree (2) Material Workspace (3) Exit Nodes (4) Displayed Material Name

Navigation Basics

Panning

To pan press Alt or Space and click in the nodal view using either left or middle mouse button. Drag in any direction to pan the view.

Zooming

To zoom the view, press Alt or Space and right click in the nodal view. Drag left or up to zoom out and right or down to zoom in. You can also use the mouse wheel to perform zooming.

Note

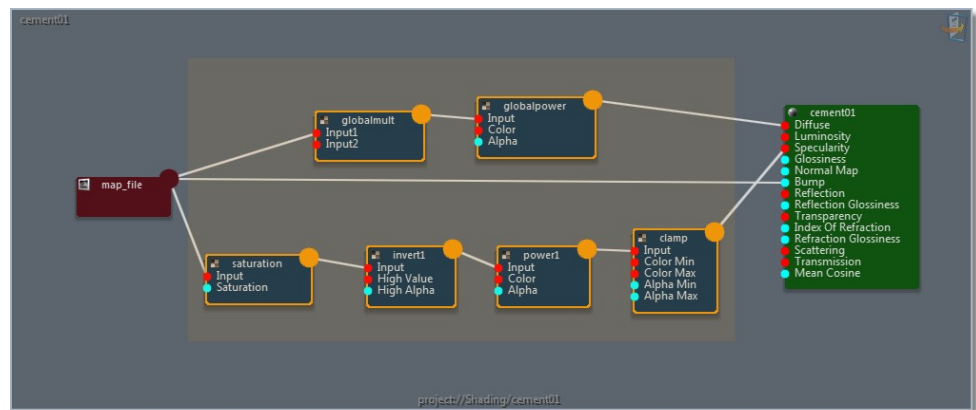
The maximum zoom value is locked to 100%.

Fitting

Press F to fit the view to the selected nodes. If no node is selected, F fits the view to all visible nodes.

Selecting Nodes

To select nodes just click on them. You can use Shift and Ctrl to add/remove nodes to selection. To deselect all click on an empty area of the nodal view. You can also perform rectangle selection. Click on an empty area and drag the mouse to draw the rectangle selection. Release mouse button to select nodes. Only nodes entirely within the rectangle are selected.

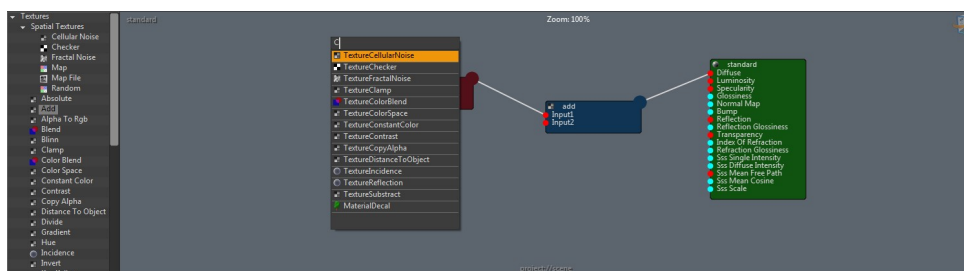


Tip

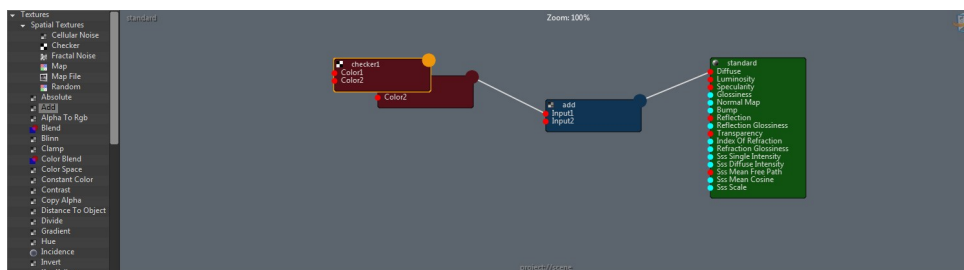
You can perform drag and drop from the Material Editor to any widget. Select your node(s), press **CTRL** and drag and drop your selection to any other widget.

Adding new Nodes

To add new nodes, select the texture or material you want to create in (1) and drag and drop it in (2). You can also press Home key and type the name of the class you're looking for:

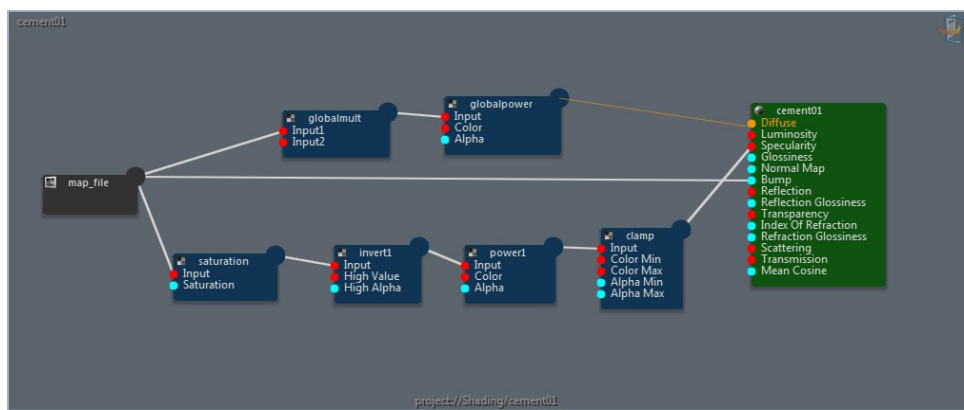


Press return to validate and create your node.



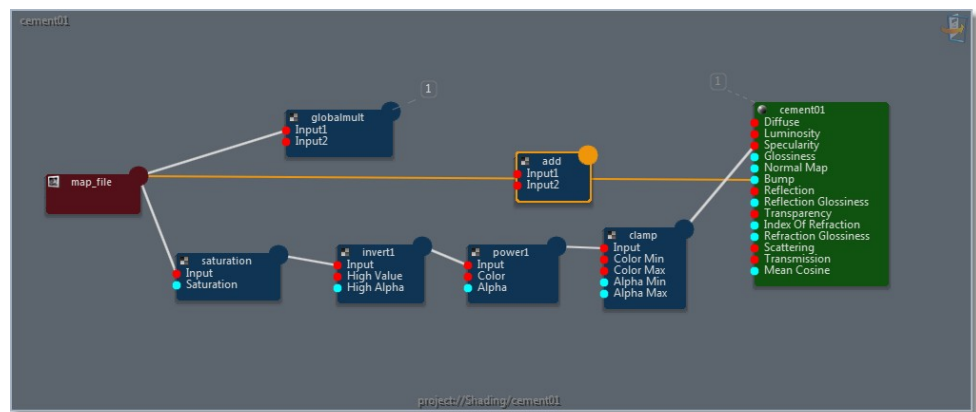
Connecting Nodes

To connect nodes, click and drag the output circle of a node and plug it to input circle of a node.

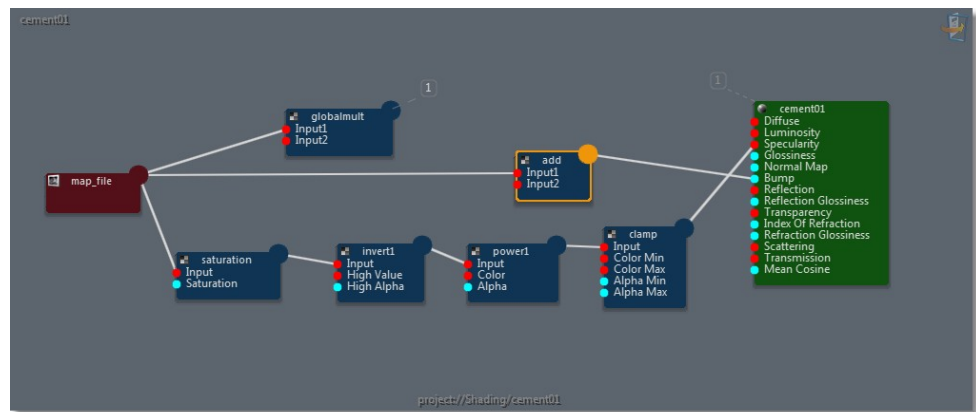


Inserting Nodes

You can insert nodes on existing connections. To insert a node between two, select and drag the node over a link.

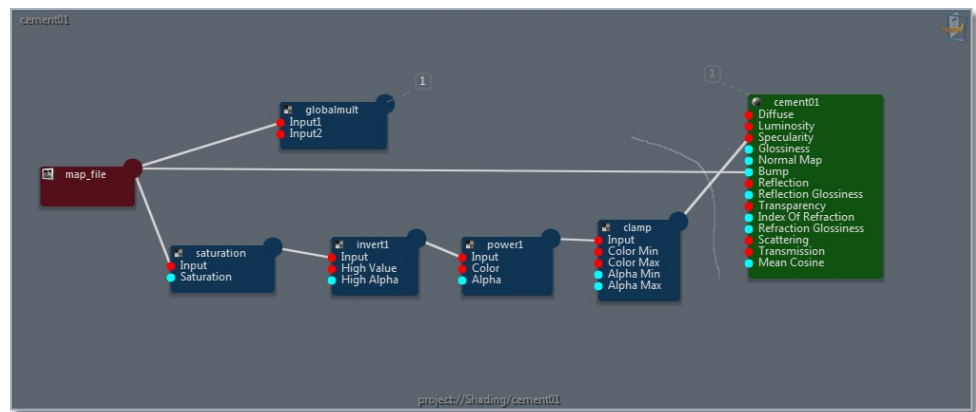


When the link is highlighted, drop the node to insert it.

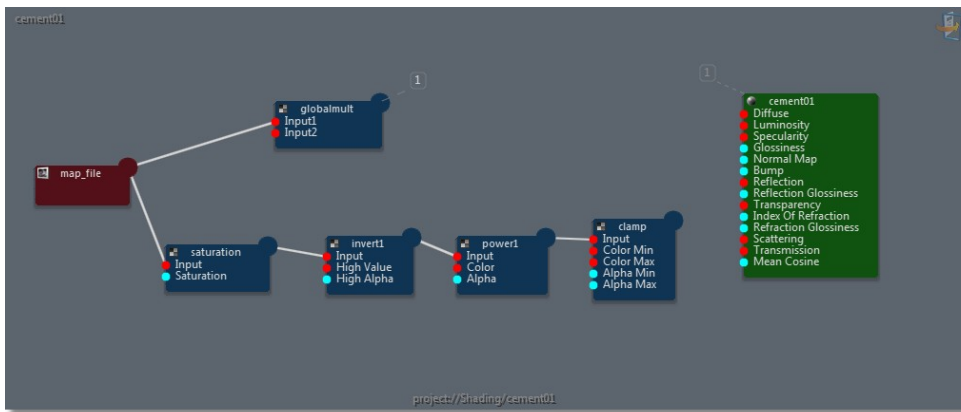


Disconnecting Nodes

To disconnect a node, simply click on an input circle and drag and drop to unplug. You can also disconnect multiple connection at the same time by right clicking and dragging to draw a path.



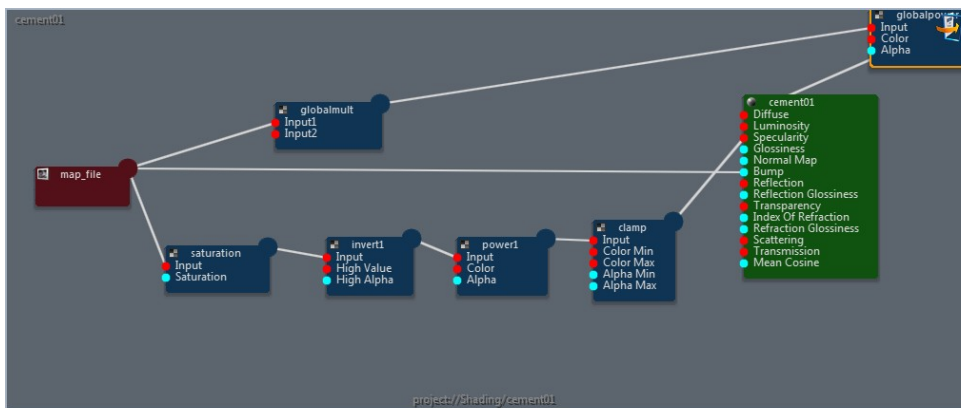
Each time the path intersects links, they get disconnected.



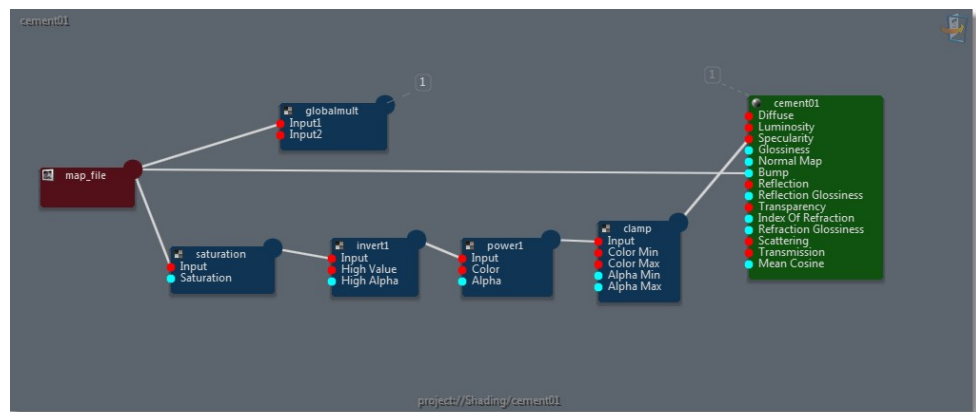
Removing Nodes

You can display and edit multiple materials and consequently multiple networks at the same time. You can remove nodes from the Material Editor by drag and dropping node(s) into (3). Nodes are then removed from the nodal view, they are not deleted. You'll be able to bring them back in the view later on.

When a node is dragged to the Exit node icon, the icons is highlighted:



Drop the node to remove it from the view:

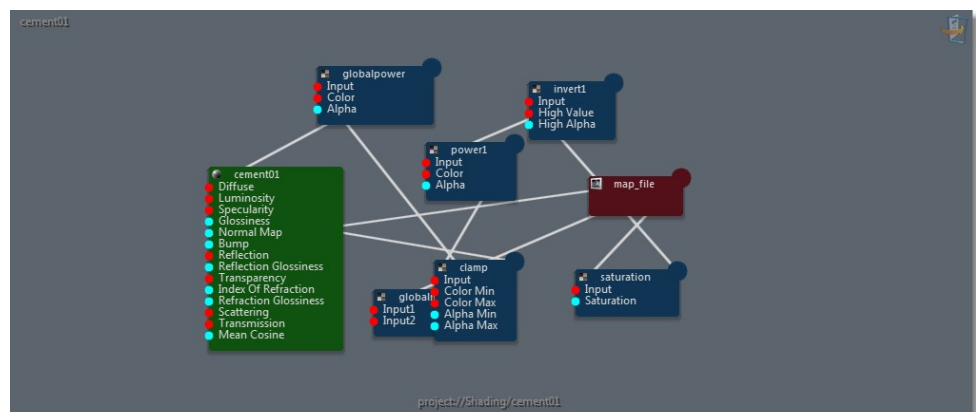


The selected node is now removed from the view. Notice the two numbers displayed coming out globalmult and coming in cement01. This visual hint means globalmult has one external output and cement01 has one external input (basically the node we've just removed). Double click on the visual hint to bring the external node.

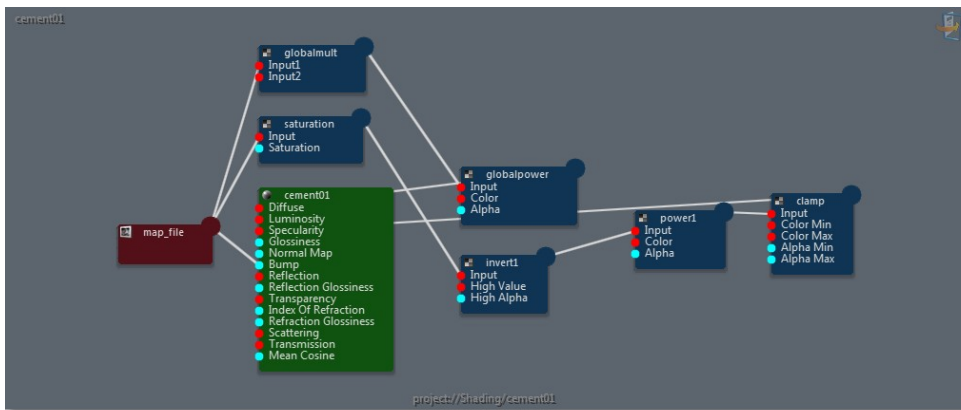
Auto Layout

The material editor provides a simple auto layout feature working on a selection of nodes. To auto layout nodes, press L on a selection of nodes. If you press L without selecting nodes it will affect all visible nodes.

A material graph before auto layout:



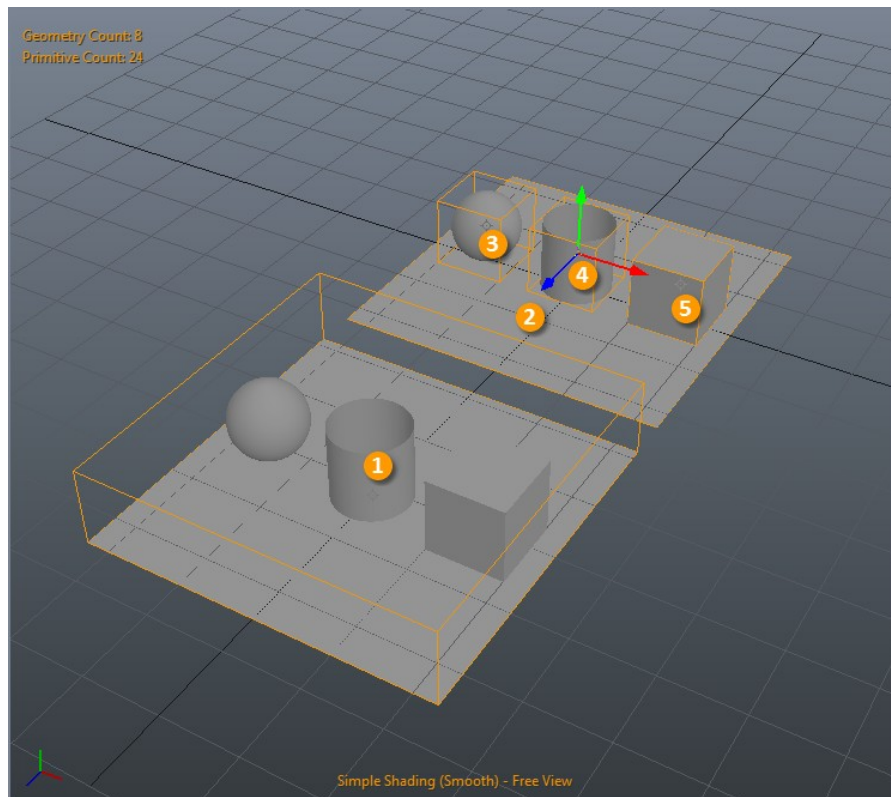
Same graph after auto layout:



Combiners

A Combiner allows you to combine multiple scene objects (that can be animated) into a new one. The combiner is basically a list of referenced scene objects that can be moved in space. Referencing is dynamic and the list can be edited at anytime. Deleting a scene object referenced in combiners removes this object from the referencing combiners. Combiners are really powerful as they can be used to create super sets.

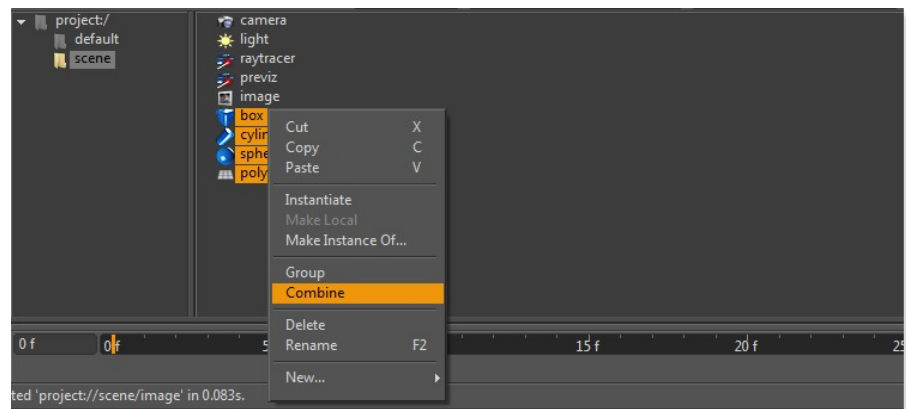
Combiners can be combined in combiners.



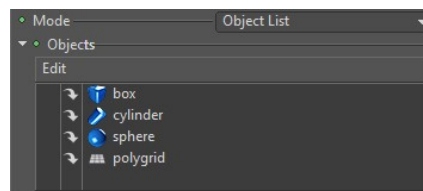
A Combiner (1) combining 4 scene objects (2) (3) (4) (5)

Combining Scene Objects

To combine selected scene objects, go to **Edit > Combine** to create a new combiner combining the selected scene objects. You can also use the right click in a Browser and select **Combine** in the popup menu.



Or finally, you create a new empty combiner, **Create > Combiner** and drag and drop scene objects in the *Objects* attribute list in the Attribute Editor.



Mode

Combiners have two different modes:

- Object List mode allows you to explicitly combine scene objects.
- Image List mode allows you to explicitly combine contents of images. The combiner implicitly references scene objects that are referenced by referenced images.
- Group List mode allows you to explicitly combine groups and their hierarchy.

Scatterers

The Scatterer is a really powerful scene object allowing you to scatter tons of scene objects from input Particles and input scene objects. Scatterers also support scattering of Combiners and scatterers. You can also access very finely

to instances during texturing using the texture Instance Color. Please refer to Instance Color section for more information.

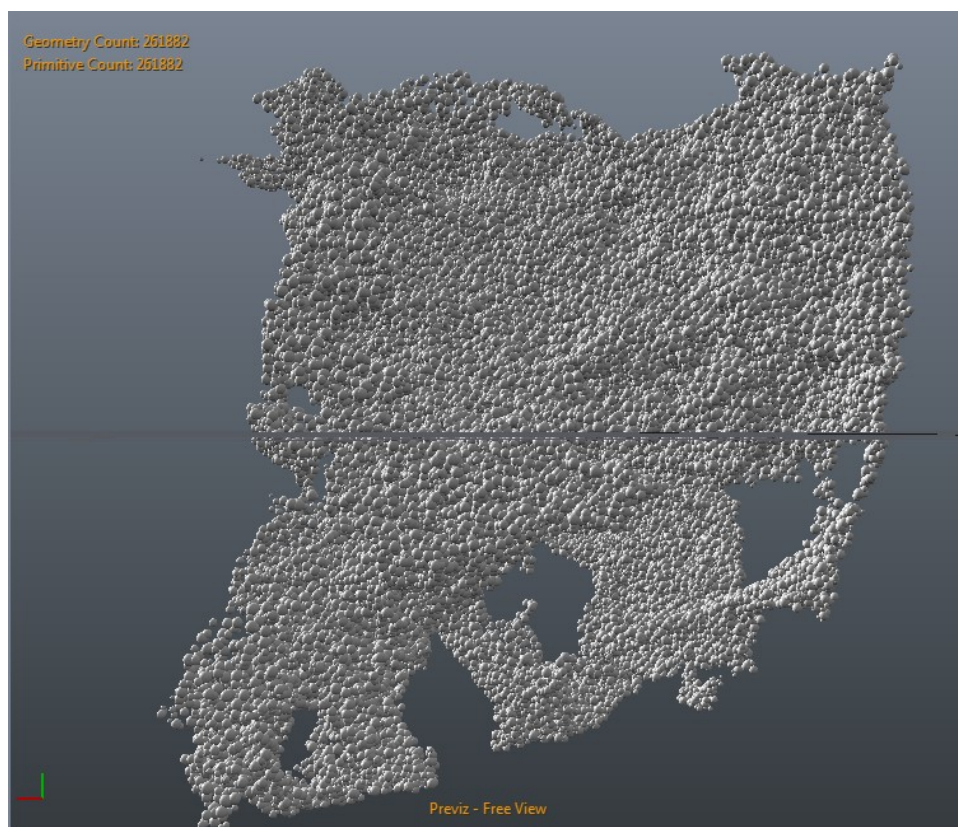


Forest created using scatterers, combiners and a few geometries.

Attribute	Description
Geometry Support	Sets the geometry the scatterer will use as particles.
Geometry	List of scene objects being scattered on particles. When multiple items are set in the list, they will be randomly picked.
Scatter Position	Position offset of all scattered item. The position offset of the scattered item is relative to the particle it is scattered on.
Scatter Position Variance	Position offset variance between scattered items.
Scatter Rotation	Rotation of all scattered item.
Scatter Rotation Variance	Rotation variance between scattered items.
Scatter Rotation Variance Step	Rotation step variance. Rotation values will be a multiple of the input step.
Scatter Scale	Scale of all scattered item.
Scatter Scale Variance	Scale variance between scattered items.
Uniform Scale Variance	Sets if the scale variance is applied uniformly through the items (conserve proportions)

Geometry Support

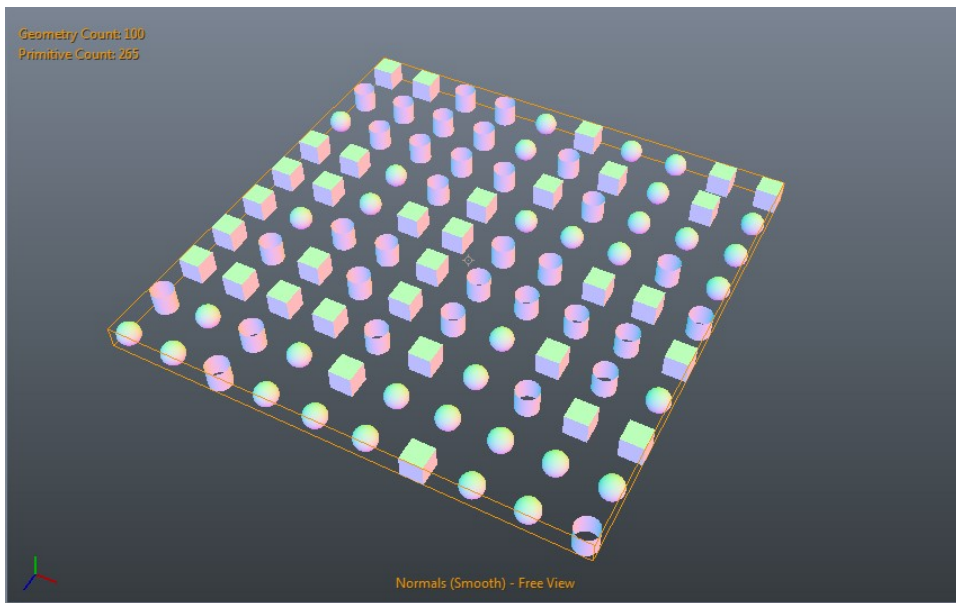
The scatterer extracts referenced geometry vertices as particles to scatter items. Geometry Support can be any geometry that defines points (Particles, Polymesh...)



Textured point volume used as geometry support to scatter spheres.

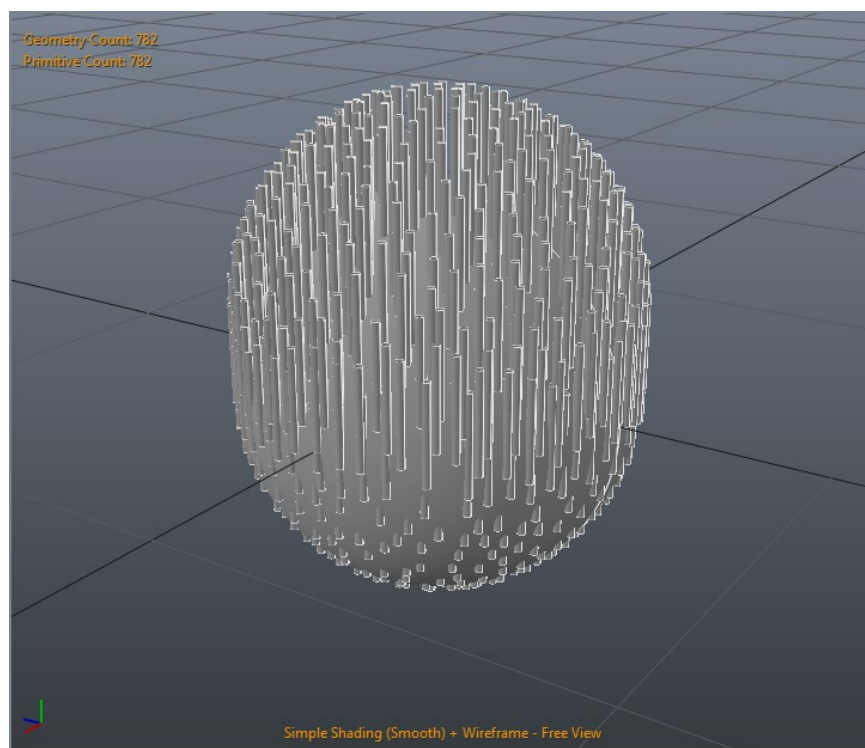
Geometry

You can set multiple scene objects to be scattered. When specifying multiple scene objects, the scatterer picks randomly an item to be scattered.

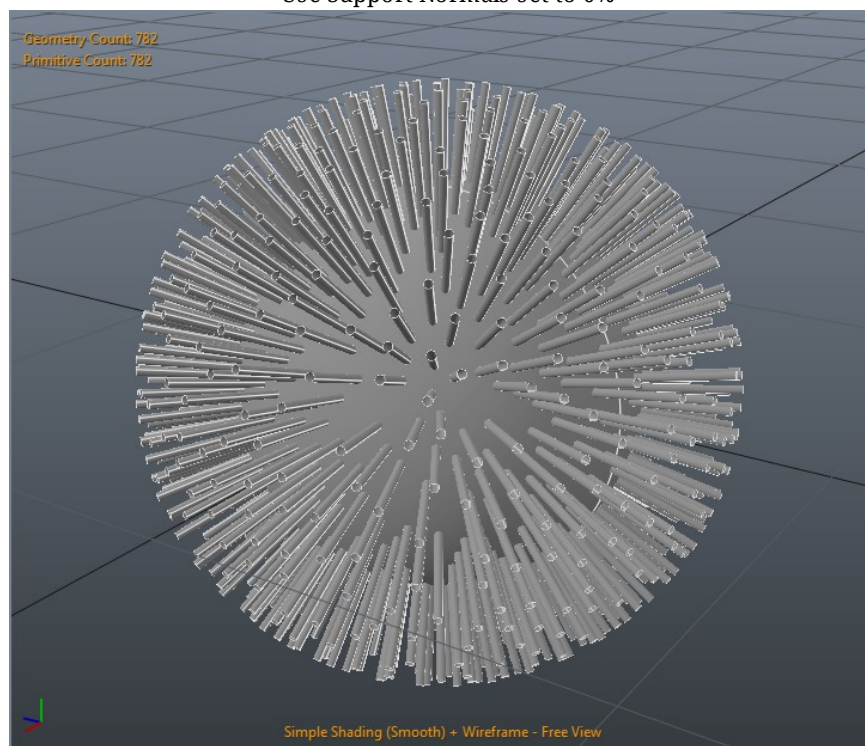


Use Support Normals

This attribute allows you to align scattered geometries according to the underlying geometry point cloud when applicable*.



Use Support Normals set to 0%

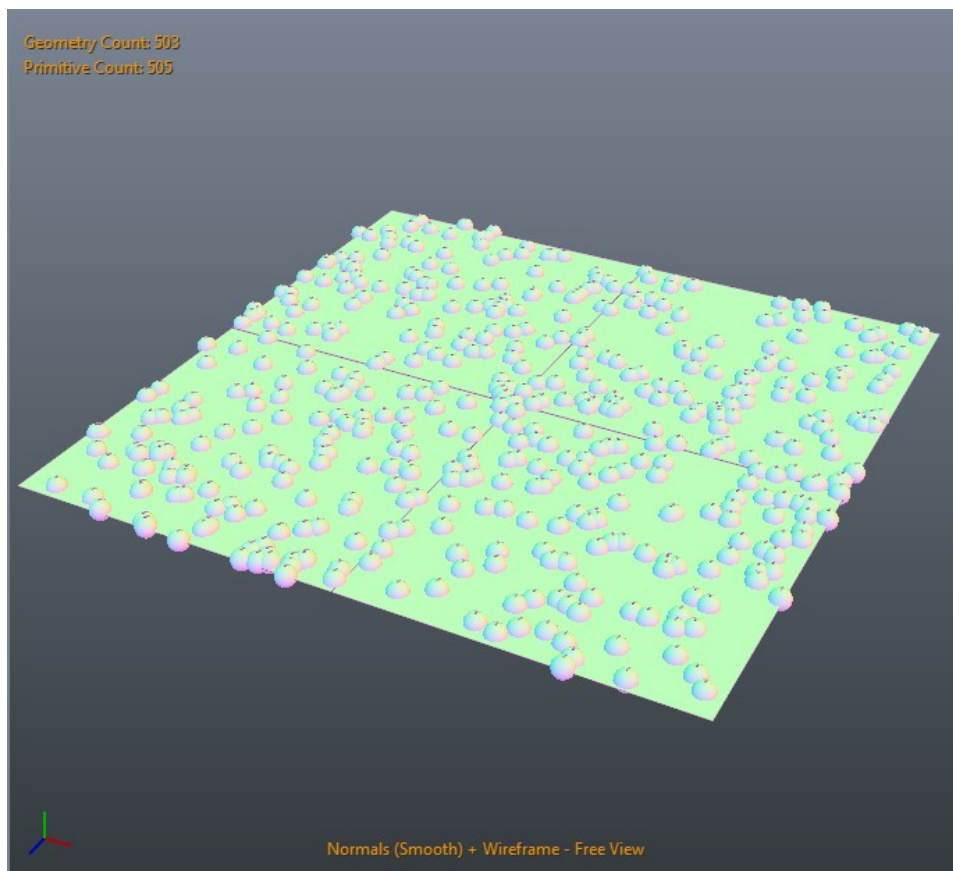


Use Support Normals at 100%

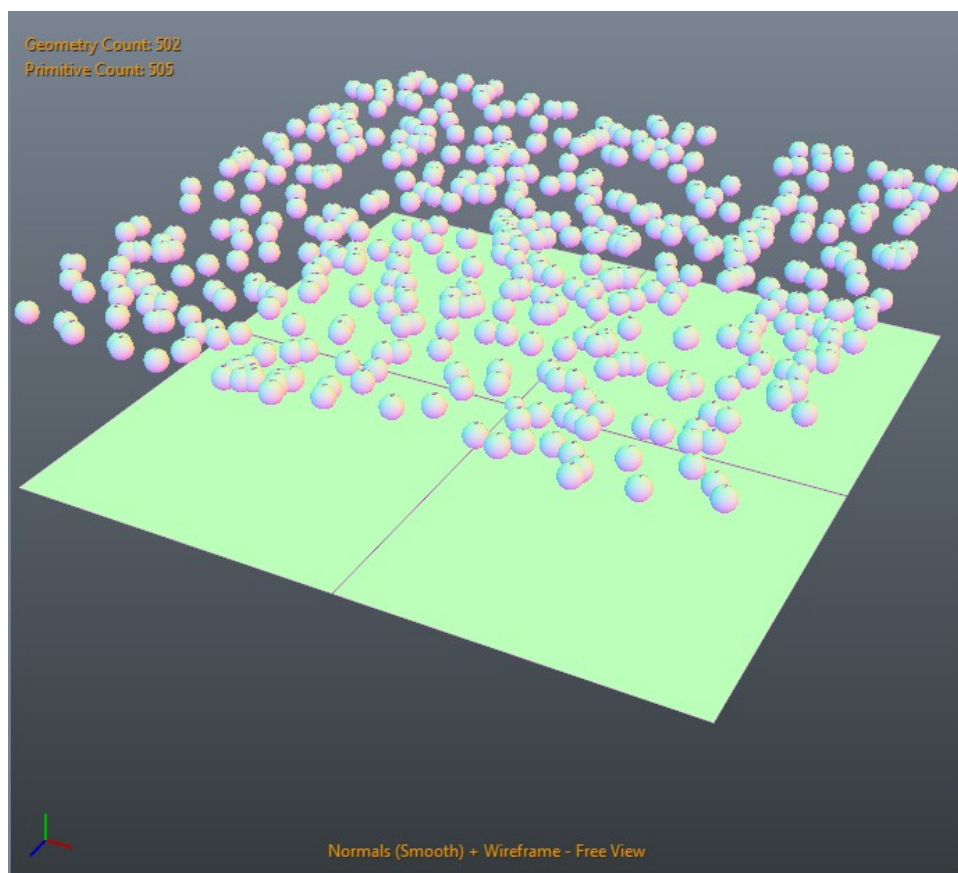
* Point clouds can originate from many different ways. If the input point cloud originates from a Point Array, then point normals are aligned with the world Y axis.

Scatter Position

Scatter Position allows you to offset the position of scattered geometries. Each scattered geometry will be offsetted by the specified *Scatter Position* value. *Scatter Position Variance* adds randomness to the offset each scattered instance will have a random offset comprised between 0 and the specified value.

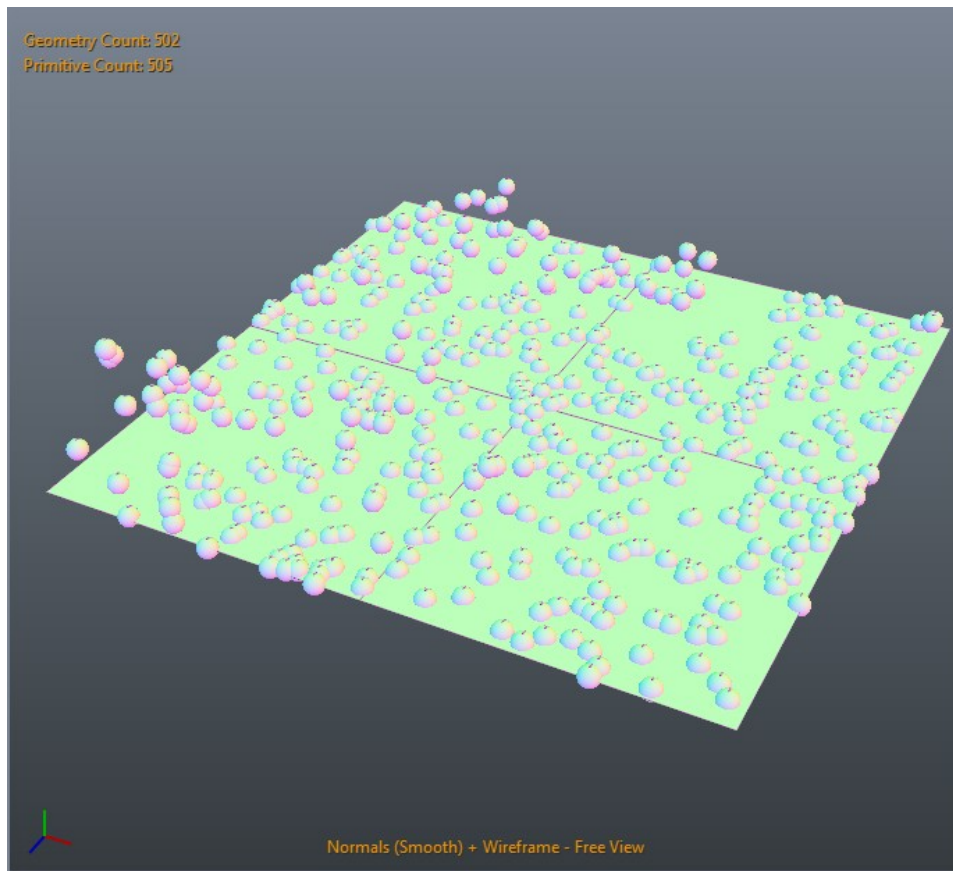


No offset



Offset in Y axis

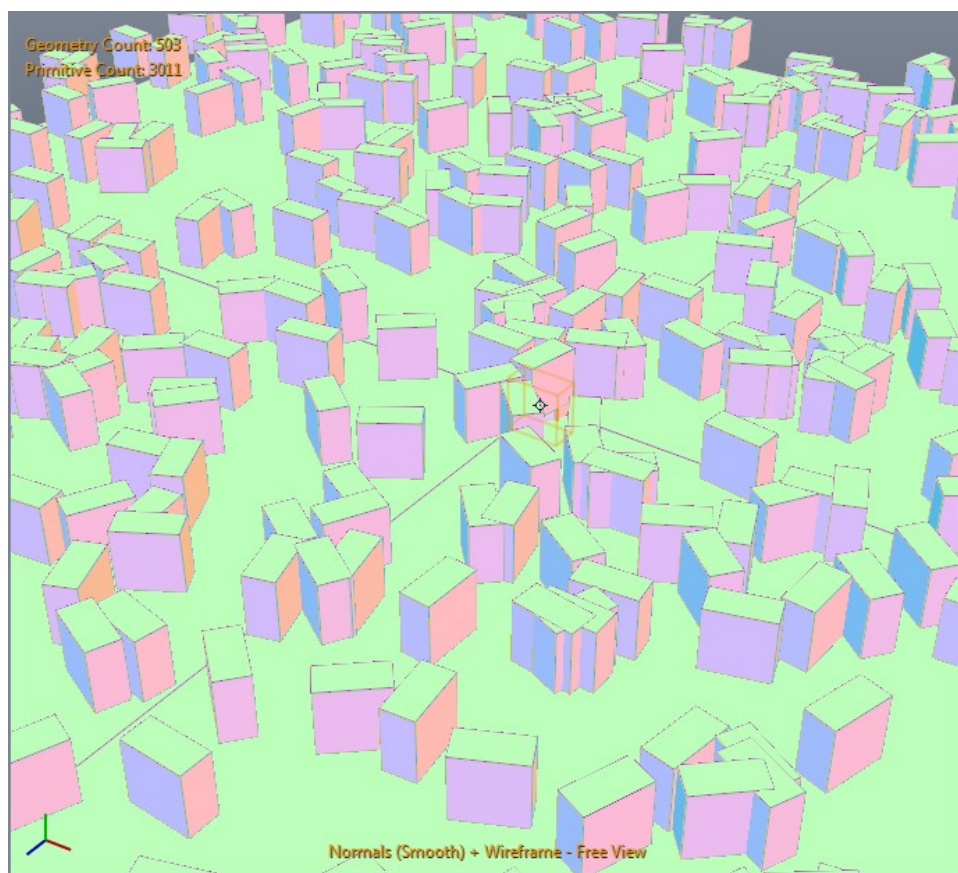
Both attributes can be textured. In that case, the final value is multiplied by the texture value evaluated at the sample point (R for X, G for Y, B for Z).



Same offset but driven by a texture

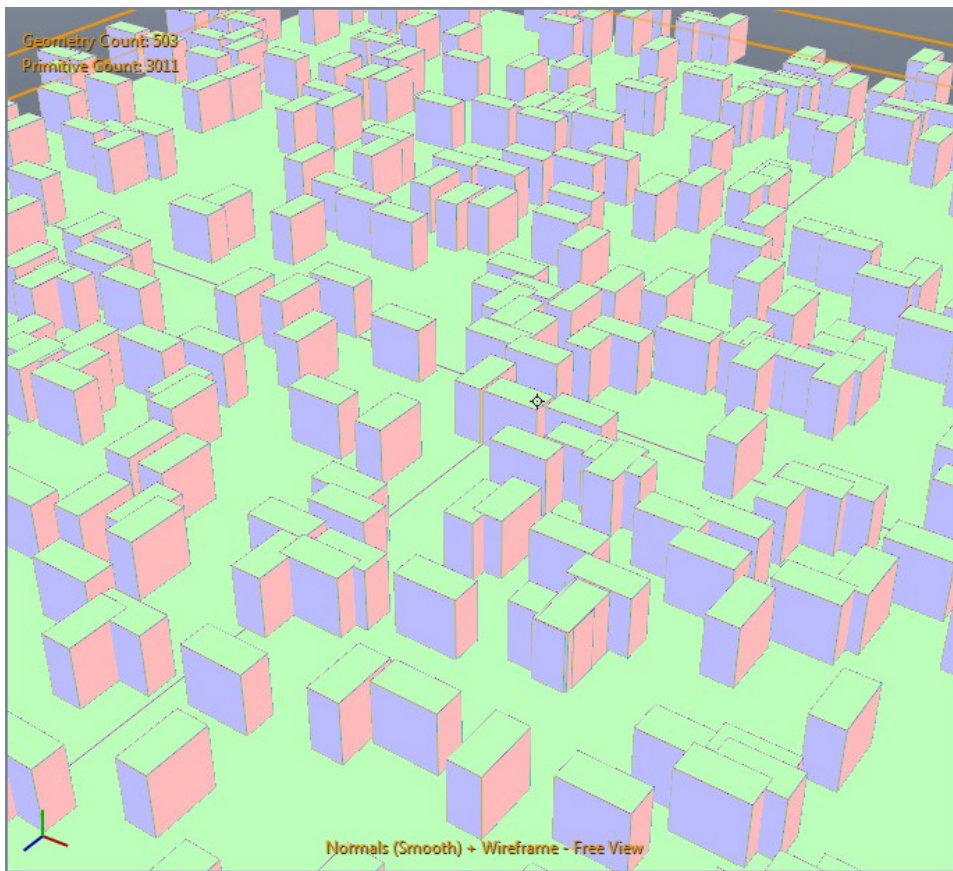
Scatter Rotation

Scatter Rotation allows you to rotate scattered geometries. Each scattered geometry will be rotated by the specified *Scatter Rotation* value. *Scatter Rotation Variance* adds randomness to the rotation on each scattered instance. Each instance will have a random rotation comprised between 0 and the specified value.



Boxes scattered using a rotation variance of (0, 360, 0)

Scatter Rotation Variance Step constraints the rotation to be a multiple of the specified value. For example, you would set 90 degrees if you were to scatter buildings.

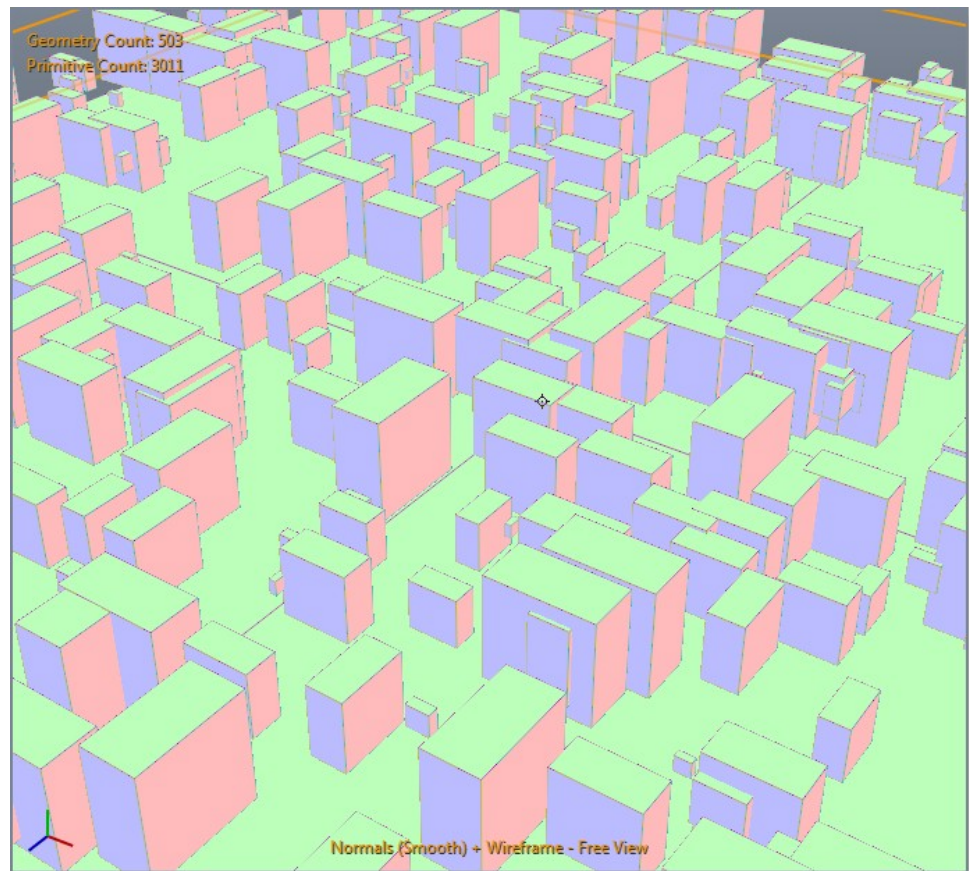


Same scatterer but with a Rotation Variance Step of 90 degrees

All those attributes can be textured. In that case, the final value is multiplied by the texture value evaluated at the sample point (R for X, G for Y, B for Z).

Scatter Scale

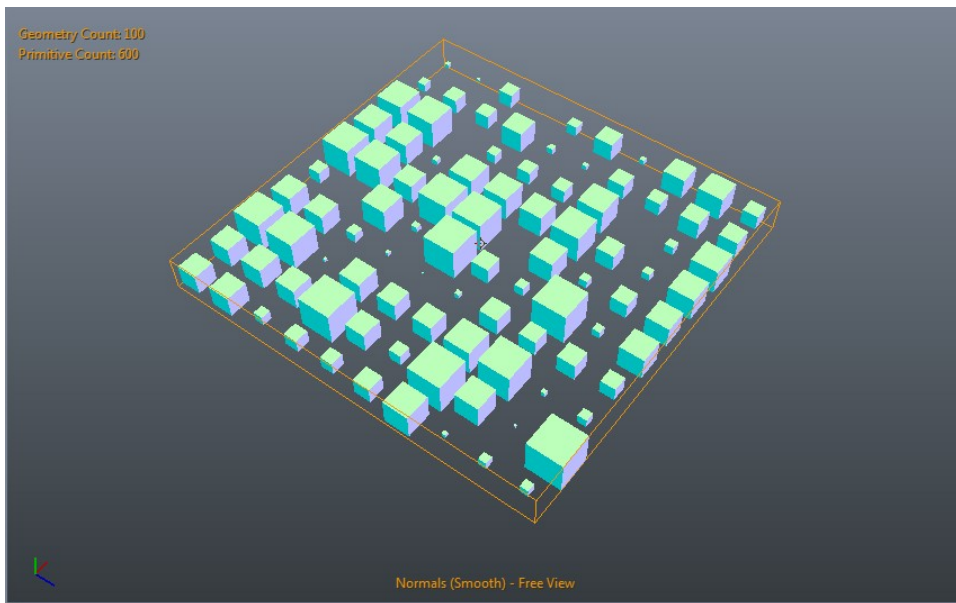
Scatter Scale allows you to control the scale of scattered geometries. Each scattered geometry will be scaled by the specified *Scatter Scale* value. *Scatter Scale Variance* adds randomness to the scale on each scattered instance. Each instance will have a random scale comprised between 0 and the specified value.



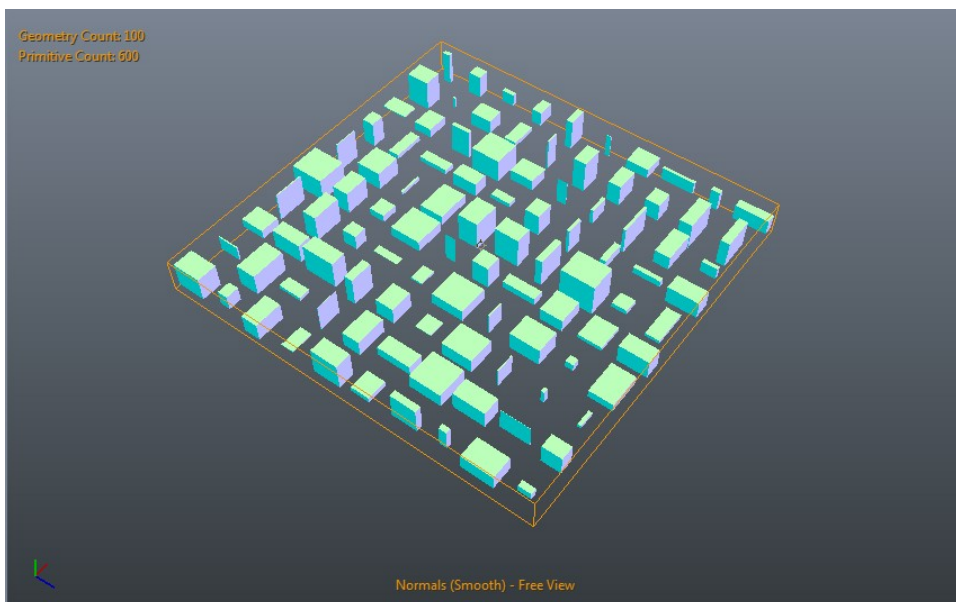
All those attributes can be textured. In that case, the final value is multiplied by the texture value evaluated at the sample point (R for X, G for Y, B for Z).

Uniform Scale Variance

By enabling this attribute, only random value is generated for all axis. If the variance is value is identical in all 3 axis the scattered scene objects will conserve original proportions.



With uniform scale variance



Without uniform scale variance

Lighting and Rendering

In this chapter we will go through all the essential elements to help you light and render 3d scenes. In Clarisse 3d scenes are 3d layers. To render a 3d scene you'll need an image with a 3d layer in it. For more information on images please refer to Working with Images.

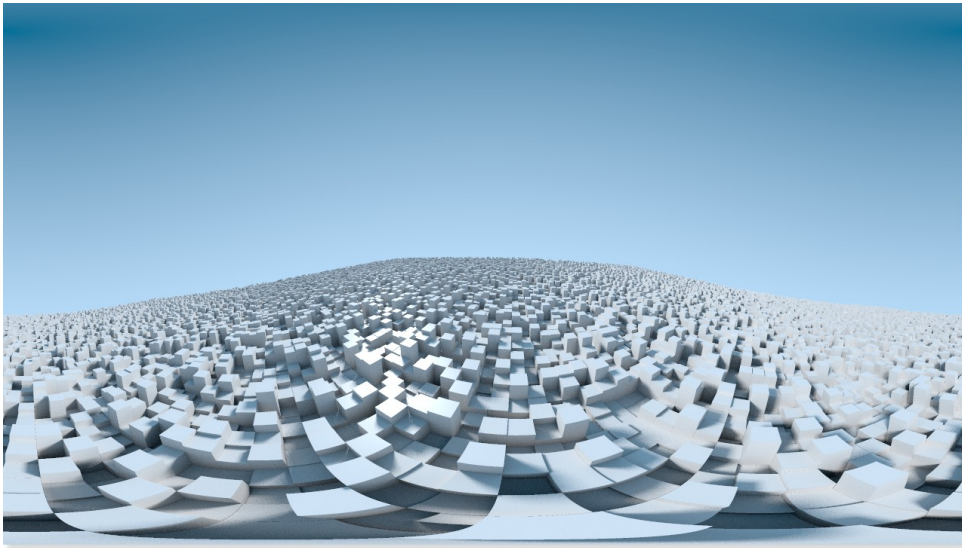
What's a Renderer?

The renderer computes a 2d image from scene objects, lights, materials and a projection (viewpoint) defined by a camera. Currently Clarisse provides a single class of renderer: Raytracer. However it may be extended in the future to provide for example third party renderers.

Raytracer

Clarisse raytracer is based on an advanced and highly optimized CPU-based raytracing engine. As a matter of fact, the software based 3d View is actually using this very same engine. The offline raytracer which supports interactive rendering, is both a path tracer and a ray tracer.

To render an image, the raytracer divides the image into buckets or tiles. On each tile, the raytracer casts rays from the viewpoint to compute the visibility. Ray origins and directions are defined by the camera projection referenced by the 3d layer. The raytracer supports arbitrary types of projections, for example you can use a Panoramic camera to create HDR spherical maps. For more information on projections please refer to Setting the Viewpoint.



Panoramic camera (360 degrees)

Attribute	Description
Anti Aliasing Samples	Sets the number of samples used to perform super sampling.
Shading Oversampling	Controls the number of shading sample used for each sub sample.
Alpha Threshold	Sets the alpha threshold before a material is considered opaque.
Alpha Depth	Sets the mode of transparency depth.
Custom Alpha Depth	Sets the number of transparent layers before the material is considered as opaque. Enabled if Alpha Depth is set to custom.
Enable Reflection	Sets if reflections are performed during shading.
Reflection Depth	Sets the mode of the reflection depth.
Custom Reflection Depth	Sets the maximum number of consecutive reflection bounces during a material evaluation. Enabled if Reflection Depth is set to custom.
Refraction Depth	Sets the mode of the refraction depth.
Custom Refraction Depth	Sets the maximum number of consecutive refraction bounces during a material evaluation. Enabled if Refraction Depth is set to custom.
Enable Motion Blur	Sets if motion blur is enabled.
Previz Mode	If enabled, disables current lighting for a simplified one.
Disable Shading	If enabled, no lights and material are evaluated. The renderer outputs a normal map like image.

Anti Aliasing Samples

Anti aliasing is a technique to eliminate jaggies and pixelated edges. The raytracer performs anti aliasing by super sampling the 3d scene. It basically divides the pixel into sub regions defining sub pixels, then casts a ray from each sub region. This simple technique leads to a smoother image.

You should be aware the higher the anti aliasing value is, the slower rendering gets. By default, the raytracer doesn't perform any anti aliasing. To enable anti aliasing just increase the number of samples. The actual number of samples used to perform anti aliasing is the specified value squared. For example, for a value of 3, the number of samples used for anti aliasing is $3 \times 3 = 9$ samples. A value of 5 will cast 25 samples...

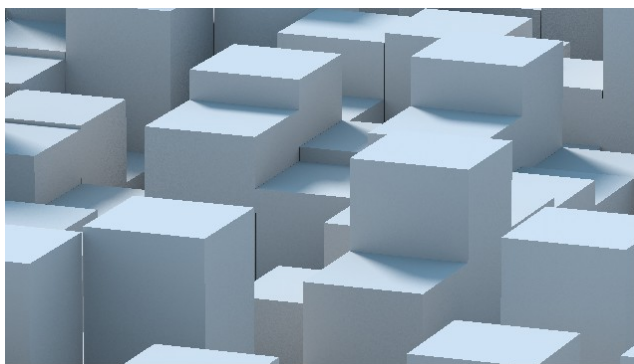
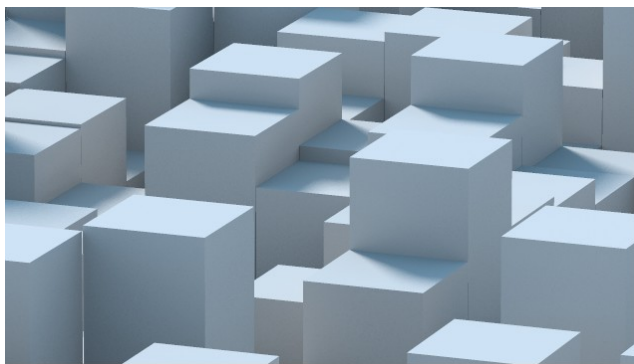


Image rendered without anti aliasing



Same image rendered with anti aliasing

Anti Aliasing Filtering

Anti aliasing filtering is applied when the renderer integrates anti aliasing samples to create the final pixel color. Anti Aliasing Filtering can greatly increase image quality, specially when dealing with fur an tiny objects. Two attributes controls filtering: *Anti Aliasing Filter* and *Anti Aliasing Filter Size*.

Filter

The *Raytracer* renderer provides several anti aliasing filters.

Mode	Description
Box	Each sample has the same weight. This is the default settings. Recommended filter size is 1.0 px, 1.0 px.
Triangle	Sub sample weights decrease linearly according to their distance to the pixel center. Minimum recommended filter size is 2.0 px, 2.0 px.
Gaussian	Apply a Gaussian curve to the sub according to their distance to the pixel center. Minimum recommended filter size is 3.0 px, 3.0 px.
Blackman-Harris	Apply a Blackman Harris curve to the sub according to their distance to the pixel center. This filter is highly recommended. Minimum recommended filter size is 3.0 px, 3.0 px.
Mitchell	Apply a Mitchell curve to the sub according to their distance to the pixel center. This filter is sharper than Gaussian but it has negative lobes that may introduce artifacts with HDR values . Minimum recommended filter size is 4.0 px, 4.0 px.
Lanczos	Apply a Lanczos curve to the sub according to their distance to the pixel center. This filter is quite good: a good balance between smoothness and sharpness, however similarly to Mitchell, it has negative lobes that may introduce artifacts with HDR values . Minimum recommended filter size is 2.0 px, 2.0 px.

Filter Size

Filter size defines (x,y) diameter of the filter used during anti aliasing. This value highly depends of the filter set in *Anti Aliasing Filter*. For example a *Gaussian* filter set to 1.0 px will decrease greatly image quality. A *Gaussian* set to 10.0 px will blur the image and greatly slowdown rendering. For recommended filter sizes please see Anti Aliasing Filters.

Note

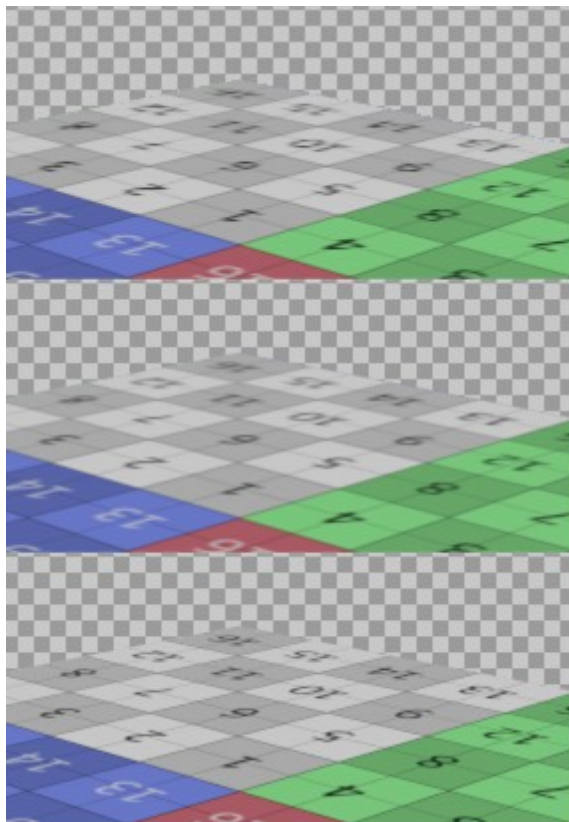
If you don't know what you are doing, a poorly set filter size can lead to a noisier image and slower rendering times. In this case, leave filter sizes to their default values.

Anti Aliasing Oversampling

When enabled, at least 4 samples are used for final pixel interpolation. This may improve image quality but it leads to slightly slower rendering. If enabled, each bucket needs to render an extra row and column of samples.

Subsample Quality

Subsample Quality controls the quality of texture filtering during anti aliasing. It basically controls the size of ray differentials for each camera sample. By default, the value is set to 0%. This means the size of each sample is about a pixel. At 100% the pixel size is about the size of the actual sub-pixel. The higher the *Subsample Quality* value the crisper the texture filtering.



(top) No Anti aliasing (AA)
(center) AA and Subsample Quality at 0%
(bottom) AA and Subsample Quality at 100%

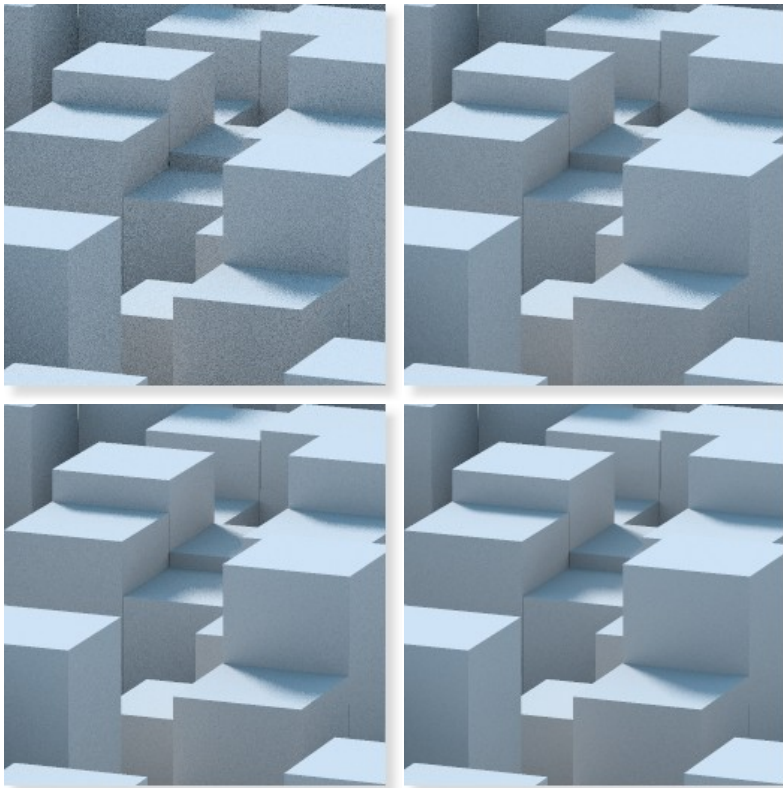
As details are crisper, high level of *Subsample Quality* can lead to two different issues:

- Texture streaming cache performance can get impacted. If it's the case, make sure to raise Texture Cache Size to reduce performance drops.
- Flickering may occur.

Shading Oversampling

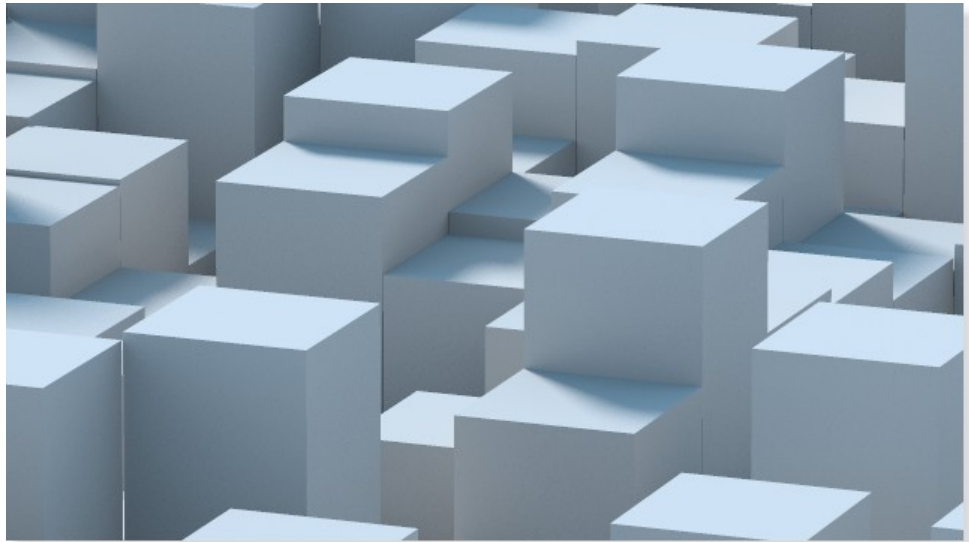
Unlike most raytracers, Clarisse decouples material and light sampling from visibility (or geometric) anti aliasing. This feature is really powerful as increasing anti aliasing does not implicitly increase the number of samples the

user has set for its lights and materials. You are then free to set the number of samples necessary to remove noise without thinking about slow render times when anti aliasing or motion blur are on. However, you might want to globally increase material and light sampling. In this case, you can control sampling quality during anti aliasing by modifying *Shading Oversampling* attribute.

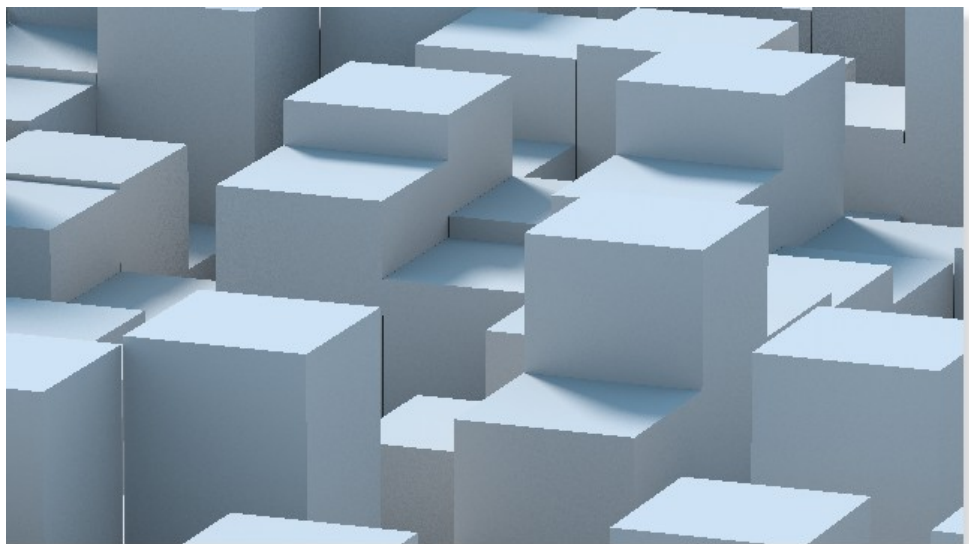


From top left to bottom right, shading oversampling at 0%, 25%, 50%, 100%

By default, the 0% *Shading Oversampling* value means anti aliasing doesn't increase material and light sampling. At 100% you'll have what you get in most raytracers. In the previous screenshots we deliberately under sampled light sampling quality to illustrate *Shading Oversampling* effect. In the next series, see how sampling quality is identical when the image is rendered with and without anti aliasing.



AA on with shading oversampling set to 0%

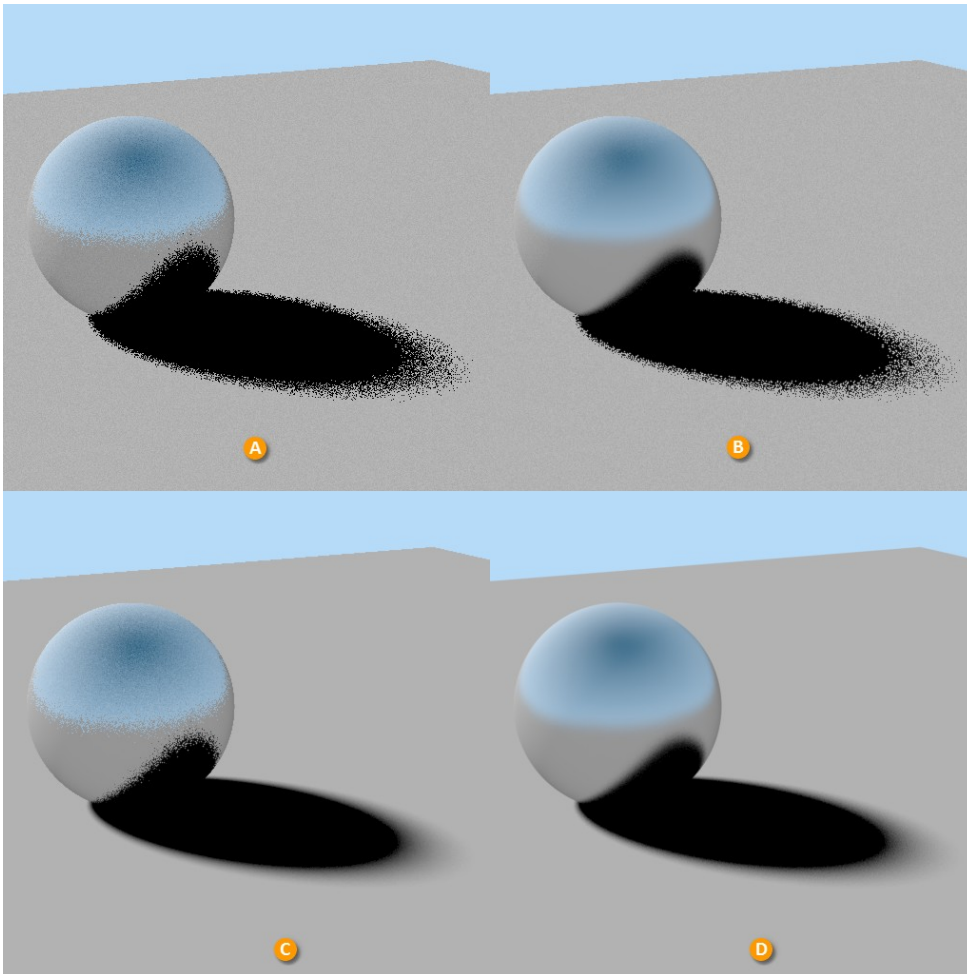


No anti aliasing

You are no longer obliged to adjust the number of samples in your lights and materials when increasing anti aliasing. Please note there's a minimum of 1 shading sample per anti-aliasing sample. If you have 64 AA rays, you'll have a minimum of 64 shading sample (materials and lights).

However, as you can see in the following example, those extra samples are put into good use.

In (A) glossy *Reflection Quality* and shadow *Quality* are set to 1. In (B) glossy *Reflection Quality* is set to 8 and shadow *Quality* to 1. In (C) glossy *Reflection Quality* is set to 1 and shadow *Quality* to 8. If *Anti Aliasing* is set to 8 and *Shading Oversampling* to 0% on each image (A, B, C) they will all lead to (D). As you can see in (D) all the extra samples fired by the anti aliasing (8) are gracefully redistributed to improve general sampling. Please note all resulting images (D) are equivalent in both quality and render time.



If you look closely to the shadow in the reflection in (B) there's something interesting happening here. Even if the shadow *Quality* is set to 1, the fact that the reflection is oversampled at 8, benefits to the reflected soft shadow quality.

Understanding Transparency

With the exception of outgoing ray directions, there is no difference between refraction and transparency in Clarisse. They're both raytraced the same way. On complex scenes that have complex materials and lighting, there's no magic,

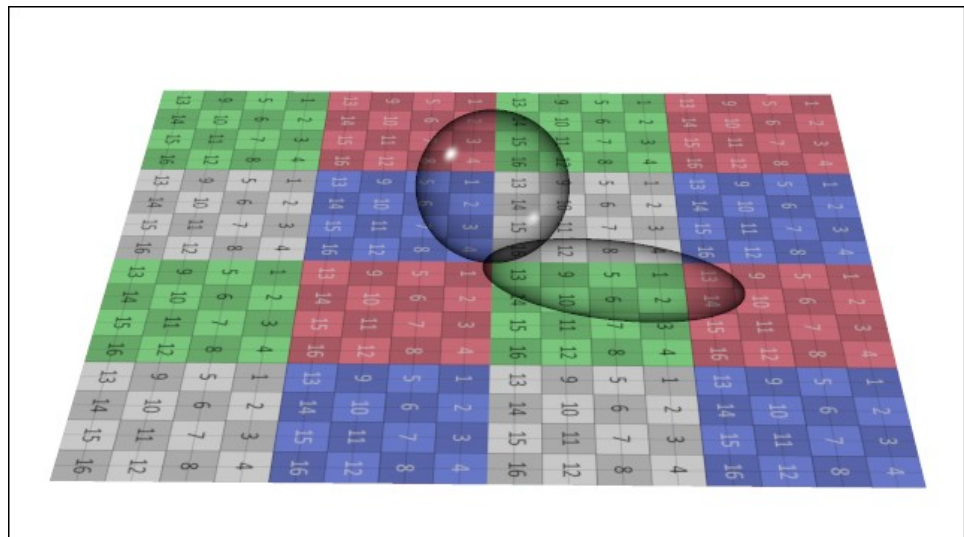
transparency increases render times. In order to keep control on render times, the raytracer provides several attributes managing transparency.

Note

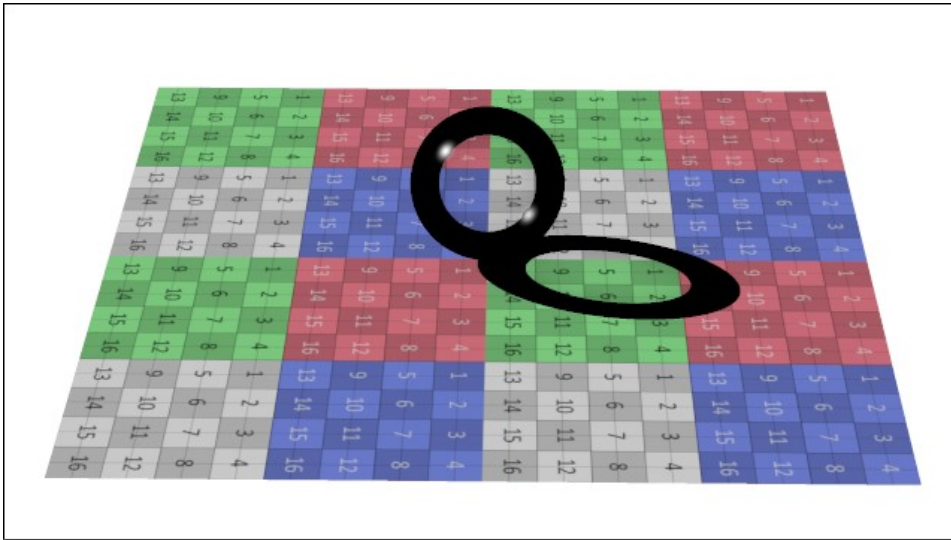
Transparency settings in the renderer can be overridden by materials. Usually the renderer settings is used as maximums whereas material ones as minimums.

Alpha Threshold

By default, when a material or the result of the transparency net reaches an alpha value higher than $1.0 - 1/255$, the resulting color is considered as opaque. Increasing this value reduces the depth of the transparency net and reduces render times.



Alpha Threshold default value



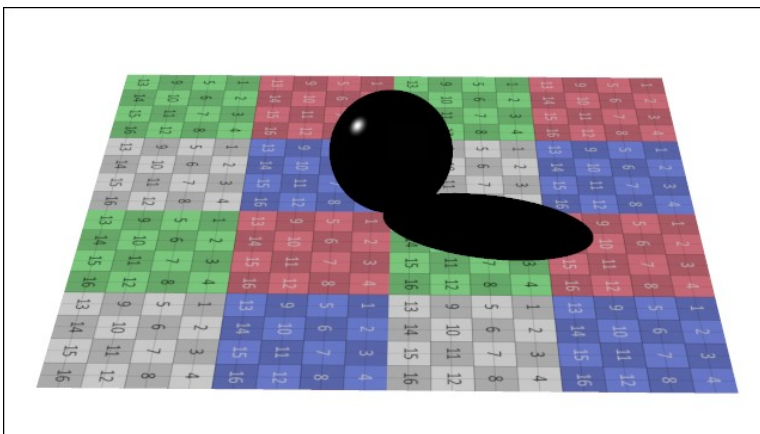
Alpha Threshold set to 75%

Alpha Depth

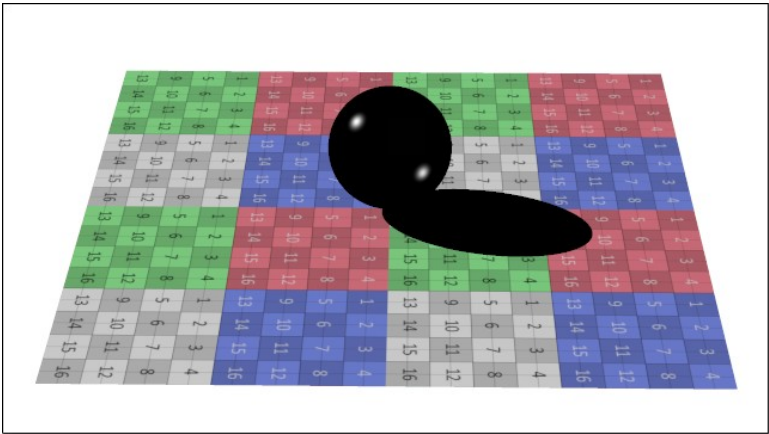
By default, the raytracer evaluates transparency up to 10 levels. After, the 11th level is considered opaque. To increase or decrease this value modify *Custom Alpha Depth* attribute. To tell the renderer to compute the full transparency net you can set *Alpha Depth* attribute to *Infinite*. In this case the raytracer virtually evaluates transparency on an infinite number of level. Please note that shadow transparency net depth uses also *Alpha Depth* value.

Note

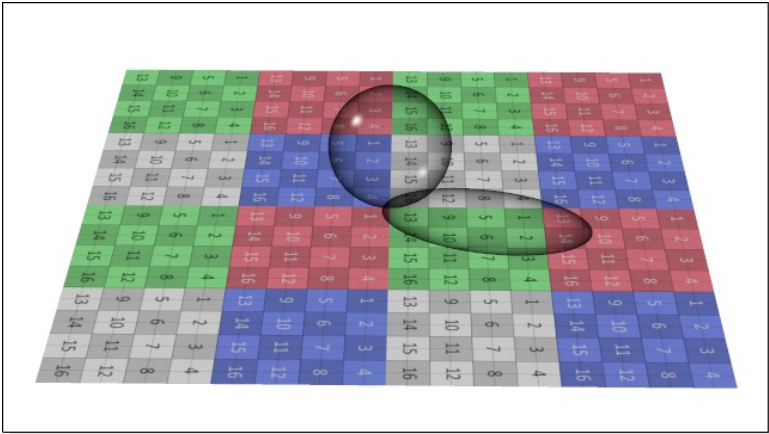
An *Alpha Depth* value of 0 disable both transparency and refraction.



Alpha Depth set to 0



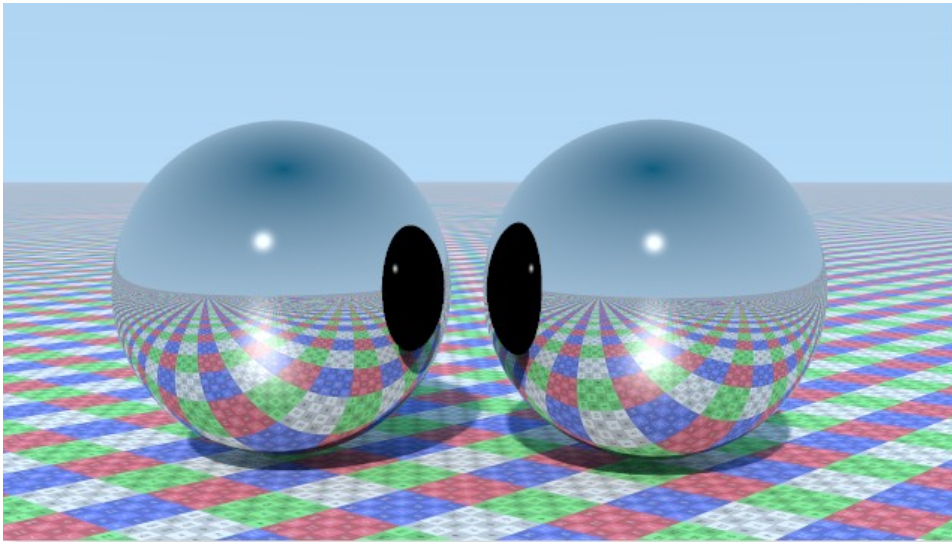
Alpha Depth set to 1



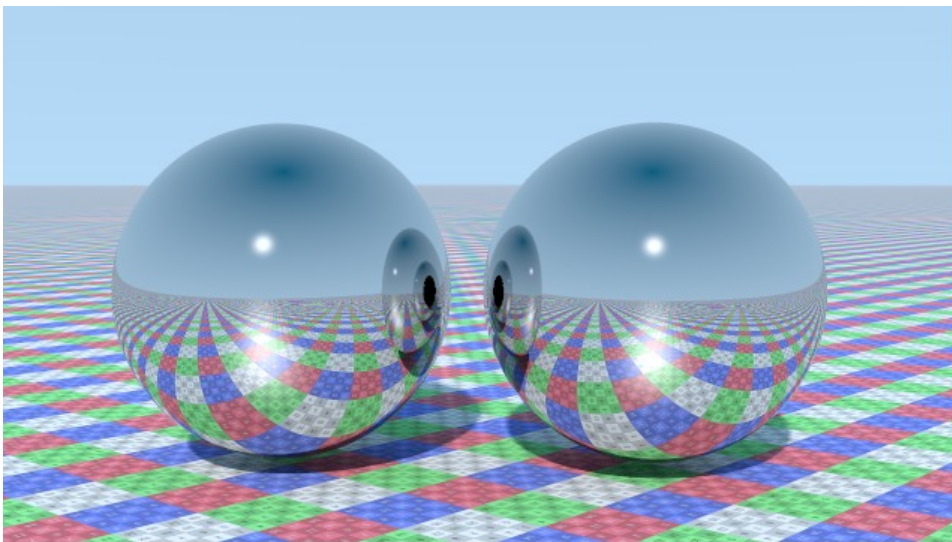
Alpha Depth set to 2

Reflection Depth

The *Reflection Depth* attribute sets the maximum number of reflection bounces before considering a ray hit material as non-reflective. To increase or decrease this value modify *Custom Reflection Depth* attribute. You can also set *Reflection Depth* attribute to *Infinite* to let the raytracer evaluate virtually an infinite number of bounces.



A reflection depth of 1



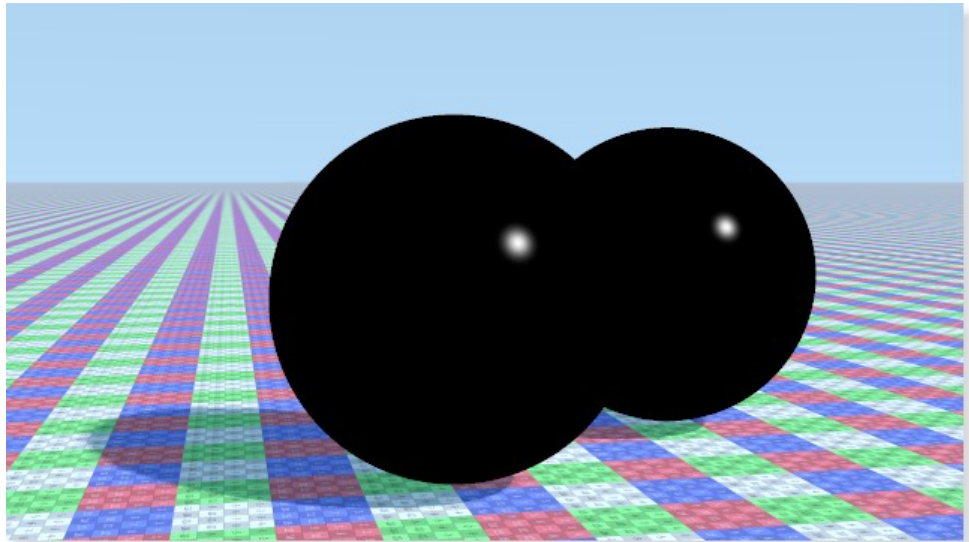
A reflection depth of 3

Note

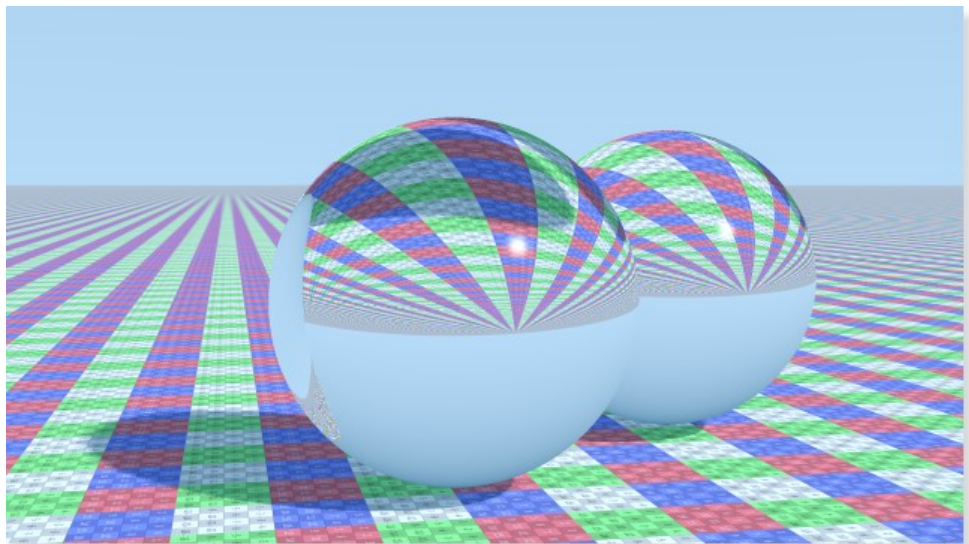
Reflection Depth settings in the renderer can be overridden by materials. Usually the renderer settings is used as maximums whereas material ones as minimums.

Refraction Depth

The *Refraction Depth* attribute sets the maximum number of refraction bounces before considering a ray hit material as non-refractive (opaque). To increase or decrease this value modify *Custom Refraction Depth* attribute. You can also set *Refraction Depth* attribute to *Infinite* to let the raytracer evaluate virtually an infinite number of bounces.



A refraction depth of 1



A refraction depth of 3

Note

Refraction Depth settings in the renderer can be overridden by materials. Usually the renderer settings is used as maximums whereas material ones as minimums.

Motion Blur

In most raytracers, 3d motion blur is a big overhead. In Clarisse, however, its overhead is minor. To enable motion blur, just activate *Enable Motion Blur* attribute. In the following screenshots motion blur overhead is about 10 percent.



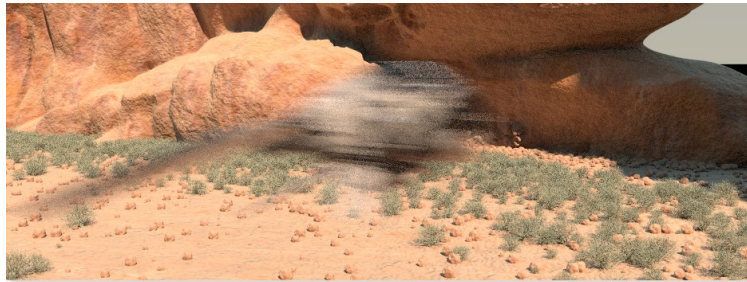
Rendered in 60 secs without motion blur



Rendered in 65 secs with motion blur

Motion Blur Settings

Clarisse supports several motion blur types: *Backward*, *Centered*, *Forward*. To set motion blur direction, go to **Edit > Preferences...** under *Rendering* tab you can change *Motion Blur Direction*. Clarisse also supports multi-segment motion blur. In the very same panel you'll find *Motion Blur Sample Count* which defines the number of times Time is sampled. To control shutter time modify *Motion Blur Length*.



Backward motion blur



Centered motion blur



Forward motion blur

New in 1.5

Motion Blur Length and *Motion Blur Direction* are global settings for the current project. Even if those settings are available from the Preferences panel, they are both saved in the current project file.

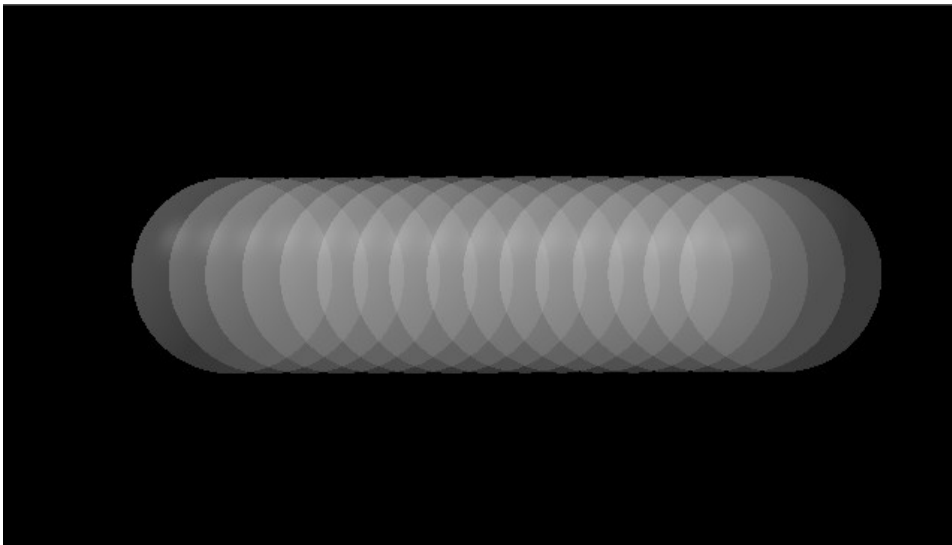
Motion Blur Sampling Mode

You can choose over several sampling strategies when rendering motion blur.

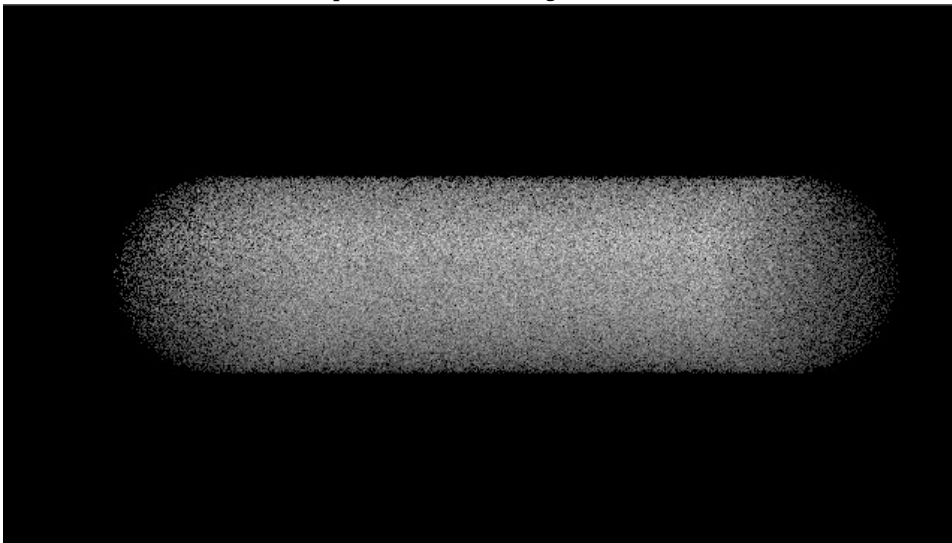
Mode	Description
Stratified	Each samples is at a fixed interval. This method introduces banding in your images.
Random	Each sample has a random time. This method introduces a lot of noise and is only useful as debug information to shader writers.
Stratified Jitter	Each sample is jittered on a grid. This method introduces noise over banding and leads to good results.

Blue Noise	Each sample has a random time. With low level of anti-aliasing, this method tends to reduce noise over Stratified Jitter. This is the recommended mode.
------------	---

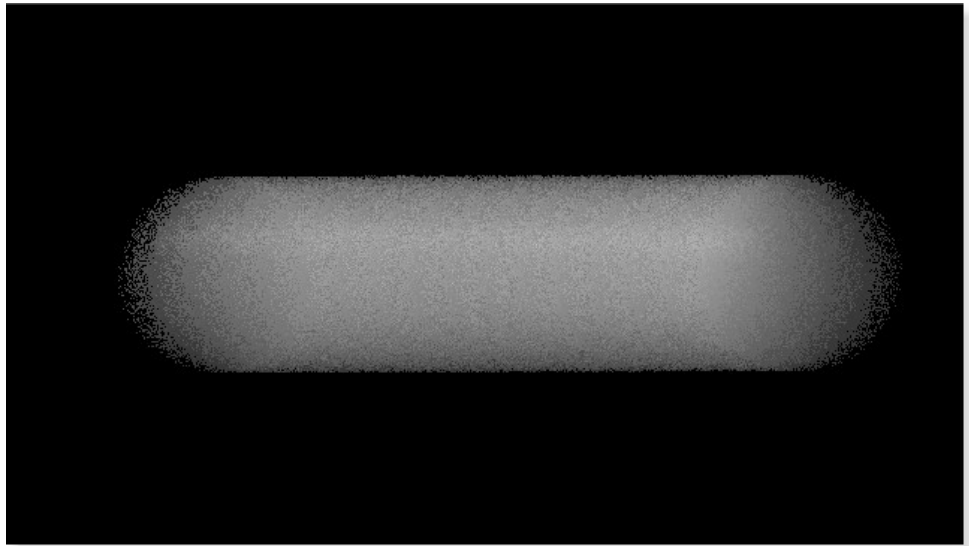
All the following images have been rendered with a *AntiAliasing Samples* set to 4 (16 samples)



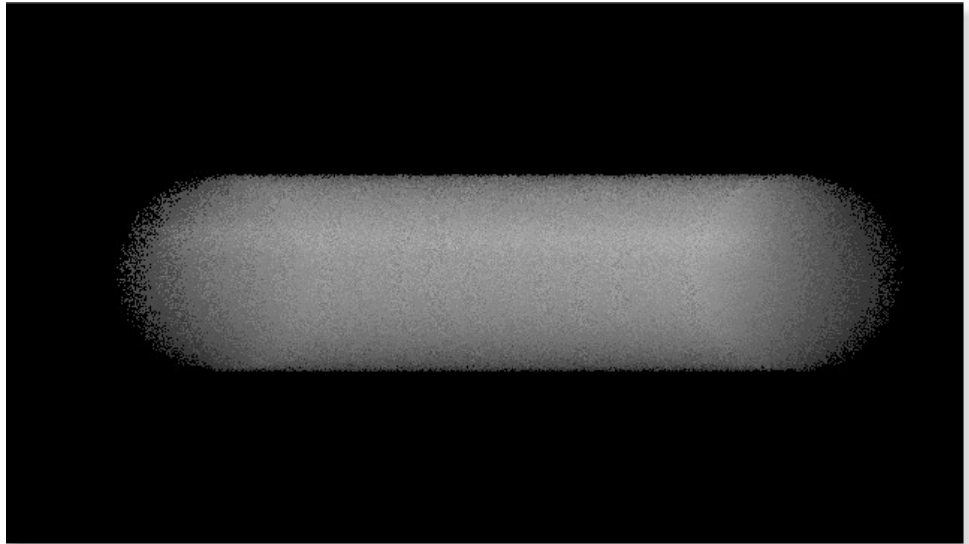
Note the pronounced banding (Stratified Mode)



Note how the noise is pronounced (Random Mode).



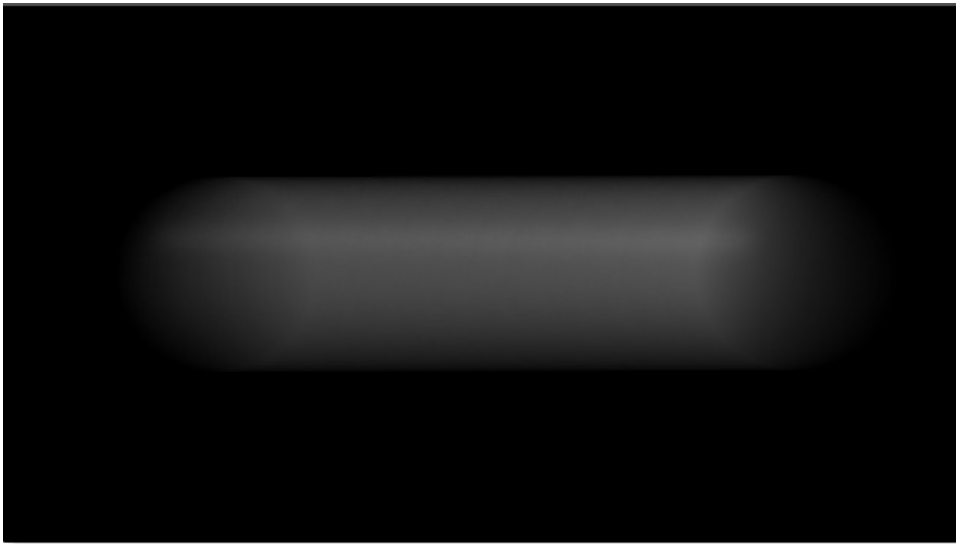
Good compromise between banding and noise (Stratified Jitter).



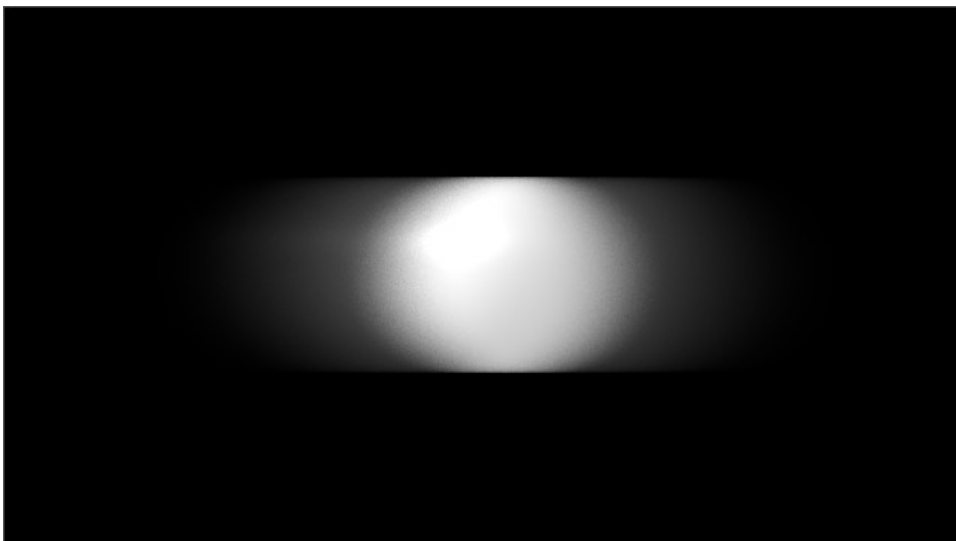
No banding and low noise (Blue Noise).

Shutter Curve

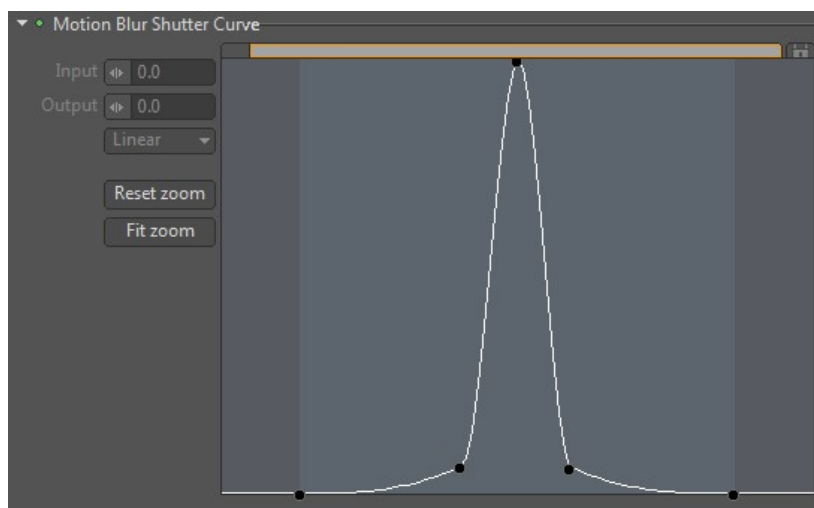
By default, samples are exposed for the same amount of time. You can finely control the aspect of the motion blur by modifying the shutter curve. This will basically simulate how fast/slow the shutter opens between each frame.



Linear (default) shutter curve



Same animation, but driven by a custom shutter curve.



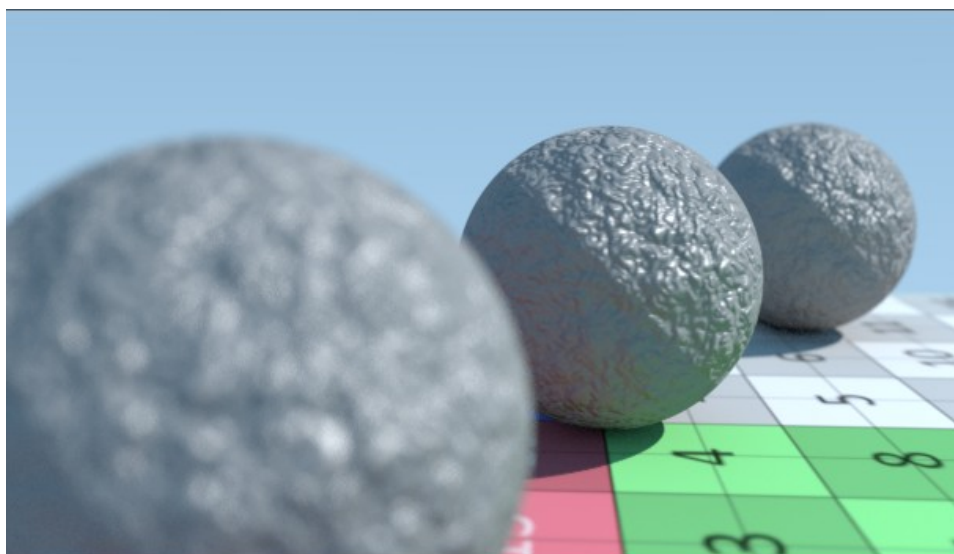
A shutter curve example

Lock Sampling Noise

Denoiser filters tend to provide better results when sampling is not time dependent. If you wish to use a post denoiser, we then recommend you to enable *Lock Sampling Noise*. *Lock Sampling Noise* locks material, light... sampling through time in a way that noise seems to stick to the camera plane. For example, if you were to render a static frame over time, all the rendered frames would be identical: resulting noise won't be animated.

Depth of Field

3D Depth of field can be achieved by enabling *Enable Dof* attribute in both Perspective and Perspective Advanced cameras. To increase the quality of Depth of Field, you must increase the number of anti-aliasing samples.

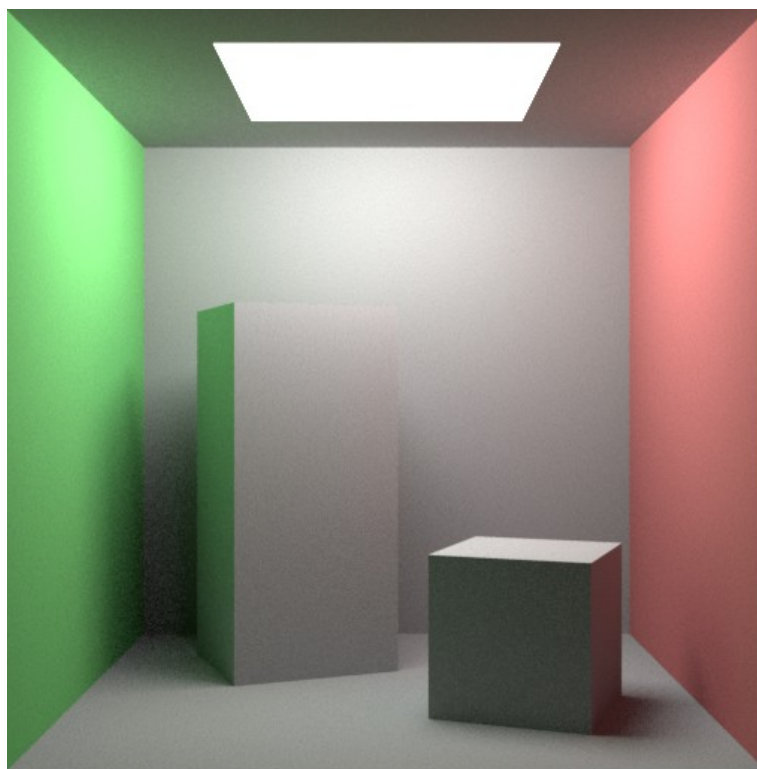


The size of the lens is controlled using the *F Stop* attribute. *Focal Distance* can either be set manually or automatically by setting a *Focus Object*. Please note, you can't currently specify the shape of the *bokeh*. It is a perfect circular lens.

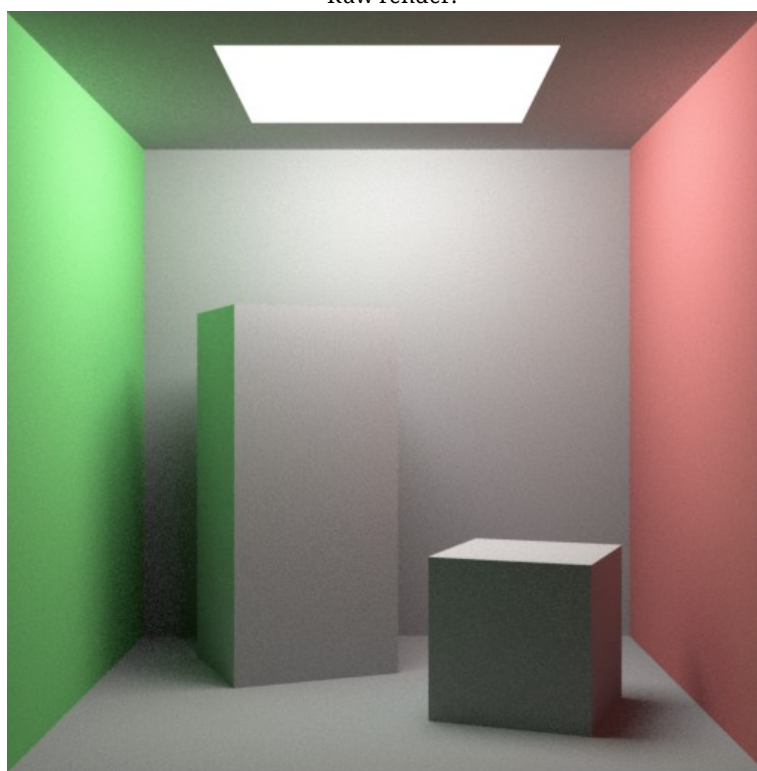
Tone Mapping

During rendering, Tone mapping allows you to remap each sample by using user defined curves. Tone mapping is applied prior final pixel integration. This means it is applied before all sub-samples are integrated to produce the final pixel color. This allows you for example to clamp high luminance value near very bright spots to improve anti-aliasing.

To activate Tone Mapping, just check *Enable Tone Mapping*. By default, the tone mapping curve clamps negative values and values higher than 1.0.



Raw render.



Tone mapping curve clamping values above 1.0.



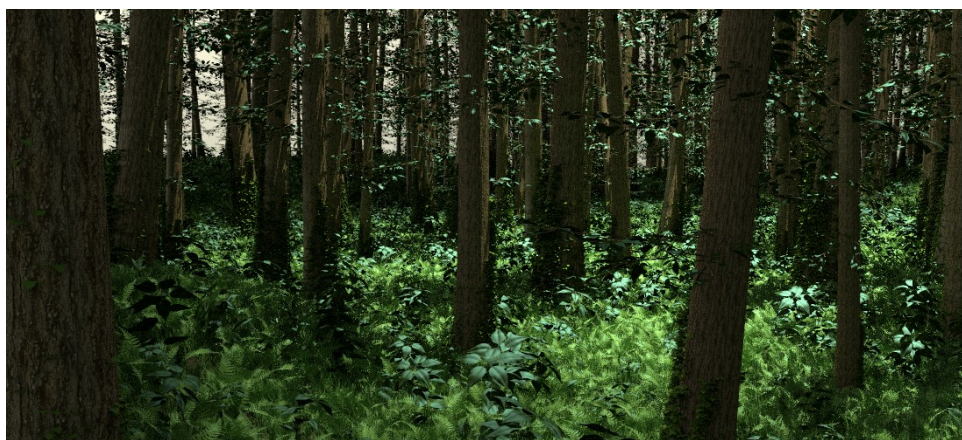
See how aliasing has been reduced on the right (tone mapping on) compared to left (tone mapping off)

Previz Mode

Sometimes you may want to work your materials with a neutral lighting rig or simply without lighting overhead. By enabling *Previz Mode* the lighting used in the 3d layer is replaced by a very fast neutral one. Please note Previz lighting rig is the same one used in the 3d View while in Previz mode.



Previz Mode enabled



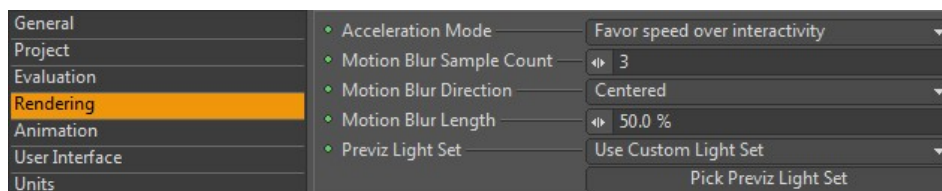
Previz Mode disabled

New in 1.5

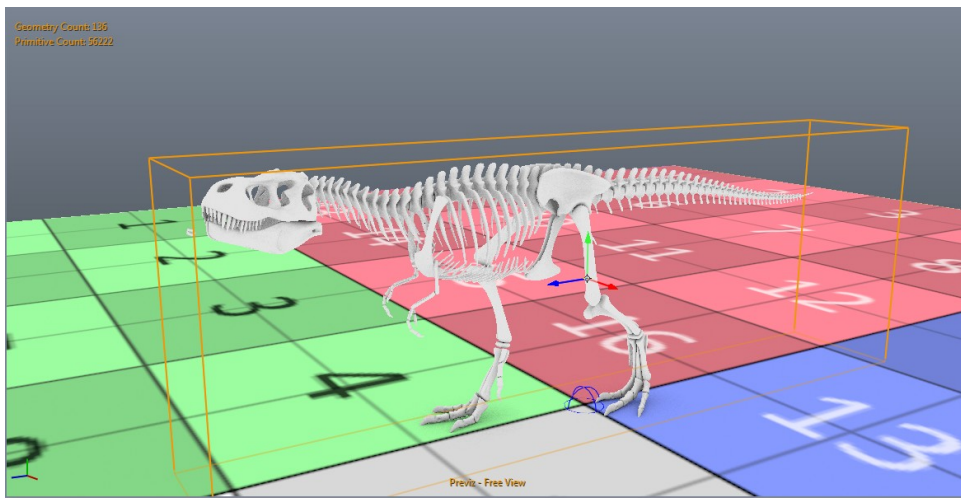
You can now create your custom Previz light sets.

Creating a Custom Previz Light Set

To create your own light set, you must create a new group and put your lights inside. Then go to **Edit > Preferences.../Rendering**. Click on *Pick Previz Light Set* and choose your group. Make sure to set *Previz Light Set* to *Use Custom Light Set*.



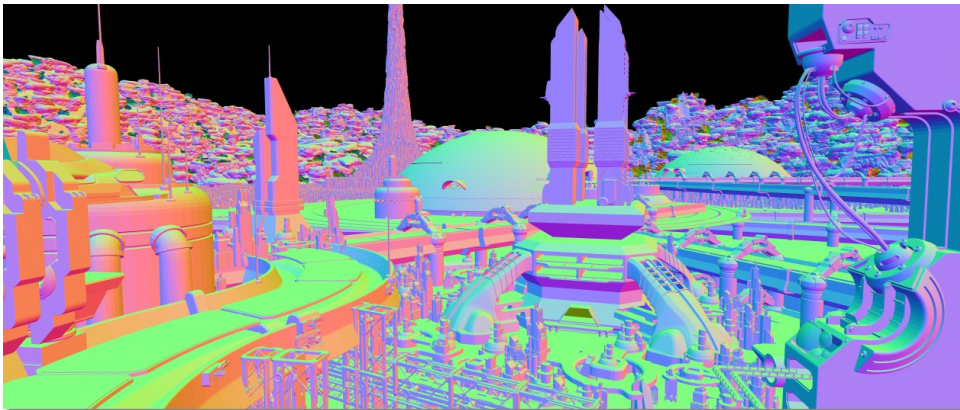
The custom light set is saved in the configuration file. If you modify a light that is referenced in the group used as Previz light set, well the light set won't be updated. When referencing a custom light set, lights are copied.



3D View in Previz mode with a custom light set.

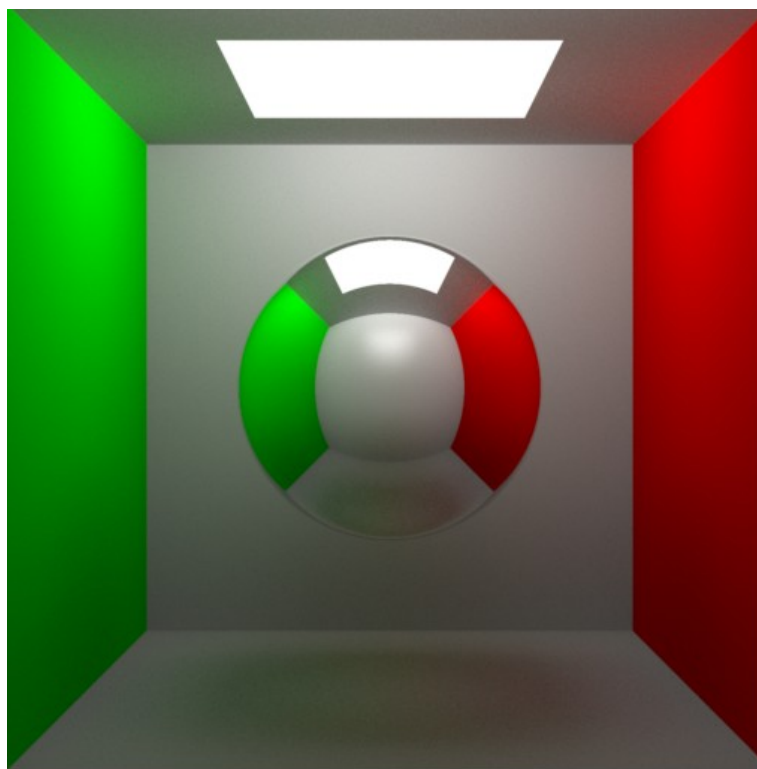
Disable Shading

When enabled this mode renders the scene as world space geometric normal pass.

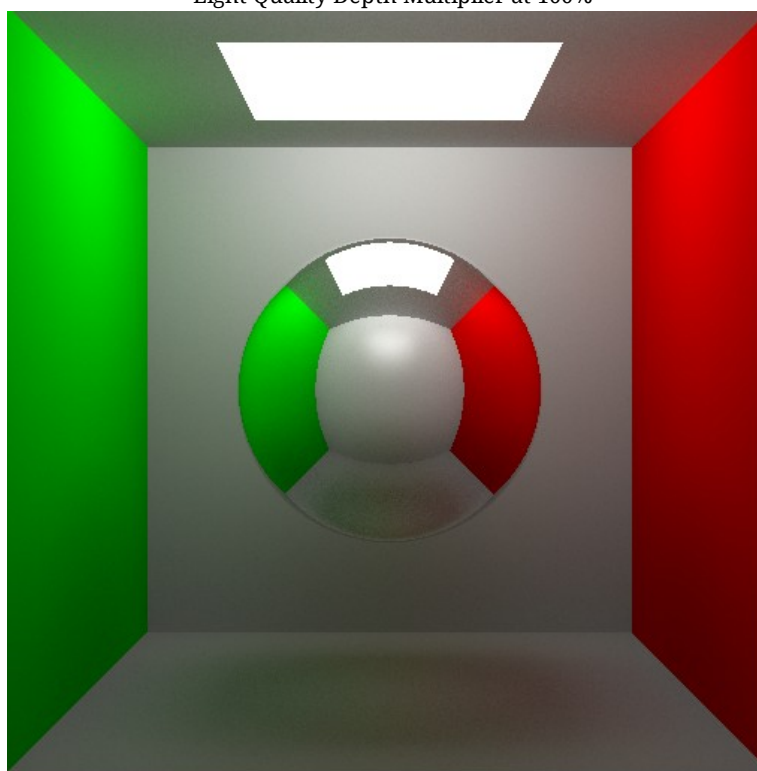


Light Quality Depth Multiplier

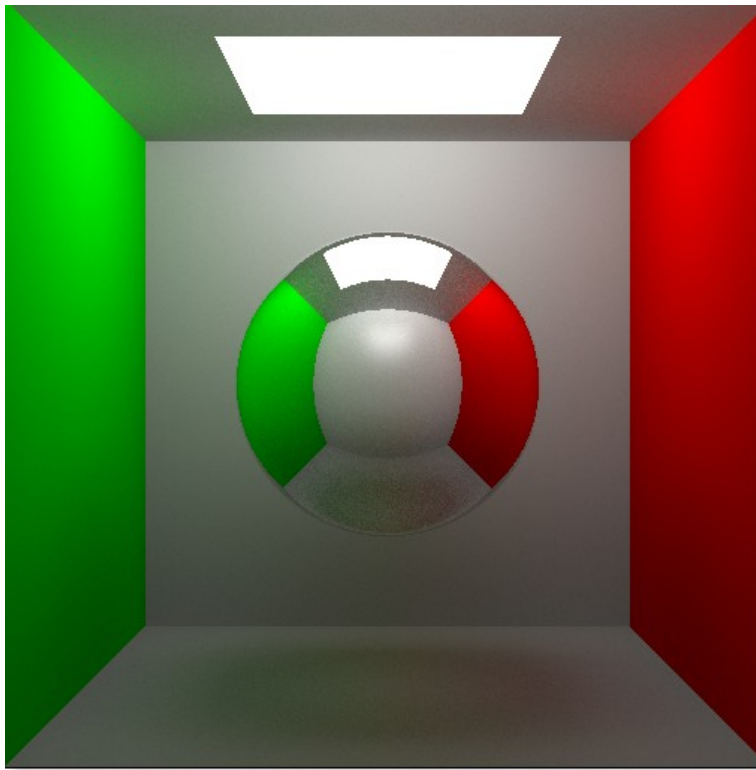
This attribute allows you to control the quality of light sampling in secondary rays. By default, when set on 100%, secondary rays uses same sampling level as primary ones. You can either increase (increase sampling quality) or decrease this value (reduce sampling quality).



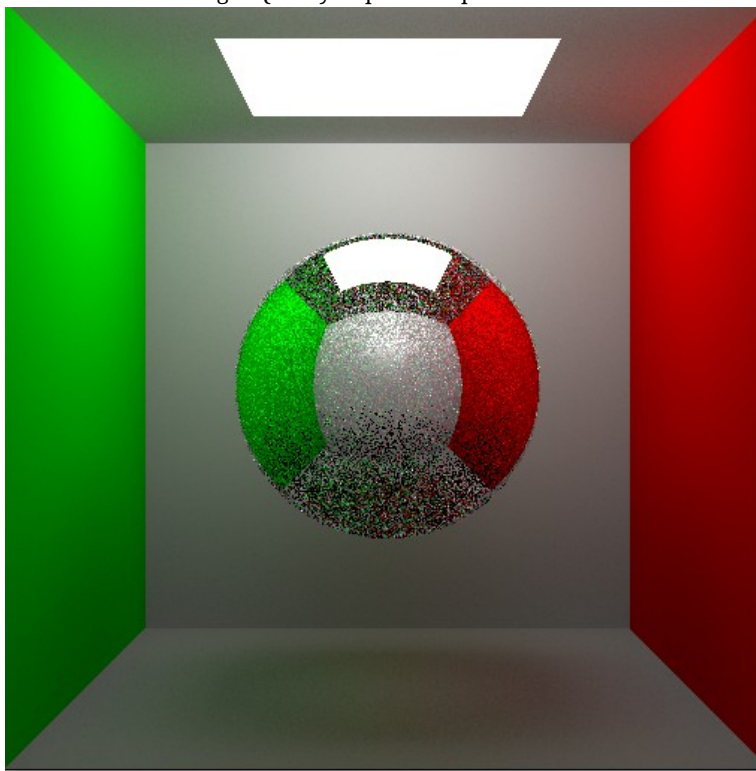
Light Quality Depth Multiplier at 100%



Light Quality Depth Multiplier at 50%



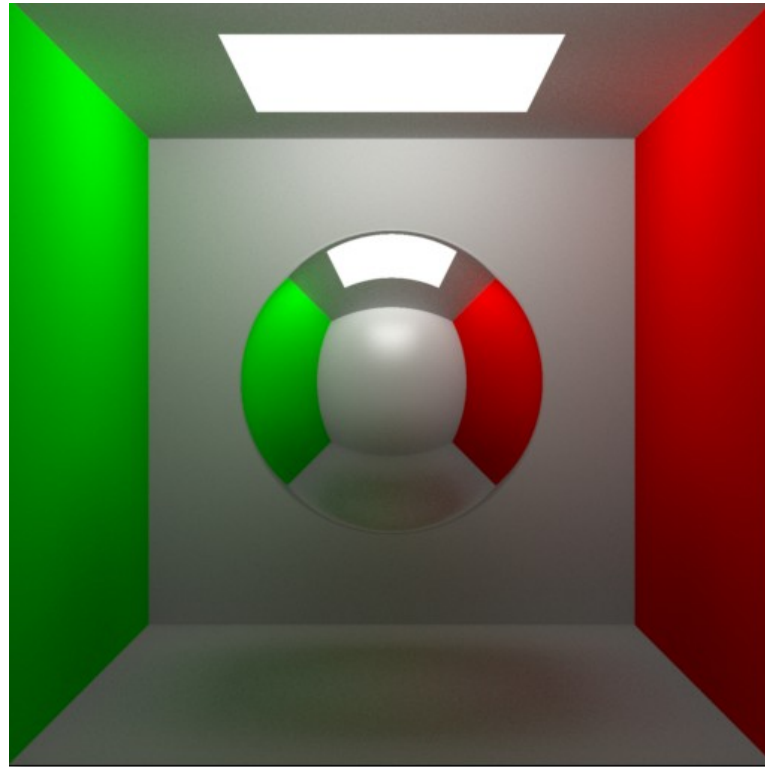
Light Quality Depth Multiplier at 25%



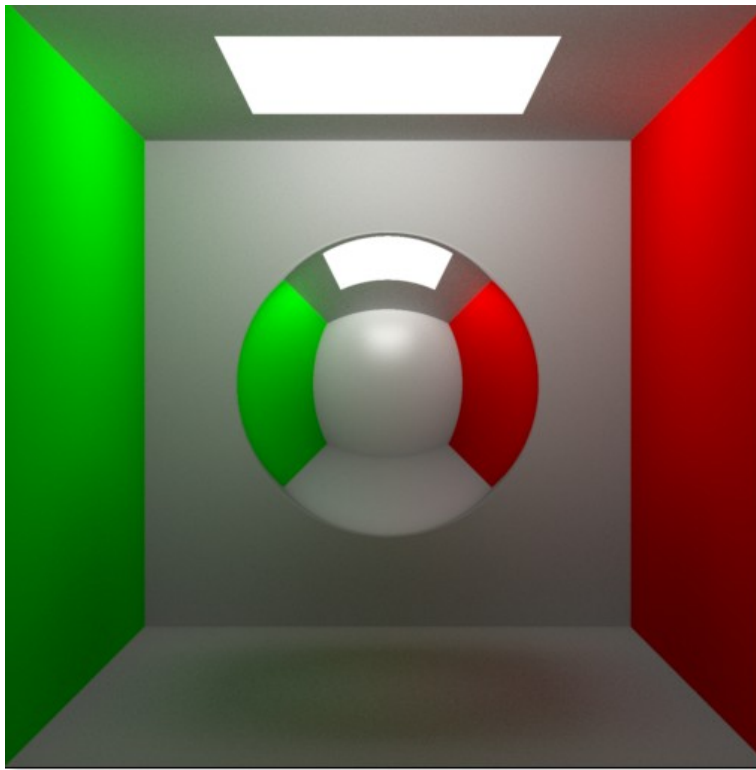
Light Quality Depth Multiplier at 0%

Max Shadow Depth

This attribute allows you to control the ray depth when shadows should stopped being computed. This is very useful to optimize your scenes. Please note, Renderer *Max Shadow Depth* can be overridden at light level.



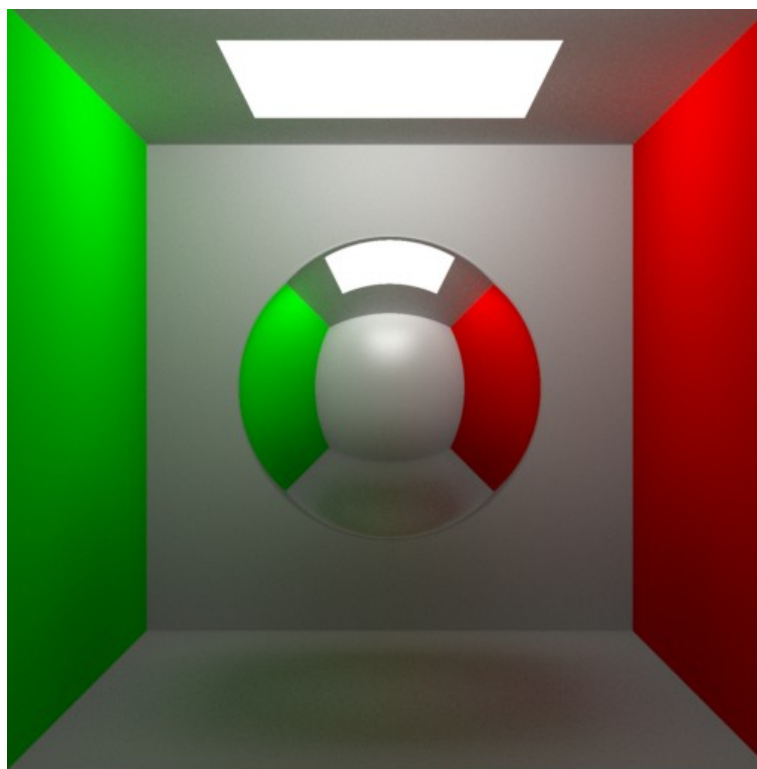
Global Illumination and Shadows by default



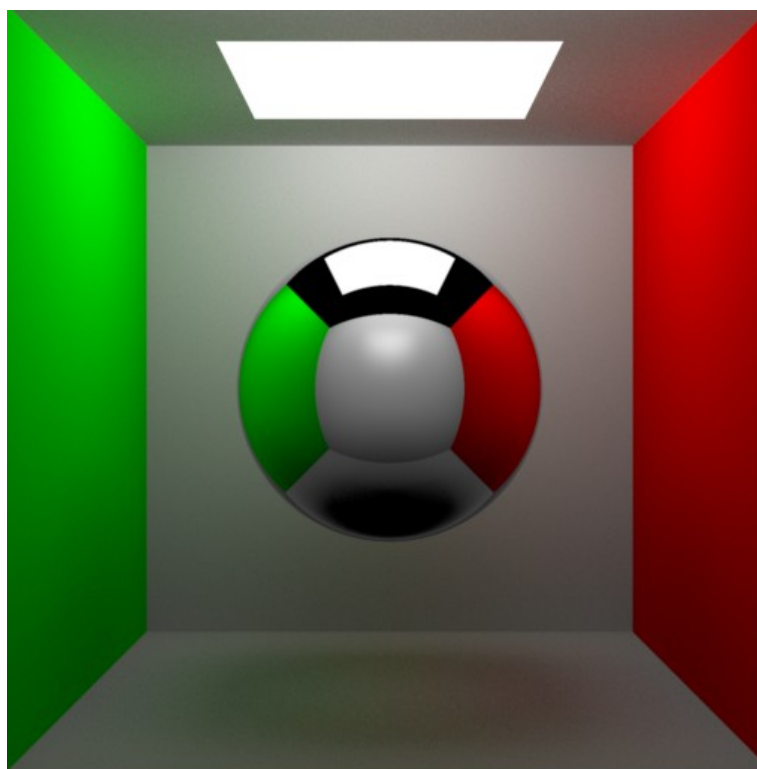
Max Shadow Depth set to 1. Sphere shadow is no longer evaluated in the reflection

Max Global Illumination Depth

This attribute allows you to control the ray depth when global illumination should stopped being computed. Please note, Renderer *Max Global Illumination Depth* can be overridden at Global Illumination light level.



Global Illumination and Shadows by default



Max Global Illumination Depth set to 1. Global Illumination is no longer evaluated in the reflection

Export AOVs

You can disable AOV evaluation by disabling *Export AOVs*. Disabling AOV evaluation can greatly speed up rendering on scenes that have many AOVs.

Lighting in Clarisse

In the following section we will go through many different lights that are available in Clarisse and we will talk about light linking. There are currently two light classes: directional and non directional lights. All Clarisse lights only use raytracing to illuminate scene objects.

Common Attributes

Most lights share a set of predefined attributes (when applicable) to control for example the illumination color, sampling quality etc.

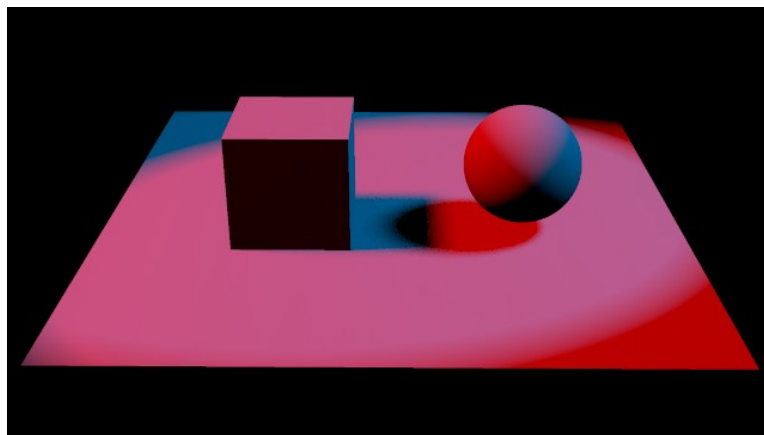
Attribute	Description
Geometry Group	Sets the scene object "seen" by the light. Please refer to Light Linking section.
Color	Sets the illumination color.
Affect Diffuse	Sets if the light affects material diffuse shading.
Affect Specular	Sets if the light affects material specular shading.
Affect Backlighting	Sets if the light affects material backlighting
Enable Shadows	Sets if the light casts shadows.
Enable Transparency Shadows	If disabled, the light considers occluders as opaque.
Quality	Controls the quality of the noise in the illumination (and shadows). The total number of samples is the square of this value. Please refer to Light Sampling section.

Light Linking

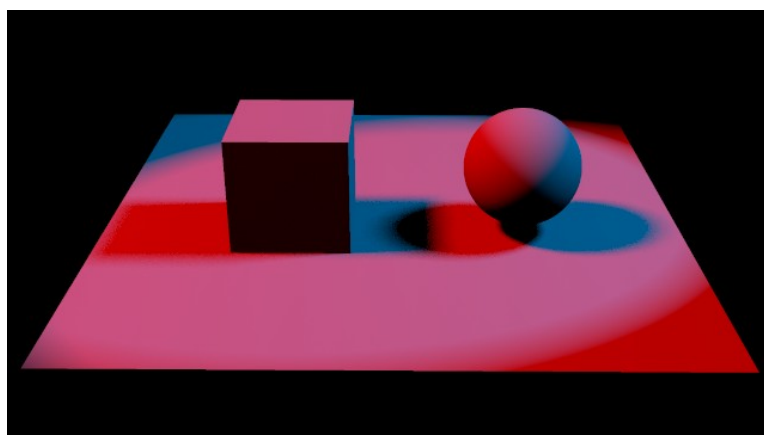
Depending on how you wish to control illumination, Clarisse offers many ways to deal with light linking. Indeed, Light linking can be achieved:

- at Layer 3d level by specifying a group of lights to *Lights* attribute. Please refer to 3D Layer - Lighting for more information
- at Scene Object level by specifying a group of lights to *Lights* attribute. Please refer to Rendering Attributes - Light Linking for more information

- at Light level by specifying a group of geometries to *Geometry Group* attribute

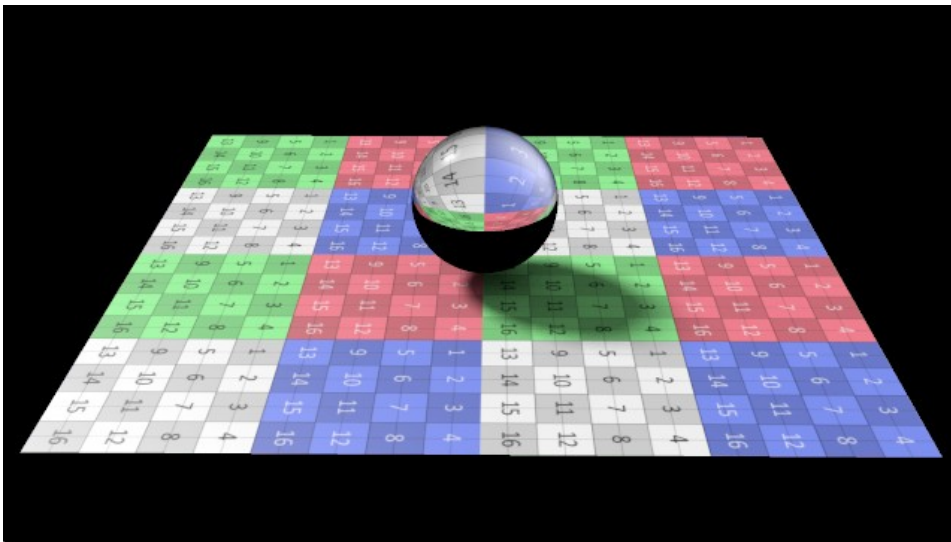


In this example the blue light has a geometry group that references the sphere whereas the red one has the box. In this scenario, the blue light is only occluded by the sphere and the red light only by the box. In the next example, the same scene without light linking. Note how the objects cast each 2 shadows.

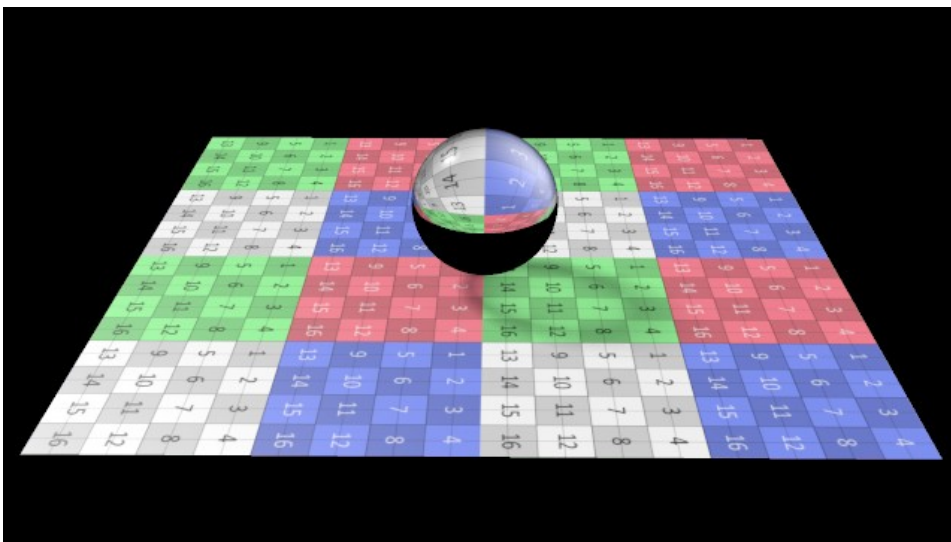


Lights and Transparency

Transparency can be a big overhead to rendering. You can control how light shadow rays deal with transparency. If *Enable Transparent Shadows* is disabled the occluder is considered as opaque. When enabled, lights evaluate material transparency along with their shadow rays until they hit background or a material that is opaque.



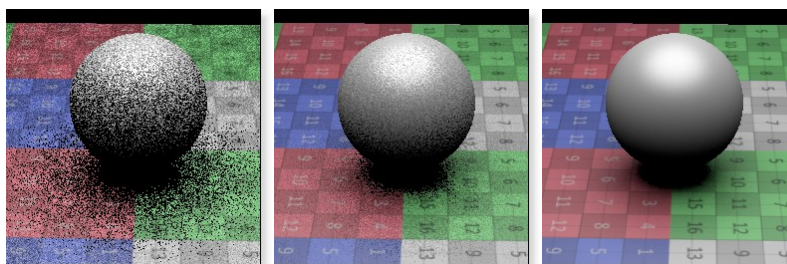
Enable Transparent Shadows disabled



Enable Transparent Shadows enabled

Light Sampling

Lights requiring multiple samples to be evaluated expose the *Quality* attribute. This attribute controls the number of samples that are used to sample lights. In fact, the actual number of samples used by lights is the square of *Quality* value. The lower the *Quality* value and the noisier shadows and illumination gets. Please note high *Quality* values increase rendering time. As a general rule, please keep in mind that in most cases to reduce sampling noise by 2 you need to increase the number of samples per 4.



Quality set to 1,2,8 (1, 4, 64 samples)

Directional Lights

Directional lights are lights that illuminate points from a major direction. For example a spot or a point light. With the exception of the *Distant* and *Linear*, lights can act as occluders for global illumination rays. To learn more please refer to Light Occlusion section.

Light Occlusion

Area lights (*Area*, *Point*) simulate surfaces that emit light. When computing indirect illumination, Clarisse evaluates those lights and then gathers rays by sampling the material of hit scene geometries.

To be physically correct, area lights, which already emitted direct lighting, must occlude gather rays. Failing to do so means shading samples may receive energy from what's behind the light.

To enable/disable light occlusion, use the attribute *Enable Light Occlusion*. This attribute should always be enabled and is on by default. However on older scenes, it is automatically turned off to match former lighting. Please note the attribute is only kept for backward compatibility and may disappear in the future.

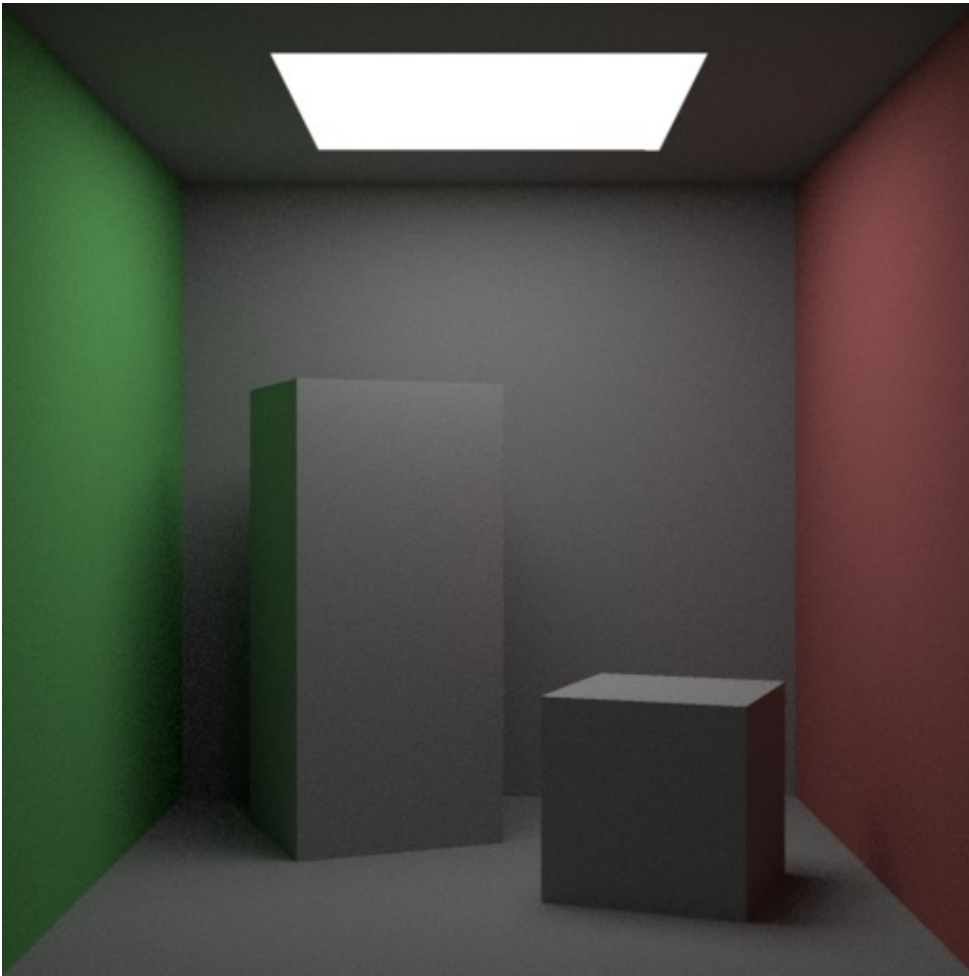


Figure A: occlusion disabled

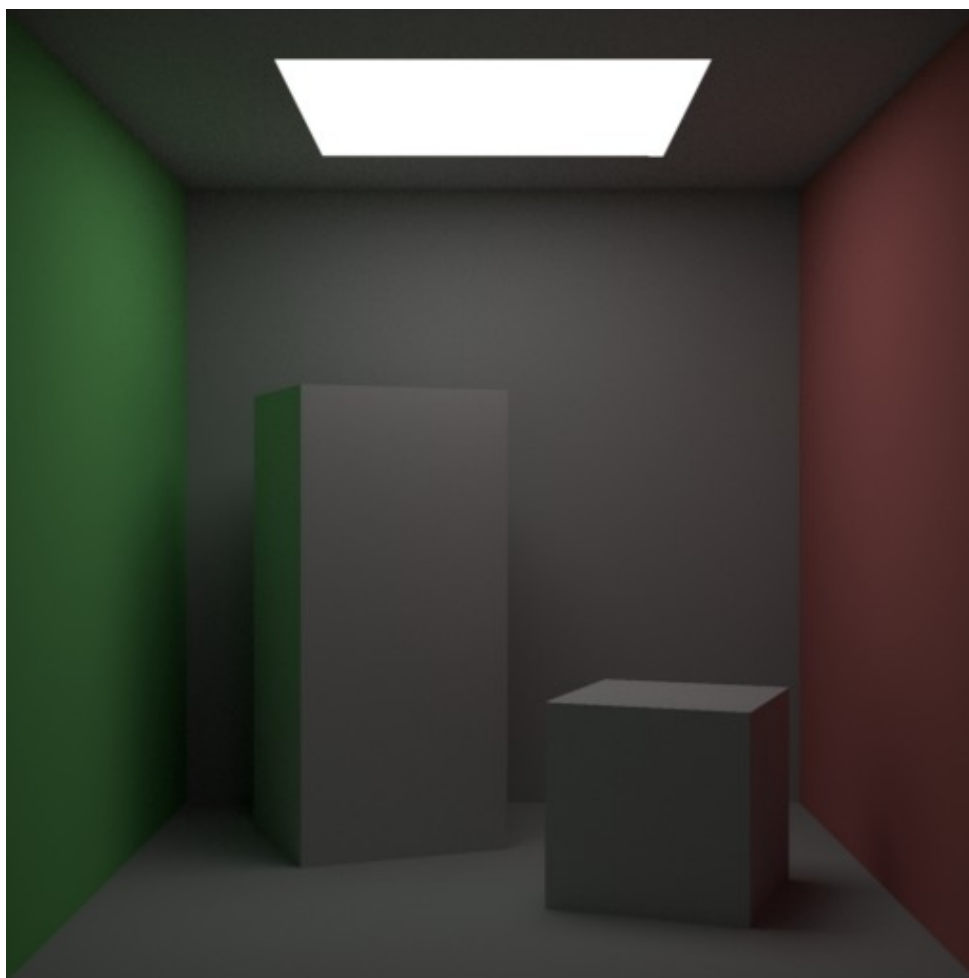


Figure B: occlusion enabled (default)

In this example, we can clearly see that without occlusion (Figure A), the image is too bright and noisier than with (Figure B).

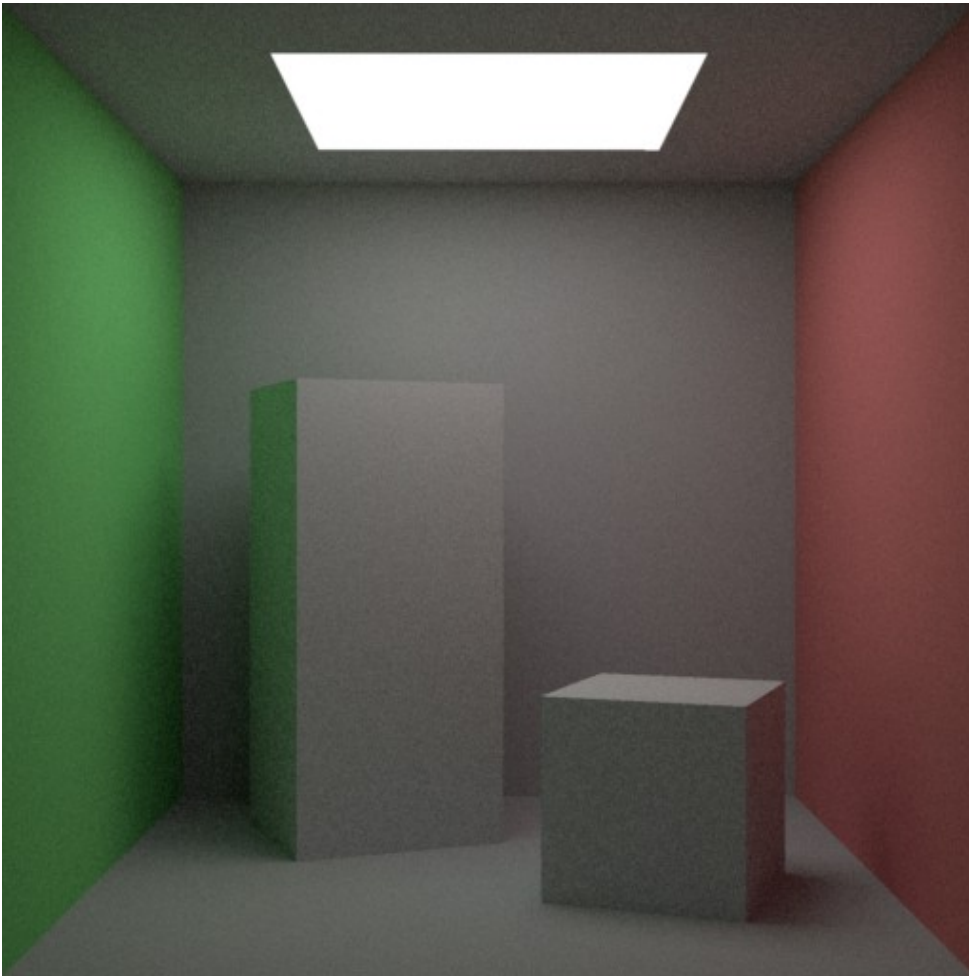


Figure C: occlusion disabled 2 bounces

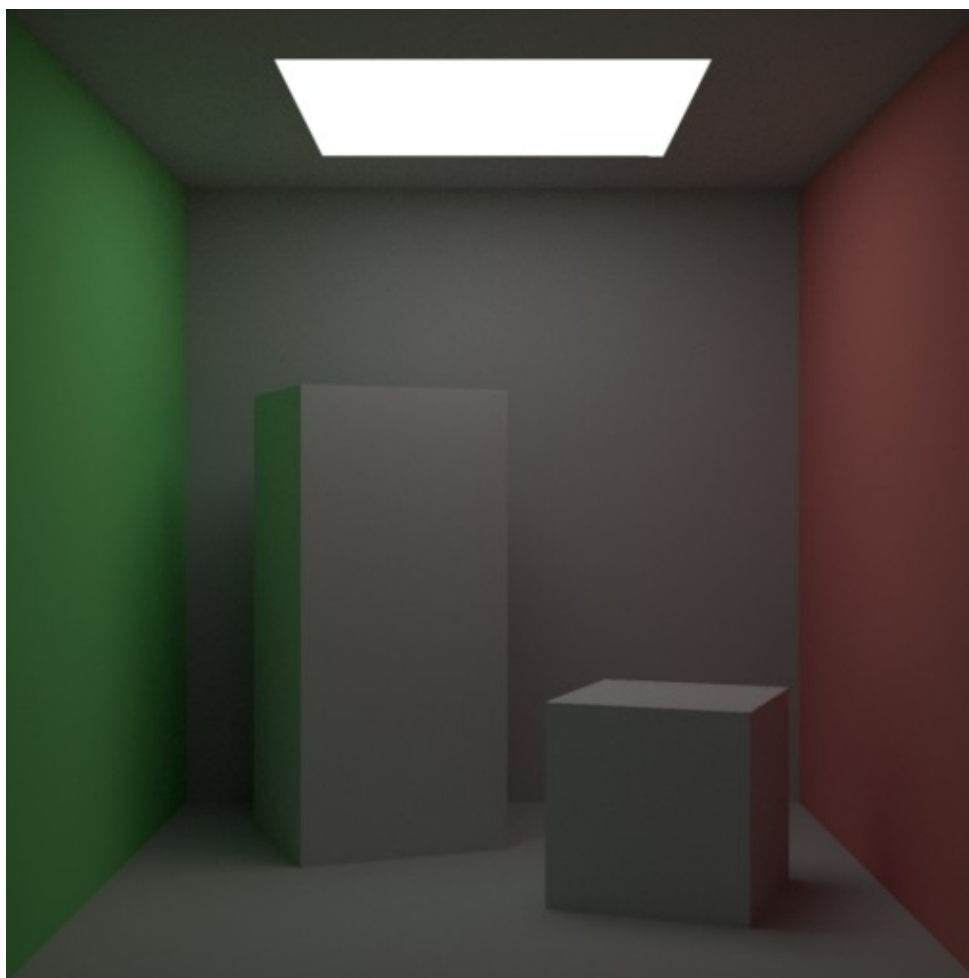


Figure D: occlusion enabled 2 bounces

The difference is even stronger with more bounces as the error (and noise) is increased. Now with 2 bounces, see how much brighter and noisier Figure C becomes over Figure D.

Light Attenuation

Some lights (Area, Point...) support attenuation of their intensity based on distance. This allows you to modulate light's intensity according to the distance of the shaded point. By default, the intensity of a light is inversely proportional to the square of the distance from the source of light (*Inverse Distance²*).

However, two issues may rise from this physical law:

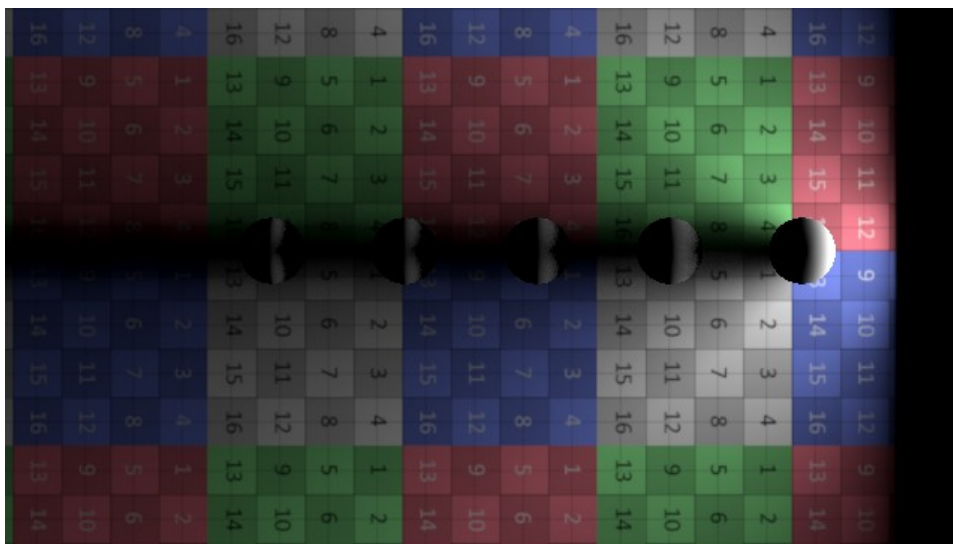
- Objects located near the source of light receive an intensity that is close to infinity.

- Lights continue to contribute to the illumination of very far away objects. Please note light intensities near a very small epsilon are automatically considered as not contributing.

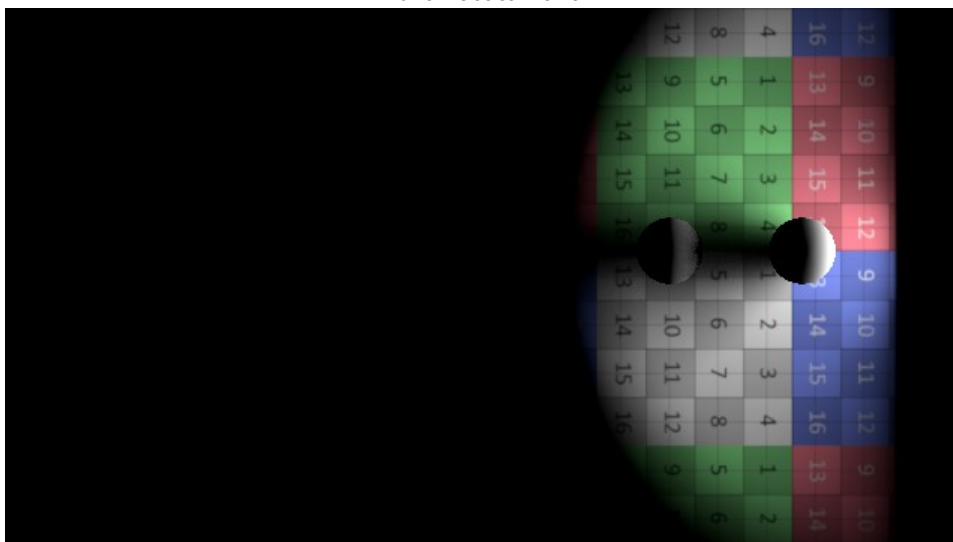
In Clarisse, you can control light attenuation in different ways. You can control its falloff. The falloff applies a predefined curve which will modulate light intensity. You can also control near and far attenuation range or completely control attenuation using a custom curve.

Falloff

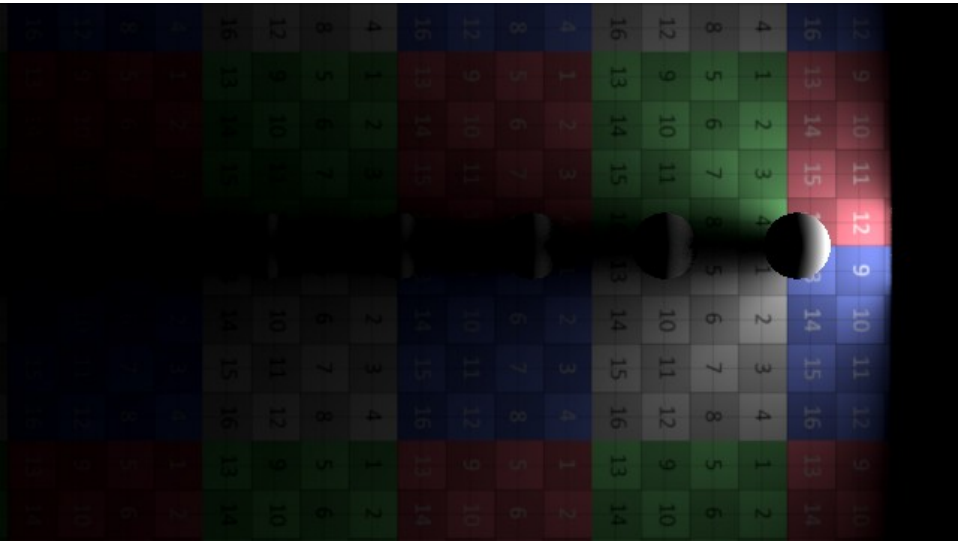
You can change the falloff of the light by modifying *Falloff Mode* attribute. By default, it's set to *Inverse Distance^2* meaning it follows inverse square law.



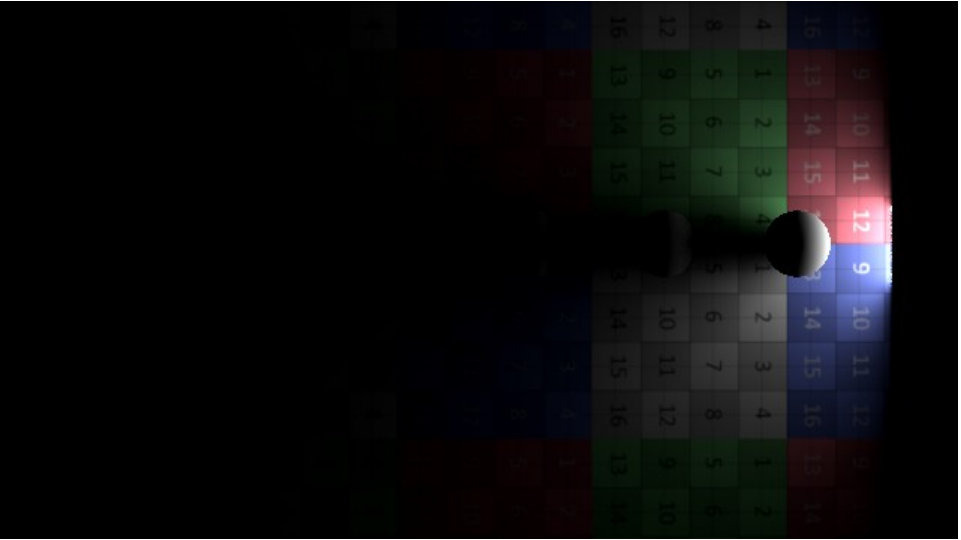
Falloff set to None



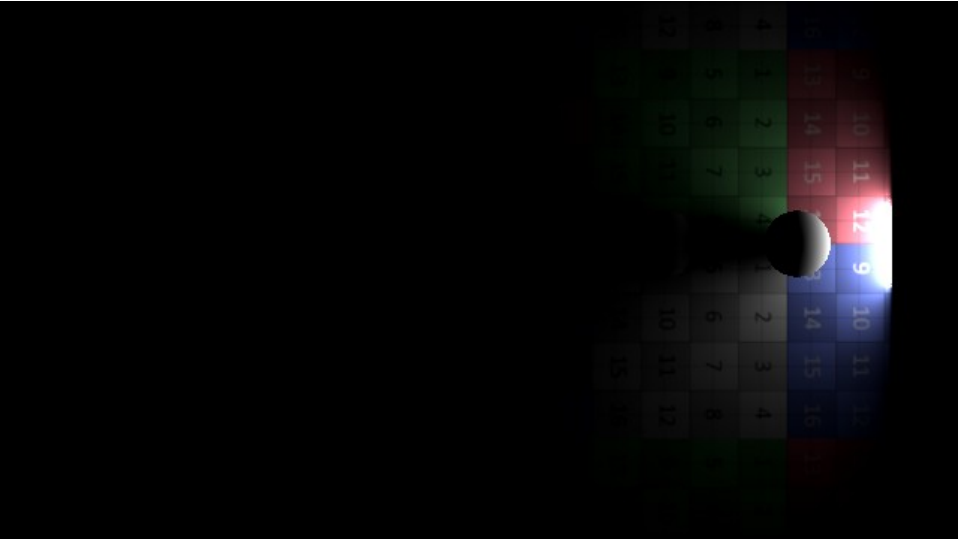
Falloff set to Linear and Falloff Range to 5 meters.



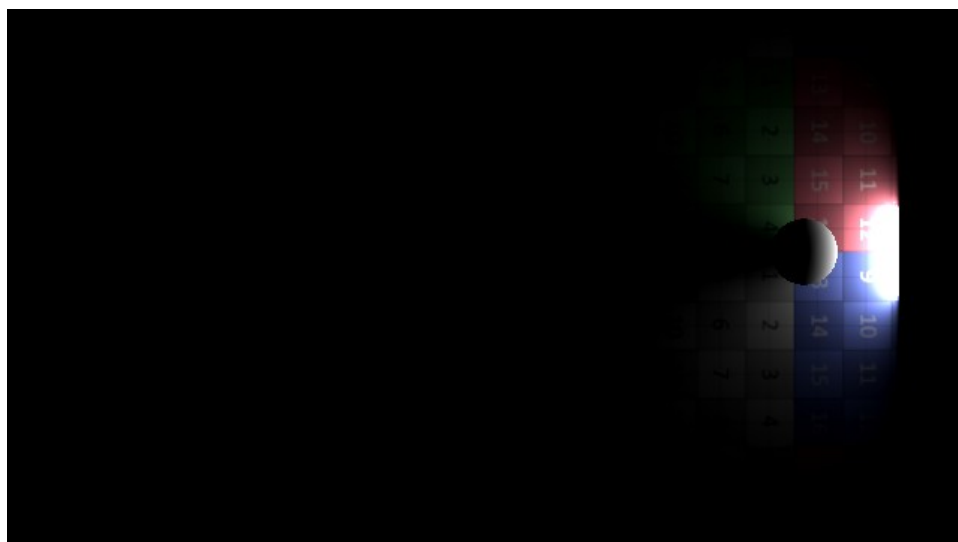
Falloff set to Inverse Distance.



Falloff set to Inverse Distance^2 (default).



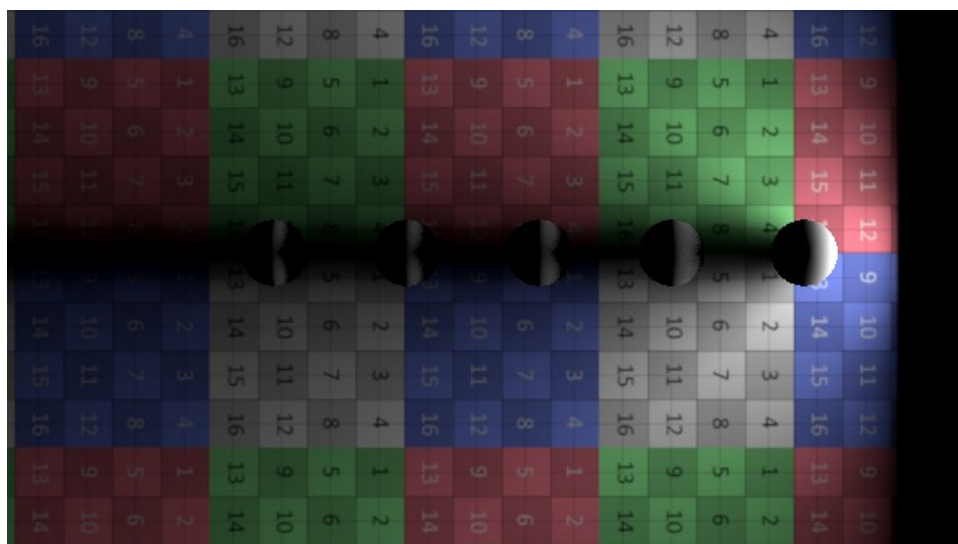
Falloff set to Inverse Distance^3.

Falloff set to Inverse Distance⁴.

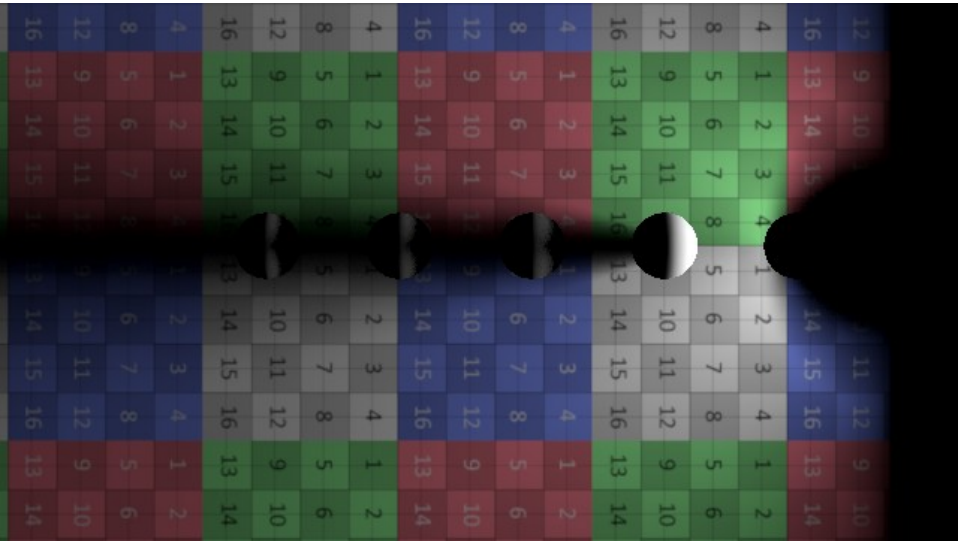
Attenuation

Using *Attenuation* attribute, you can control near, far, or both near and far attenuation. *Attenuation Near Start* and *Attenuation Near End* allow you to control the range at which the light starts its illumination. *Attenuation Far Start* and *Attenuation Far End* allow you to control the range at which the light ends its illumination. Please note the transition between start and end is a predefined smooth gradient.

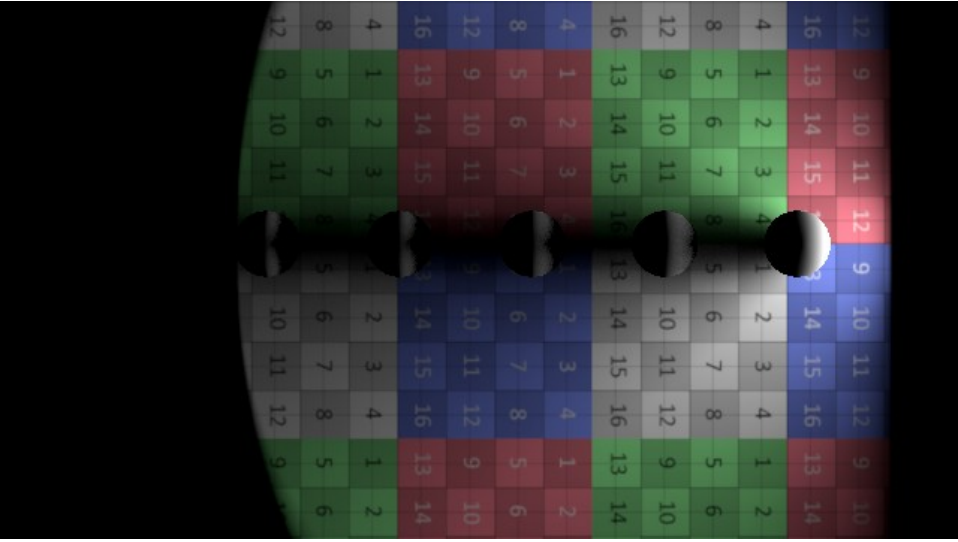
The attenuation is multiplied to the falloff value. In other words, if your light has a falloff, the attenuation doesn't replace it it will be multiplied to the falloff.



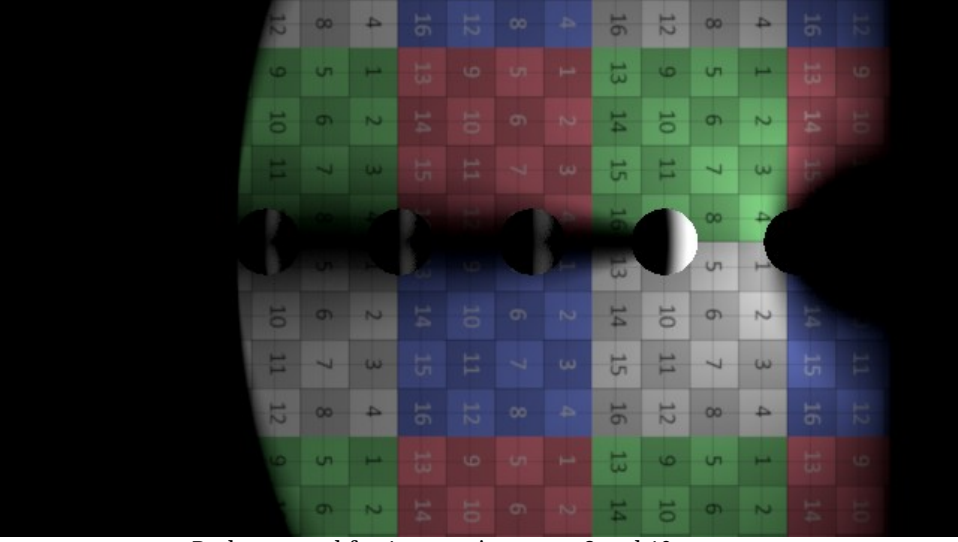
No Attenuation and No Falloff



Near Attenuation set to 2 meters.

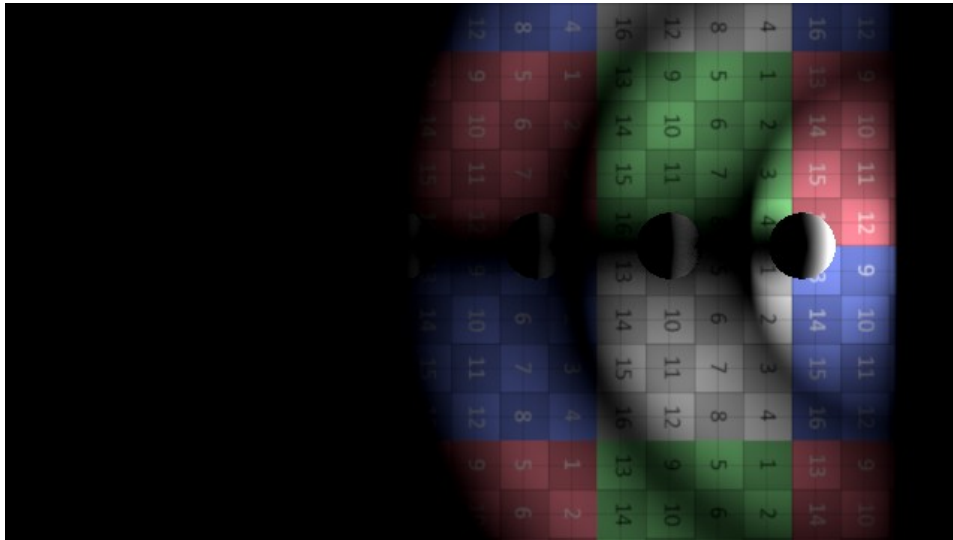


Far Attenuation set to 10 meters.



Both near and far Attenuation set to 2 and 10 meters.

You can also set *Attenuation* to be driven by a custom curve.



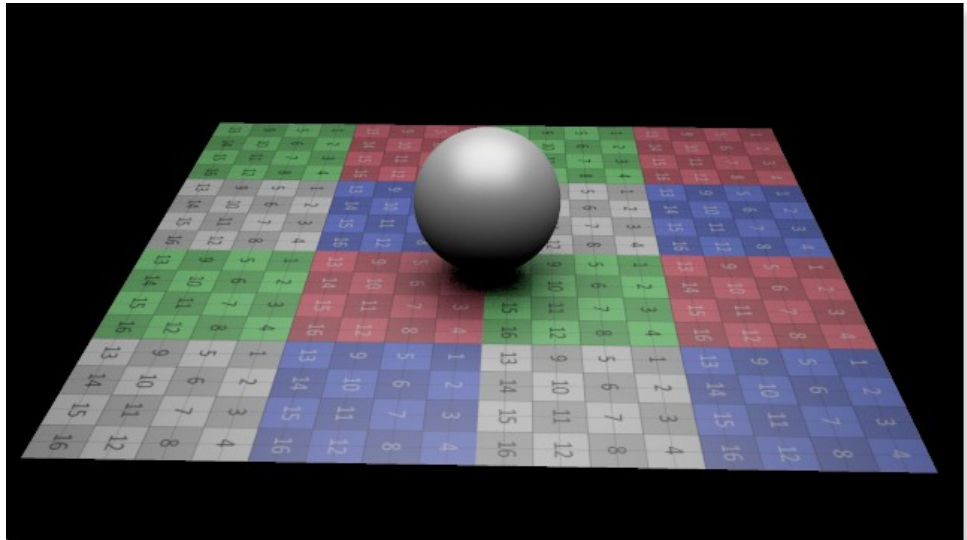
Attenuation driven by a curve.

Tip

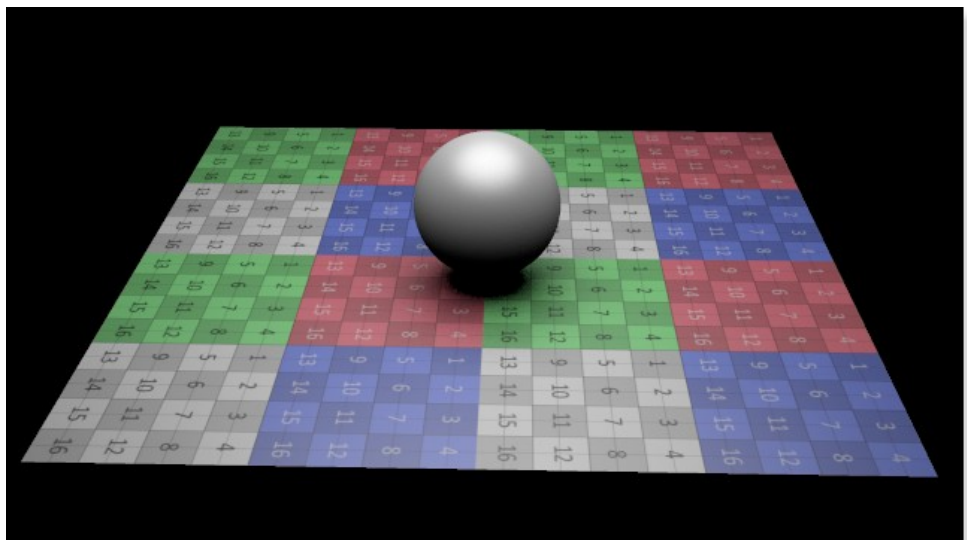
Setting Near and Far attenuation optimizes rendering speed. If a light is out of contribution range, its evaluation is skipped.

Area

An area light simulates the light that would come from a flat quad surface. This light is double sided and illuminates forward and backward. The area light can also simulate light coming from a flat disk-shaped surface by enabling *Disk Area*.



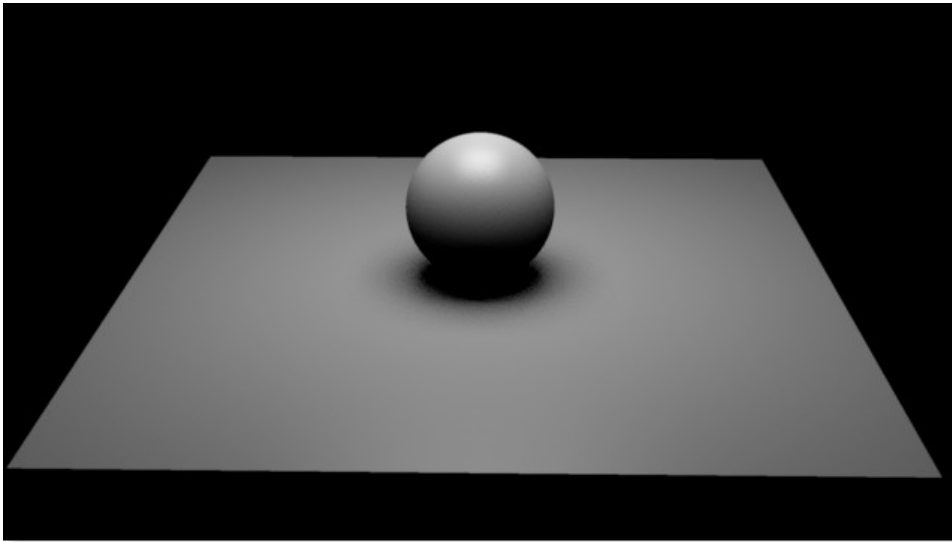
Disk Area disabled



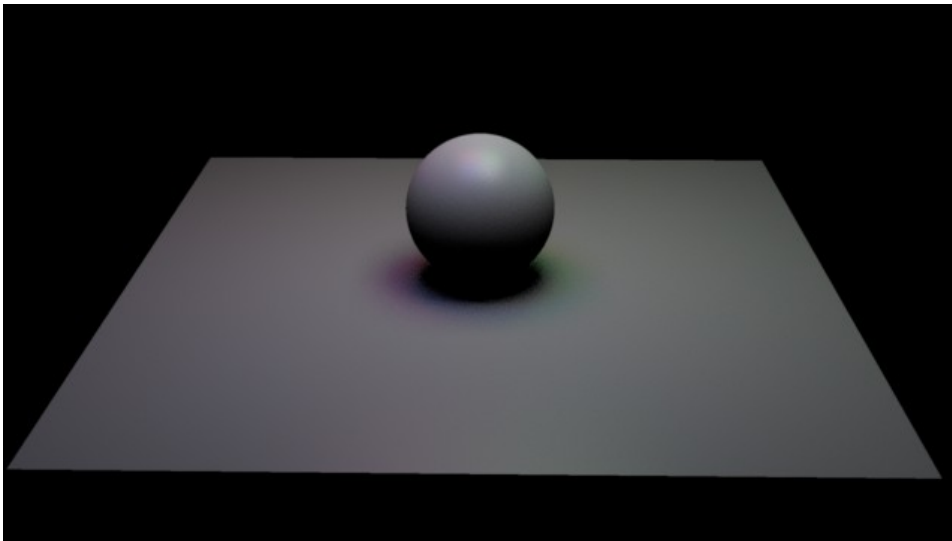
Disk Area enabled

Texturing

You can texture both *Color* and *Shadow Color* attribute. Remember to set your texture *Projection* to *Parametric*.



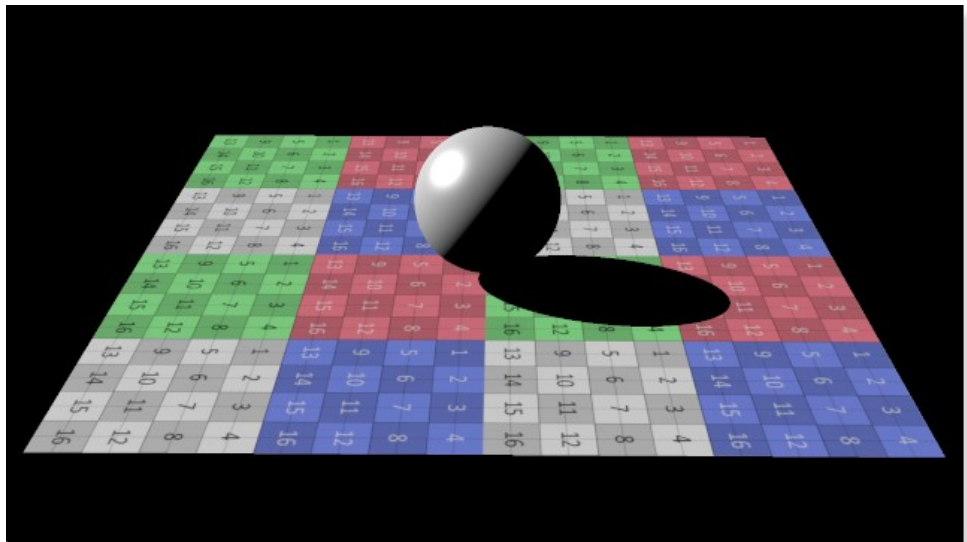
No texture.



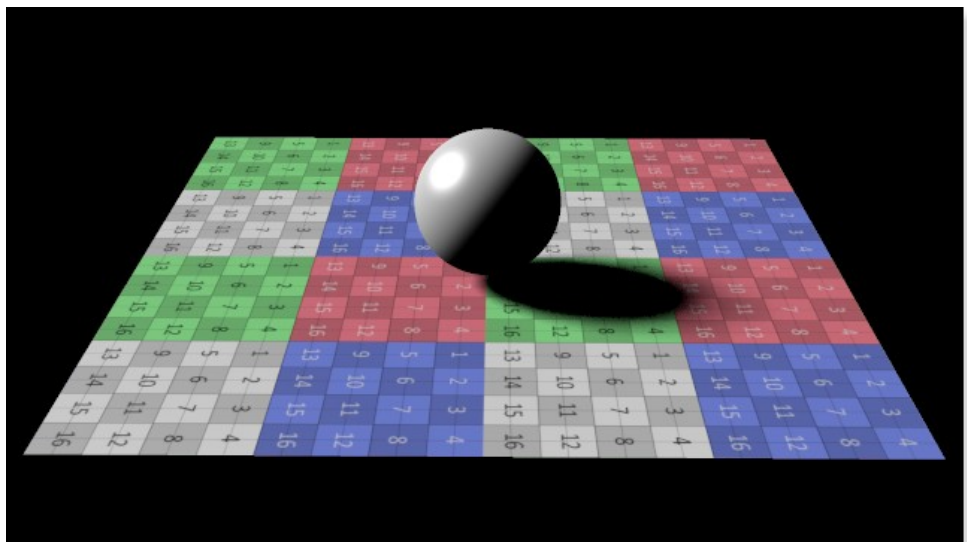
Texturing color.

Distant

A distant light simulates an infinite light such as the Sun. You can turn the distant light into a soft light by increasing *Angle* attribute. *Angle* measures the visual diameter (or angular diameter) of an object as an angle. If you wish to simulate the Sun or the Moon you should set an *Angle* value of 0.5 degrees.



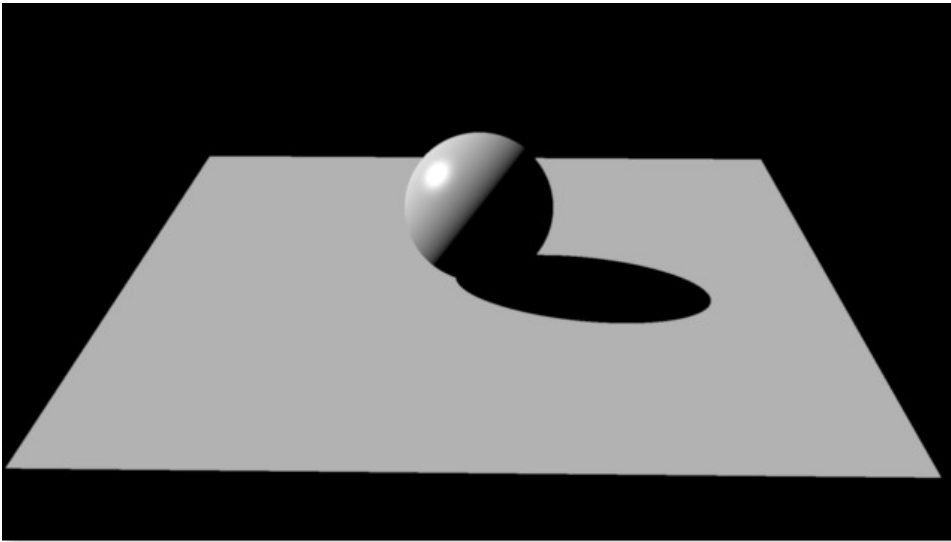
A distant light with an Angle of 0.0



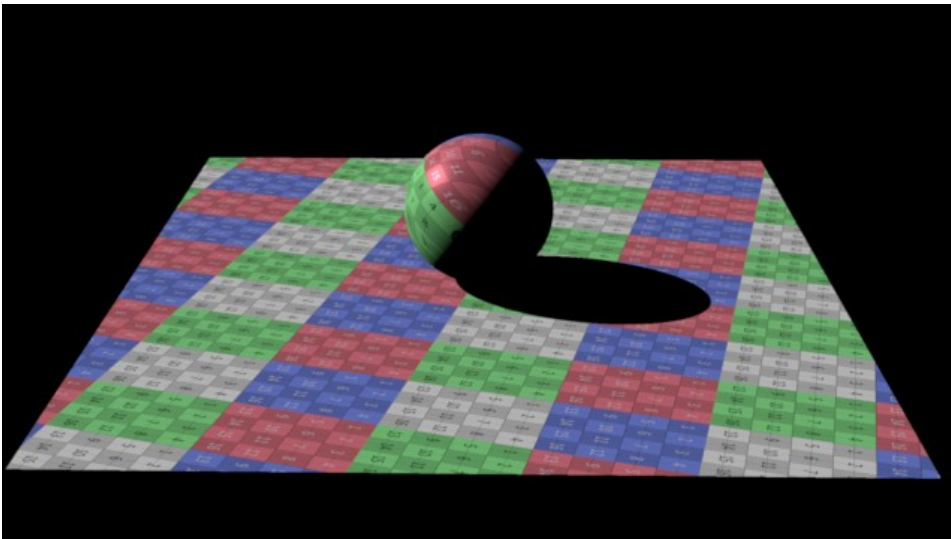
Same distant light with an Angle of 15.0

Texturing

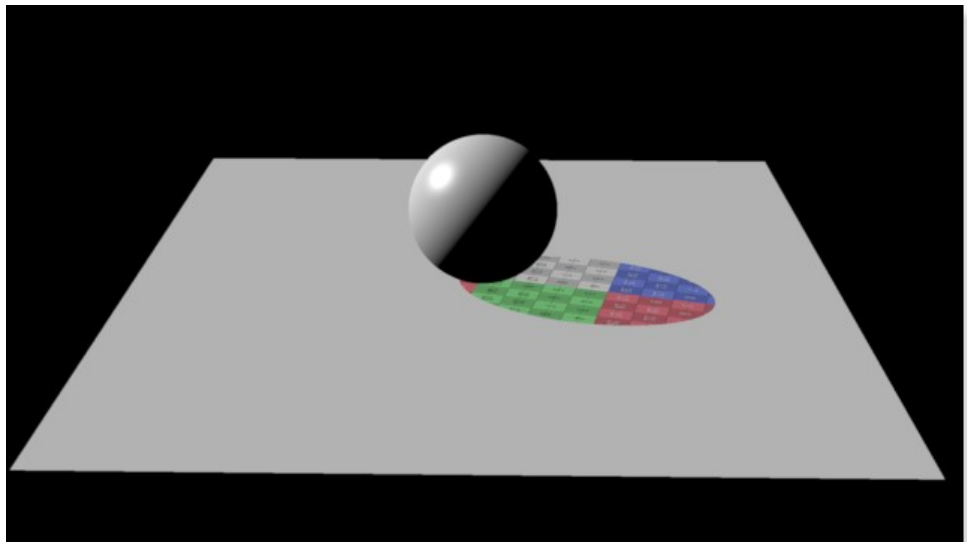
You can texture both *Color* and *Shadow Color* attribute. Remember to set your texture *Projection* to *Parametric*.



No texture.



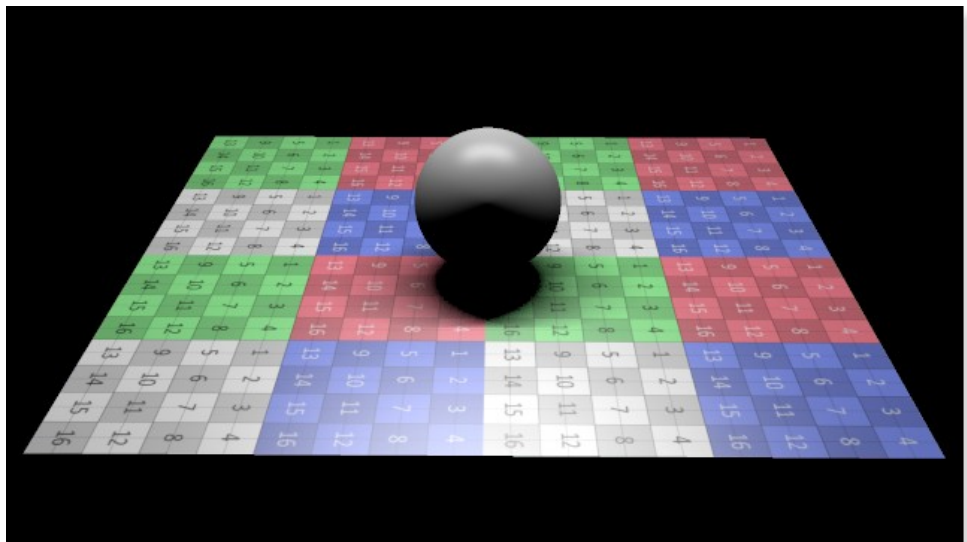
Texturing Color.



Texturing Shadow Color

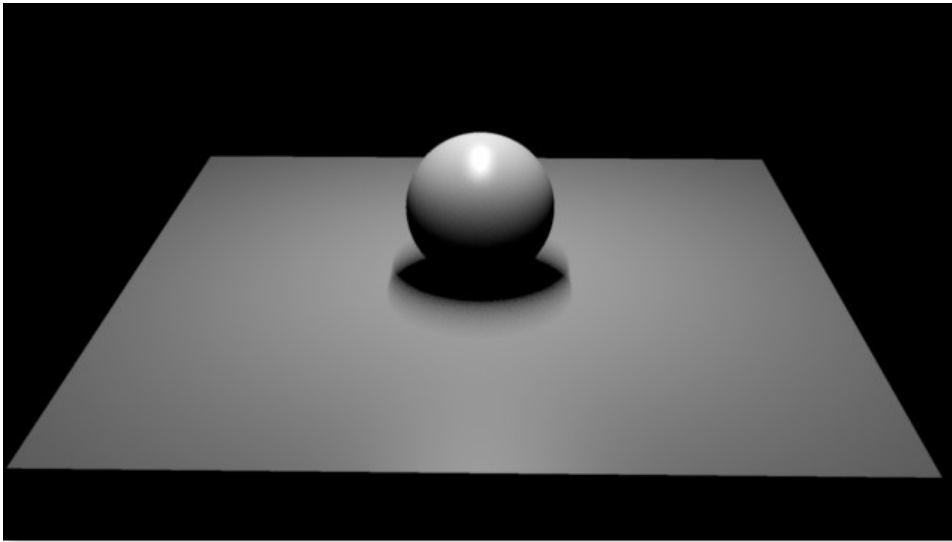
Linear

A linear light simulates an neon light.

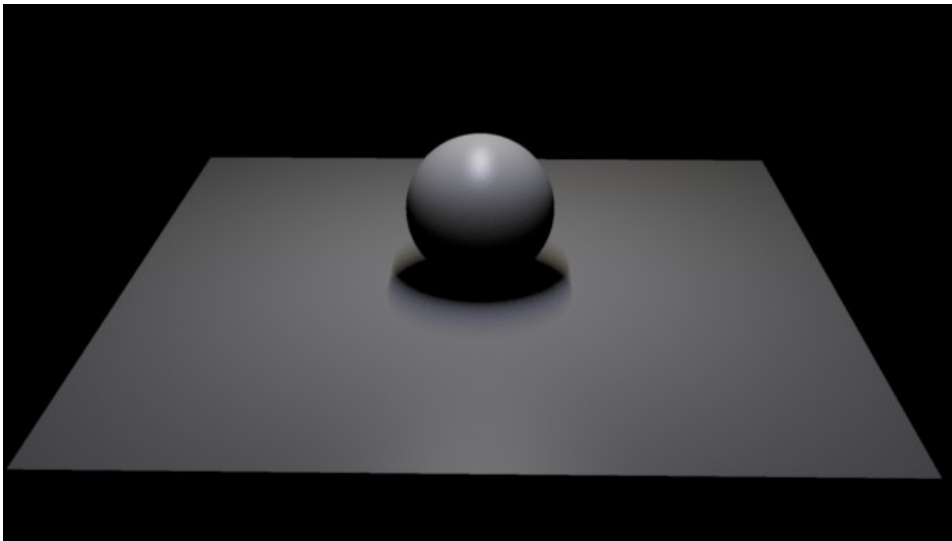


Texturing

You can texture both *Color* and *Shadow Color* attribute. Remember to set your texture *Projection* to *Parametric*.



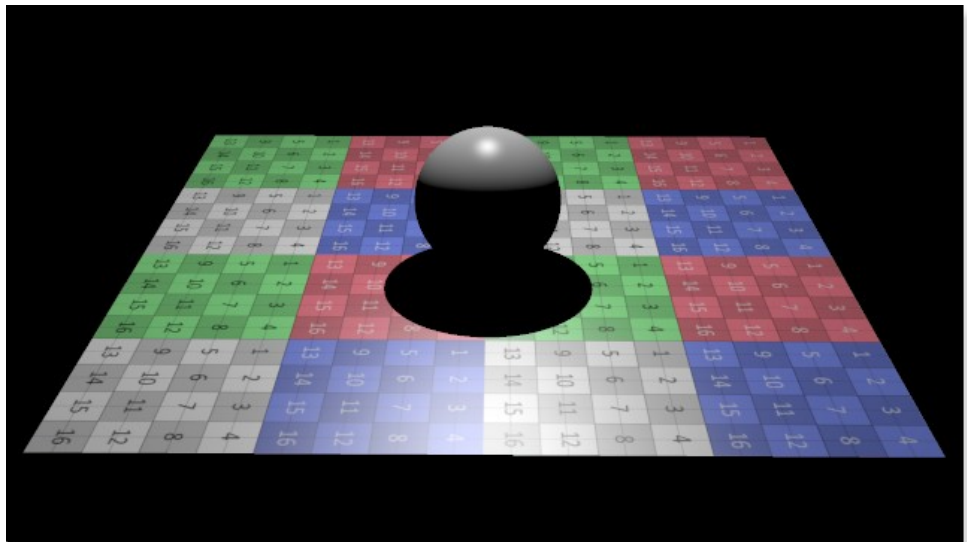
No texture.



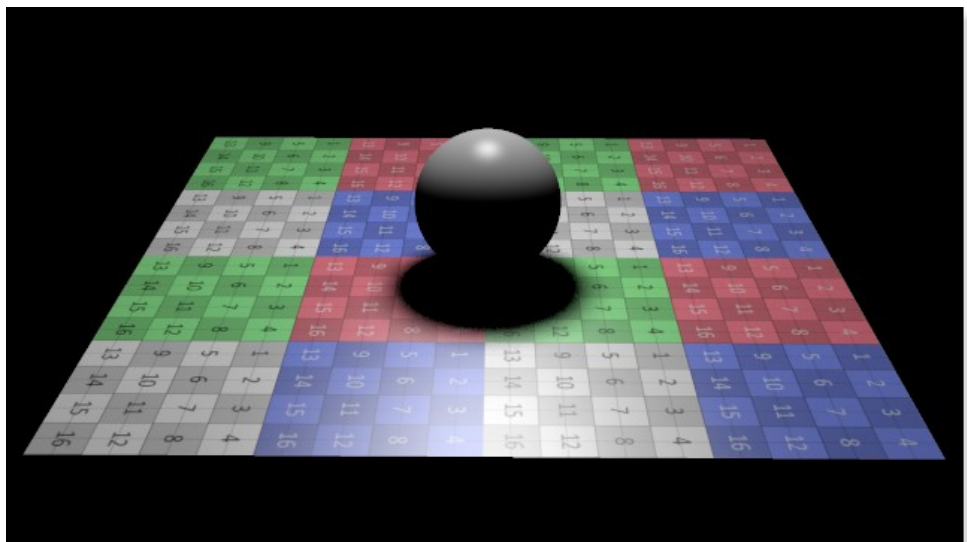
Texturing color

Point

A point light simulates a point source. It illuminates in all directions away from the light source.

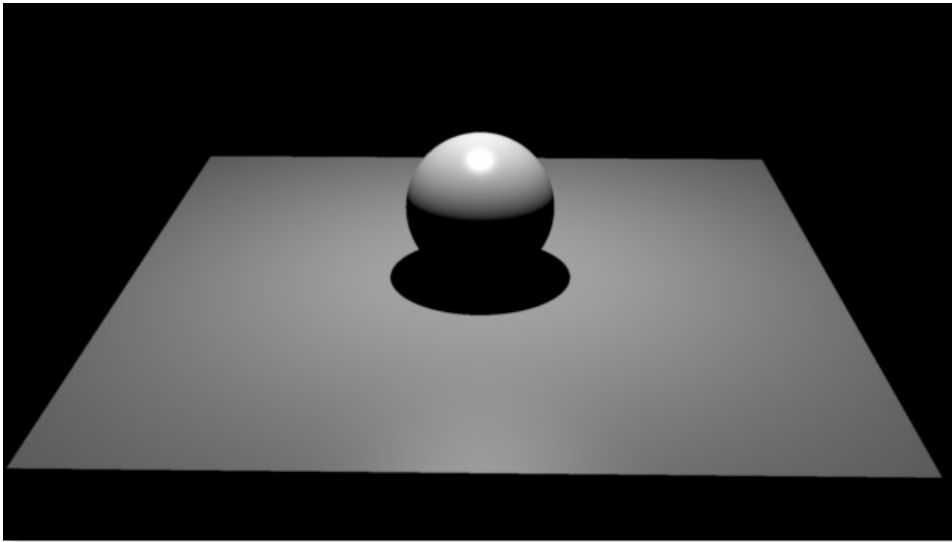


You can turn the point to a spherical light and thus have soft lighting by specifying a value to *Radius* attribute.

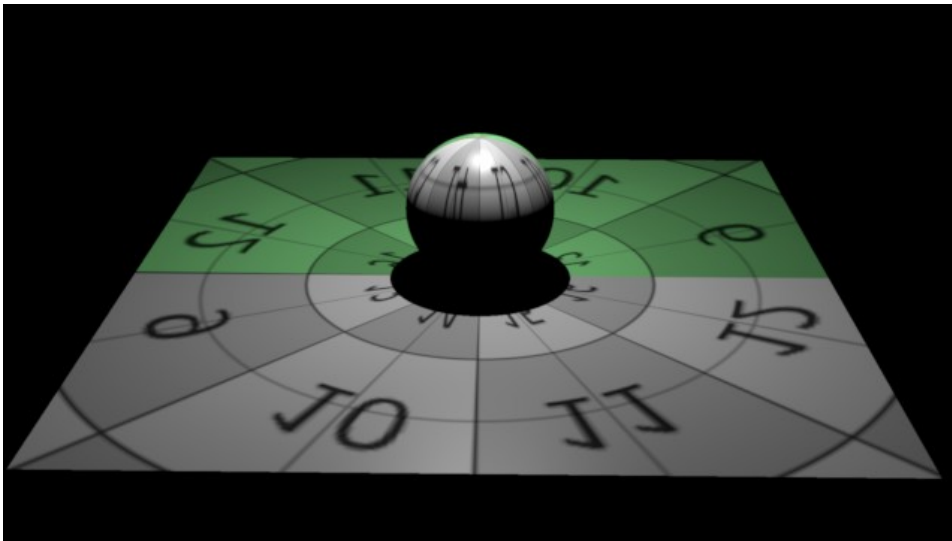


Texturing

You can texture both *Color* and *Shadow Color* attribute. Remember to set your texture *Projection* to *Parametric*.



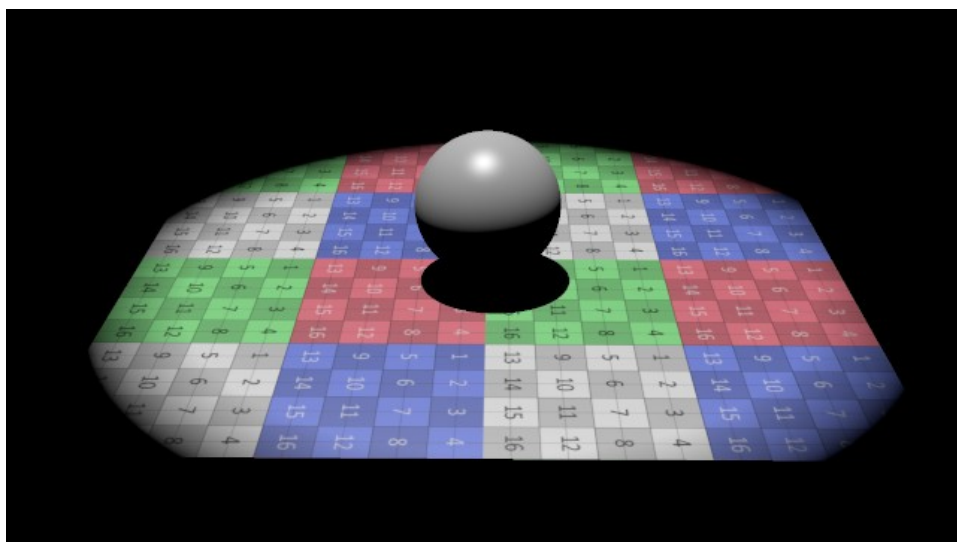
No texture.



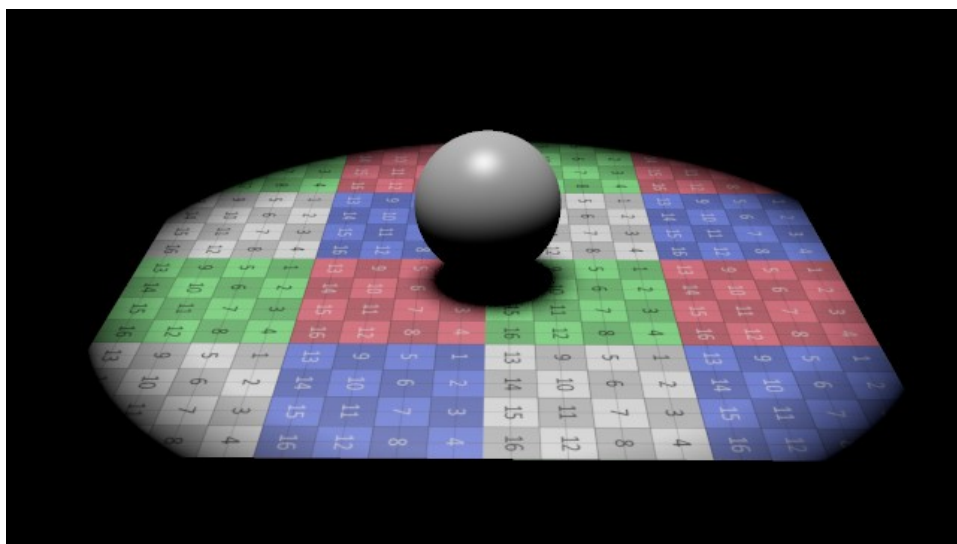
Texturing Color

Spot

A spot light is very similar to a point light. The main difference is its illumination is constrained within a cone. You can control both *Cone Angle* and *Falloff Angle*.

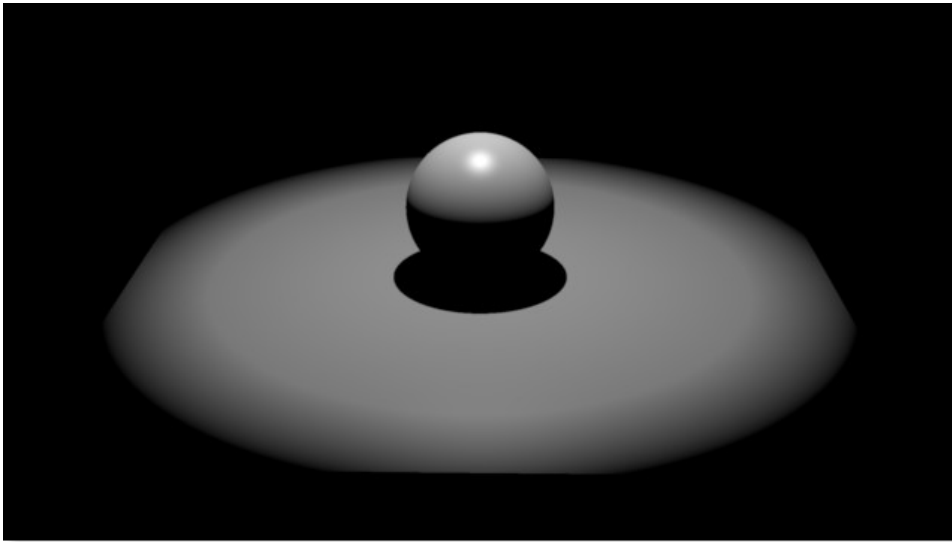


You can enable soft shadows by modifying *Radius* attribute.

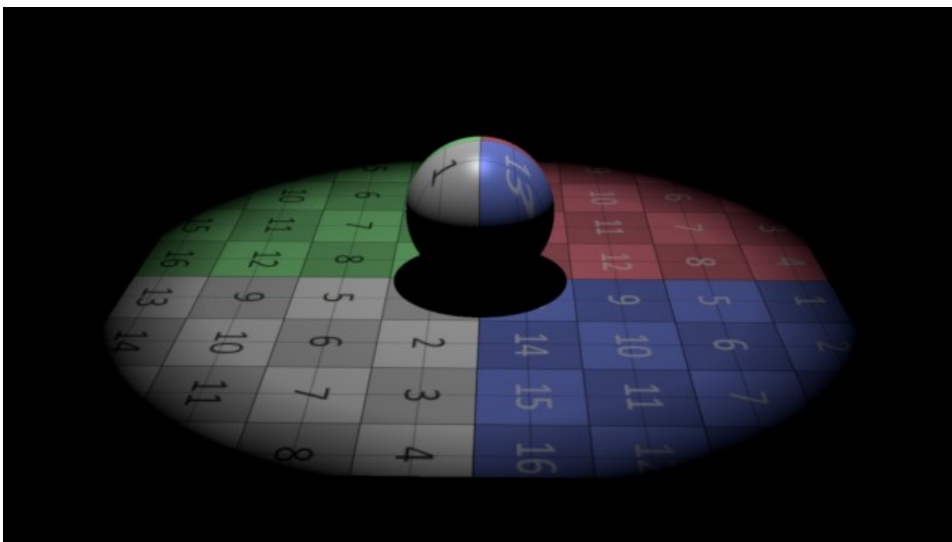


Texturing

You can texture both *Color* and *Shadow Color* attribute. Remember to set your texture *Projection* to *Parametric*.



No texture.



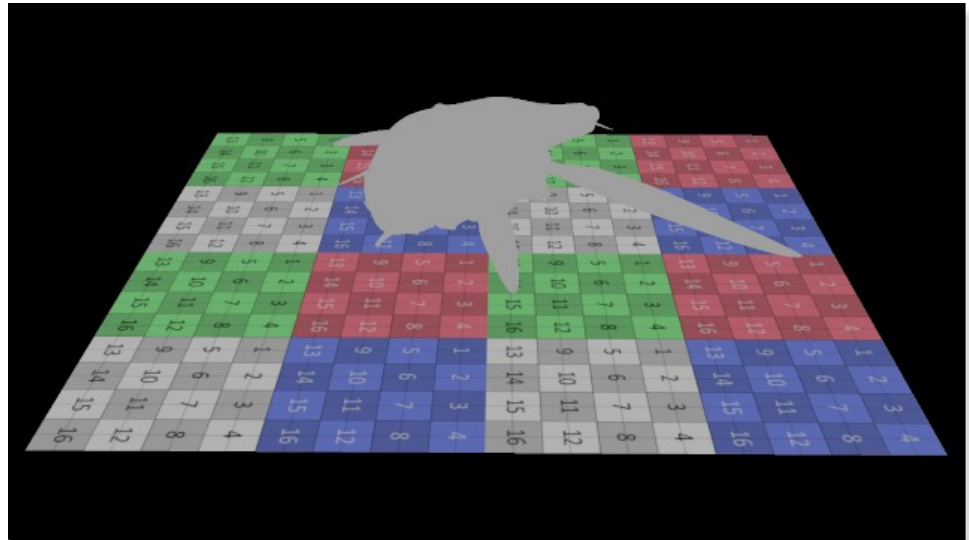
Color texturing.

Non Directional Lights

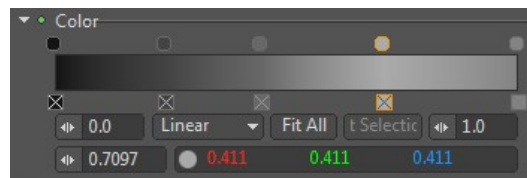
Non Directional lights are lights illuminating points from no major direction. For example an ambient or a dome light. Some shading operations do not evaluate non directional lights. For example in subsurface scattering, non directional lights are skipped during single scattering gathering as it doesn't make any sense. However both directional and non directional lights are evaluated during diffuse scattering.

Ambient

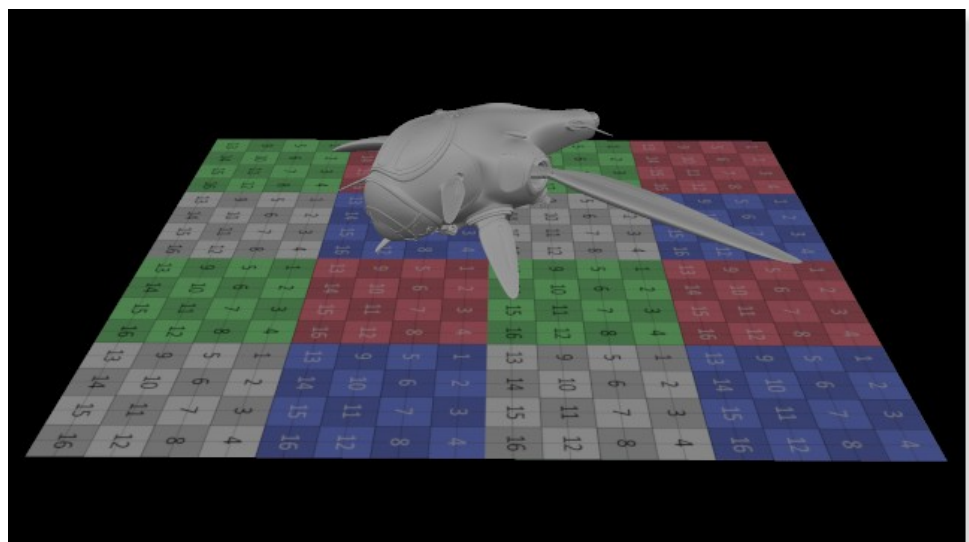
Ambient light affects all scene objects equally. No shadow computation is performed.



Most of the time ambient lighting is prohibited in light rigs as it flattens the image. In Clarisse, however, Ambient *Color* attribute is of type gradient. It allows to control the light color according to scene object surface normals. This is quite handy as it improves the point of using ambient.



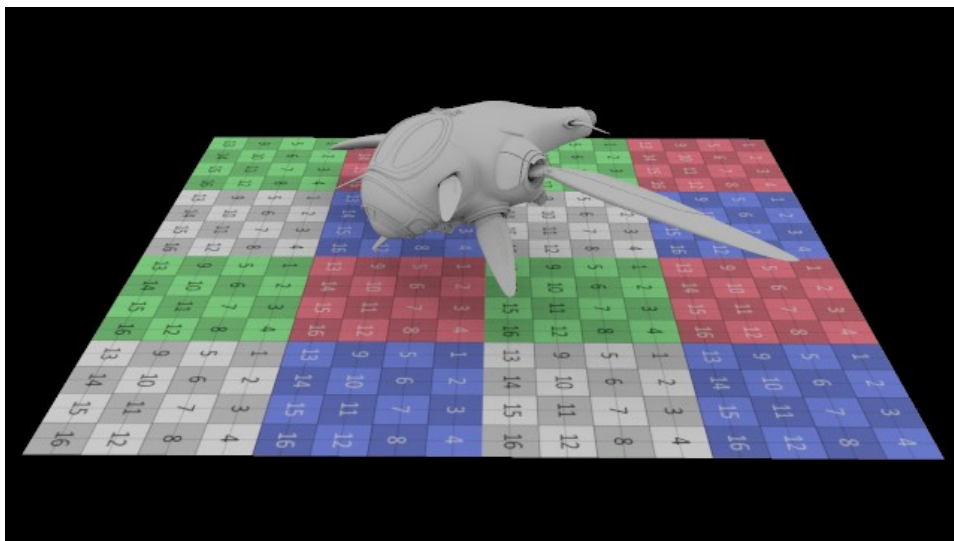
Ambient color gradient



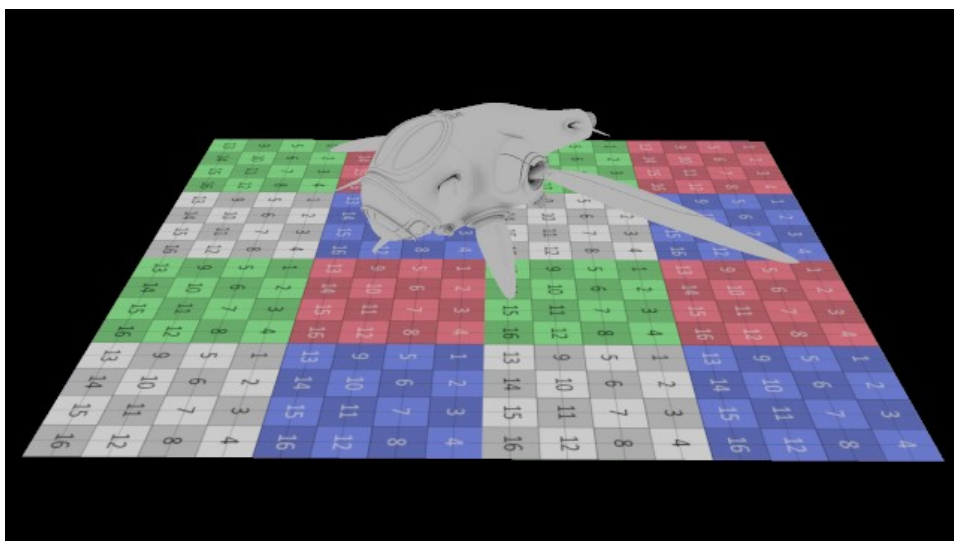
Illumination with ambient only driven with the color gradient

Ambient Occlusion

Ambient Occlusion light is a crude approximation to global illumination. It basically casts occluding rays from an hemisphere located on each surface normal.

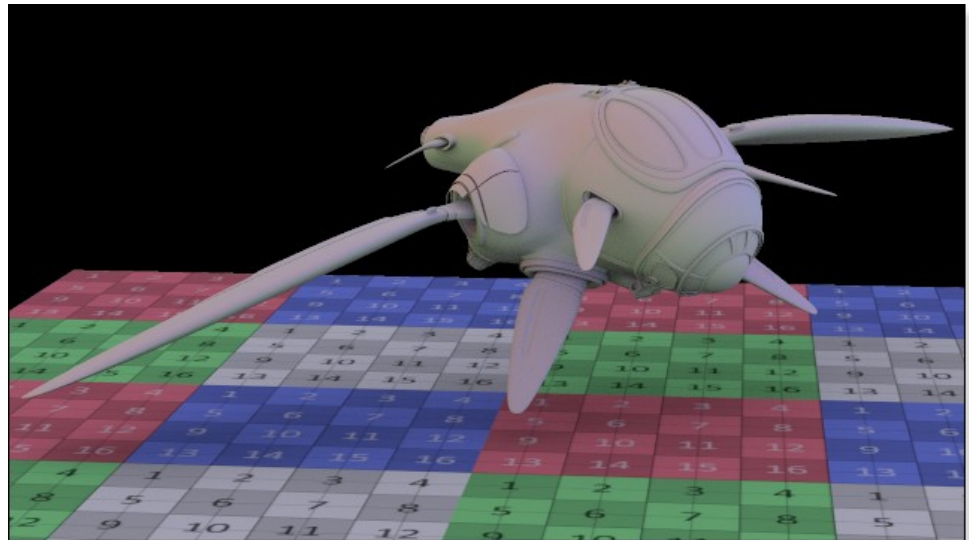


Occluding ray length can be constrained by enabling *Enable Distance Clipping* and setting a max ray length to *Max Distance*. This makes shadows look shorter and also reduce render time.



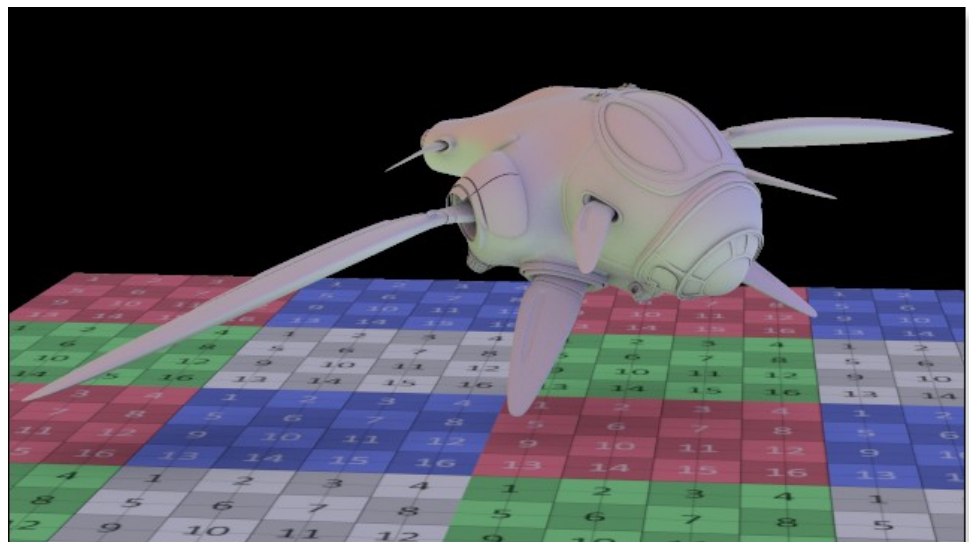
Ambient occlusion with a Max Distance set to 15 cm

Ambient Occlusion *Color* attribute can be textured to be used as an environment light. To properly use a texture for ambient occlusion, make sure texture *Projection* attribute is set to *Parametric*.

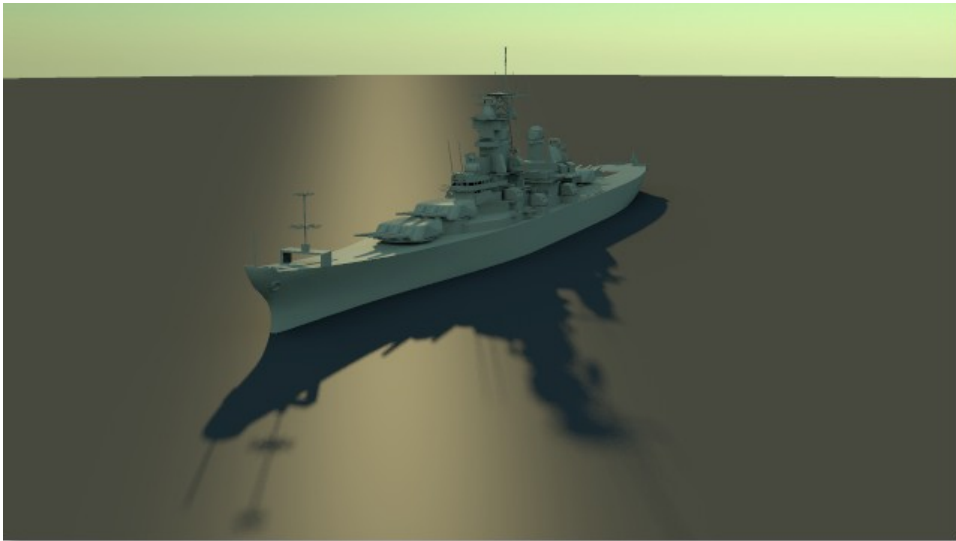


Texture ambient occlusion using the floor texture

In the next example, see how a carefully set Max Distance value gives the illusion that light is bouncing from the floor.



By default, to improve performance, the *Color* texture is only sampled once using surface normal as direction. This approximation works quite well on homogenous texture maps and it supposes that the texture has mip maps. Even though sometimes this approximation can be too crude. Enable *Enable Sample Color Texture* to evaluate the texture for each sample instead.



In this example the ground is under sampled



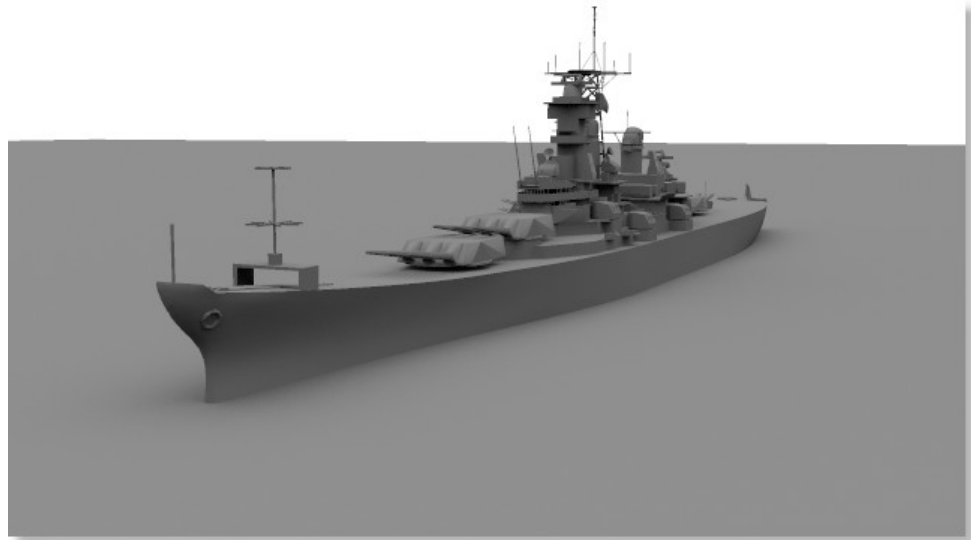
Sample Color Texture on

Tip

To speed up texture evaluation, switch *Mipmap Filtering Mode* to *Trilinear*

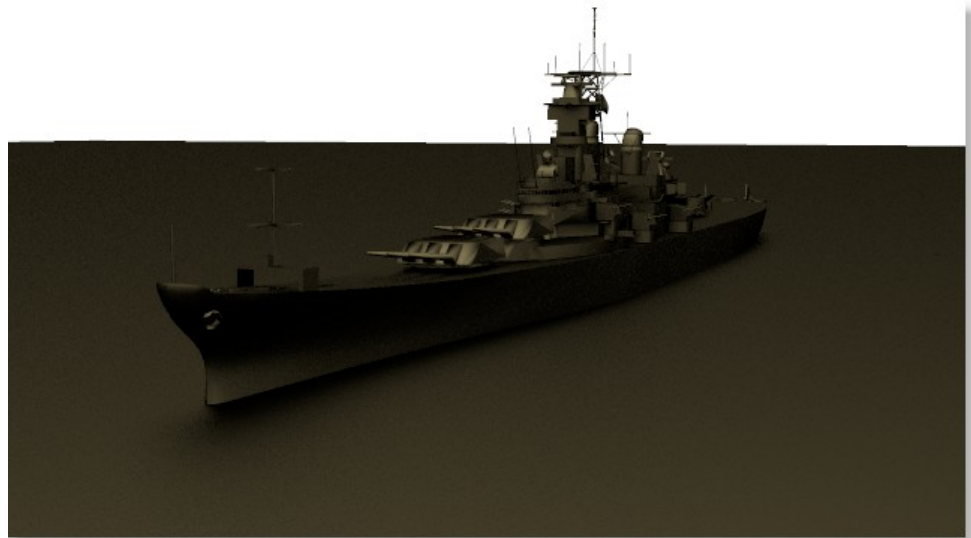
Dome

Dome light simulates illumination coming from the sky. In other packages, this type of light is also called a sky light. It casts occluding rays from an hemisphere aligned to world Y axis.



Dome light diffuse only

The dome light can affect specular lighting.



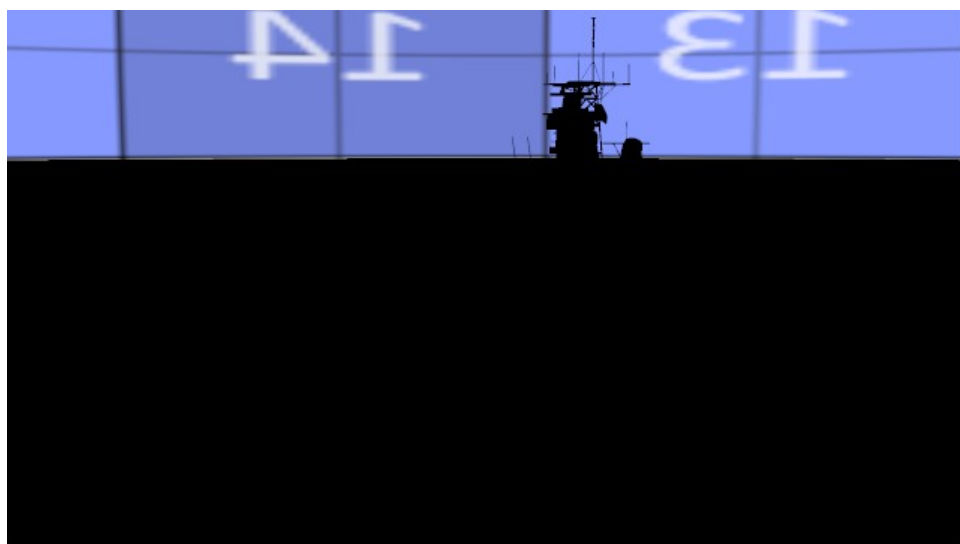
Dome light specular only

Monte Carlo Global Illumination

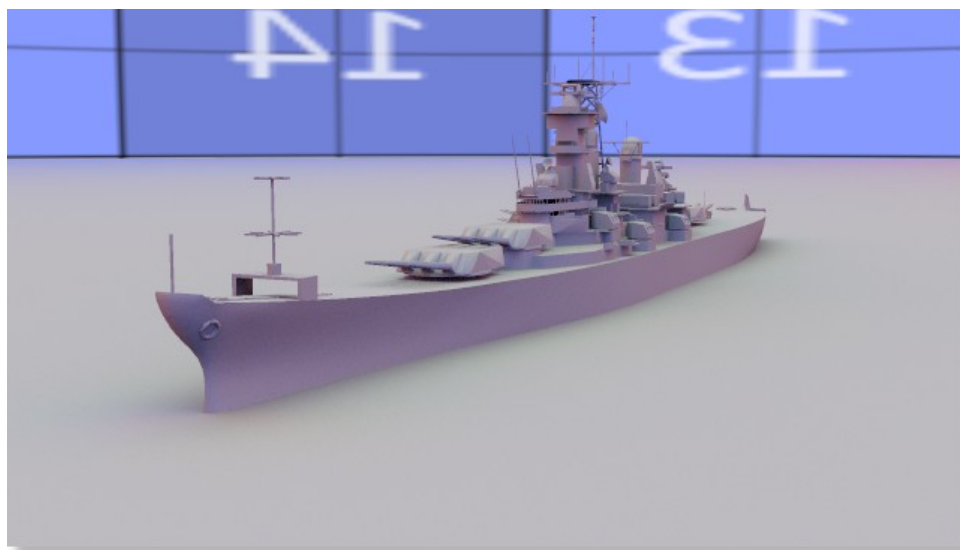
Global illumination tries to simulate realistic lighting by taking into account both direct and indirect lighting. The basic algorithm is to evaluate direct

illumination on a surface and gather indirect illumination by sampling the 3d scene and recursively shading each new sample point to simulate diffuse inter-reflections.

In Clarisse, global illumination is performed by a light. Similarly to ambient occlusion, Monte Carlo GI casts rays from an hemisphere aligned to each surface normal. Each time a ray hits a new surface, it is shaded. If the maximum number of bounces isn't reached (*Max Bounces*), this operation is performed again and again. Global illumination is by definition a slow process which directly depends on lighting, materials and geometric complexity.

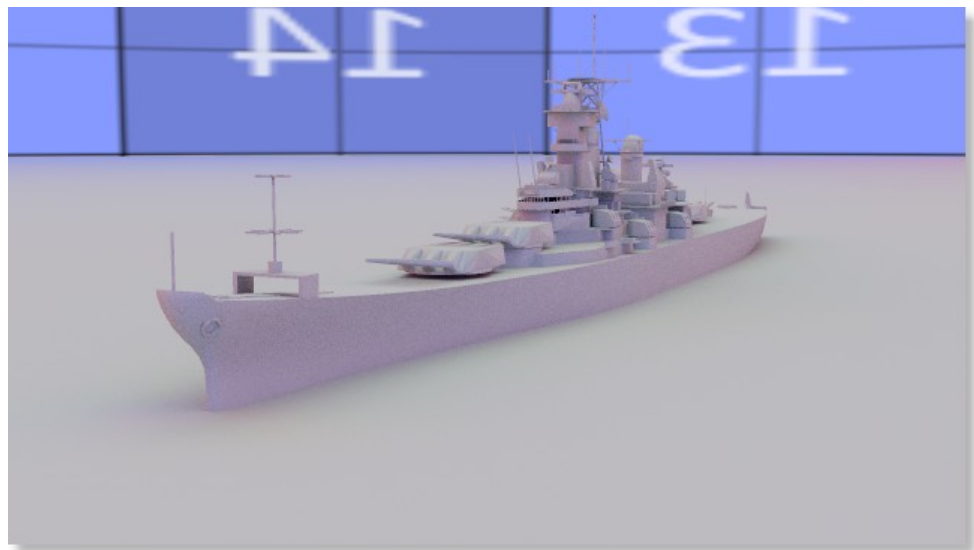


No direct lights, global illumination off

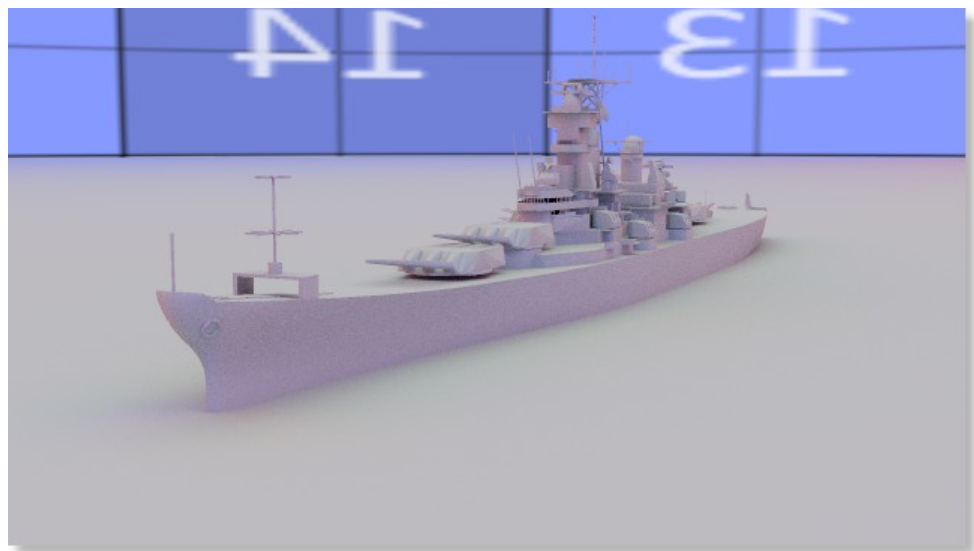


No direct lights, global illumination on

To increase the number of GI bounces, increase the value of *Max Bounces*. Increasing the number of bounces doesn't improve necessary image quality. On the contrary, this can lead to a noisier image and slower rendering times. In most cases, when illumination comes mainly from direct lighting (such as exteriors), a *Max Bounces* value of 1 should be enough. Worst case scenario is *"the light at the end of the tunnel"* where illumination becomes mainly indirect. These are the kinds of cases where *Max Bounces* should be increased.



With 2 bounces



With 8 bounces

Adding direct lights is the best possible way to reduce global illumination noise. It's definitely faster than increasing global illumination sampling.

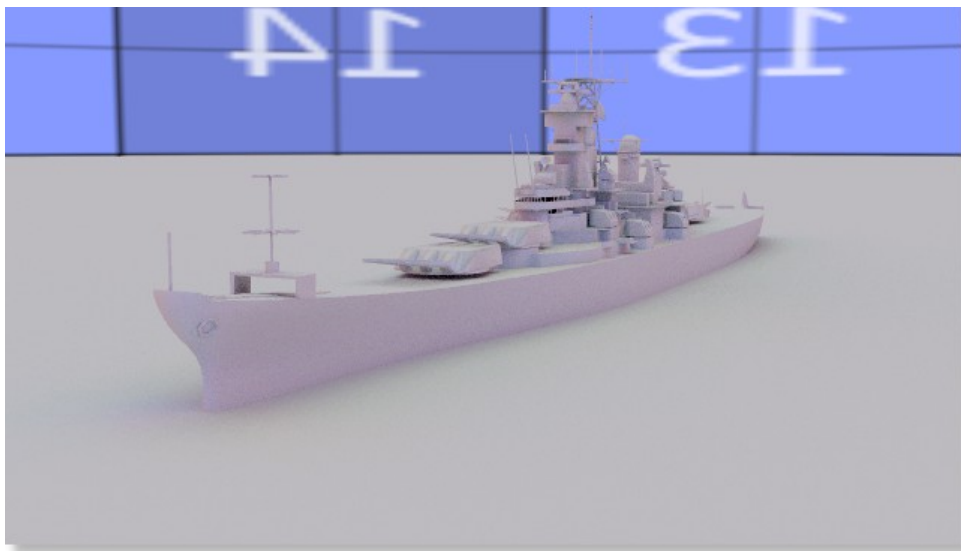


Image based lighting + 1 GI bounce means faster rendering

To speed up rendering you can use light linking to boost global illumination sampling on specific scene objects. You can also specify a geometry group with scene objects assigning simpler material shader to speed things up.

Global illumination makes use of the smart sampling system Clarisse uses. Unlike most raytracers, enabling features such as glossy reflections/refractions, SSS doesn't increase rendering time by an order of magnitude. Clarisse renderer is an hybrid path-ray-tracer.



Two lights: one distant and one Monte Carlo GI rendered in 124s



Same settings with glossy reflections on both floor and blackboard in 158s

Shading and Texturing

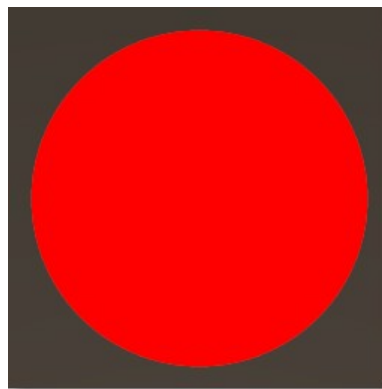
In the following section we will go through the shading and texturing basics. In Clarisse, shading is performed by evaluating materials that are connected to a network of textures. For more information on how to create material shading networks please refer to Using the Material Editor.

Standard Material

There are many different kinds of materials available in Clarisse. Some are specialized for hair rendering, mattes, while Standard material can be of general use. For more information on each material please refer to the Reference section of the HTML documentation.

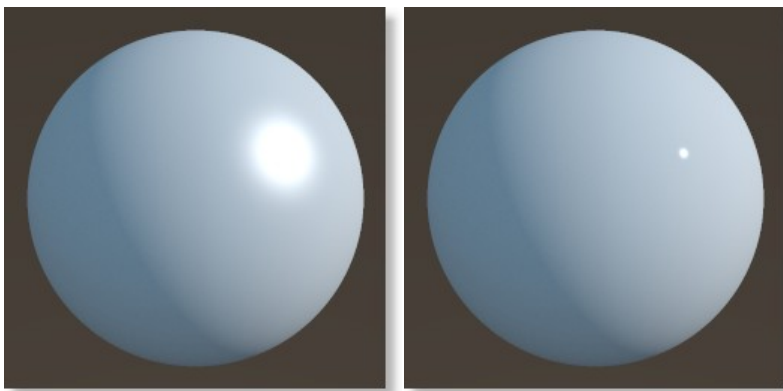
Luminosity

Luminosity color simulates the emission or self illumination color of the material: typically the material of a light.



Specular Reflection

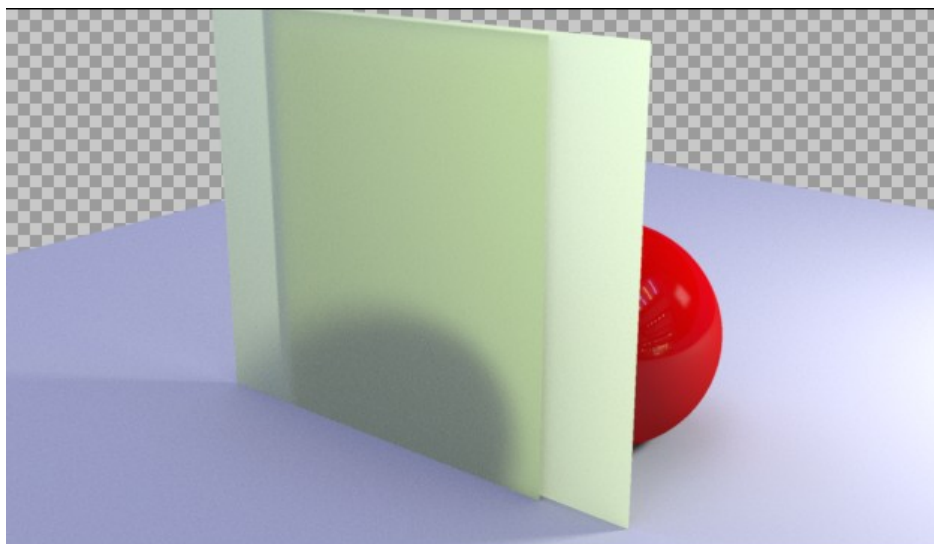
The specular reflection model used in the Standard material is Blinn-Phong model.



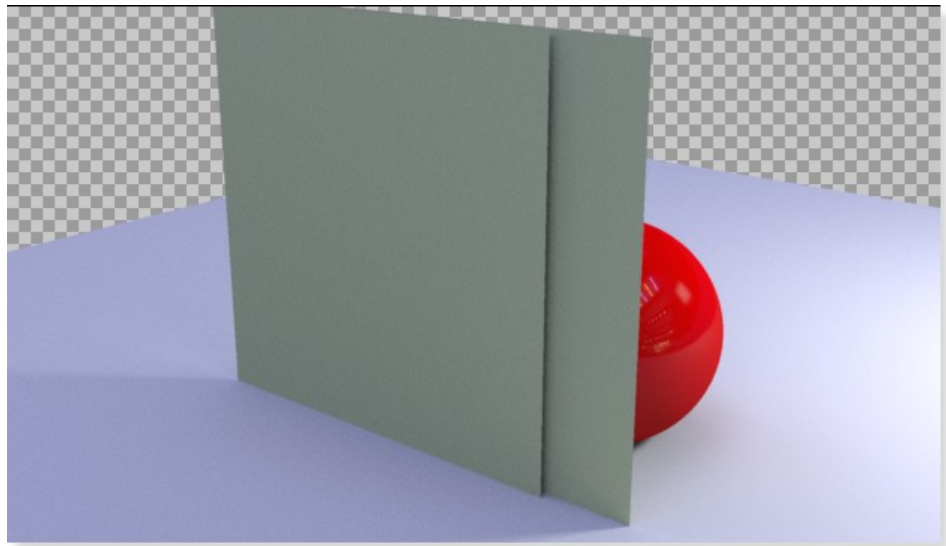
Set *Specularity* to control the color intensity of the specular reflection. *Glossiness* controls the exponent which approximate the surface glossiness. These settings only approximate reflection of lights.

Backlighting

Backlighting simulates thin translucency such as the one that occurs on a piece of paper or a leaf. When you activate back-lighting, illumination is evaluated from both side of the geometry. By default all lights are evaluated during back-lighting computation. You can exclude each light by disabling its *Enable Backlighting* attribute.



Backlighting enabled to simulate 2 pieces of paper



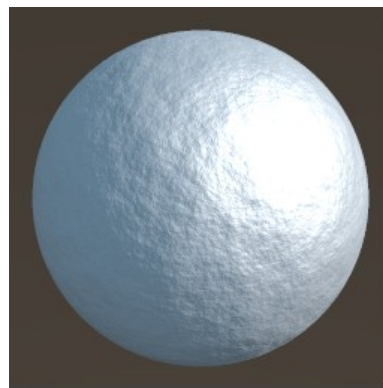
Backlighting disabled

Note

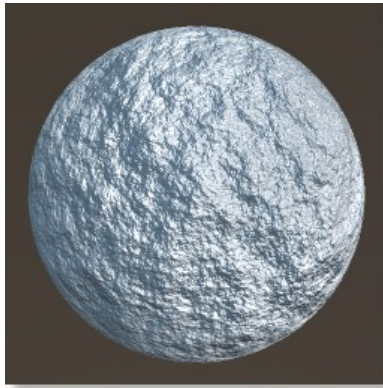
As back-lighting evaluates illumination from both side of the geometry, materials having back-lighting enabled are twice as slow to render.

Bump Mapping

The bump mapping approximates small displacements occurring on a surface. It only affects surface normals, it doesn't actually geometrically displace surfaces. *Bump* value is a height multiplier of textures used for bump mapping.



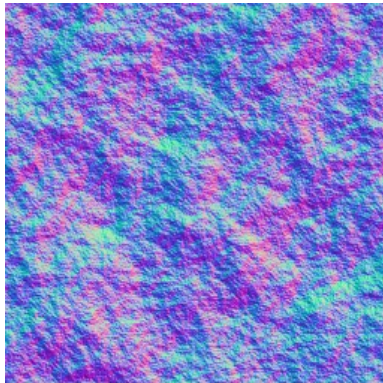
A bump texture of 1 cm



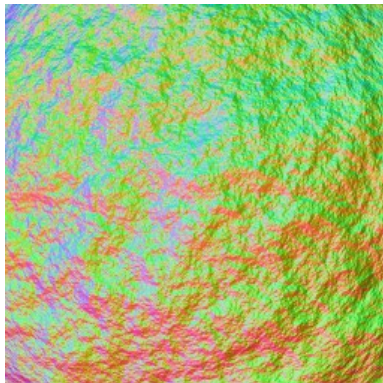
Same bump texture with 10 cm value

Normal Map

You can use normal maps to alter the surface normals of your material. Two of the most widely used modes are supported: tangent and object space maps. Choosing the appropriate mode depends only on the application used to generate the normal map. *Normal Map* attribute blends between surface normal (0%) and normal mapped one (100%)



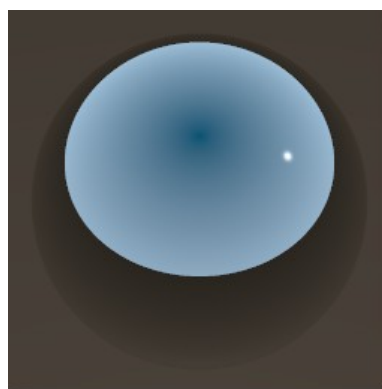
A Tangent space map



An Object space map

Reflection

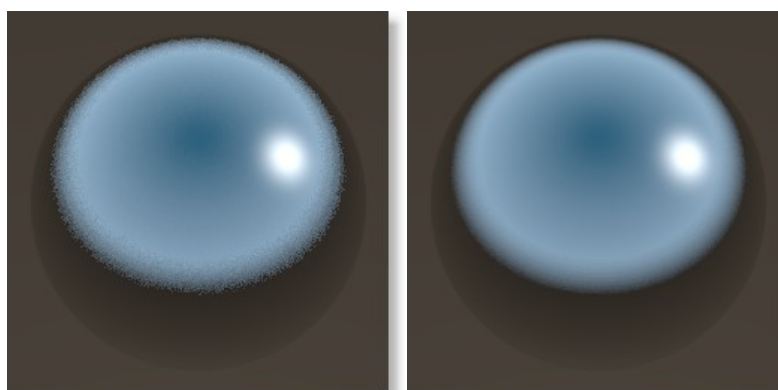
Reflection is performed using raytracing. The reflection model of the Standard material can be controlled using *Energy Mode* attribute. Please refer to Energy Mode section for more information. The number of maximum reflection bounces can be controlled using the attribute *Max Reflection Depth*. If this value is set higher than the one set for the renderer, renderer value will clamp the material one.



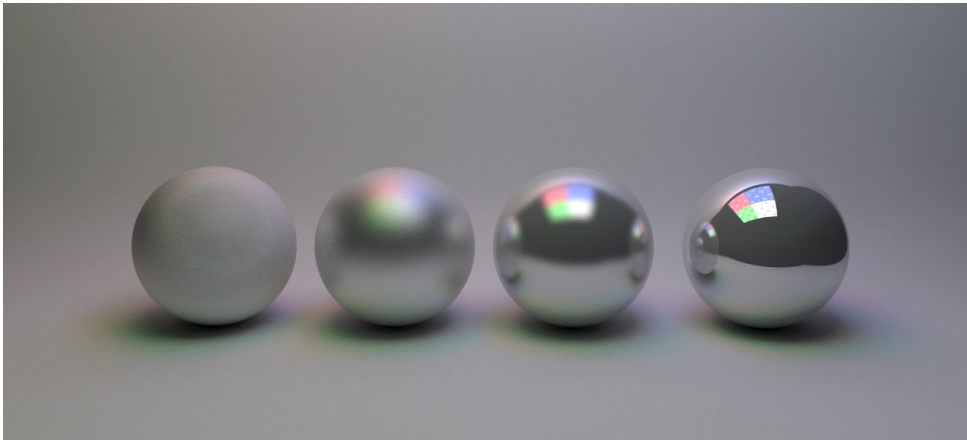
Raytraced reflection

You can simulate glossy reflection by modifying *Reflection Glossiness* attribute. *Reflection Glossiness* controls how glossy the surface is. At 100% the reflection behaves like a perfect mirror, where at 0% the surface is completely rough (microfacets model).

Reflection Quality controls the number of samples used for the reflection. The actual number of samples used for reflection is the square of the *Reflection Quality* value. Please note the higher number of samples, the slower the rendering gets.



Reflection Quality of 3 (9 samples) on the left and 8 (64 samples) on the right

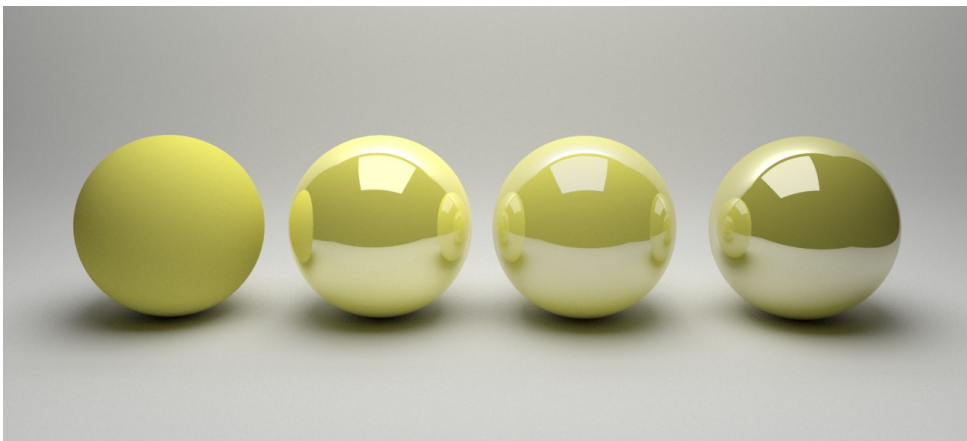


Glossiness at 0%, 33%, 66%, no glossy (from left to right)

Energy Mode

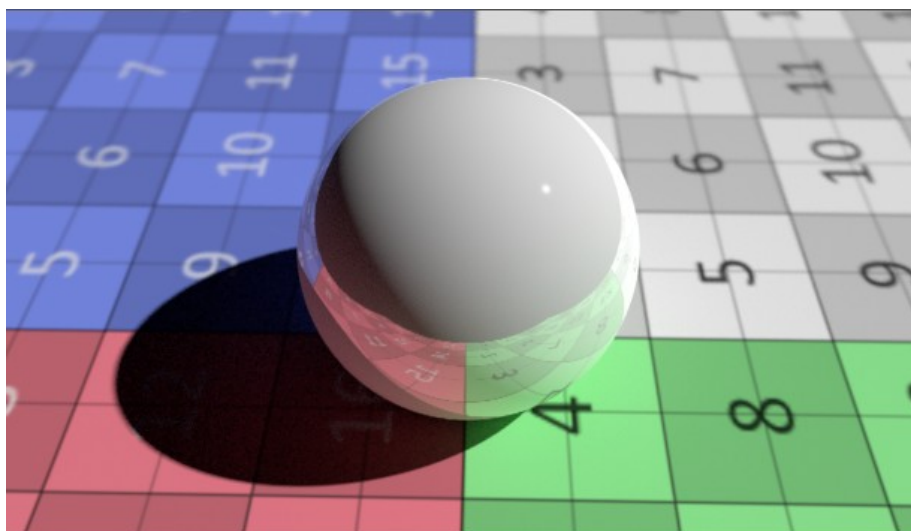
Energy Mode allows you to control how reflection and diffuse blend. There are several modes available and by default reflection is set to *Conservative (Value)*

Mode	Description
Additive	Simply adds the result of reflection to the diffuse.
Conservative (Raytrace)	Evaluated diffuse (final emitted diffuse) is altered during rendering to ensure the material doesn't emit more light than it receives.
Conservative (Value)	Diffuse value intensity is altered during rendering to ensure the material doesn't emit more light than it receives.

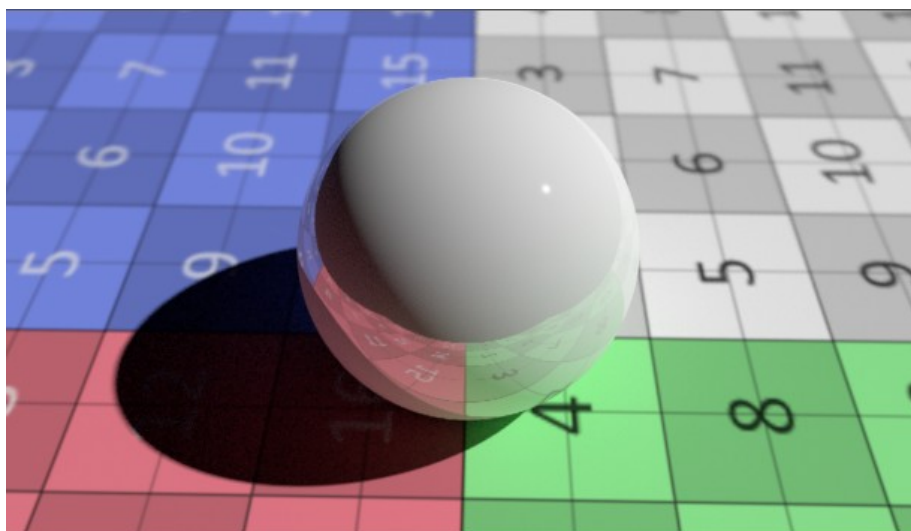


Diffuse reference, Additive, Conservative (Raytrace), Conservative (Value) (from left to right)

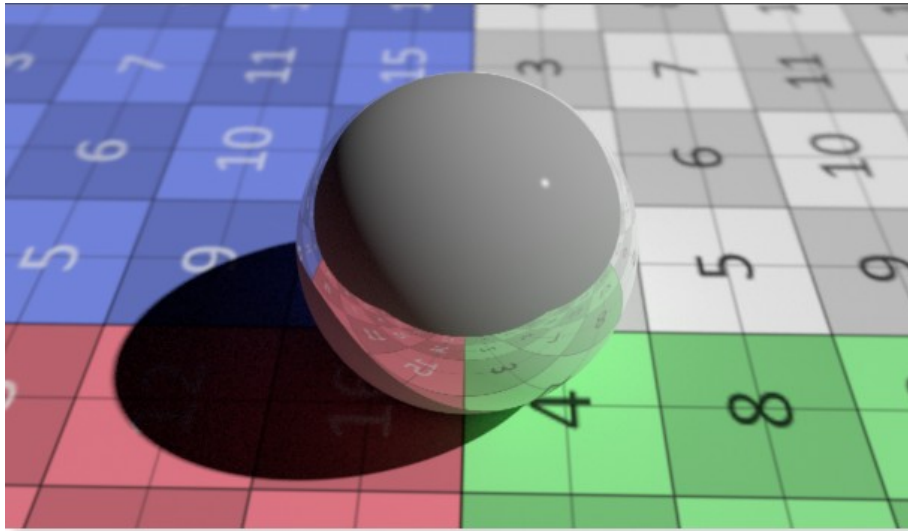
The following images have all their reflection set to 50% and their diffuse set to 78%.



Energy mode: Additive. The sphere is overexposed on its right



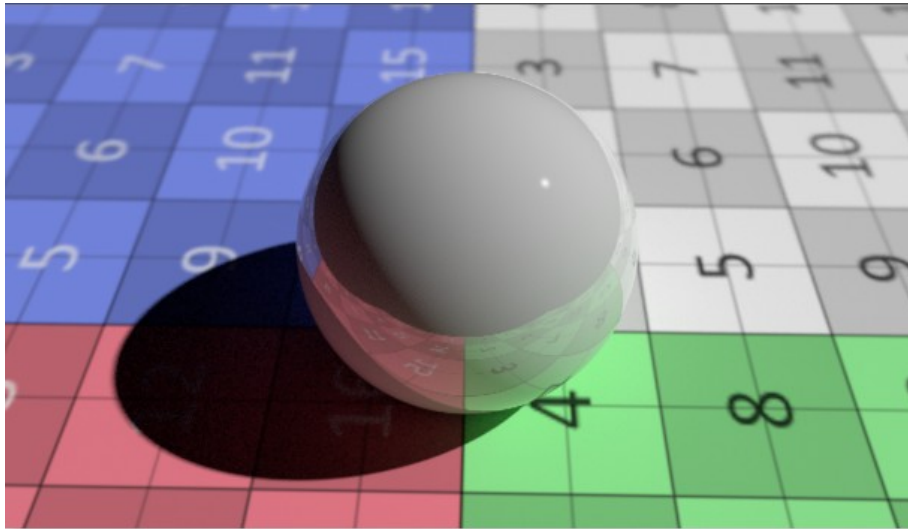
Energy mode: Conservative (Raytrace). The sphere overexposure is gone.



Energy mode: Conservative (Value). Sphere diffuse value has been automatically reduced.

BRDF and Fresnel

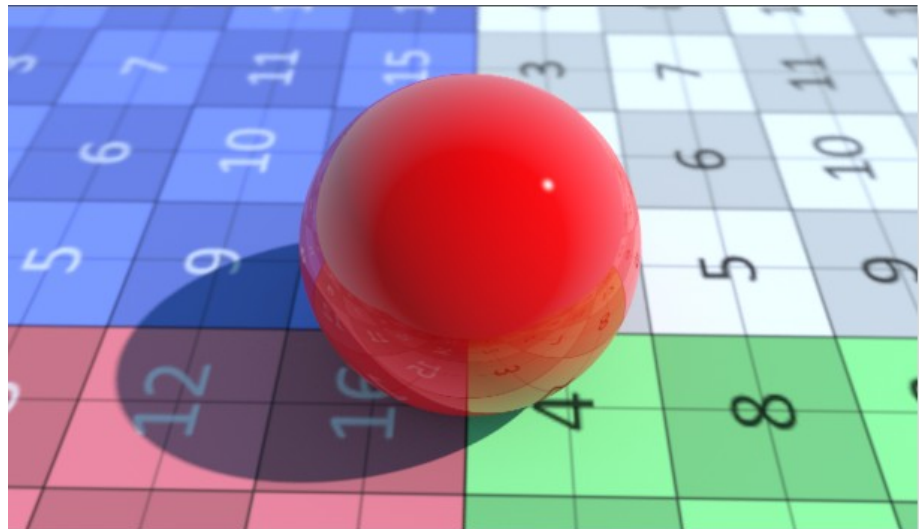
You can control the BRDF of your materials to affect both reflection and refraction. By checking *Use Fresnel BRDF*, the BRDF (dielectric model) is automatically computed according to the camera ray incidence angle and to the *Index Of Refraction*. The dielectric model is more fitted to refractive materials, to you use it for reflection, you'll have to set a high *Index Of Refraction* values.



Fresnel BRDF on a reflective material (IOR set to 3.0)



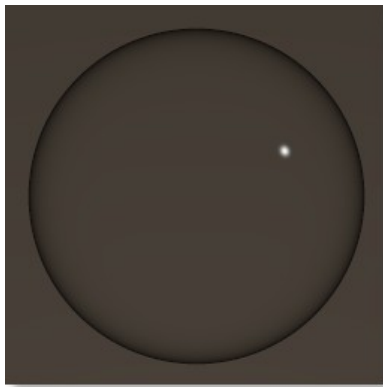
Fresnel BRDF on a refractive material (IOR set to 1.5)



Custom BRDF on a reflective material

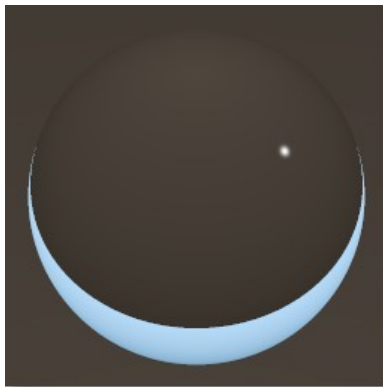
Transparency and Refraction

Transparency and refraction is performed using raytracing. If the *Index Of Refraction* is set to one, then the material is performing transparency. Transparency depth is controlled through the renderer settings. Please refer to Understanding Transparency section.



Transparency on incidence

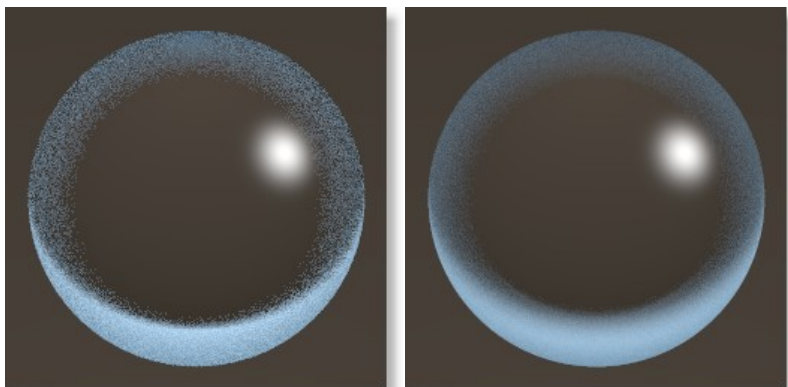
When *Index Of Refraction* is different from one, material performs refraction. The number of maximum refraction bounces is controlled using *Max Refraction Depth*. If this value is higher than the one set in the renderer, the renderer value will clamp the material one.



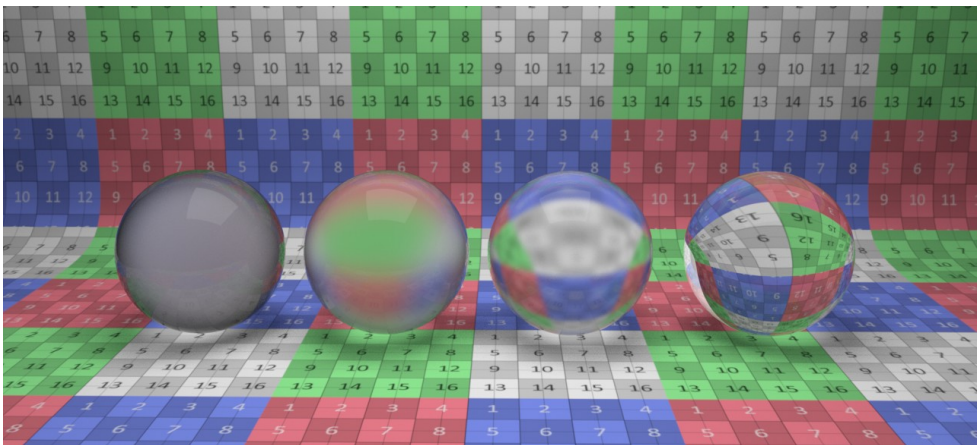
Raytraced reflection

You can simulate glossy refraction by modifying *Refraction Glossiness* attribute. *Refraction Glossiness* controls how glossy the surface is.

Refraction Quality controls the number of samples used for the refraction. The actual number of samples used for refraction is the square of the *Refraction Quality* value. Please note the higher number of samples, the slower the rendering gets.



Refraction Quality of 3 (9 samples) on the left and 8 (64 samples) on the right

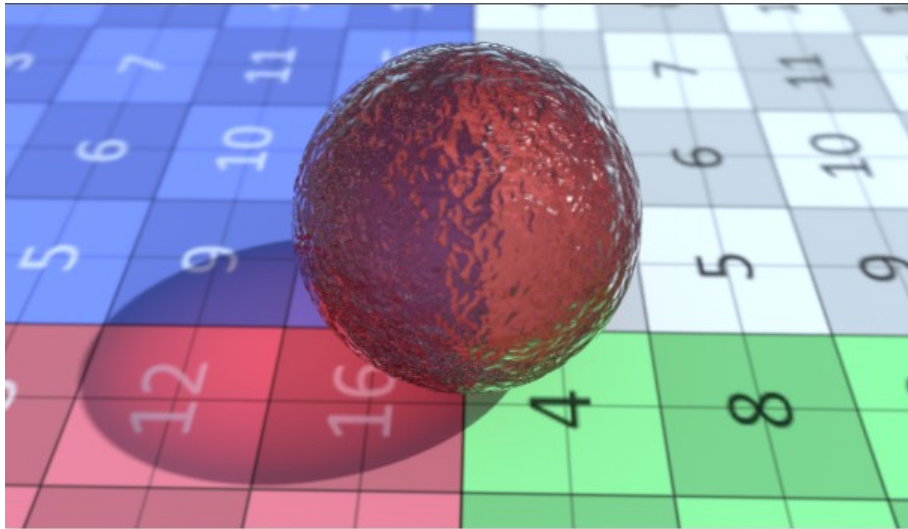


Glossiness at 0%, 33%, 66%, no glossy (from left to right)

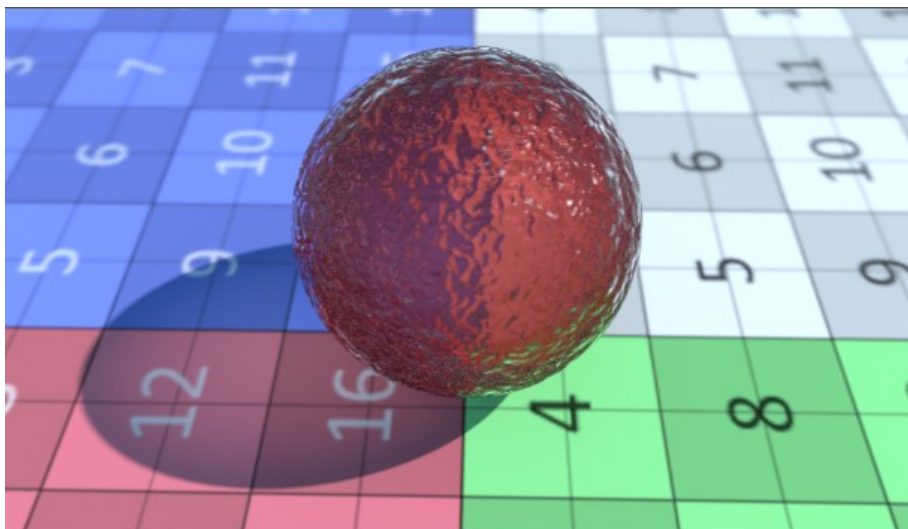
Transparency Mode

Transparency Mode allows you to control how transparent (or refractive) shadow computations are performed at the material level. There are several modes available and by default it is set to *Alpha Only*.

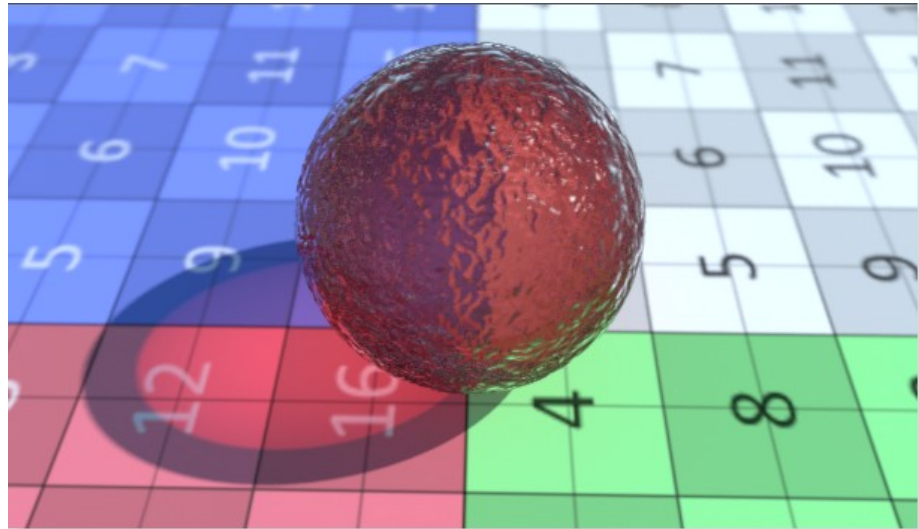
Mode	Description
Legacy	Simply evaluate the opacity of the material. This mode is deprecated since Clarisse 1.5.
Alpha Only	Evaluate the opacity and the color of the transparency. (Fastest)
Use BRDF	Evaluate the BRDF and the color of the transparency.
Use BRDF with bump	Evaluate the bump and normal map channels then the corresponding BRDF and color. (Slowest)



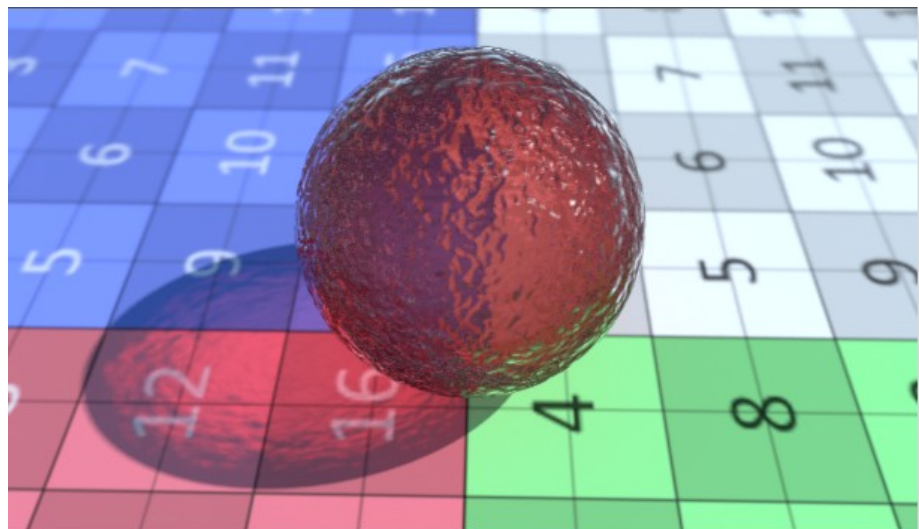
Alpha Only. The shadow is colored according to the transparency texture.



Legacy. The shadow matches the opacity but isn't colored.



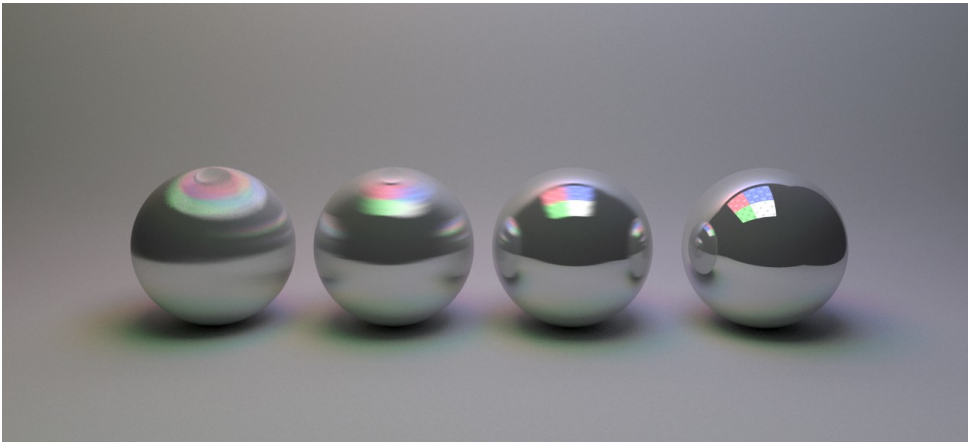
Use BRDF. The shadow opacity matches the BRDF.



Use BRDF with bump. Note how the bump affects the shadow.

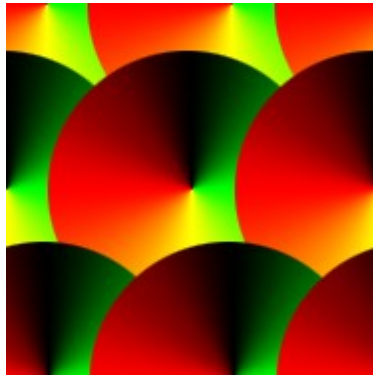
Anisotropy

Anisotropy and *Anisotropy Rotation* allow you to simulate microfacet model. *Anisotropy* specifies the stretch factor in local axis and *Anisotropy Rotation* specifies the major direction of microfacets. *Anisotropy* affects reflection, specular and refraction.

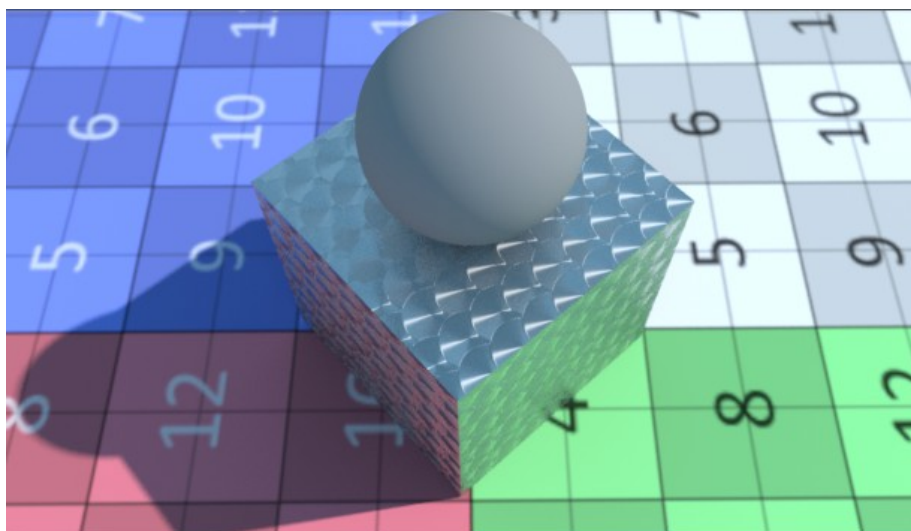


Anisotropy at 100% glossiness at 0%, 33%, 66% off (from left to right).
Note how the size of the anisotropy depends on the glossiness value.

Anisotropy Rotation can be textured with a special texture map in which its Red and Green channels (X and Y) provide the anisotropy directions.



Anisotropy direction map



Glossy Anisotropic Reflection



Glossy Anisotropy Refraction

Note

When importing anisotropy maps, remember to disable both linearize and pre-multiply flags.

Subsurface Scattering (SSS)

Subsurface scattering simulates the transport of light through a translucent

surface. The light is then scattered and reflected several times under the surface before exiting. For thin translucency such as the one that occurs on a leaf or a piece of paper, we recommend you to use Backlighting instead.

SSS is performed in pure brute force raytracing. It supports both single and diffuse scattering and it is quite expensive.

Attribute	Description
Enable SSS	Enables subsurface scattering
SSS Single Intensity	Sets the intensity of the single scattering
SSS Diffuse Intensity	Sets the intensity of the diffuse scattering
SSS Ior	Sets the SSS index of refraction
SSS Mean Cosine	Sets the main direction to which the light is scattered inside the material. A negative value defines a backward direction whereas a positive value defines a forward direction. A null value defines an isotropic scattering
SSS Scale	Sets the surface thickness scale
SSS Light Oversampling	Multiplies the quality of lights for each SSS samples. The higher the value, the slower the rendering.
SSS Single Quality	Sets the number of samples used for single scattering.
SSS Diffuse Quality	Sets the number of samples used for diffuse scattering. Diffuse scattering is noticeably noisier than single scattering. To get the same amount of noise as for single scattering, the quality value needs to be doubled (4 times the number of samples)



Global Illumination and SSS enabled (model and texture courtesy of Ten 24)



Same settings without SSS

Single and Diffuse Scattering

The SSS model provided in the Standard material is based on the BSSRDF model (Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy and Pat Hanrahan: "A Practical Model for Subsurface Light Transport". Proceedings of SIGGRAPH'2001).

This model is the combination of two solutions: an exact single scattering with an approximate dipole diffusion for multiple scattering.



Both Single and Diffuse scattering.

Single scattering simulates light bouncing once inside the surface. The Single scattering only evaluates directional lights. In other words, non-directional lights such as GI, do not contribute to the single scattering component.



Single Scattering only (SSS Diffuse Intensity at 0%)

The Diffuse scattering component (multiple scattering) evaluates all lights. It approximates light bouncing under the surface several times. This component is more computationally expensive. It requires twice the number of samples needed by the single scattering component.



Diffuse Scattering only (SSS Single Intensity at 0%)

SSS Quality

It can be really tricky to identify why there's noise on a material that has SSS. The noise may either come from a too low value of light quality, Single SSS or Diffuse SSS Quality. In the worst case scenario, it can be a bit of everything. Following are recommended guidelines to help you fighting noise while keeping low render times.

The best way to start fighting excessive noise in an image is to disable SSS. As a general rule, if there's noise in the diffuse, then lights are under sampled. No matter the value you'll set in SSS Quality, you'll likely have noise in your render. Moreover SSS tends to accentuate subtle noise in the diffuse.

Note

Raising SSS Quality may improve the noise due to under sampled lights. However it comes at a great cost. Improving light sampling during SSS is about 10 times more expensive than raising light quality.



Figure 1 (diffuse only)

In the previous image rendered in 35s (figure 1), we can see that lights are under sampled. Noise is clearly visible. In the next image (figure 2), rendered in 60s, increasing light sampling makes the noise more acceptable.



Figure 2 (diffuse only)

Now, you can enable SSS to start tweaking SSS Quality. We highly recommend to set Single and Diffuse separately. This can be easily done by setting either *SSS Single* or *SSS Diffuse Intensity* to 0%. In the next image series we've already set *SSS Single Quality*. We focus only on *SSS Diffuse Quality*. The trick is to start with a low *SSS Diffuse Quality* value. Then, continue raising the value until there is a very small visual difference between two sets of values.



Figure 3 SSS Diffuse Quality 4 (101s)



Figure 4 SSS Diffuse Quality 6 (104s)



Figure 5 SSS Diffuse Quality 8 (109s)



Figure 6 SSS Diffuse Quality 10 (115s)

Here, a value of 8 seems to be a good one. The noise difference between figure 5 and figure 6 is pretty small.

SSS Light Oversampling

As you can read in Shading Oversampling section, Clarisse renderer tries to avoid ray explosion (as well as explosion of render time). This is why the renderer recycles samples and rays as far as possible.

However, depending on the number of light samples, SSS samples and anti aliasing samples, this behavior can lead, in some rare cases, to poor sampling. This is why we've introduced the *SSS Light Oversampling* attribute. Quite similarly to *Shading Oversampling*, this attribute allows you to control the number of samples used by SSS samples when sampling lights.

SSS Light Oversampling should be increased only when there's no noise in the diffuse but noise in the SSS. Please refer to SSS Quality section for more information.

At 0% light samples are distributed for each SSS samples. At 100% each SSS sample samples lights using the number of samples specified in light *Quality*. Obviously, the closer you get to 100%, the higher the render time and the total number of samples. They basically increase exponentially.

However, please be aware that it's faster to reach a noise free image using this attribute than artificially increase SSS *Quality*.



Figure 1 no oversampling (rendered in 109s)



Figure 2 oversampling at 50% (rendered in 162s)



Noise comparison zoomed in.
Figure 1 (top) and Figure 2 (bottom)

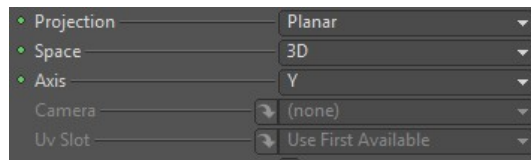
As you can see there's less noise in Figure 2 using *SSS Light Oversampling* at 50% than in Figure 1 with no light oversampling. Render time may have increased but it is lower than if you had increased *SSS Quality* or *Light Quality*.

Texturing

Clarisse provides many different kinds of textures. A texture is an operator computing a vector from a geometric fragment. The output vector can be anything from a color, a value to a direction. Clarisse provides also a subclass of textures: Texture Spatial. Those textures handle spacial projections such as parametric UVs, UV maps, implicit projection...

Spatial Projections

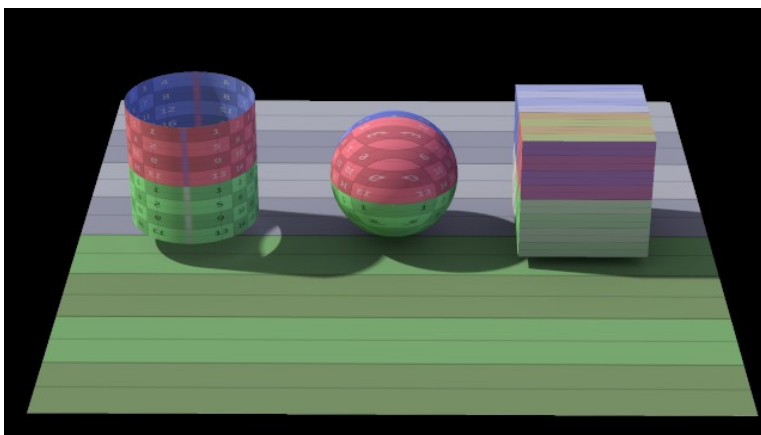
Texture Spatial provides many different kinds of projections. It supports parametric, implicit (planar, cylindrical, spherical, cubic), camera, UV mapped projections that can be either 2D or 3D.

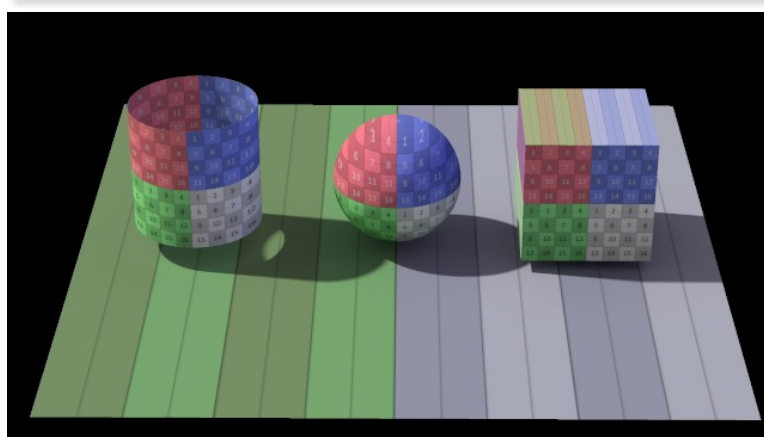
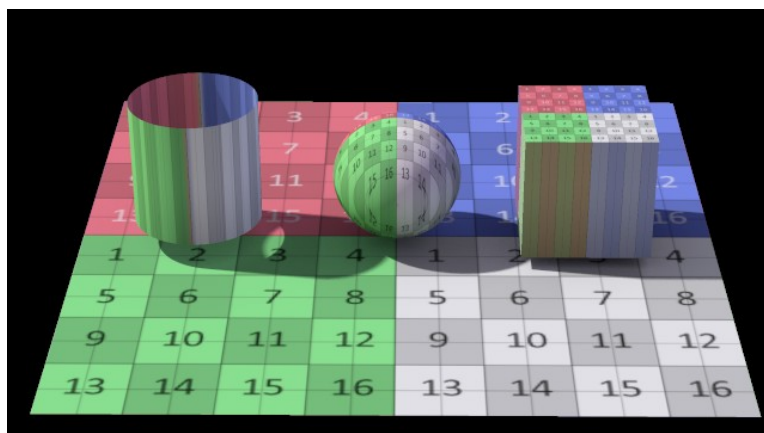


Attribute	Description
Projection	Sets the implicit projection mode
Space	Sets texture mode as 2D or 3D texture. In 3D space all components are used to evaluate the texture (useful for 3d noise textures)
Axis	Sets the projection axis.
Camera	Sets the camera used for projection when in Camera projection mode
UV Slot	Sets which UV Slot to use
World Coordinates	Sets if the texture is local/global. In global mode object transformations affects its texturing.
Projection Translate	Translates the position of the texture
Projection Rotate	Rotates the texture
Projection Scale	Scales the texture
UV Translate	Translates the uv coordinate
UV Rotate	Rotates the uv coordinate
UV Scale	Scales the uv coordinate

Projection Planar

Planar projection generates on the fly a planar UVs map.

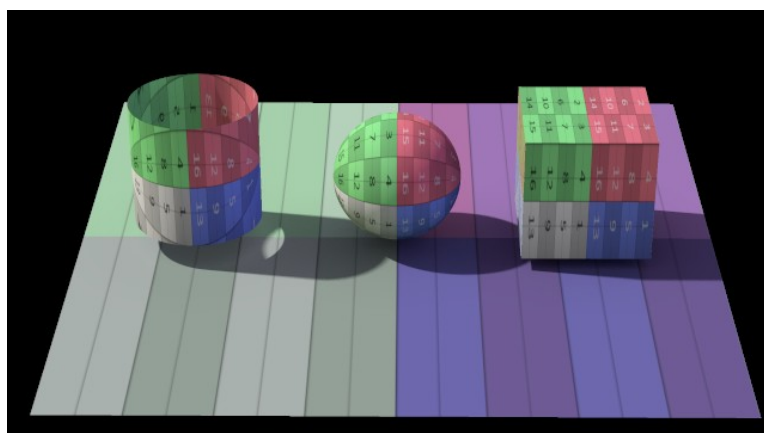


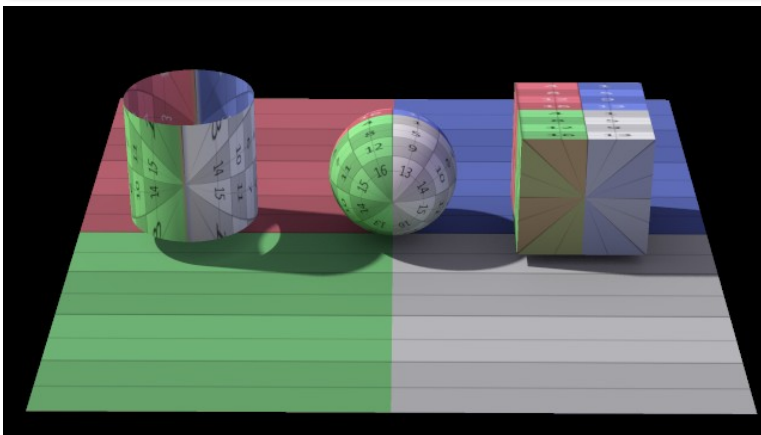
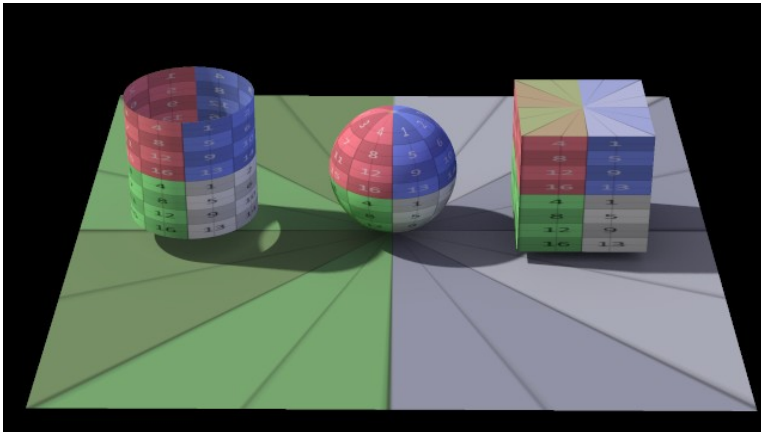


Planar X, Y, Z projections

Projection Cylindrical

Cylindrical projection generates on the fly a cylindrical UVs map.

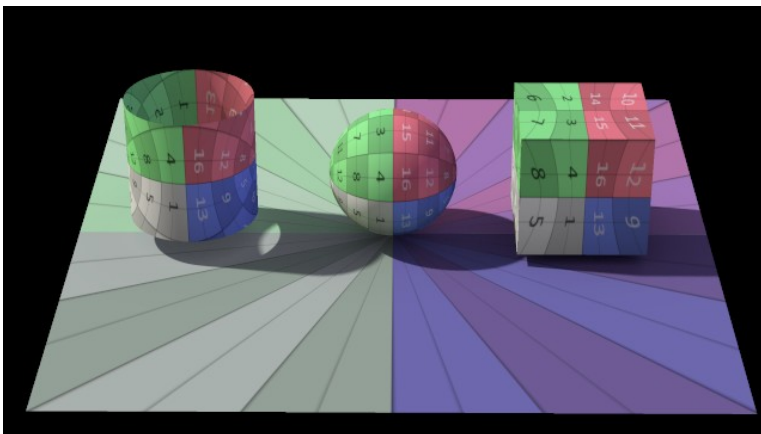


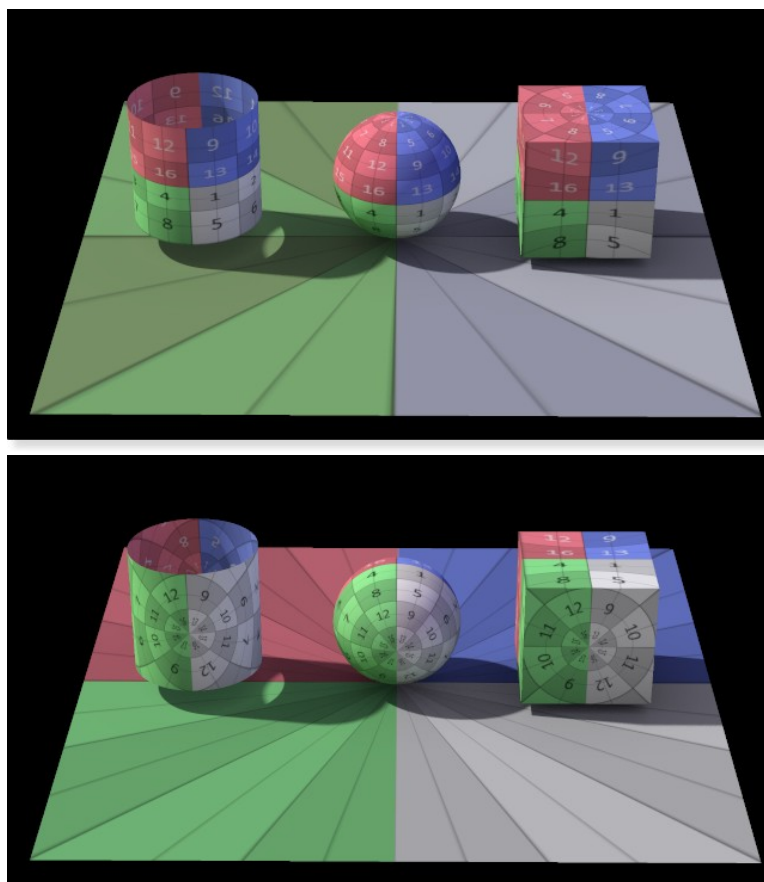


Cylindrical X, Y, Z projections

Projection Spherical

Spherical projection generates on the fly a spherical UVs map.

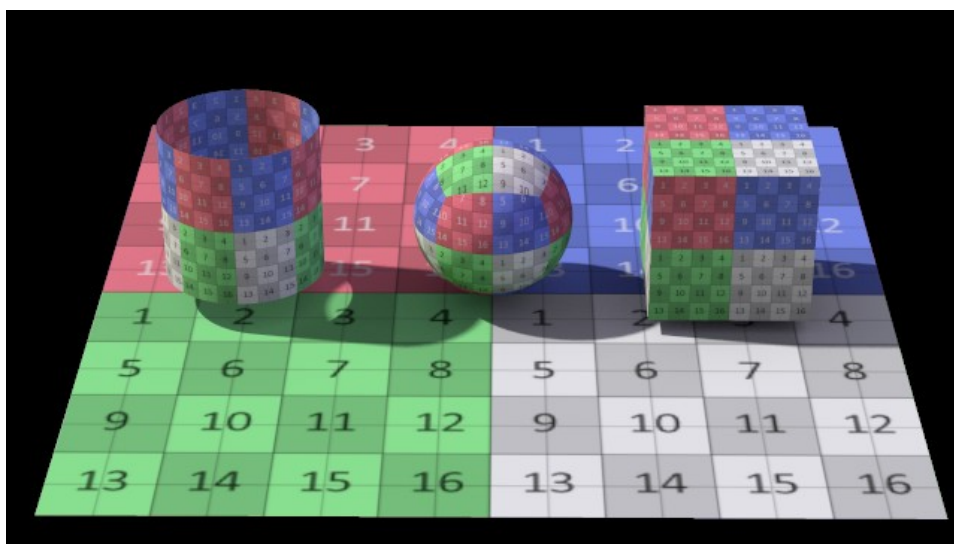




Spherical X, Y, Z projections

Projection Cubic

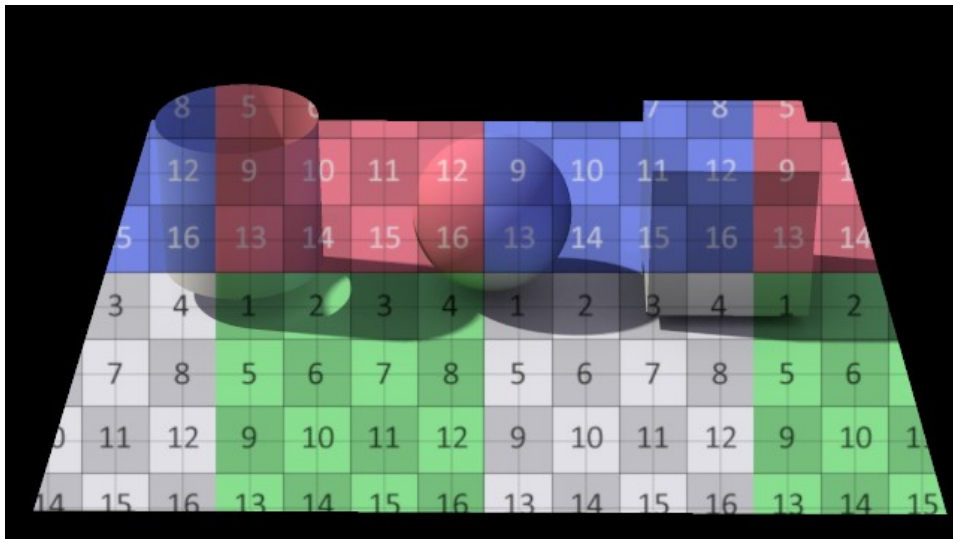
Cubic projection generates on the fly a cubic UVs map. In Cubic mode, *Axis* attribute has no effect.



Cubic projection

Camera

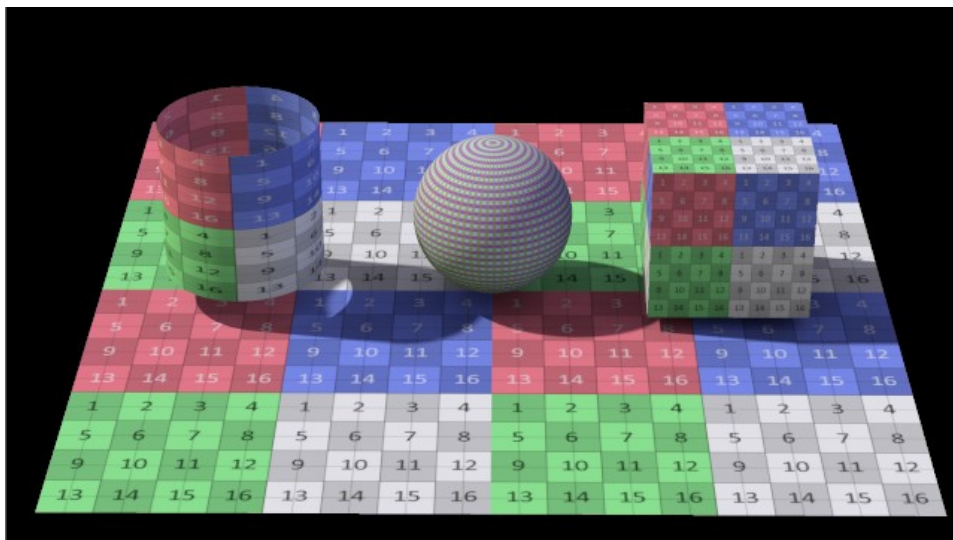
Camera projection allows camera mapping. The viewpoint and the projection are defined by the camera referenced by *Camera* attribute. The texture uses the real projection of the camera it references as its projection. This way, by referencing a Panoramic camera, it creates panoramic camera mapping.



Same camera used both camera projection and rendering

Parametric

Parametric projection uses the natural UV coordinates of primitives.

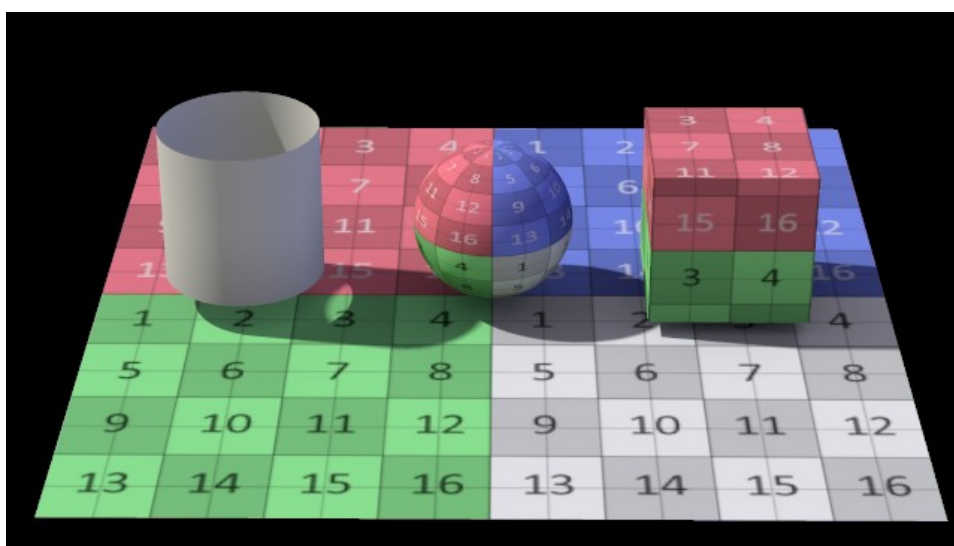


UV

UV uses a UV map as projection. In Clarisse, textures can be shared between materials and geometries. This is why UV maps are defined as application wide objects called UV Slots.

A UV Slot is globally mapping UV Maps. They are statically generated project items that are not saved in the project. You can find UV Slots in the `project://default` context. The maximum number of UV Slots in your project is equal to the maximum number of different UV Maps that are in your geometries. For example if one of your geometries has 10 UV Maps while others have less to none. The total number of UV Slots created in your projects will equal to 10.

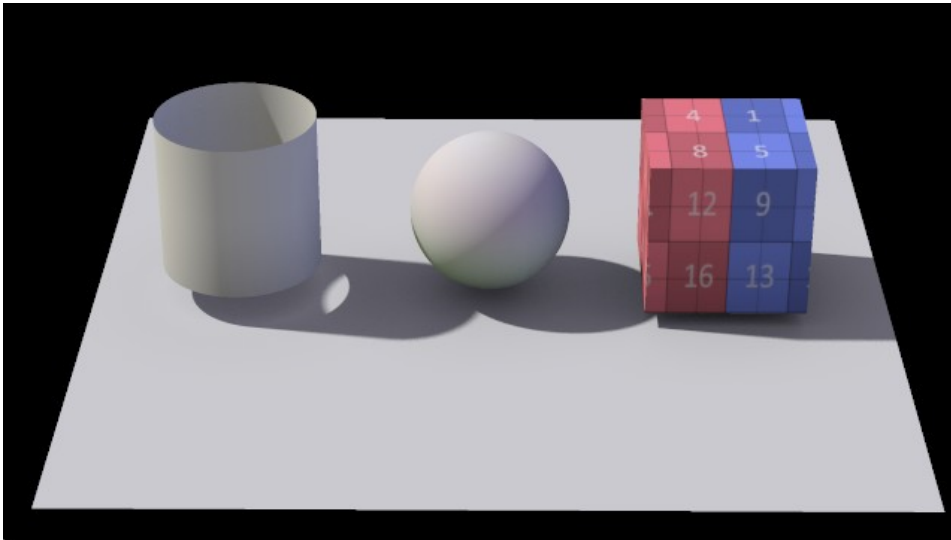
If a material references a texture, with a UV Slot mapping a UV Map number that doesn't exist in the applied geometry, the texture returns pure black (0.0, 0.0, 0.0, 0.0).



UV Mode and UV Slot set to Use First Available (default)

In the previous example, every geometry shares the same textured material. Here, the cylinder has no texture as Implicit Cylinders don't provide UV Maps.

In the next example, we've referenced explicitly `project://default/uv_slot_1` to the shared Map File texture. As you can see here, only the cube is textured as this is the only geometry that provides such UV Map.



UV Slot referencing project://default/uv_slot_1

Image Maps

In Clarisse, 3 kinds of image map texture nodes are available:

- *Map* which uses a Clarisse image as a texture. This is the node you should use if you wish to use a render as a texture.
- *Map File* which can load any supported image file to be used as a texture.
- *Streamed Map File* which is basically the same as the *Map File* except it streams from disk the input image file. Note it only works on EXR (tiled), TIFF (tiled) or TX files. An utility called maketx can be used to create your own TX files. You can find this utility inside Clarisse binaries directory.

Map and Map File

Texture Map and *Texture Map File* both share a same set of attributes. Some control image pre-processing and others control the quality of the image filtering.

Color Space and Pre Multiplication

Clarisse uses internally an pre-multiplied linear image pipeline. Images must be in linear color space and RGB values must be pre-multiplied by the alpha channel. Both *Map File* and *Streamed Map File* provide a set of attributes to pre-process the input image file. Please note *Texture Maps* are already in the right color space.

Attribute	Description
-----------	-------------

Linearize	Consider the input image as in sRGB color space. If checked (default), the image is automatically linearized. Note: When you import HDR, EXR... or any non sRGB images (essentially grayscale images) then you must uncheck attribute.
Pre Multiplied	Consider the image as pre-multiplied. If the input image isn't pre-multiplied, you must uncheck this attribute.

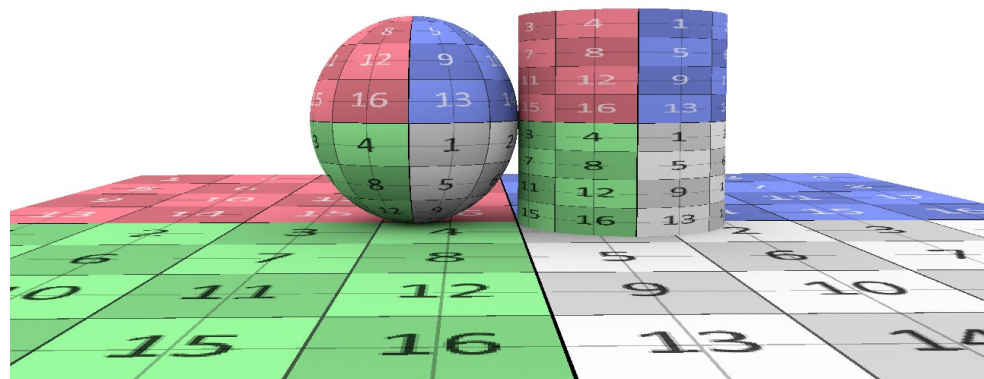
Note

Understanding these pre-processes is fundamental. If you get artefacts such as bright highlights, wrong colors etc... make sure to check the value of those attributes.

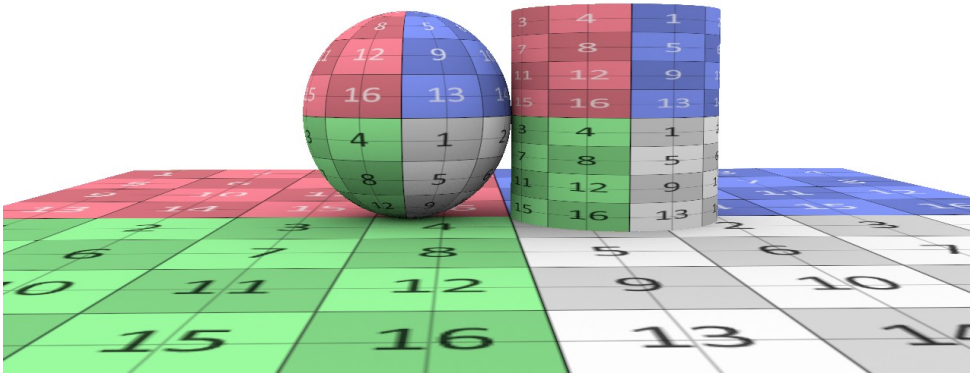
Filtering Quality

Clarisse provides several ways to filter image maps.

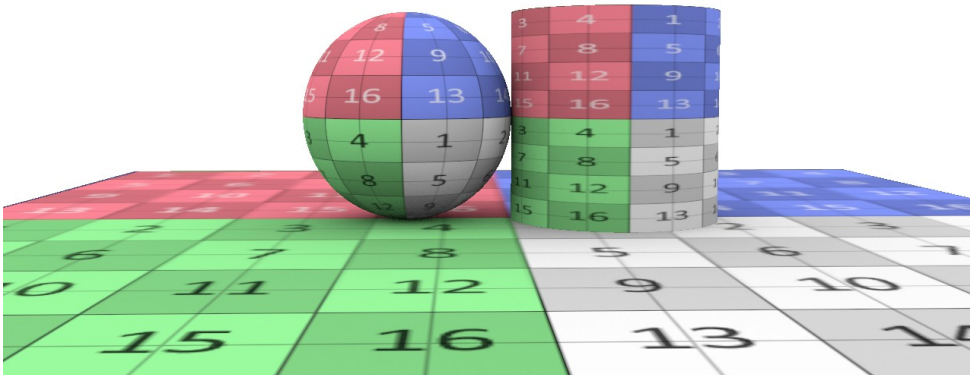
Filtering Type	Description
Nearest Neighbor	No filtering is performed. The nearest corresponding pixel is taken from the image. This returns the poorest results.
Bilinear Filtering	The nearest 4 pixels are used to compute the final color.
Trilinear Mipmapping	Create a image pyramid of mip maps if not available, performs two bilinear interpolation to compute the final color.
EWA Mipmapping	Same as Trilinear but performs an elliptical weighted average to compute the final color. This returns the best results specially with anisotropic projections.



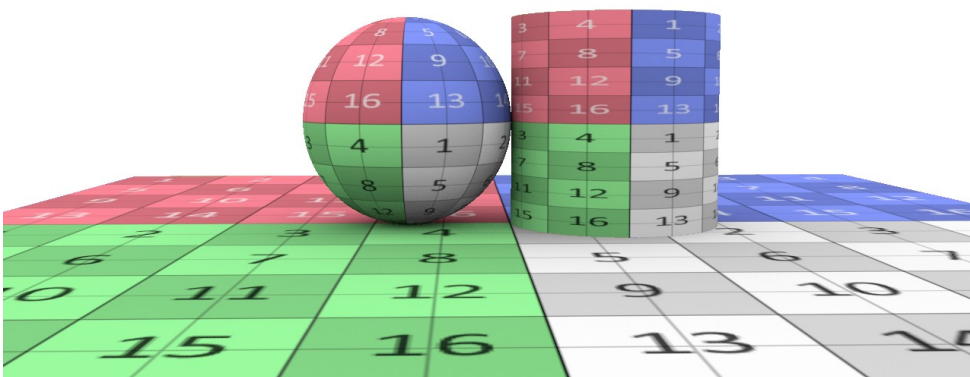
Nearest Neighbor (pixelated result)



Bilinear Filtering smooth results when textures are magnified. Poor results at a distance



Trilinear Mip Mapping, good results except at glancing angles where it gets too blurry



EWA gives the best results but it is more expensive to compute.



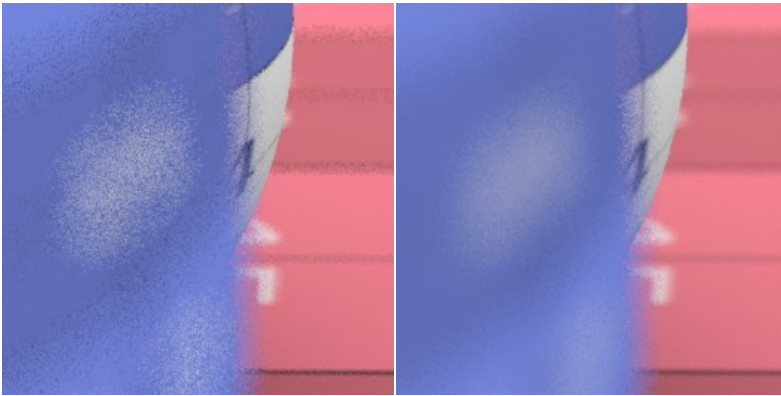
(A) Nearest (B) Bilinear (C) Trilinear (D) EWA

The mip mapping technique requires the creation of a mip map pyramid. This needs some processing power and an increase of memory usage (an increase of up to 33% of the image memory usage).

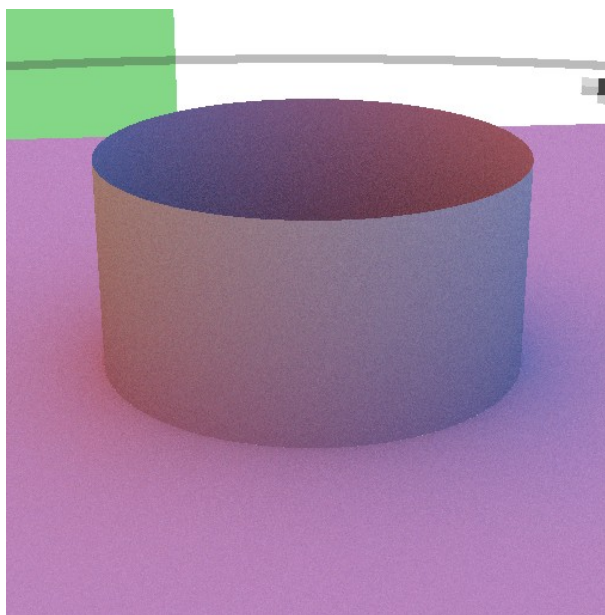
However, when you use *Stream Map File*, the image is not loaded in memory, it is streamed from disk. It is even recommended to use mip mapped images with *Stream Map File* as texture cache efficiency is improved.

Role in Sampling

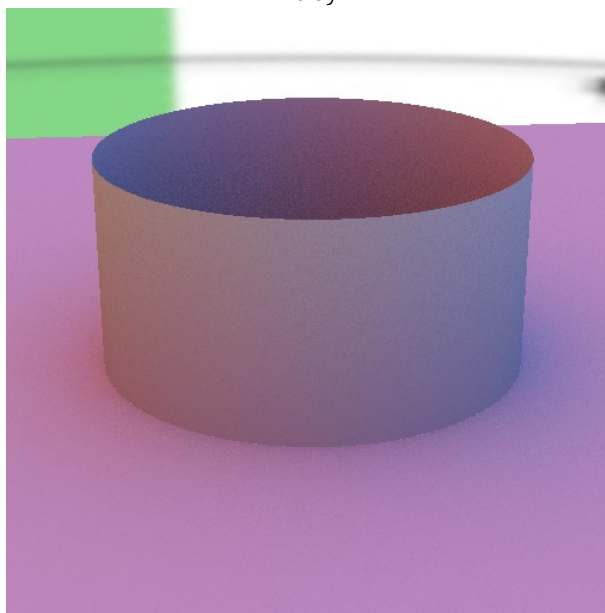
Using mip-mapped textures both improves the quality of your textures and the quality of depth of field, lights, global illumination, reflections and refractions. With the same number of samples you can get far less noisy images.



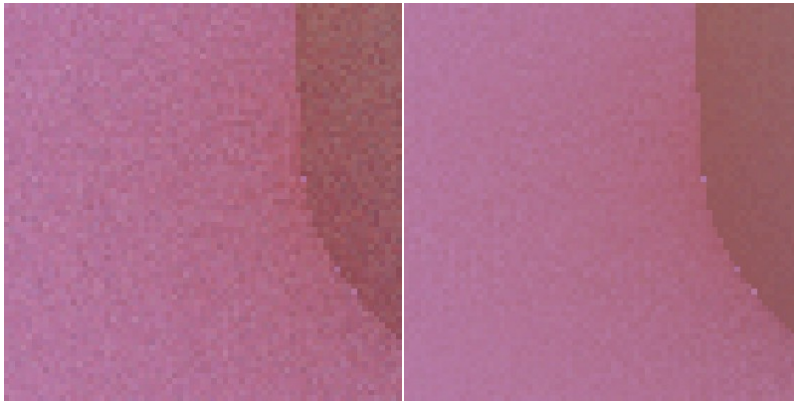
Depth of field with nearest textures (left) and with EWA (right)



Texture in nearest neighbor, the global illumination is noisy



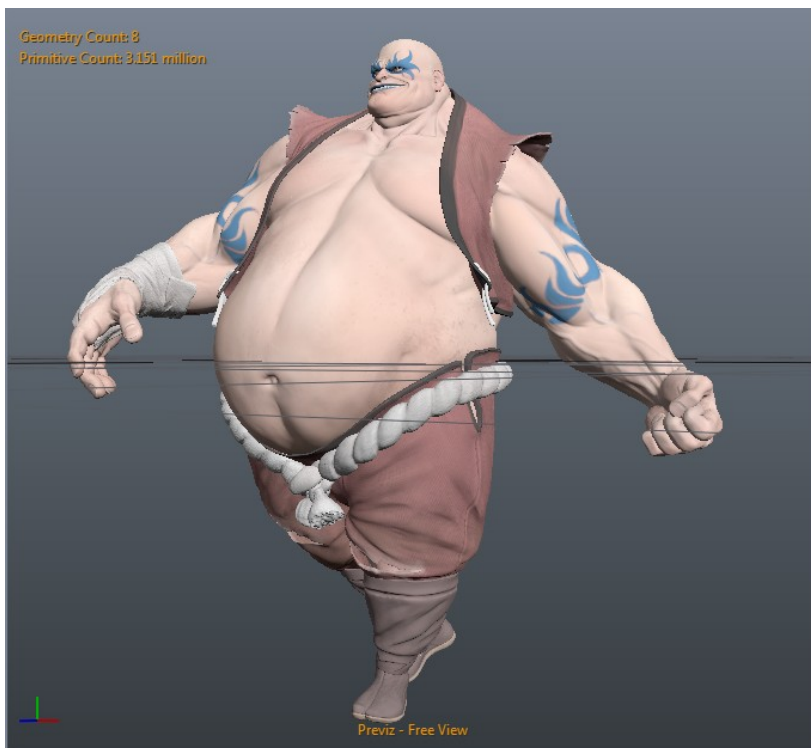
In EWA, global illumination (same settings) is far less noisy



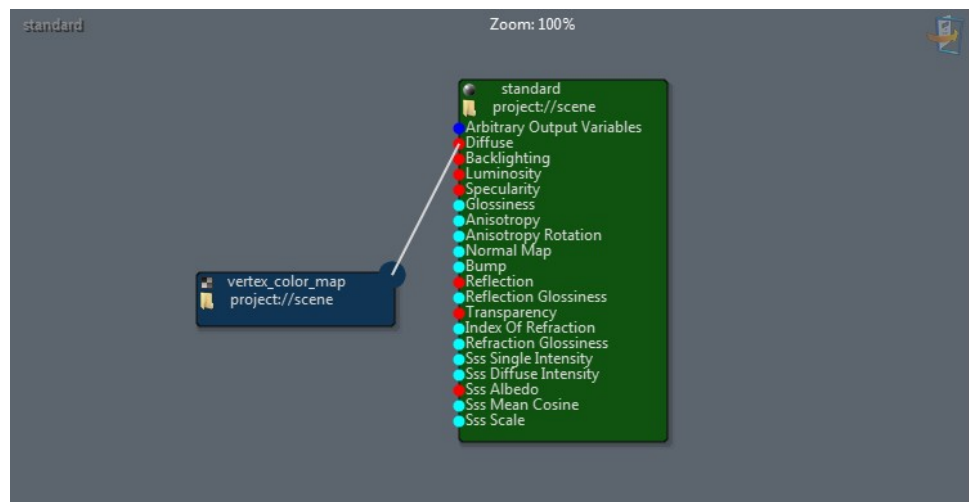
Nearest left, EWA right

Vertex Color Maps

Clarisse supports vertex color maps when available in a geometry. To access to vertex color map information, you must use a Vertex Color Map texture.



Geometry exported from ZBrush containing polypaint (vertex color map)



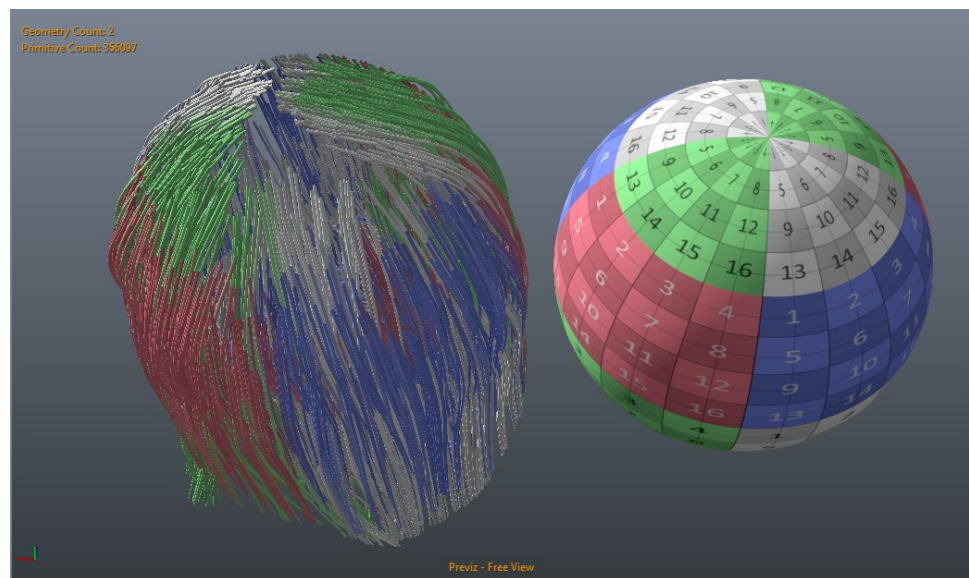
Vertex color map plugged into Diffuse

Note

If no vertex color maps is found by the texture, then it returns black.

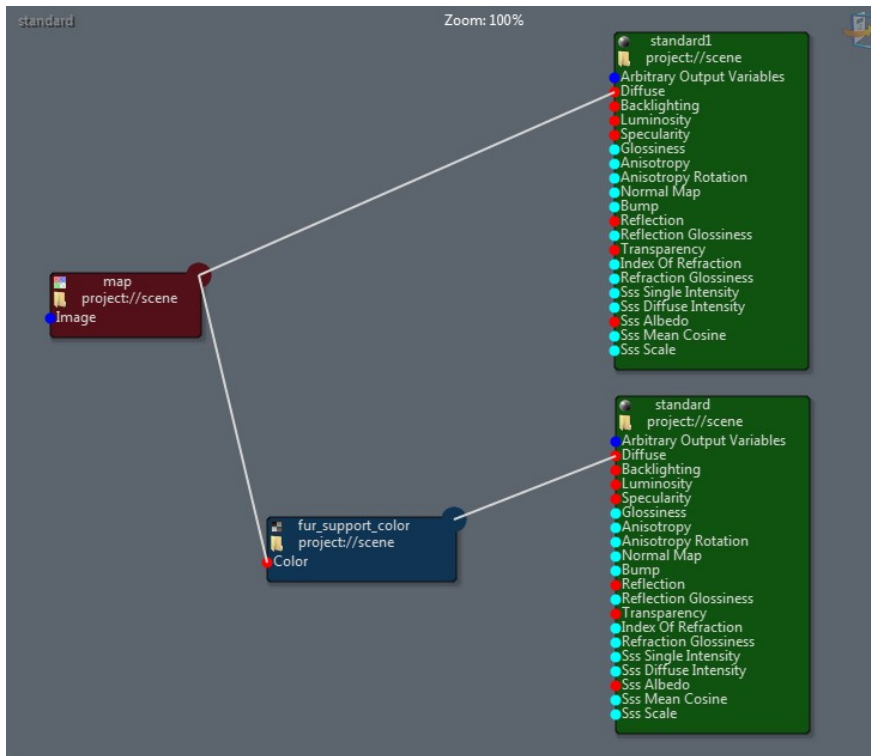
Fur Support Color

Fur Support Color texture allows you to extract texture color as it was shaded from the underlying geometry. Basically, the projection is based on the support geometry. Fur Support Color only works on materials attached to fur interpolate geometries.



Fur using Fur Support Color (left), support geometry with the same texture (right)

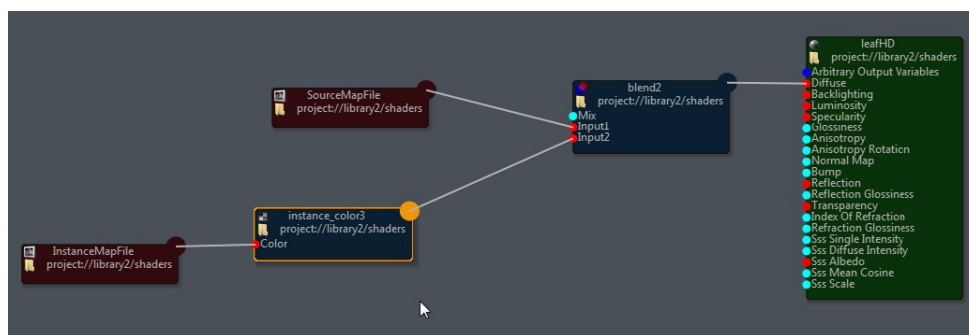
In the previous image, we clearly see each fur curve getting the color of its root as if it was on the sphere. Using the Fur Support Color is fairly simple. First, work out your texture on the support geometry. Then insert a Fur Support Color in between. Fur Support Color acts as a support geometry to fur projection bypass.



The same texture is directly used on the sphere material and indirectly used on the fur material

Instance Color

Instance Color texture allows you to control independently the color (or any other shading attribute) of each scattered object instance. It works with the instance's support geometry (typically the point cloud used by the scatterer), to generate the projection coordinates needed by the input.



Typical setup of the Instance Color texture

Note

If the object is not instantiated, the instance color texture returns a black color. This is why it is generally used before a blend node, so the shader can still be used for non scattered objects.

Clarisse instancing/scattering system is hierarchical. Indeed, you can have a scatterer scattering another scatterer and so on. Using *Parent Level* attribute, you specify the hierarchical level of your instance. The Instance Color will then use for its texture coordinates the correct point cloud support.

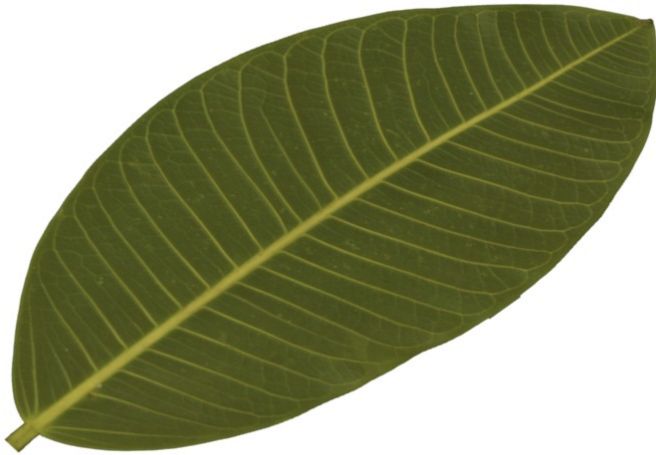


figure 1



figure 2



figure 4

A single leaf scattered on a tree. Each leaf receives a different color. Colors are based on the texture (figure 4) mapped using the texture coordinates of the point cloud used to scatter the leaves on the tree.

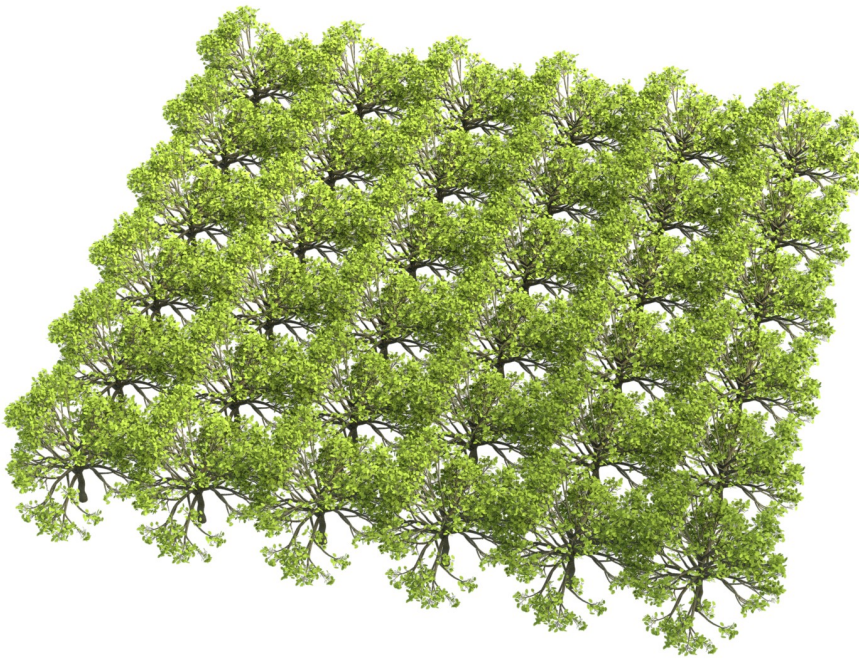


figure 5

The resulting tree is then scattered on a grid. You can see the same pattern which is repeated on each tree. This is because the *Parent Level* of the instance color texture is set to 1. It then uses the same texture coordinates as in figure 2 to drive the color of each leaf.

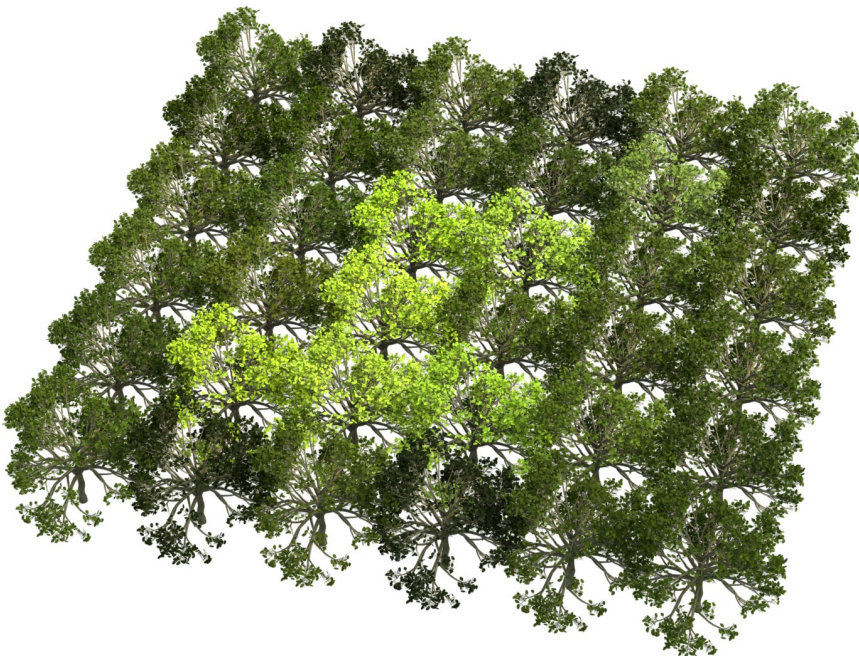
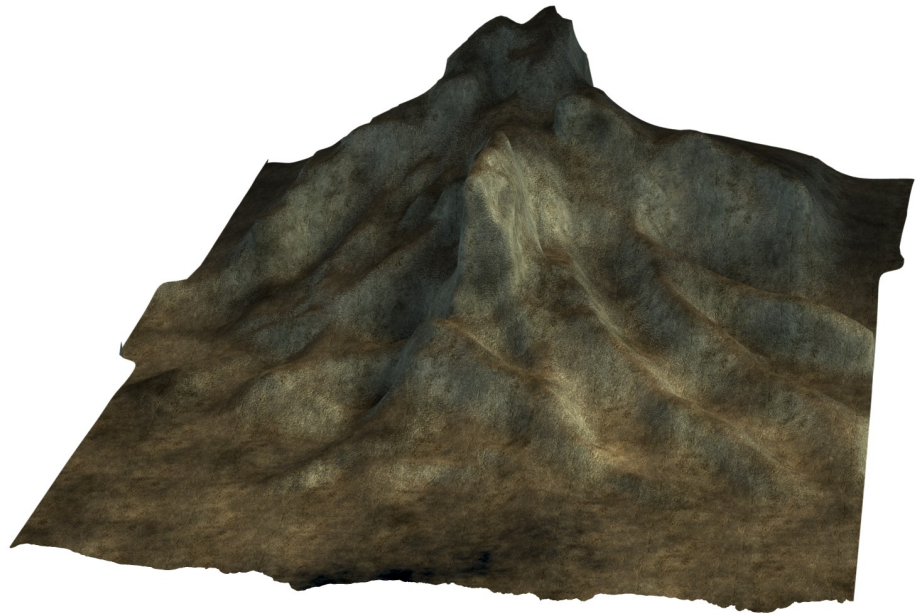


figure 6

The same setup, but now with *Parent Level* set to 3. The texture coordinates are now extracted from the array point cloud (level 2 being the combiner containing each tree).

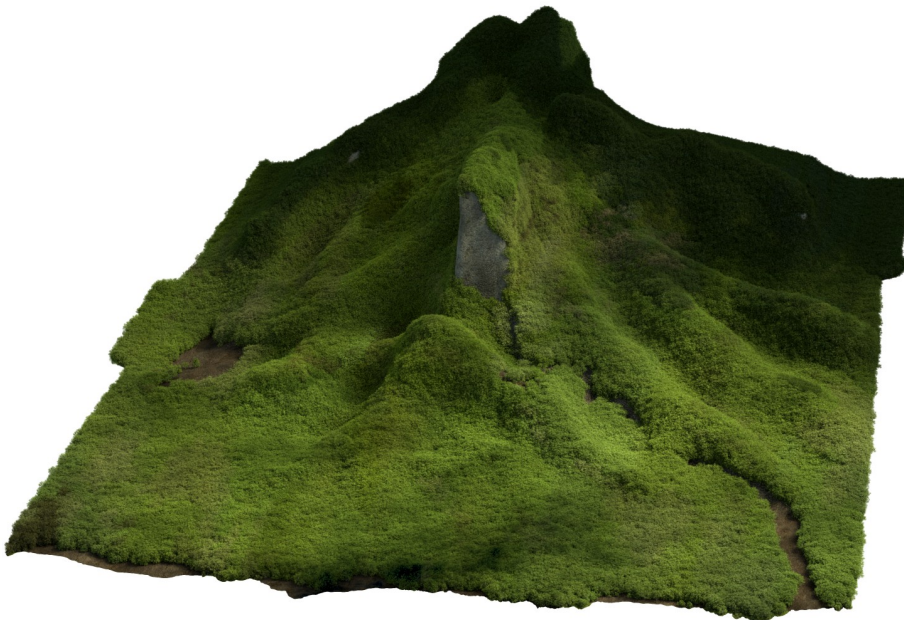
Thanks to the hierarchical scattering system, you can get infinite variations using this powerful texture node. For example you can draw patterns on crowds or create very realistic cities using instance color texture to switch different versions of same assets controlled by a drawn texture map. The possibilities are endless. Here is an example using the *Instance Color* texture on a forest:



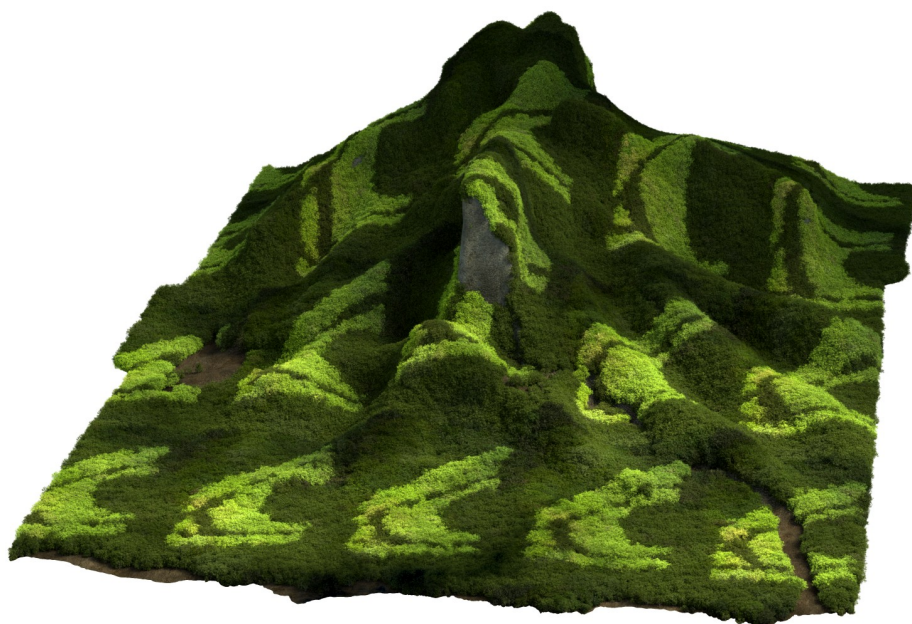
The ground support object (shaded)



The texture map applied to the ground used to check the texture coordinates



The same texture map connected to the instance color texture used on the leaf material



Same render using the figure 4 texture



Final shot

Utility

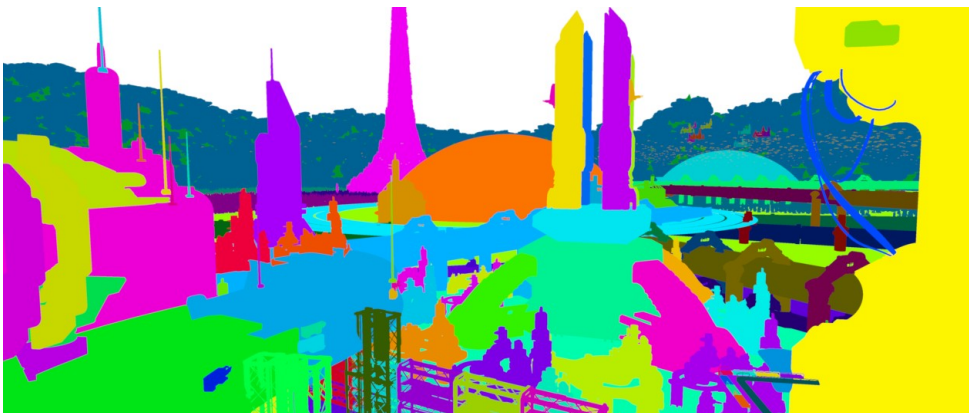
The *Utility* texture is very useful if you wish to extract geometric and rendering information. It can also be useful when exporting AOVs. For example, with the *Utility* you can extract world positions, shutter time, normals etc... Its single attribute *Output* sets what information the texture will output as color.

The output value is not normalized. For example, if you output *World Position*, this will be the real world position value and not a normalized color.



Utility set to Area (Ray Differentials)

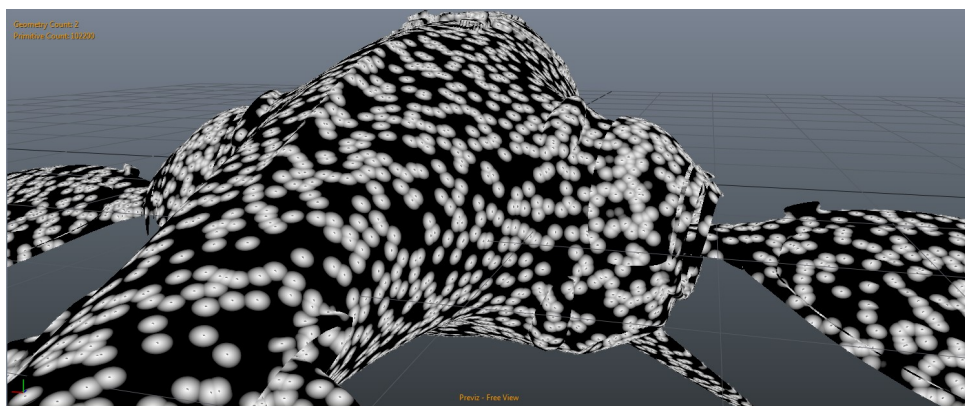
However some values are output as a normalized color such as *Object ID* or *Shutter Time*.



Utility set to Object ID

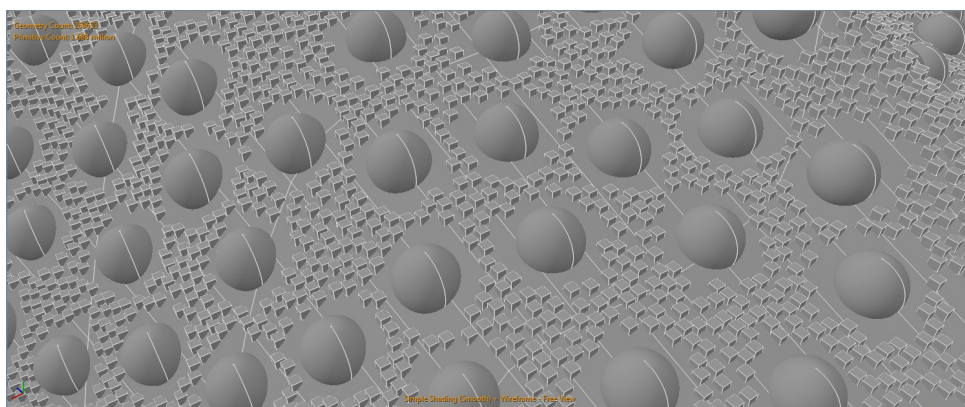
Point Cloud

The *Point Cloud* texture is a powerful feature. It allows to return a color based on the distance of the current shaded surface from an input *Point Cloud* geometry.



A point cloud geometry generated from the underlying used as input of a point cloud texture

This texture can be used for many different purposes. One of its primary purpose though is to use it to create masks for points clouds used in scatterers.



The point cloud used for spheres is used as decimate texture for the one used for boxes

Reorder

The *Reorder* texture allows you to reorder channels of an input color. By default, *Channel Order* is set to *rgba* which means it leaves the input color unmodified.

To modify the reorder of the input, simply type in *Channel Order* the new channel order. For example *rrrr* will output the content of the input red channel for each *rgba* color output components. *Channel Order* also supports 0 or 1 in place of the channel name. For example if you want to output a white color you would type *1111*, and *0000* for a black one.

For example, let's say we have an input color (0.25, 0.8, 1.0, 0.17) and we specify as *Channel Order* *a0r1*. The output color will then be (0.17, 0.0, 0.25, 1.0)

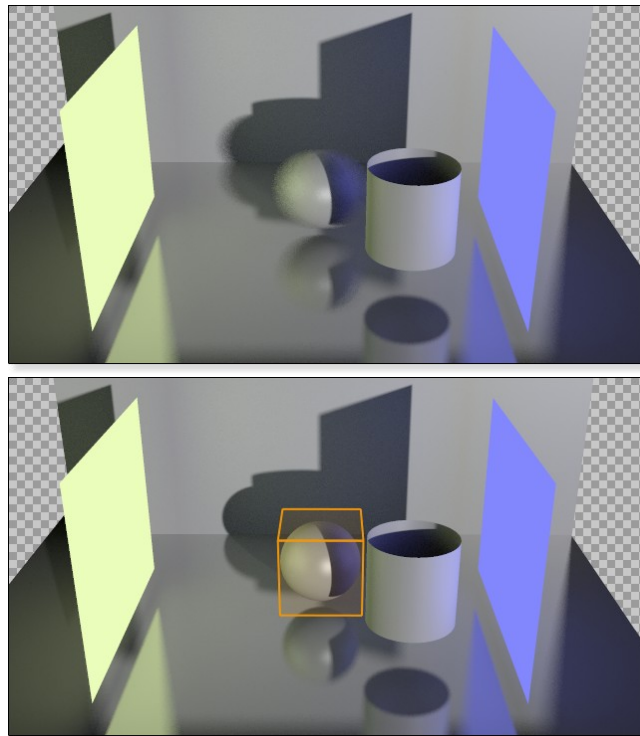
Rendering Attributes

As Scene Objects are meant to be rendering, they provide many attributes that are related to rendering.

Attribute	Description
Raytrace Offset	Sets the minimum distance for ray intersection. If the ray hit distance from the ray source position is smaller than this value, then the intersection is skipped. This attribute is mainly used to prevent self intersection artifacts.
Enable Motion Blur	If unchecked the item is considered as static during rendering. Typically if the item is animated, it won't get blurred. Note the object can still be blurred if the camera is moving.
Unseen By Camera	The item will be invisible to camera rays (primary rays) but will still cast shadows and scene in raytracing.
Unseen By Rays	The item will be invisible to raytracing (secondary rays) but still be visible to camera.
Cast Shadows	Sets if the item is casting shadows.
Receive Shadows	Sets if the item is receiving shadows.
Is Emitter	Sets if the item emits light (during global illumination computation).
Lights	Specifies a group of lights illuminating the item. By default, the item is lit by the lights used in the Layer 3D.
Override Material	Sets a material to override the material used by the item.
Matte Object	Sets if the item is a matte object.
Matte Color	Sets the matte color.
Matte Alpha	Sets the alpha value.

Enable Motion Blur

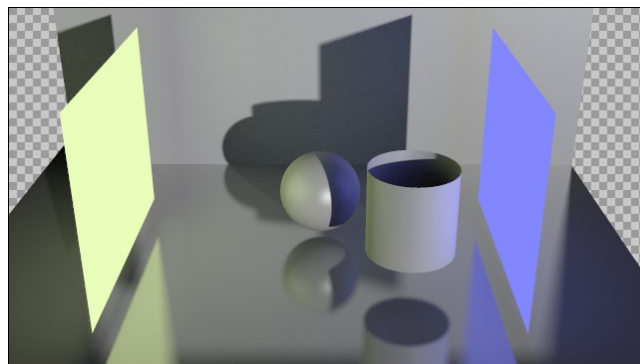
When the renderer has motion blur enabled, by default, every scene objects are blurred according to their motion. You can disable motion on a scene object by having *Enable Motion Blur* attribute unchecked. Please note this doesn't affect camera blur.

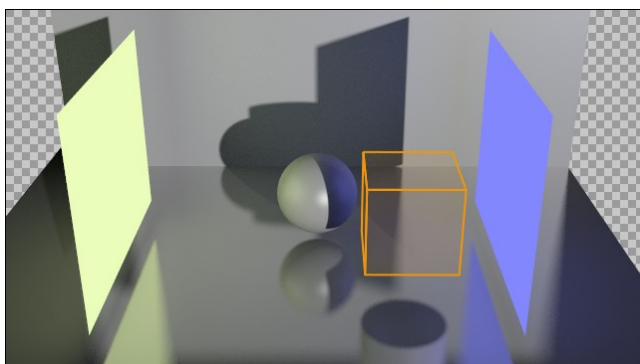


Motion blurred Sphere (top), Enable Motion Blur attribute unchecked on the sphere (bottom)

Unseen by Camera

By default, all scene objects are visible to camera. You can hide items from camera by checking *Unseen By Camera* attribute. Note items are only hidden from camera rays (primary) they still cast shadows and are still seen during raytracing.

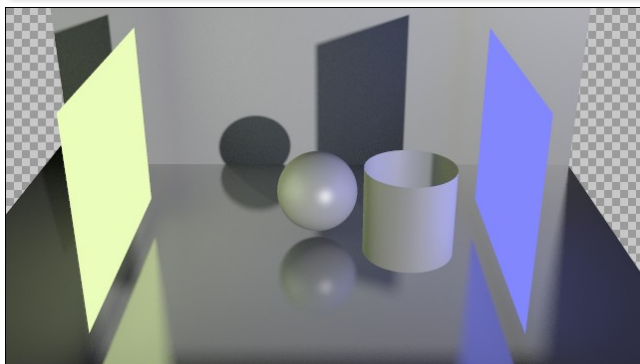
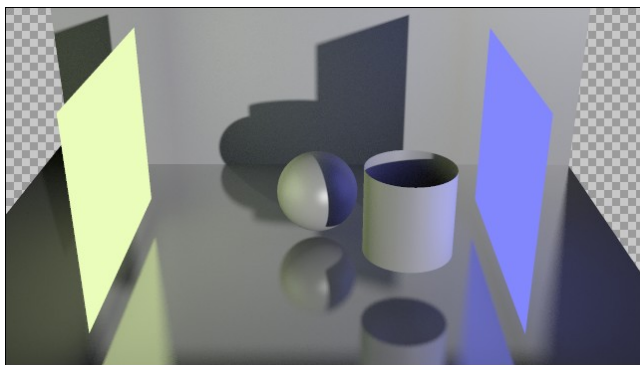




Original scene (top), Unseen by Camera checked on Cylinder (bottom).

Unseen by Rays

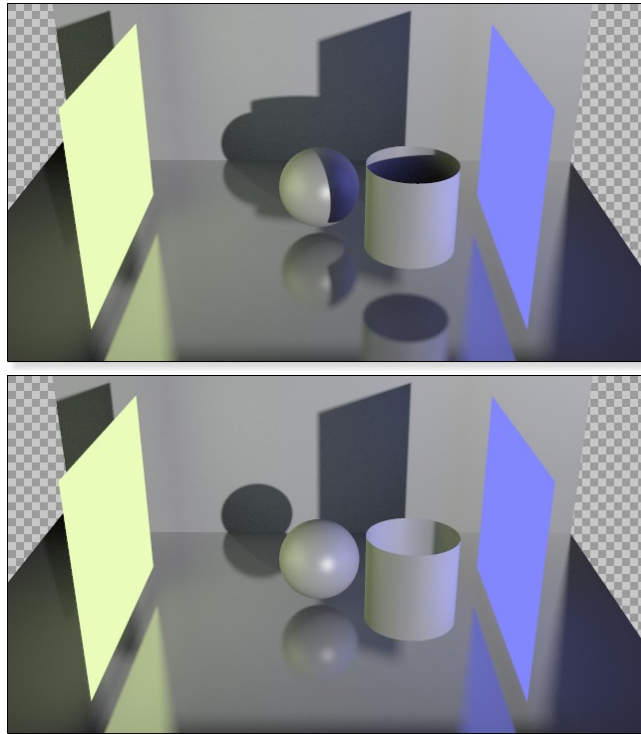
By default, all scene objects are visible during raytracing. They are reflected/refracted, cast shadows... You can hide items from raytracing by checking *Unseen By Rays* attribute. Note the item will be still visible to the camera.



Original scene (top), Unseen by Rays checked on the Cylinder (bottom).

Cast Shadows

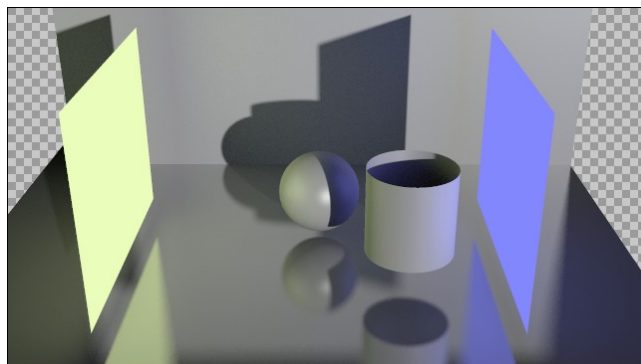
By default, all scene objects cast shadows. You can remove items from the list of shadow casters by unchecking *Cast Shadows* attribute.

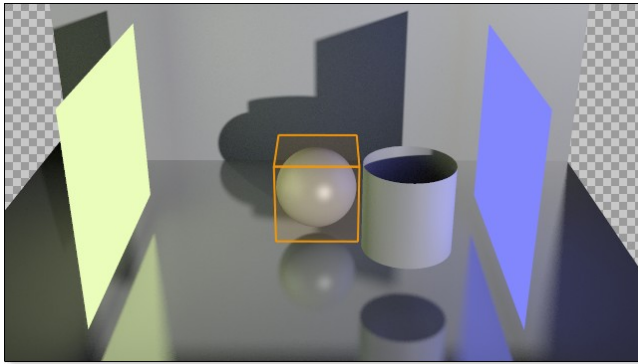


Original scene (top), Cast Shadows unchecked on the cylinder (bottom).

Receive Shadows

By default, scene objects receive shadows from shadow casters. You can change this behavior by unchecking *Receive Shadows* attribute. In this case scene objects don't receive any shadows.

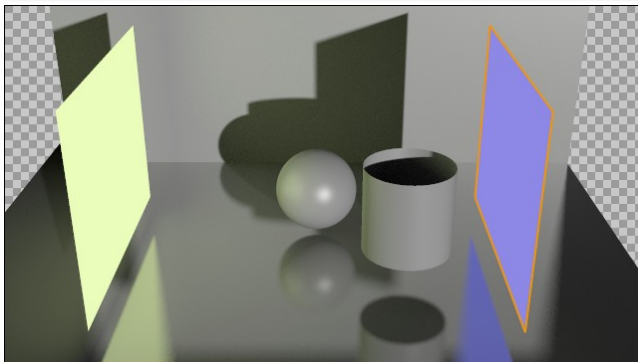
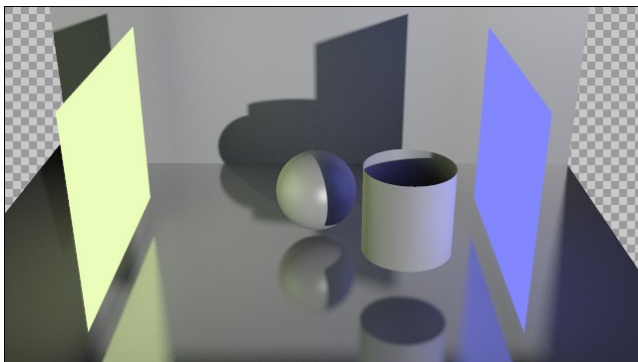




Original scene (top), Receive Shadows unchecked on the sphere (bottom).

Is Emitter

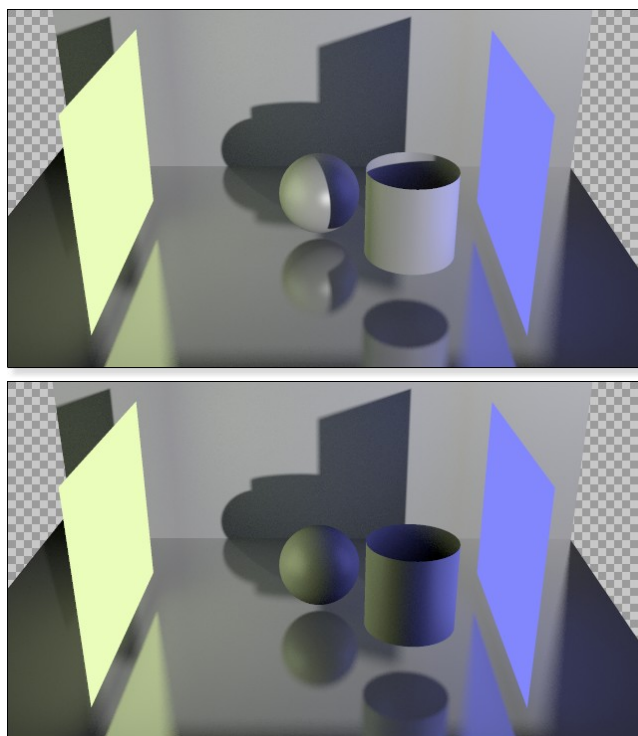
By default, all scene objects emit light during global illumination computations. To stop objects from emitting light, uncheck *Is Emitter* attribute.



Original scene (top), Is Emitter unchecked on the right plane (bottom).

Light Linking

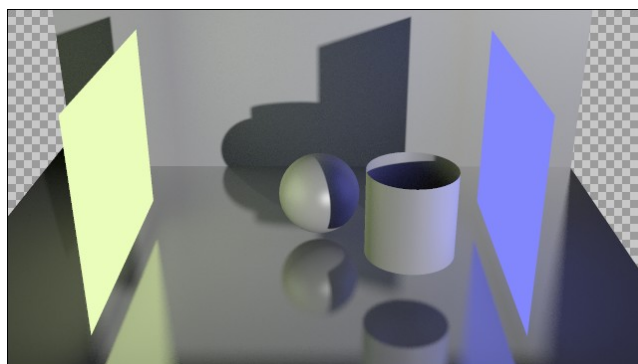
By default, scene objects are illuminated by lights referenced in the 3D Layer they are rendered in. You can specify the lights illuminating items by referencing a group of lights in their *Lights* attribute.

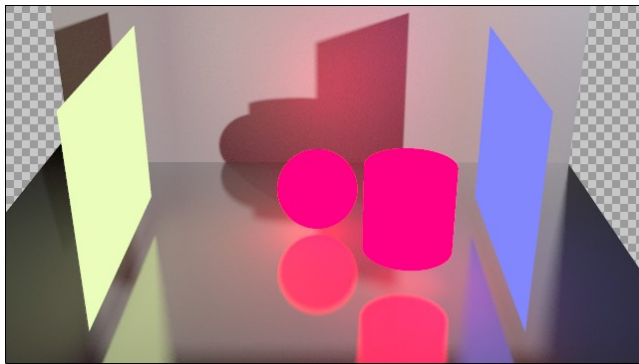


Original scene (top), sphere and cylinder are excluding the key light (bottom).

Override Material

By default, scene objects are using materials attached to their geometry shading groups. You can entirely override those materials by referencing a new material in *Override Material* attribute.



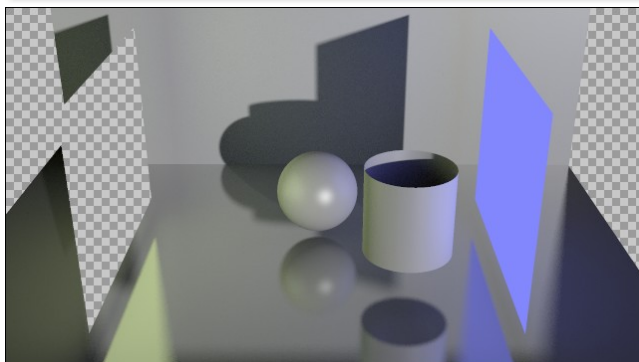
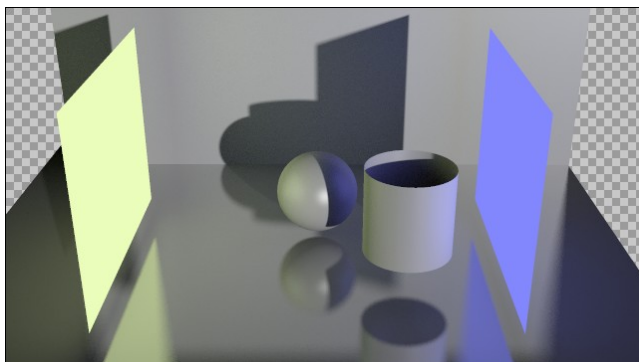


Original scene (top), sphere and cylinder are excluding the key light (bottom).

Working with Mattes

When you work in layers, you'll probably need to work with matte objects to compose your layers. Every scene object has a dedicated set of attributes to manage mattes and alphas.

Attribute	Description
Matte Object	Sets if the item is a matte object.
Matte Color	Sets the matte color.
Matte Alpha	Sets the alpha value.



Original scene (top), left plane is a matte object with a matte alpha value of 0% (bottom).

Rendering Image Sequences

There are two ways to render final images in Clarisse:

- Using the command line tool CNODE (Clarisse for render nodes).
- Directly from within Clarisse user interface.

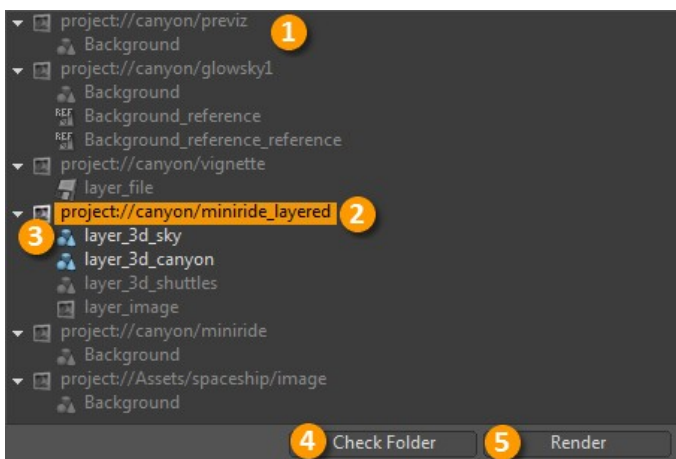
If you wish to render your images using CNODE, please refer to Using CNODE section. Otherwise, please continue reading this topic.

In Clarisse, you don't need to explicitly launch a render to start rendering. Displaying an image in the Image View starts the rendering of all layers and display the fully composited image. When working on animations scrubbing the timeline will eventually re-render the displayed image for the current frame.

While, this workflow is pretty simple when working interactively on images, Clarisse also provides a more convenient workflow for outputting image sequences using the Render Manager.

Using the Render Manager

The Render Manager is a dedicated widget designed to render and output final image sequences as well as individual layers to disk. It displays all images and layers existing in your project. Grayed out items are not exported to disk.



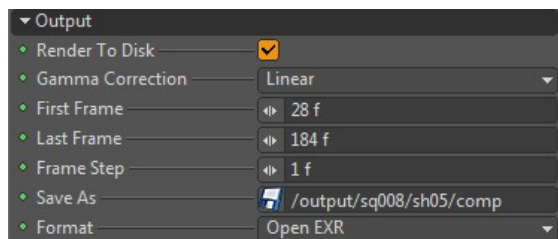
(1) Image and layer tree (2) Image marked for output (3) Layer marked for output (4) Check Folder tool (5) Render

Note

An item name displayed in red has an incorrect filename set in its *Save As* attributes.

Disk Output Attributes

Images and Layers provides a set of dedicated attributes managing their outputs to disk. You can find these attributes under the *Output* category of images or layers when selected in the Attribute Editor.



Attribute	Description
Render to Disk	Sets if the layer/image is outputted to disk
Gamma Correction	Sets if gamma correction is performed before saving to disk.
First Frame	Sets the first frame to render
Last Frame	Sets the last frame to render
Frame Step	Sets the step between two rendered frame
Save As	Sets the file output name
Format	Chooses the output image file format

By default images and layers are not output to disk.

Render To Disk

To set images or layers to be saved to disk, select images and layers and enable their *Render To Disk* attributes in the Attribute Editor. Once renderable to disk, they'll appear such as (2) and (3).

Gamma Correction

You can apply a gamma correction to your renders. If you want to save your images without any gamma correction, then set *Gamma Correction* to *Linear*.

Setting Frame Range

You can set individual frame range for your renders by setting *First Frame*, *Last Frame* and *Frame step* attributes.

Filename

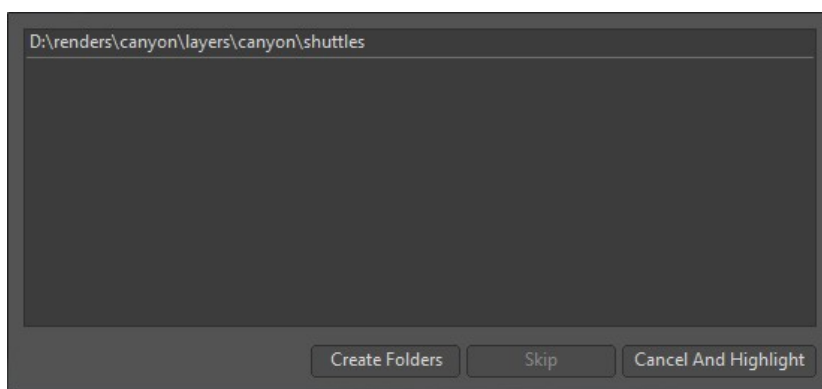
Set the image sequence filename in *Save As* attribute. Padding and image file extension are automatically appended to the specified filename.

Format

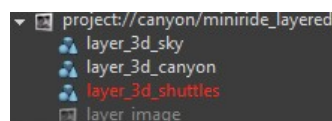
Choose image file output format by setting *Format* attribute.

Check Folder

You can check and create the folders of your exported image by pressing Check Folders button. If problems are detected then a popup window will appear displaying the invalid detected path. This window helps you create the missing folders. Select any invalid path and press Create Folders button.



If you press Cancel And Highlight, images or layers with invalid file path will get their names highlighted in red in the Render Manager tree.



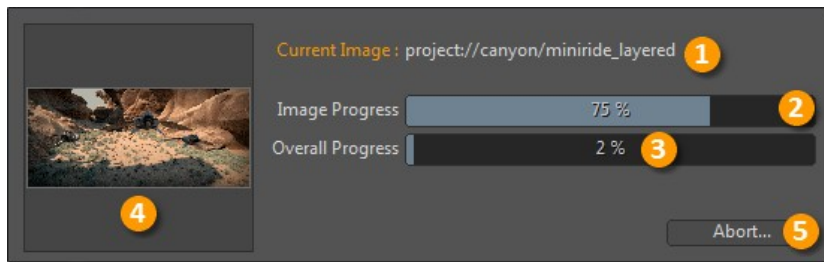
layer_3d_shuttles is invalid

Note

Images or layers with invalid path are not rendered.

Rendering

To render all selected images, press Render button. When rendering image sequences, images and layers may have different frame range settings. The Render Manager will automatically skip unnecessary frames and images to only render what is required.



Render Manager rendering window

(1) Current Rendered Image (2) Current Image Progress (3) Overall Progress (4) Last Rendered Image thumbnail (5) Abort Rendering

To interrupt rendering, press Abort button (5). This will interrupt all pending renders.

Working with Shading Layers

Traditionally, materials are explicitly assigned to sets of primitives defined by geometries and called shading groups. This type of workflow makes it very difficult to replace geometries in shots or to switch their set of materials. To resolve these issues, Clarisse introduces the Shading Layer.

What's a Shading Layer?

A Shading Layer is a project item dedicated to dynamically assign materials at a 3D Layer level or in a 3D View. It has been designed with this simple idea: instead of explicitly storing material assignments at the geometry level, assignments should be stored in a high level item to make them geometry-independent and re-usable.

In a nutshell, this is what shading layers are all about.

How it works?

The shading layer is based on rules and each one has properties. In a shading layer, rules are used to identify items in the project. More importantly, rules are soft links: this means properties set in shading layers are not transferred to items.

For example, if one of your geometry has already a material applied using either an *Override Material* or the Material Linker, its properties won't be physically modified. The soft link is only resolved during evaluation.

Creating a Shading Layer

To create an empty shading layer use **Create > Shading Layer**. You can also create a pre-filled one from the selection (Scene Object, images...). In a browser make your selection (can have multiple items), right click to bring the Browser pop up menu and select **Create Shading Layer > Use Relative Names** or **Create Shading Layer > Use Absolute Names**.

When creating a shading layer from a selection, the resulting shading layer contains all extracted shading group and material associations. This makes the shader layer ready to use.

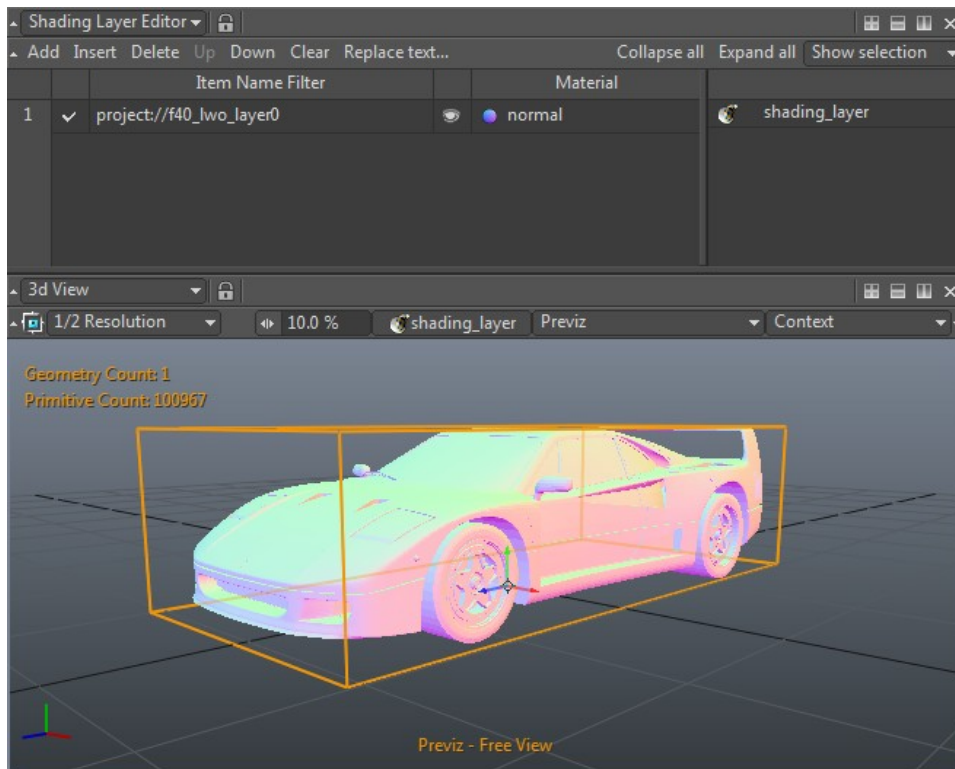
Assigning Shading Layers

Shading Layers are assigned to the attribute *Shading Layer* in 3D Layers. You can also assign shading layers to the 3D View. Remember to set the 3D View rendering mode to Previz.

The Rule Concept

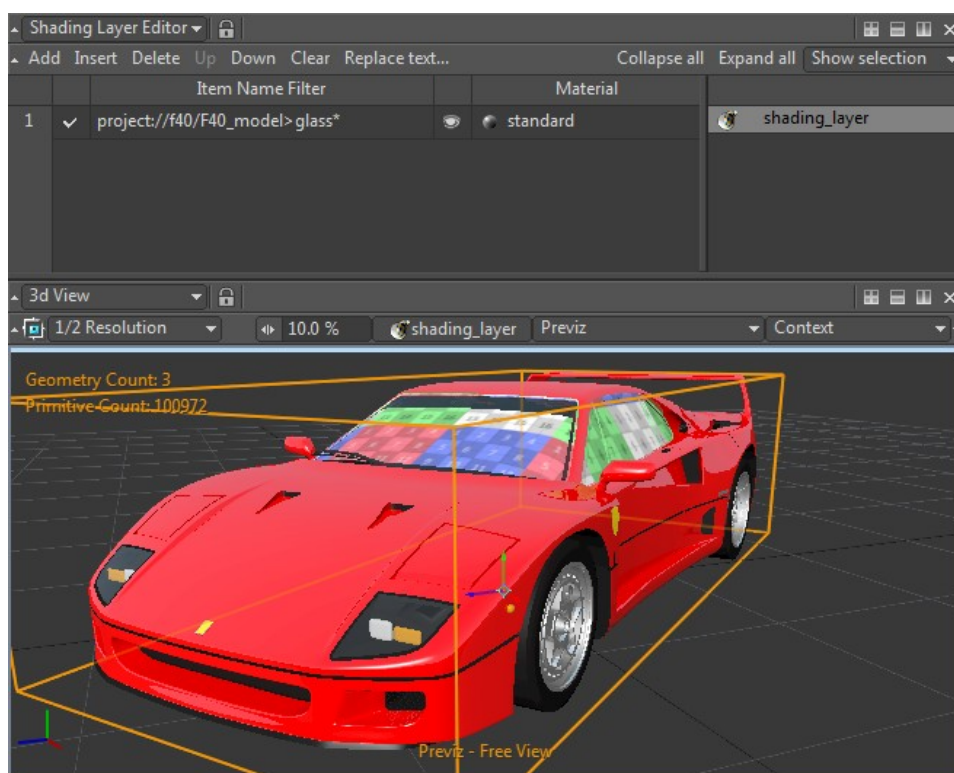
A rule is basically an item path that locates one or multiple geometries or geometry components such as shading groups. Each rule has also a set of properties (more, are likely to come in the future) such as visibility state or material assignment.

Now let's say, you have in your scene an object named `project://f40_layer0` and you would like to assign a material `normal` to all of its shading groups. To do that, you would just need to add a new rule in the shading layer. Set it to its item path and select the normal material.



project://f40_two_layer0 with a normal material applied

One fundamental thing to understand about shading layers is that rules act as overrides. Your geometries can have their materials defined via the material linker or via a *Material Override*, without being affected by a shading layer. **If no rule affects a specific item, it's left unchanged.**



Only the F40 glass shading group is affected by our shading layer.

Item Name Syntax

Item name paths can either be relative or absolute. They support also wildcards and variables. For more information on variables please refer to Working with Variables section.

Relative Path

Unless specified otherwise (project:// or *), all input paths are relative. You may explicitly set a relative path by starting the path with ./

A path is relative when it is relative to the shading layer location in the project. For example, when your shading layer name is project://asset/geometry/f16/f16_shading_layer. When expanding the item path the shading layer will consider its root to be project://asset/geometry/f16.

Relative paths are extremely useful as they are project location independent: you can freely move the context f16 anywhere else in the project and rules won't be affected.

Absolute Path

Absolute path always starts by either `project://` or the wildcard `*`

Absolute paths starting with `project://` are recommended when the context hierarchy structure of your projects is static. Indeed, if you define a rule starting with `project://shot001` and you move `shot001` context into `project://seq001`, then all rules starting with `project://shot001` won't find any item unless you explicitly edit them.

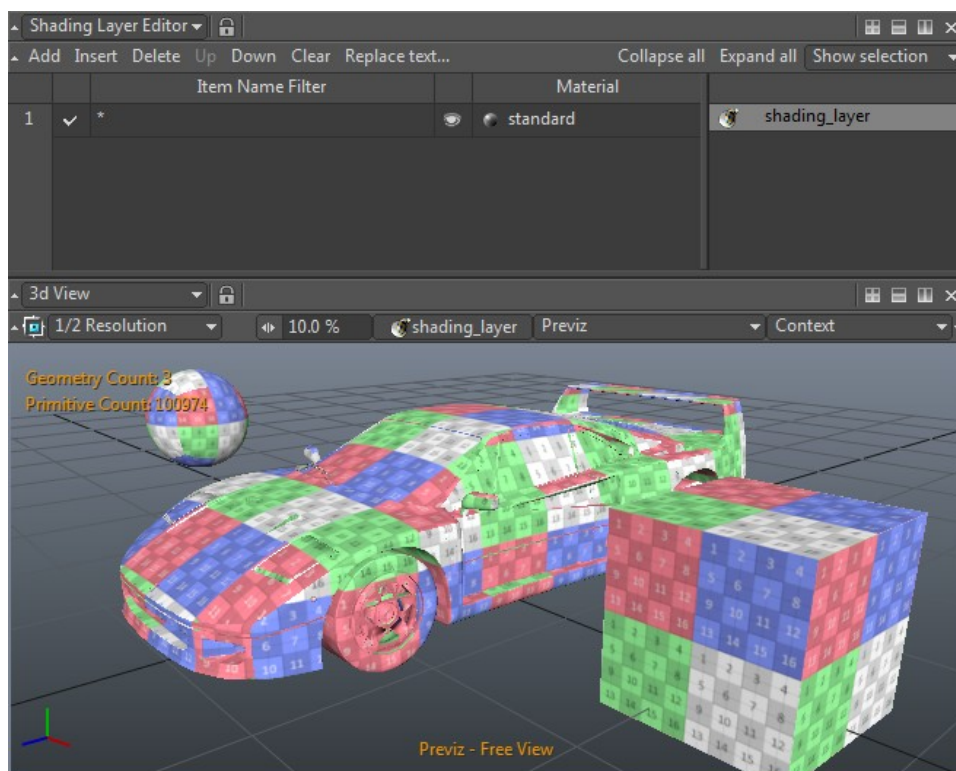
Specifying Shading Groups

A shading group is an item component. To specify a component in a path you need to use the character `>`

For example, if you wish to specify a rule locating a specific shading group, you would write `project://f40_layer0>car_paint`

Wildcards

Item name path supports wildcards. The wildcard character is `*`. `*` and means literally everything so that the simplest valid item name path is `*`. For example, if you wish to assign the same material to every single item in your project you would create a rule with `*` as item name path.



The wildcard * used item name path. Every geometries in the project gets the same material

You can use as many wildcards as you want in an item name path.

- */scene/*item locates any item which name ends with item and that is in a context named scene somewhere in the project.
- */scene/*item* locates any item which name has item in it and that is in a context named scene somewhere in the project.
- */scene/*item*>*paint* locates any item which name has the word item with a shading group name having the word paint and that is in a context named scene somewhere in the project.

Accessing to item's sub-elements

Sometimes, items are made of a group of items. This is typically the case for combiners or scatterers as they are made of a list of referenced items.

Shading layers support material overriding for those objects. You can, for example, assign or change the material of a specific geometry referenced in a combiner.

The syntax is pretty straight forward. For example, to access an item referenced in a combiner, you need to use the subscript operator []. The rank of the item referenced in the combiner is used to identify which item you wish to access.

For example, to access the first item referenced in a combiner you would write: `project://combiner[0]` or `project://combiner[1]` to access the second one.

Note

In scatterers you don't have access to the individual instance. You only have an access to the geometry list used as instances.

This syntax is also recursive. If you would have a combiner referencing another combiner you could write: `project://combiner[0][5]` to access the 6th item of the 1st item in the combiner.

Evaluation Order

Rules are evaluated using a top down scheme. The first rule is evaluated before the second one and so on. Rule evaluation stops when no more items are available.

Basically, the first rule gathers all items in a list. When the rule is processed the remaining unprocessed items are passed to the next rule and so on until no more item or rule are in the list.

For example, if you have 10 different rules with the first one being * the 9 others won't be even processed. In this example, the first rule affects all items in the project.

Now if the * rule comes in second position, the first and the second rules are processed and the evaluation stops at the third one.

Shading Layer Hierarchy

On typical production scenes, Shading layers can be easily made of several thousands of rules. Having thousands of rules in a single shading layer can get very difficult to manage.

For this reason, Shading Layers have been designed to support the concept of hierarchy. This way, you are free to fragment your work in any number of shading layers you like. Then, assemble your final shading layer by parenting shading layer fragments.

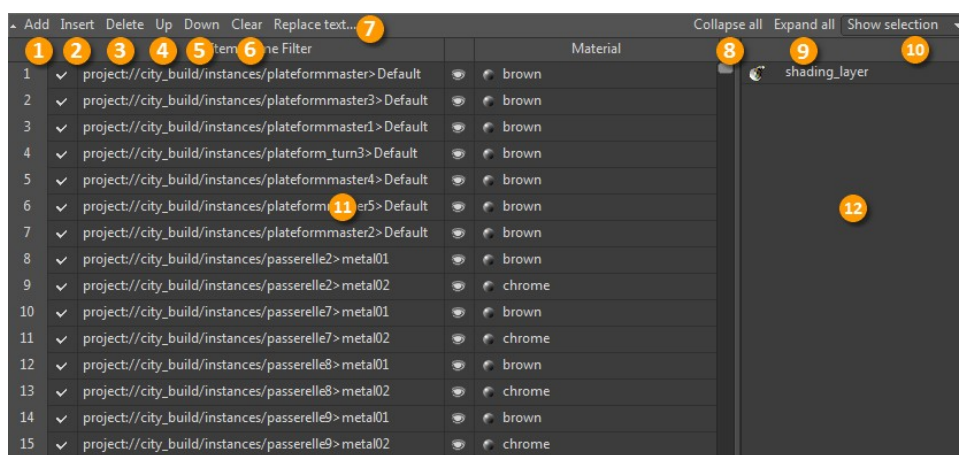
Please note the shading layer hierarchy is completely unrelated to the scene hierarchy. The simplest way to fragment shading layers is to define a single shading layer for each geometry.

Using the Shading Layer Editor

Shading layers are managed almost exclusively thanks to a dedicated widget called the Shading Layer Editor.

Overview

The Shading Layer Editor, is divided in 3 sections: the toolbar, the rule view, the tree view.



The Shading Layer Editor

(1) Add Rule (2) Insert Before Selection (3) Delete Selection (4) Move Selection Up (5) Move Selection Down (6) Clear All (7) Replace Tool (8) Collapse All Hierarchy (9) Expand All Hierarchy (10) Shading Layer Tree Mode (11) Rule View (12) Tree View

Adding New Rules

To add a new rule, press **Add** (1). New rules are appended at the end of the rule list (11).

Inserting Rules

To insert a new rule, press **Insert** (2). New inserted rules are inserted before the rule list selection.

Deleting Rules

To delete rules, select any rule in the rule list (11) and press **Delete** (4).

Shifting Rules Up

To shift rules up, select any rule in the rule list (11) and press **Up** (5).

Shifting Rules Down

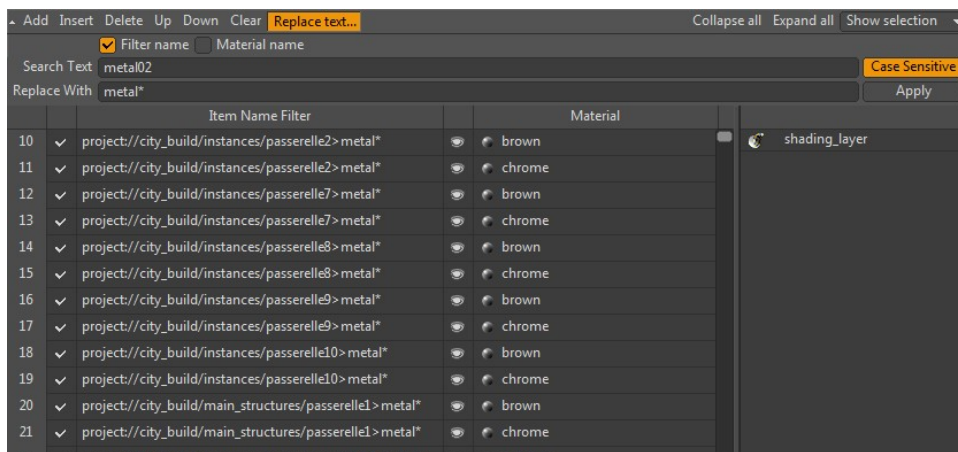
To shift rules down, select any rule in the rule list (11) and press **Down** (5).

Clear All Rules

To clear all rules, press **Clear** (6).

Replace Text

The Shading Layer Editor provides a handy replace tool allowing you to replace item name paths or materials in batch. To bring the replace tool, press **Replace Text...** (7)



Shading Layer Editor Replace Tool

Type what you want to replace in the Search Text field and the replacement in Replace With text field. You can select the replacement to affect item names and or materials. Hit **Apply** when you are done or hit again **Search And Replace** to cancel. By default the search is case sensitive. You can disable the case sensitivity by pressing **Case Sensitive**.

Tree View

The tree view is dedicated to shading layer hierarchy management. Alternatively, you still can use the Attribute Editor to specify shading layers children.

Display Mode

You can customize Tree View display mode using **Shading Layer Hierarchy Tree Mode** (10).

Mode	Description
Show Selection	Display currently selected shading layer with its hierarchy. (Default)
Show Roots Only	Display all shading layer hierarchies.
Show All	Display all shading layers that are in the project. Note: a same shading layer can appear several times within different hierarchies.

Collapse All

To collapse all hierarchies, press **Collapse All** (8).

Expand All

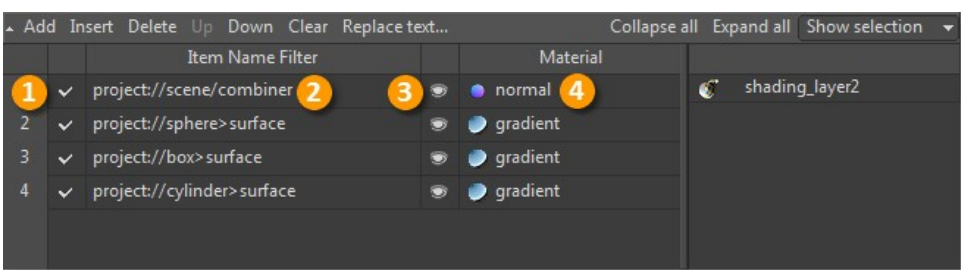
To expand all hierarchies, press **Expand All** (9).

Parent

To parent a shading layer to another one, simple drag and drop the shading layer children.

Rule View

The rule view (11) displays all rules from the currently selected Shading Layer. A rule is composed of an item path and a set of properties. For more information on rules please refer to The Rule Concept section.



Rule View

Enabling/Disabling Rules

To enable/disable rules, click on the check icon (1).

Editing Item Paths

To edit an item path (2), double click on it.

Setting a Material

To set a material, double click on the material field (4) and browse for the material you want to reference. You also can drag and drop the material to the material field.

Toggling Shading Group Visibility

To toggle shading group visibility click on (3).

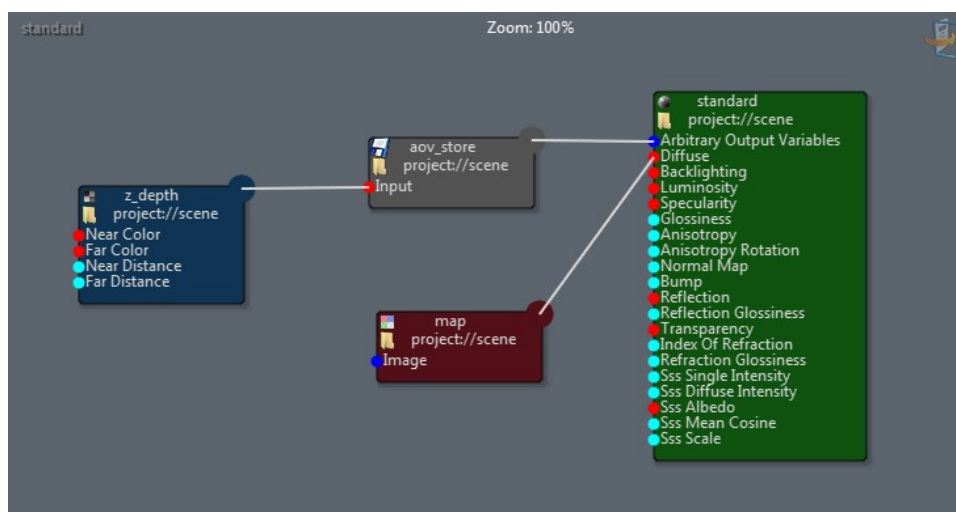
Using AOVs

Clarisse gives you the ability to output arbitrary variables (AOVs). Basically, you can output any texture (or a texture sub graph in your material) result into a unique buffer stored in a rendered image.

AOVs are extremely useful as they allow to easily create separate passes and masks that will be used for compositing.

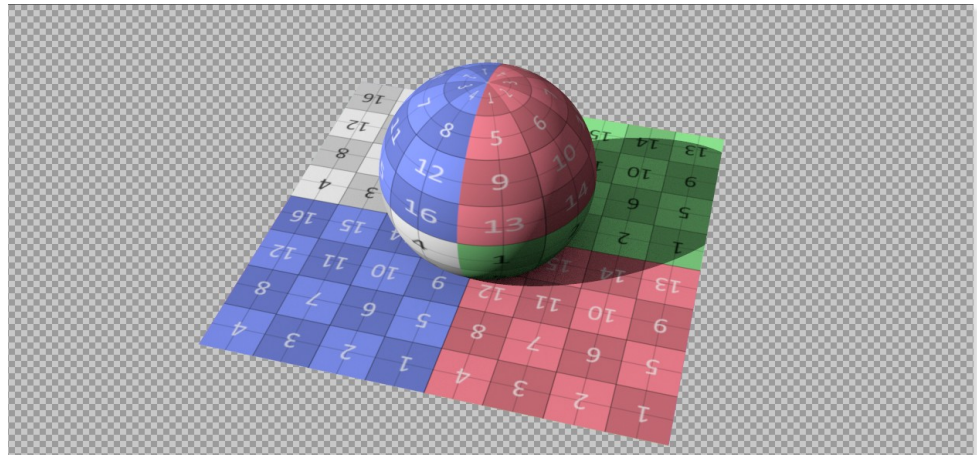
General Workflow

To set up AOVs, you need to use a special item called AOV Store.



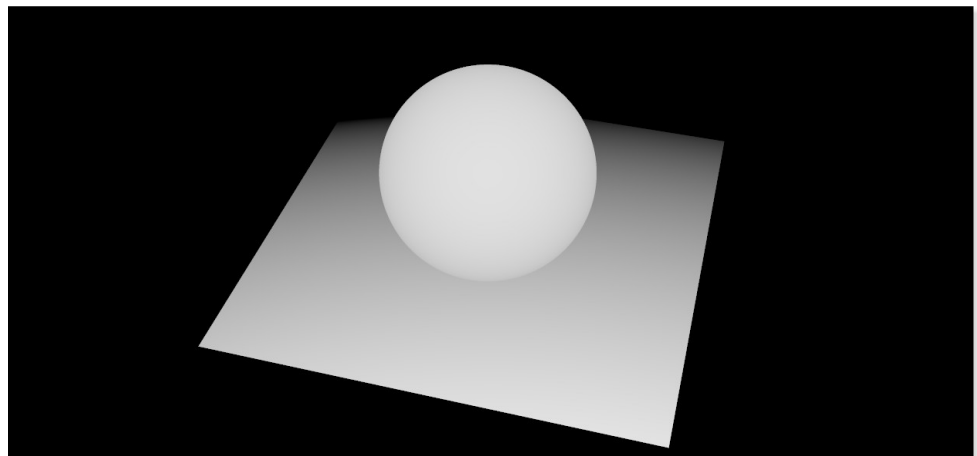
In this simple example we output the Z depth in the pre-defined depth channel

The main idea is to plug the output of a texture into the AOV Store. Choose which channel to output to and then plug the AOV Store to the material(s) you wish to export.



rgba channel

When rendering, the renderer will evaluate the beauty into the pre-defined channel rgba and any AOVs.



depth channel where the result of the AOV is stored

Tip

If you wish to export geometrical information such as world position, normal etc... Please refer to Texture Utility section

AOV Store

The AOV Store is super simple. It has only 2 attributes *Input*, where you plug the texture you want to output, and *Output* where you specify a pre-defined or a custom channel layer. By default, AOV Store returns a black transparent color (0, 0, 0, 0)

Channels and Layers

The concept of layers and channels is really simple. A channel is a named single floating point number where the layer is a named group of channels. Clarisse provides several pre-defined channels and layers such `rgba.red`, `rgba.green`, `rgba.blue` and `rgba.alpha` or `depth.z`. For example `depth` is the layer name and `z` the channel one.

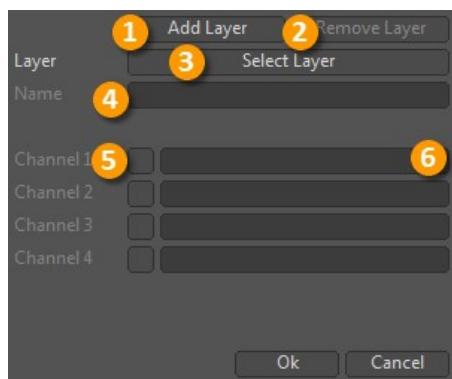
You can create as many channels and layers as you want. While this may change in the future, currently you can't put more than 4 channels in a layer. Finally channel layers definition is saved in the project file.

Managing channels and layers is done using the Channel Layer Editor. To bring Channel Layer Editor go to **Buffers > Edit Channel Layers...** Please refer to Using Channel Layer Editor section for more information.

Using the Channel Layer Editor

The Channel Layer Editor is dedicated to managing channels and layers. It allows you to create, remove, rename layers and channels.

Overview



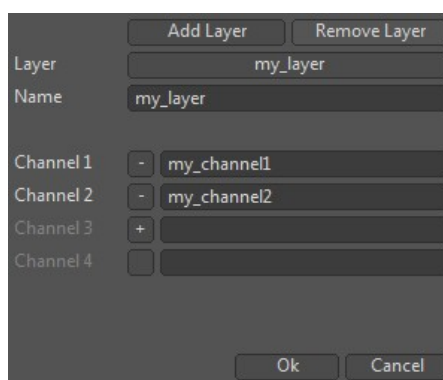
Channel Layer Editor

(1) Add New Layer (2) Remove Current Layer (3) Layer Selection (4) Selected Layer Name (5) Add/Remove Channel (6) Channel Name

Modifications are only applied when pressing **Ok** button.

Adding New Layer

To add a new empty Channel Layer press **Add Layer** (1). This will create a new empty layer named `new_layer`. To rename the layer just change its name using the text edit (4). Please note, Layer names must be unique, you can't have 2 layers sharing the same name. To Add a new channel, press **+** button (5). By default the new channel should be named `channel1`. To rename the channel, just use the text edit (6).



A new layer, `my_layer`, having 2 custom channels.

Removing a Layer

To remove a layer, press (3) to select your layer in the popup menu, then press **Remove Layer** (2). Please remember that all modifications (adding removing...) are only applied if **Ok** is pressed.

Clearing Unused Channels

When you remove a channel in a layer or remove a layer, channels aren't cleared: they are flagged as unused. To remove unused channels just go to **Buffers > Clear Unused Channels**.

Disabling AOVs

You can disable the evaluation of AOVs at several level:

- at material level by disabling *Export AOVs*
- at rendered level by disabling *Export AOVs*
- at Layer 3D level by disabling *ChannelAOVs*

Extracting AOVs

Extracting AOVs from a texture is pretty straight forward if the layer has already been created in Clarisse. Texture Map nodes such as Map File, Steamed Map File, Map have an attribute *Output Layer* in which you can choose the layer you want to output out of the node to the shading tree.

If a layer is not available in Clarisse, you'll need to create it in the Channel Layer Editor to extract it from the texture.

Exporting AOVs to Disk

AOVs are automatically exported to disk when the rendered image is saved. There are two different behaviors though:

- if the output image file format supports an arbitrary number of channels (or layers), a single image containing AOVs is output to disk
- if the output image file format does not support multiple channels, multiple luminance images named after the AOV name and channel are output.

For example, if you choose EXR or TIF then only one file will come out of Clarisse/CNODE. However, if you choose TGA, you'll get multiple images named after:

```
name.padding.layer_name.channel_name.tga
```

Animating in Clarisse

Clarisse provides many tools to animate your items, object attributes and images. You can also create rigs using constraints, deform geometries using deformers and work with image sequences. Please note Time is global to the application and it can be controlled with the timeline widget.

Working with Scene Items

The Scene Item class is an abstract class that defines items that have a transformation. This class also defines the concept of pivot point, item parenting and constraints.

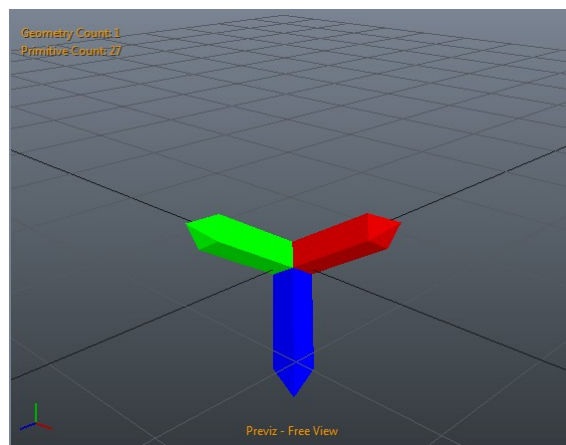
Moving Items

To move an item, simply modify the *Translate* attribute of scene items. The *Translate* attribute is expressed in local space.

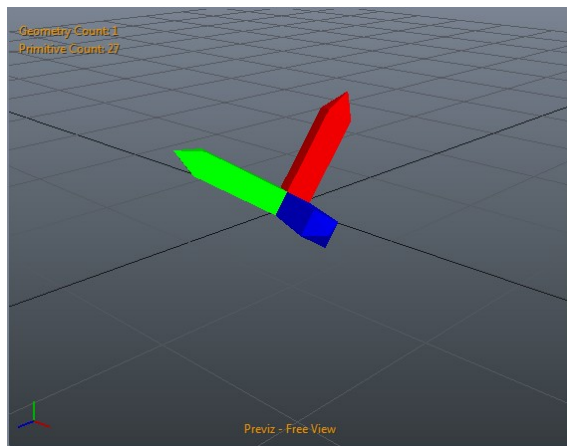
Rotating Items

To rotate an item simply modify the *Rotate* attribute of scene items. By default, the rotation order is in YXZ. You can change the order of rotation by modifying the *Rotation Order* attribute. The *Rotate* attribute is expressed in local space.

Please note modifying rotation order only reinterprets the actual order of rotation. It doesn't recompute the actual rotation order for the new one.



Rotation (45,45,45) in YXZ order



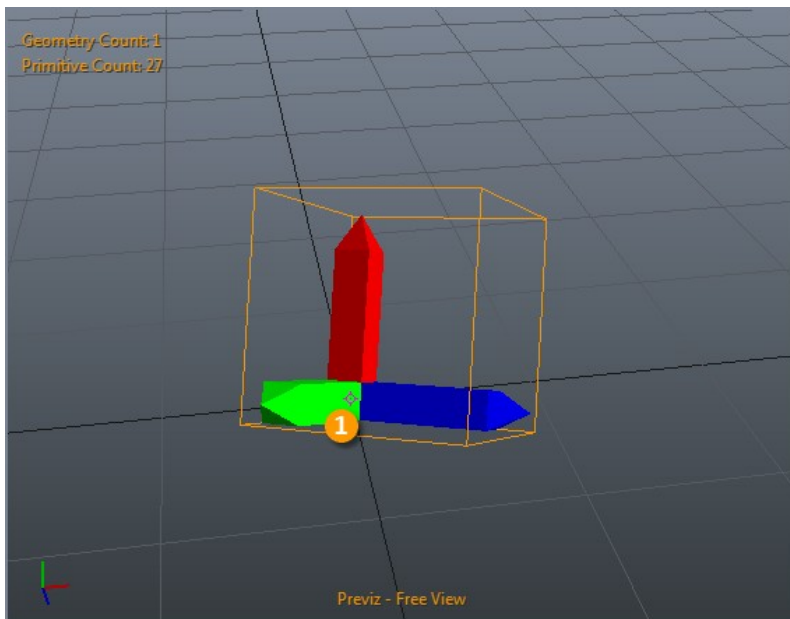
Same rotation but in ZXY order

Scaling Items

To scale an item simply modifies the *Scale* attribute of scene items. The *Scale* attribute is expressed in local space.

Pivot Points

Pivot points control how scene items are translated, rotated or scaled. Scene Items Transformations are always relative to their pivot points. You can edit the pivot of your scene item by using the Attribute Editor or by pressing the Home key while using a translate, rotate or scale tool. When items are selected, their pivot points (1) are displayed as a little overlaid target icon in both 3d and Image View.



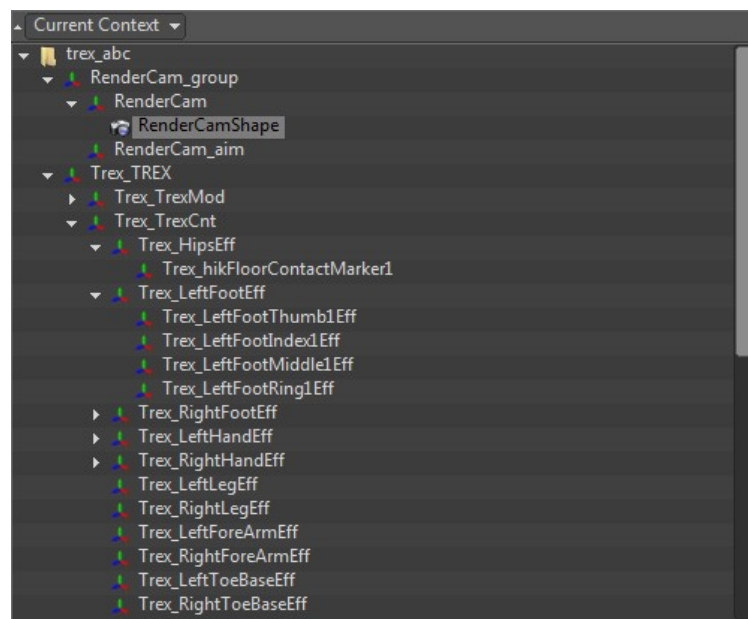
Scene Item pivot point

Parenting Items

To parent an item simply reference it as parent in the *Parent* attribute of the scene item. To unparent an item simply unreference the item.

Using the Hierarchy View

The Hierarchy View allows you to explore and manage scene items by displaying the parenting relationship between items.



Using Constraints

A Constraint allows you to constrain the position, rotation and scale of scene items. Clarisse provides a set of constraints that can be used to create basic rigs. A constraint operates on Scene Item kinematics. It can be used to point an item to another one or to override entirely items motion.

Adding Constraints

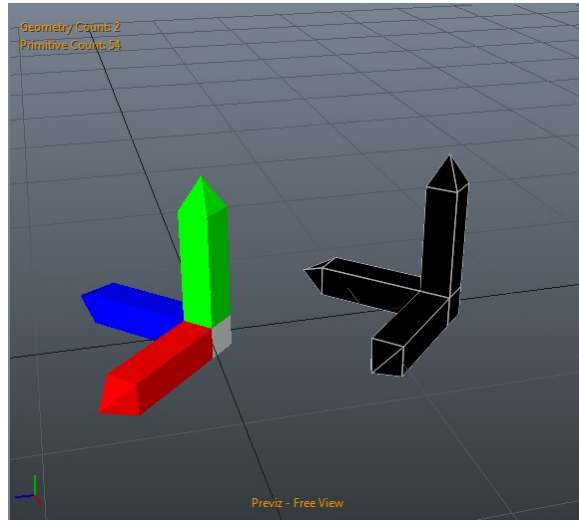
You can add constraints through the Attribute Editor. Look up for the *Constraints* attribute list. Click on the **Add** menu and choose the constraint you want to add to the stack.

Removing Constraints

To remove a constraint, select it in the constraint list in the Attribute Editor and press **Delete** Key.

Orient

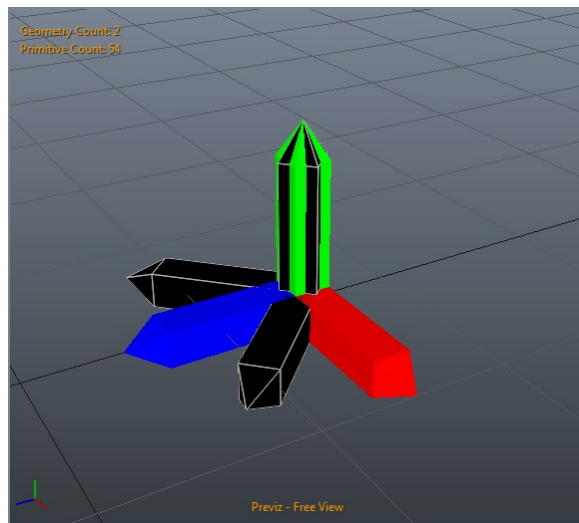
The orient constraint, constrains the item rotation to the rotation of the target.



The colored axis orientation is constrained to the rotation of the black axis.

Point

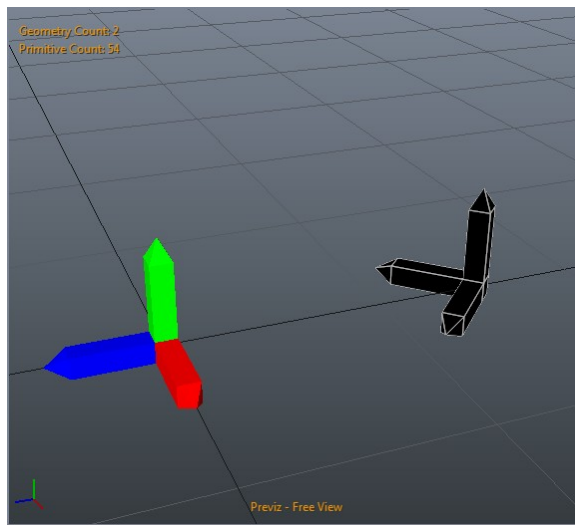
The point constraint, constrains the item position to the position of the target.



The colored axis position is constrained to the position of the black axis.

Scale

The scale constraint, constrains the item size to the size of the target.

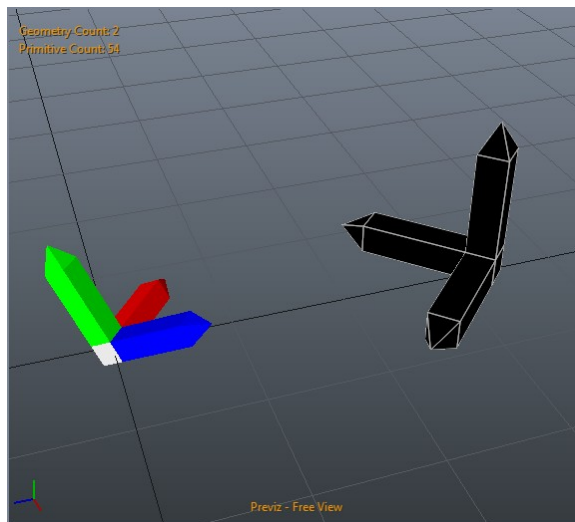


The colored axis size is constrained to the size of the black axis.

Target

The target constraint, constrains the item orientation to target the position of a referenced item. To target an item, you need to specify both up and aim vectors. This constraint is generally used to rig lights and cameras to a center of interest.

Attribute	Description
Target	Sets the item to target
Aim Vector	Sets the vector aiming the target
Up Vector	Sets the up vector



The colored axis orientation is targeting the black axis.

Parent

The parent constraint parents the item to the target. It's basically equivalent of having all 3 orient, point and scale constraints referencing the same target.

MOT Player

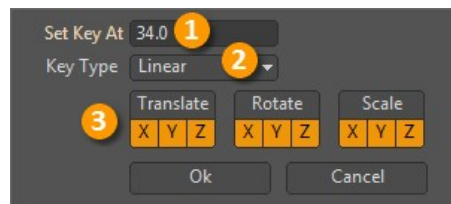
MOT is an ascii motion file format introduced by LightWave 3D. Originating from LightWave 3D, it has been since democratized by the popular commercial suite of plugins called Point Oven. For more information on Point Oven, please visit http://www.ef9.com/ef9/PO1.5/PointOven_15.html

ABC Player

Plays a baked motion from an Alembic file. This constraint is automatically added to scene items when importing an Alembic scene. To import an Alembic scene, go to **File > Import > Scene...** You should only edit its attribute to reference a newer file or to offset the animation.

Set Motion Key

You can add motion keys to the selected scene items by using **Animate > Set Motion Key...** or pressing the Enter key.



(1) Set Keys at (2) Keys Interpolation Type (3) Keyed Motion channels

Delete Motion Key

In the same way you can add motion keys, you can delete motion keys to the selected scene items using **Animate > Delete Motion Key...**

Using Deformers

Any geometries defining vertices can be deformed through time by a stack of deformers. A deformer operates and modifies geometry vertices and can't modify the topology of a geometry. The number of builtin deformers is currently mainly limited to texturable vertex displacement and point cache deformers.

Adding Deformers

You can add deformers through the Attribute Editor. Look up for the *Deformers* attribute list. Click on the **Add** menu and choose the deformer you want to add to the stack.

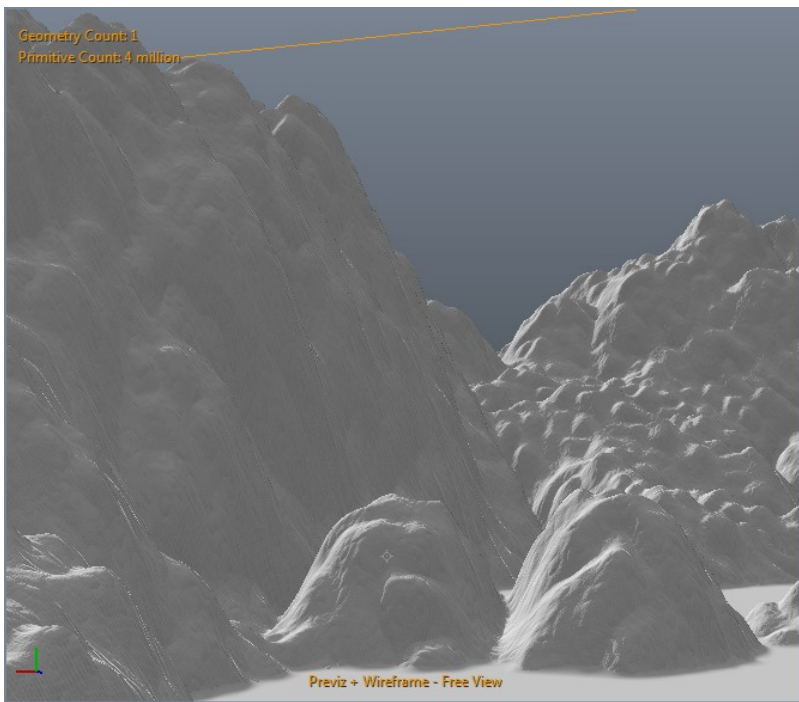
Removing Deformers

To remove a deformer, select it in the deformer list in the Attribute Editor and press **Delete** Key.

Texture Displacement

The Displacement deformer applies a textured displacement to geometry vertices (before actual tessellation). **The displacement deformer isn't a render time geometry displacement.** It has no bounding constraints and is applied before tessellation. For example, it can be used to deform a terrain or displace and animate particles.

Attribute	Description
Weight	Weight of the deformer
Local Deformation	Sets if the deformer is should read local transformation
Scale	Scales the input texture value
Texture	References the texture used for displacement
Displacement Axis	Sets the displacement axis



Displacement deformer on a subdivision surface polygrid

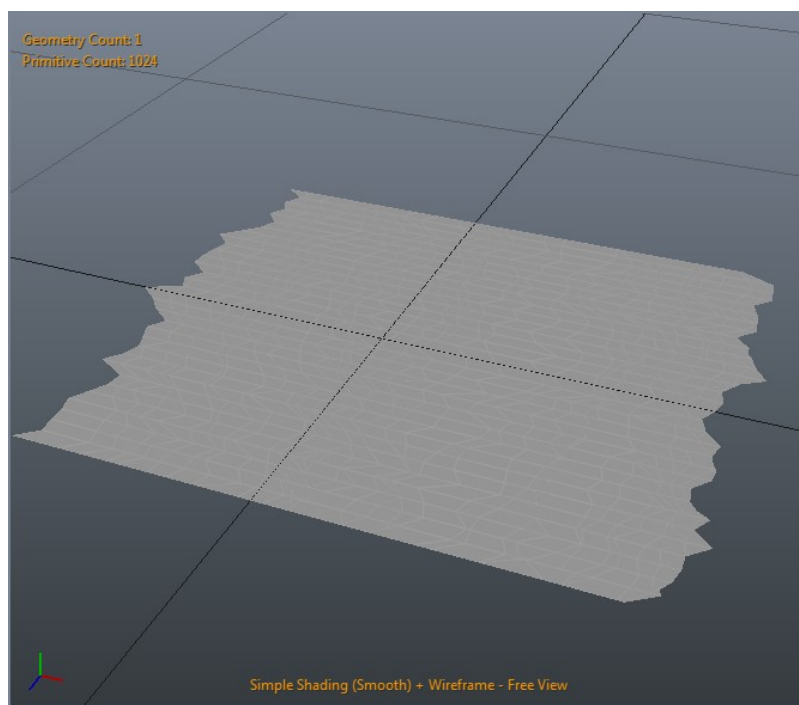
Note

Texture Displacement is applied on vertices. It currently doesn't support UV projection.

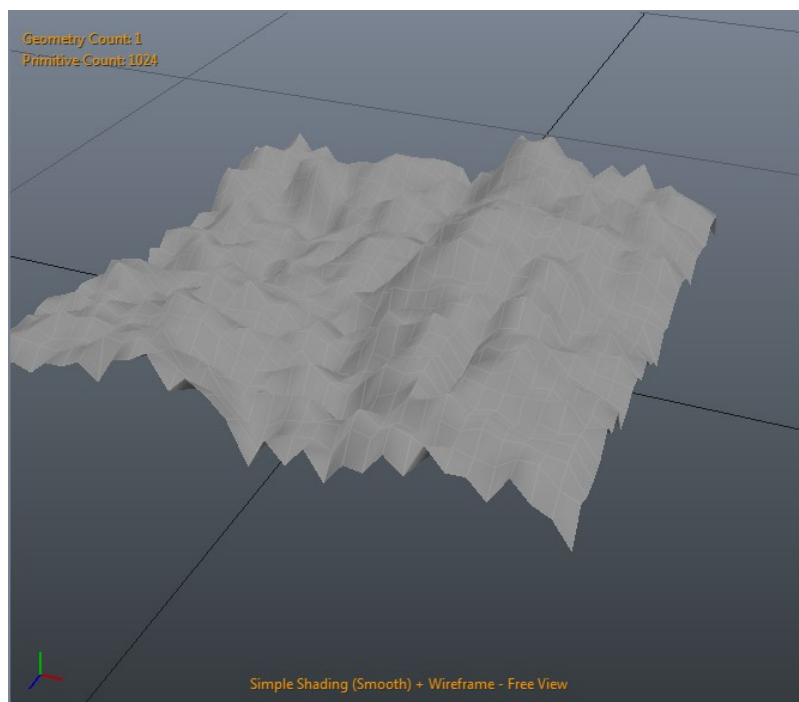
Displacement Axis

You can control the axis of displacement applied to vertices. By default, the deformer is set to *Normal* which displaces vertices along vertex normals. Some geometries such as particles don't provide vertex normals. In this case you can change *Displacement Axis* attribute to displace vertices instead along X, Y, Z or XYZ axis.

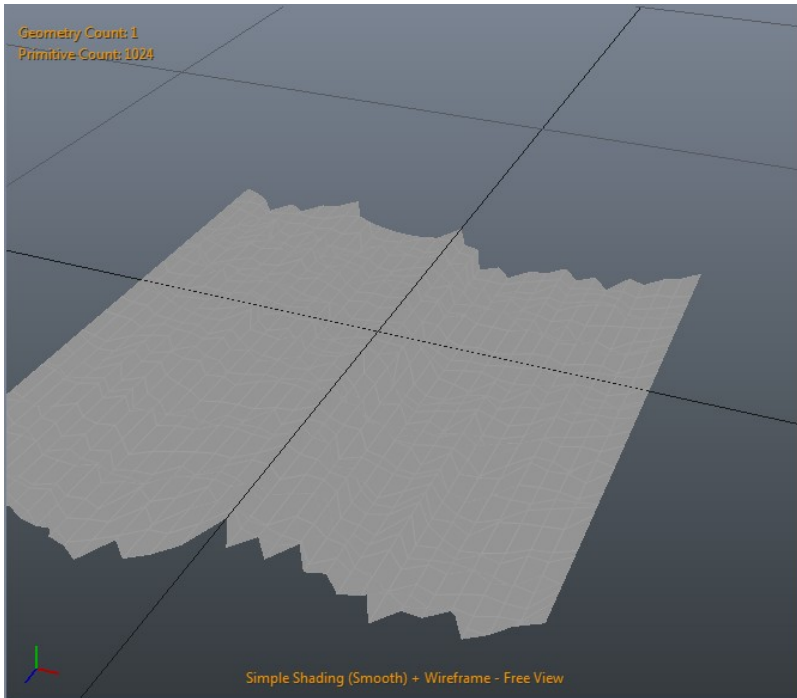
X Axis



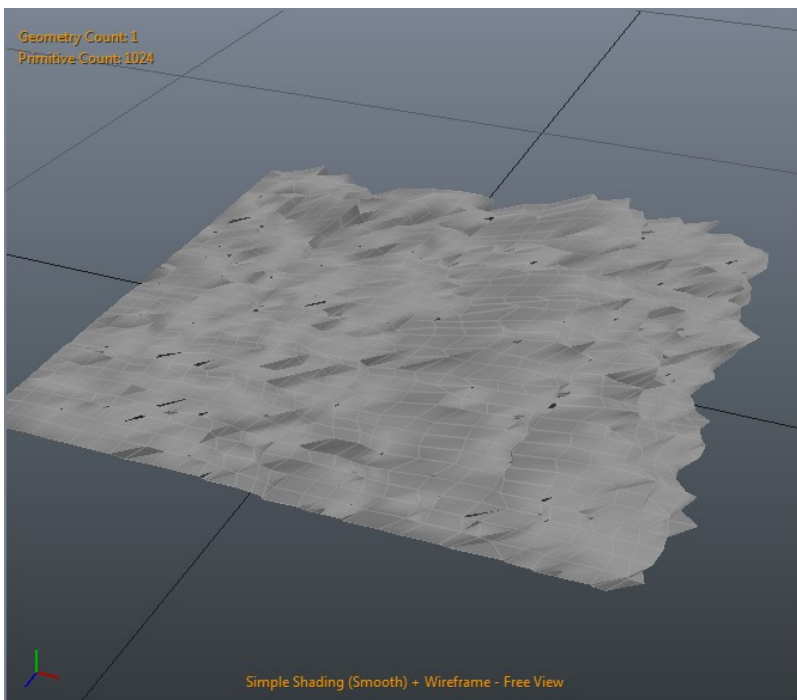
Y Axis



Z Axis



XYZ Axis



MDD Player

MDD is one of the first point cache format available in the industry. Originating

from LightWave 3D, it has been since democratized by the popular commercial suite of plugins called Point Oven. For more information on Point Oven, please visit http://www.ef9.com/ef9/PO1.5/PointOven_15.html

Clarisse implementation of the MDD player streams the MDD file from disk.

Attribute	Description
Filename	Sets the MDD file path
Data Mode	Sets if the MDD point cache data is interpreted as local or global to base geometry.
Start Frame	Offsets the MDD start frame.



Point Oven export from Softimage|XSI

Note

Motion blur on MDD is currently disabled.

ABC Player

Plays a deformation from an Alembic file. This deformer is automatically added to geometries when importing an Alembic scene. To import an Alembic scene, go to **File > Import > Scene...** You should only edit its attribute to reference a newer file or to offset the animation.



Imported Alembic scene rendered in Clarisse (image courtesy of Cluster Studio)

Note

This deformer is deprecated.

Using the Timeline

The Timeline lets you modify the application time. Using the Timeline, you can control playback range and play animations.








Overview

The Timeline is divided in 3 areas: Time Slider, Playback controls and Time fields.



(1) Time Slider (2) Current Time Indicator (3) Start Time Range (4) End Time Range (5) Current Time Field (6) Playback Controls

Playback Controls

Button	Description
	Goes to the start of the playback range
	Goes to previous frame
	Plays in backward the animation between the time range .
	Pauses the animation
	Plays the animation between the time range.
	Goes to next frame
	Goes the the end of the playback range

While playing an animation, you can still work and edit on your project.

Time Slider

To change current time, click on the Time Slider (1) and drag left to move time backward and right to move time forward.

Time Fields

You can set start and end time range directly by typing time value in (3) and (4) fields. To set the current time, use the field (5).

Time Units

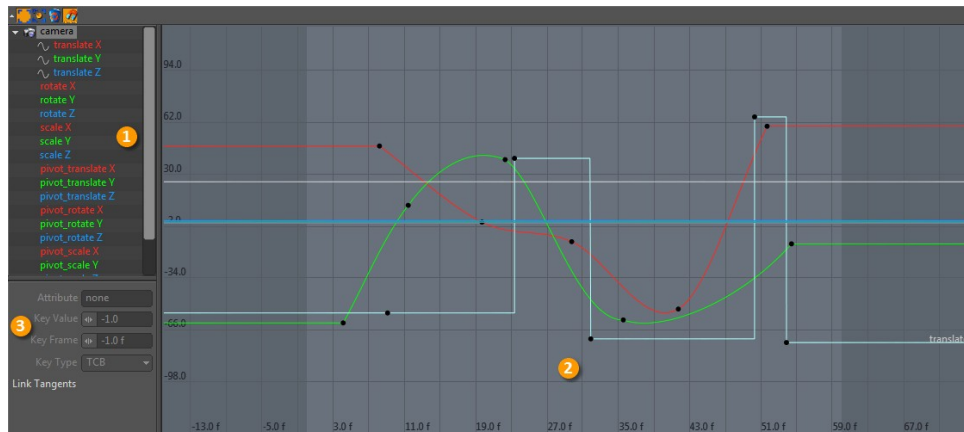
By default Clarisse displays time in frames. You can change time units in the Preferences Panel. To change time units, press **CTRL + K** or go to **Edit > Preferences...** Under *Units* tab change the value of *Time*.

Format	Description
Frame	Displays time in frames (ex: 50 f). Clarisse defaults to measuring time as 25 frames per second. One frame is then 1/25th of a second. You can change <i>Frames Per Second</i> value in the Preferences Panel under the <i>Animation</i> tab
Second	Displays time in seconds (ex: 12.000 s)
Time Code	Displays time in time code format (ex: 00:01:24:12)

Using the Graph Editor

Even if you can animate and set keys using the Attribute Editor, we provide a Graph Editor which is a dedicated widget to create and edit visually FCurves.

Overview



(1) Attribute Tree (2) FCurve Editor (3) Key editor

Attribute Tree

The Attribute Tree manages FCurves visibility in the FCurve Editor. The tree displays the items (1) selected in the application. Under each item you can find the list of animatable attributes (2).



When an attribute is animated, a curve icon (3) is displayed next to its name.

The FCurve Editor displays FCurves according to the Attribute Tree selection. When an item is selected, displays all its FCurves are displayed. To display individual FCurves, individually select attributes instead.

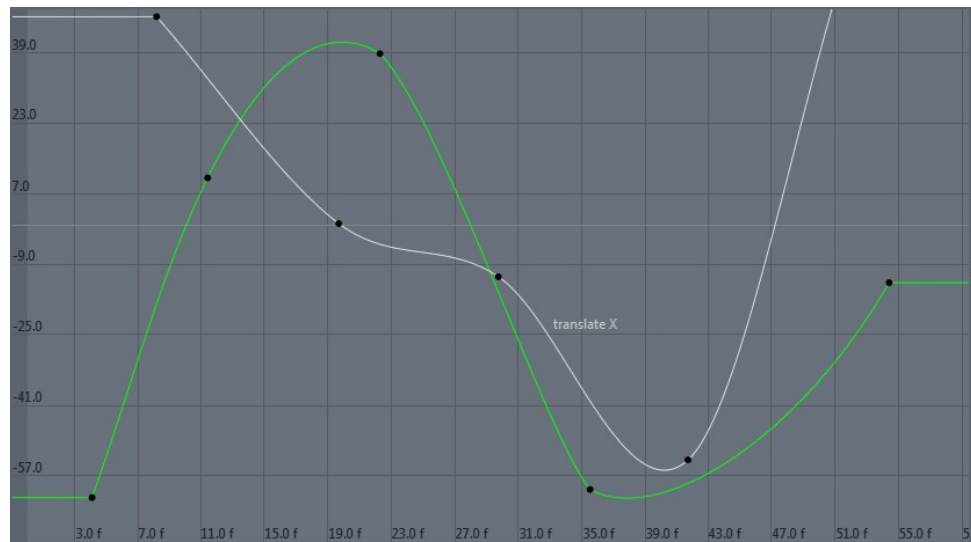
You can select multiple items using Shift or Ctrl Key. You can also freely mix items and attributes in your selection.

Note

The Attribute Tree doesn't affect application selection.

FCurve Editor

The FCurve Editor is where you view, select and edit curves and keys.



To find the curve you are looking for, hover the mouse on any curve to display attribute name the curve is attached to. In the very same way, hovering the mouse on keys displays key information.

Navigation Basics

Panning

To pan press Alt and click in the FCurve Editor using middle mouse button. Drag in any direction to pan the view.

Zooming

To zoom the view, press Alt and right click in the FCurve Editor. Drag left or up to zoom out and right or down to zoom in. You can also use the mouse wheel to

perform zooming.

Fitting

Press F to fit the view to the selected curve or selected keys. If nothing is selected F fits the view to all visible curves.

Selecting

Curves

Click on a curve to select it. When selected, the curve color turns white. To deselect, click on an empty area of the editor.

Keys

To select Key, click on it. You can also perform rectangle selection. Click on an empty area and drag the mouse to draw the rectangle selection. Release mouse button to select keys. To deselect all, click on an empty area of the editor.

Inserting Keys

You can only insert keys on a selected curve. Select a curve and middle click anywhere in the editor to insert a new key. By default, newly inserted keys are using Linear interpolation on an empty curve. This behavior can be changed in the Preferences Panel. Go to Edit > Preferences... Under *Animation* tab set *Default Key Interpolation Type*.

Editing Keys

To edit keys select one or multiple keys and drag. Depending on the key interpolation mode, you can also edit tangents. To edit tangents, simply click and drag the handle.

Removing

Keys

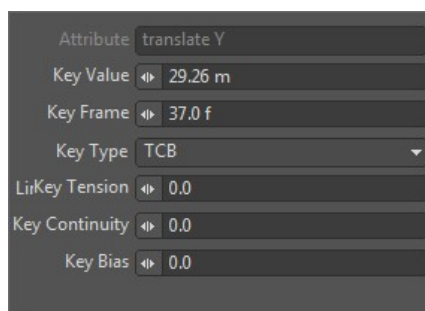
To remove selected keys, press Delete key.

Curves

To remove a curve, delete all its keys.

Key Editor

The key editor allows you to edit key attributes.



You can edit Key value, key time and Key interpolation type. The Key editor works on a selection of keys. Key options are displayed according to the key interpolation type.

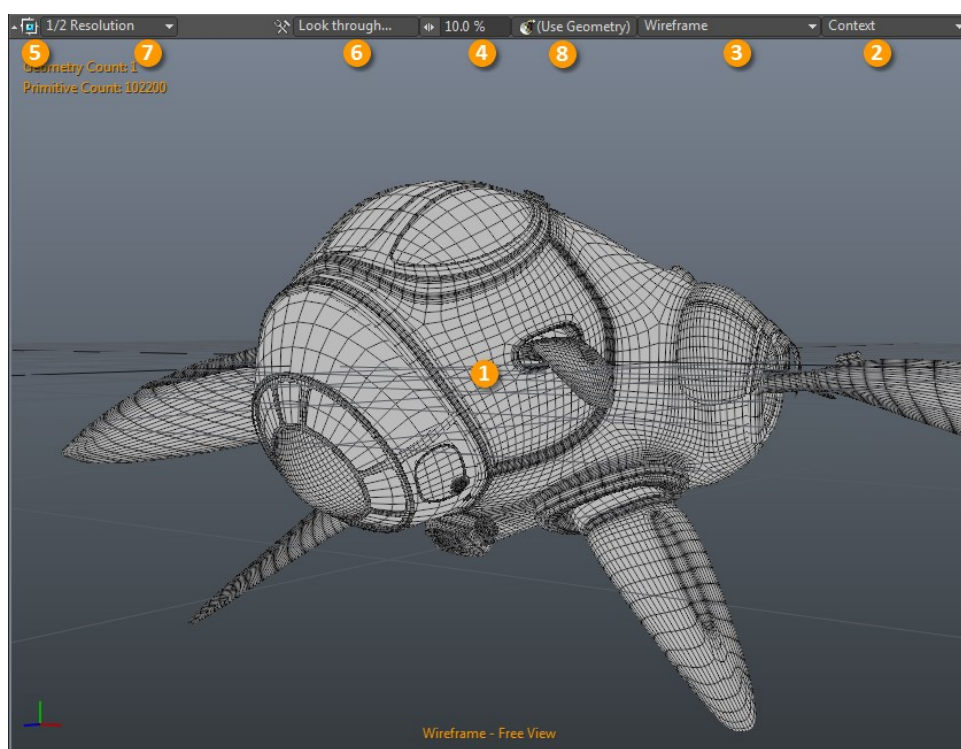
Using the 3D View

The 3D View allows you to freely navigate in your 3D scenes. You can edit, animate items and even display materials. As most widgets, it performs color correction in real-time according to the application settings. This widget relies on Tools for item manipulation. Please refer to Using Tools section.

This widget requires OpenGL 2.0.

Overview

As opposed to 3D packages, Clarisse requires basic graphics card to manipulate and display very complex scenes. The 3D View uses a realtime CPU based raytracer and displays Scene Items. It provides interactive progressive rendering, material display and anti-aliased wireframes.



3D View displaying a geometry

(1) 3D View (2) Display Selection Mode (3) Rendering Mode (4) Shading Quality Multiplier (5) Autofit Selection Mode (6) Look Through Item (7) Display Quality (8) Shading Layer Chooser

Navigation Basics

You can navigate in your 3D scene using the Free View. The Free View is a dedicated perspective camera allowing you to freely navigate in your 3D scenes.

Orbiting

To orbit the viewpoint, press Alt and Click in the 3D View. Drag in any direction to orbit around the center of interest.

Moving

To move the viewpoint, press Alt + Middle click in the 3D View. Drag in any direction to move both the viewpoint and its center of interest in the view plane. To move forward or backward, press Alt + Right click in the 3D View. Drag right or down to move forward, and left or up to move backward.

Fitting

You can fit the view to its content by pressing F. If no item is selected, it fits the view to the bounding box defined by the displayed items. On the other hand, if one or multiple items are selected, it will fit the view to bounding box defined by the selection. To frame all items, press A.

Autofit

By default the 3D View auto fits to selection each time the selection changes. Turn on/off this feature by clicking on (5).

Look Through Item

You can also see through selected lights or cameras by pressing Space key. You can also press **Look through...** browse for an item. If you wish to switch back to the free view, just press Space key again.

Display Selection Mode

By default, the display selection mode is object (Scene Item) mode. The 3D View displays selected items. You can change this behavior by clicking on (2).

Mode	Description
Object	Displays latest selected scene items.
Context	Displays the content of latest selected contexts
Group	Displays the content of latest selected groups
Layer	Displays the content of latest selected layers
Image	Displays the content of latest selected images

The display selection mode is sticky. While in image mode for example, the content of the 3D View will update when image selection changes. You can also explicitly lock the view to a specific image, layer or group using the item in the popup menu.

Rendering Mode

Clarisse requires only basic graphics card to manipulate and display very complex scenes. Built upon a realtime CPU raytracer, you can display from antialiased wireframe to material shaders. To change the display mode, simply click on (4) and select the rendering mode.



3D View in Previz

Mode	Description
Wireframe	Constant color and wireframe
Normal (Flat)	Geometric normal shading
Normal (Flat) + Wireframe	Geometric normal shading and wireframe overlay
Normal (Smooth)	Smooth normal shading
Normal (Smooth) + Wireframe	Smooth normal shading and wireframe overlay
Simple Shading (Flat)	Lambert shading using geometric normals
Simple Shading (Flat) + Wireframe	Lambert shading using geometric normals and wireframe overlay
Simple Shading (Smooth)	Lambert shading using smooth normals

Simple Shading (Smooth) + Wireframe	Lambert shading using smooth normals and wireframe overlay
Previz	Material shading
Previz + Wireframe	Material shading and wireframe overlay

If you switch the 3D View rendering mode to Previz, every items should be displayed in Previz. You can control display level per scene item. In the attribute editor, look for the *Display Mode* attribute. By default, *Display Mode* attribute is set to Previz.

Note

Scene Items have many visibility attributes dedicated to the 3D View. In the Attribute Editor, look for the *Display* category. You will be able to set, for example, items color or items visibility.

Display Quality

Thanks to progressive rendering, you can change the display quality and balance between display speed and quality. To change the current display mode quality, press **Display Quality** (7).

Mode	Description
Fps Driven	The display is driven by its average frame per second. To adjust the 3D View refresh rate, press CTRL + K or go to Edit > Preferences... Under <i>Evaluation</i> tab adjust <i>Layout View Refresh Rate</i> . A lower refresh rate increases quality but has slower feedback.
1/4 Resolution	The 3D View will display at least the display result in quarter resolution before interrupting its rendering.
1/2 Resolution	The 3D View will display at least the display result in half resolution before interrupting its rendering. (Default)
Full Resolution	Display result in full resolution before interrupting its rendering.

Sampling Quality

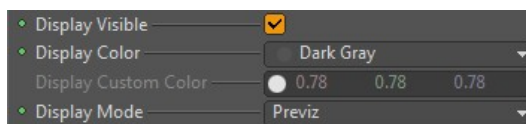
You can control the sampling quality (4) of all materials rendered in the 3D View (Previz Mode). By default, the quality is set to 10%. Shaders are evaluated using only a tenth of the original samples. This is a display only value, it doesn't modify the actual sampling values of the materials.

Note

This value can also be numerically increased to go beyond 100%. In that case original sampling values are increased.

Scene Item Visibility

Scene items have dedicated attributes driving their visibility in the 3D View. These attributes are found in the scene items Display category in the Attribute Editor.



Mode	Description
Display Visible	Toggles item visibility in the 3d View.
Display Color	Item color in the 3d View.
Display Custom Color	Sets a custom color to the item. Requires <i>Display Color</i> to be set to <i>Custom</i> .
Display Mode	Sets the item display mode.

Hiding/Unhiding

Scene item visibility can be controlled by enabling or disabling *Display Visible* attribute. However, Clarisse provides a few shortcuts to help things out:

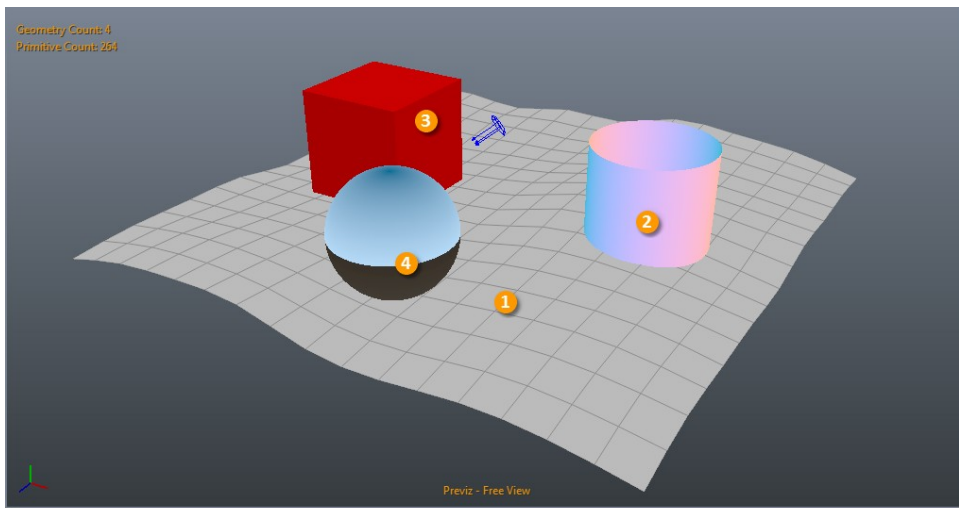
- To Hide selected items, go to **Edit > Hide Selection** or press CTRL + H.
- To Unhide selection, go to **Edit > Unhide Selection** or press Alt + H.

Note

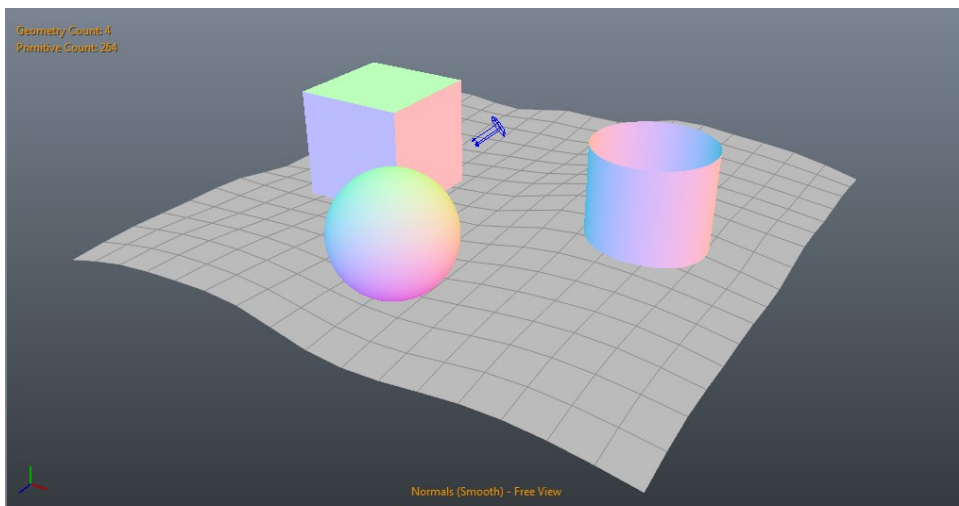
A selected hidden item will always display its bounding box in the 3D View.

Display Mode

Scene items can define their visibility in the 3D View. To change the display mode of a scene item, modify *Display Mode* attribute. The display mode attribute doesn't override 3D view display mode. There is a priority between modes: if an item is set to Previz and the 3D View is set to Simple Shading, the item will be kept displayed in Simple Shading. However, if the item is set to wireframe, it will be displayed in wireframe.



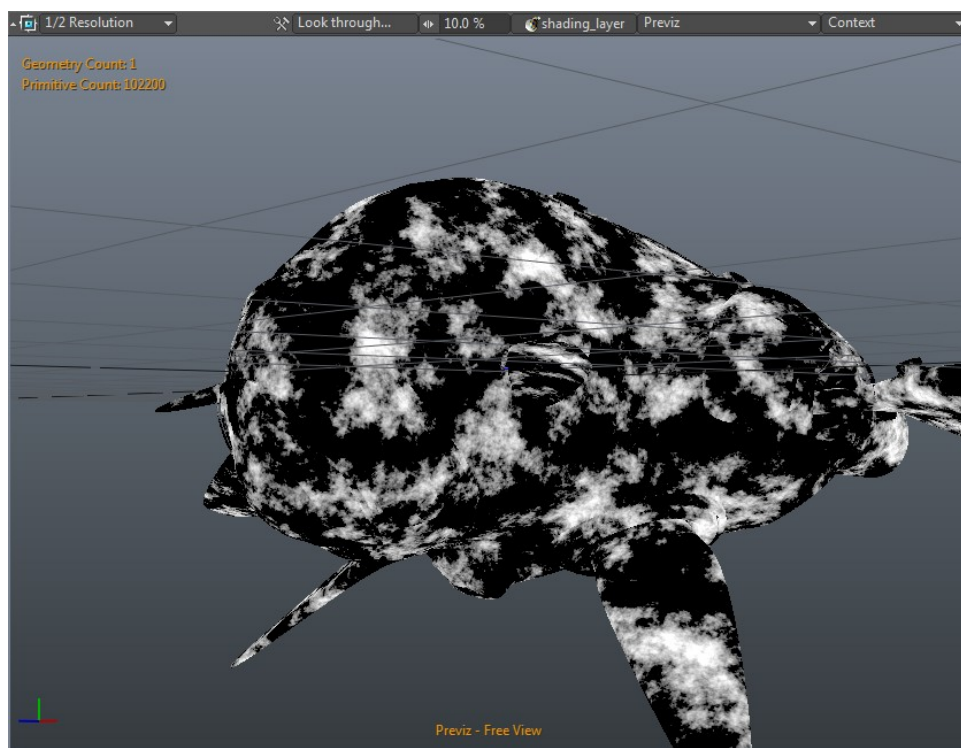
(1) Wireframe Mode (2) Normal Mode (3) Simple Shading Mode (4) Previz Mode



Same scene item settings but with 3d View set to Normal

Shading Layer

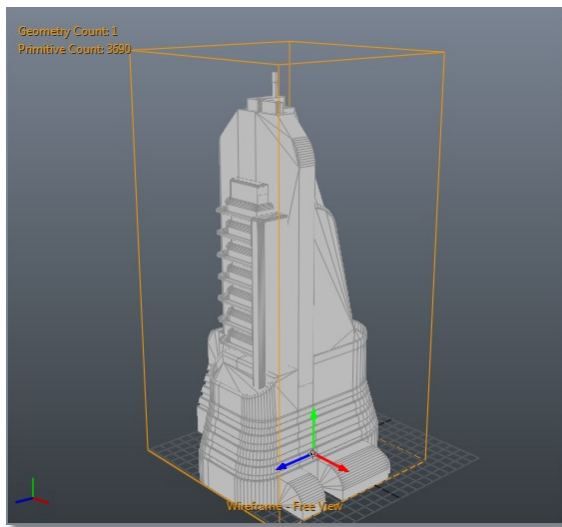
By default, while in Previz mode, the 3D View displays materials that are assigned to geometries or their *Material Override*. You can change this behavior by setting instead a shading layer. To use a shading layer, click on the Shading Layer Chooser button (8) and browse for a shading layer.



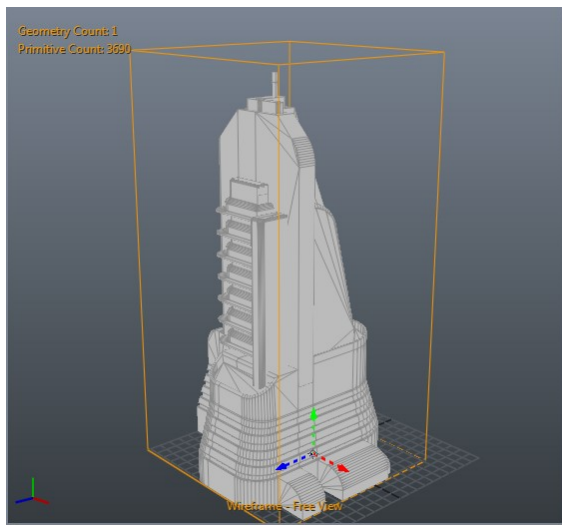
3D View with a shading layer applied

For more information on Shading Layers please refer to Working with Shading Layer section.

normal plane.



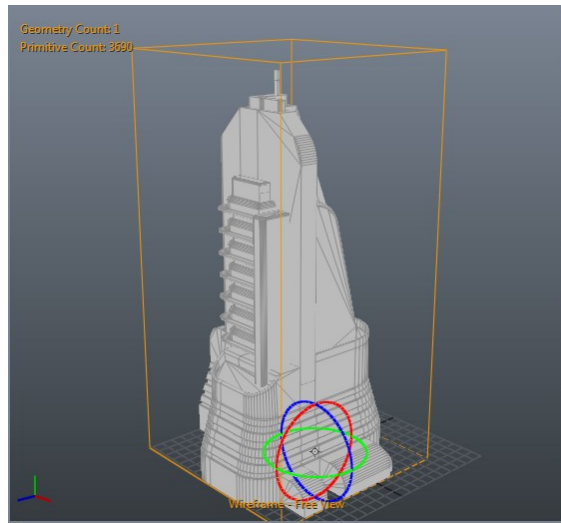
You can also toggle to translate pivot mode by pressing **Home** key. While in pivot mode axis lines are drawn using dashed lines.



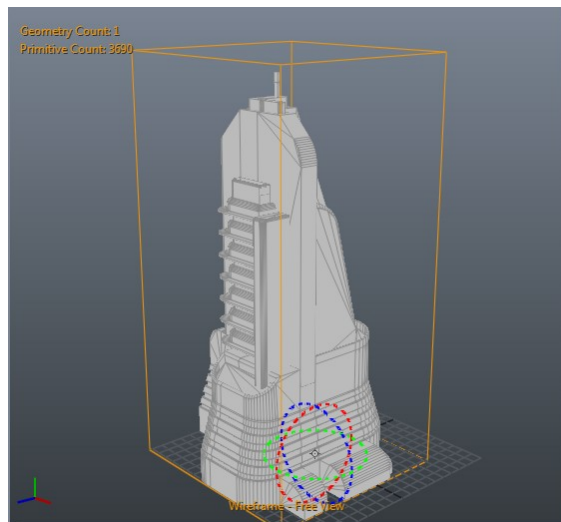
This tool operates on both the Image View and the 3D View.

Rotate Item

This tool allows you to rotate items using gizmos. Pick an axis and drag to rotate the item.



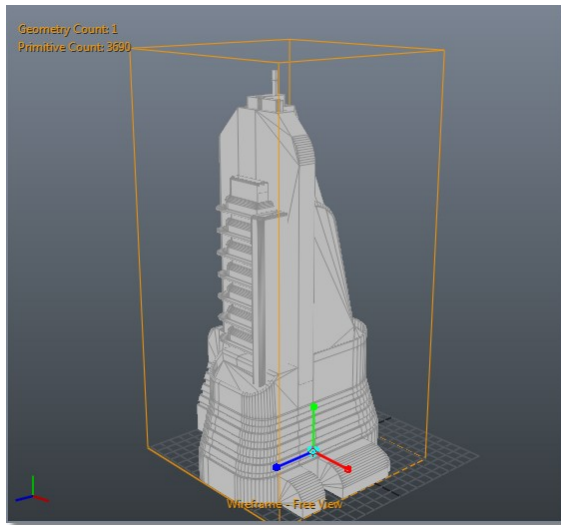
You can also toggle to rotate pivot mode by pressing **Home** key. While in pivot mode axis are drawn using dashed lines.



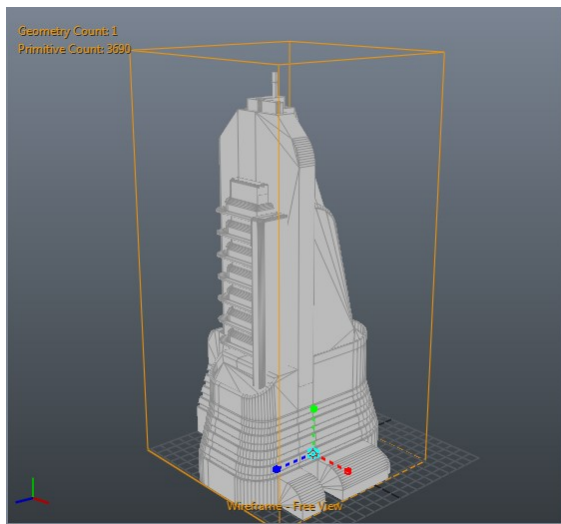
This tool operates on both the Image View and the 3D View.

Scale Item

This tool allows you to scale items using gizmos. Pick an axis and drag to scale the item. Click and drag the middle box to uniformly scale in all axis.



You can also toggle to scale pivot mode by pressing **Home** key. While in pivot mode axis lines are drawn using dashed lines.



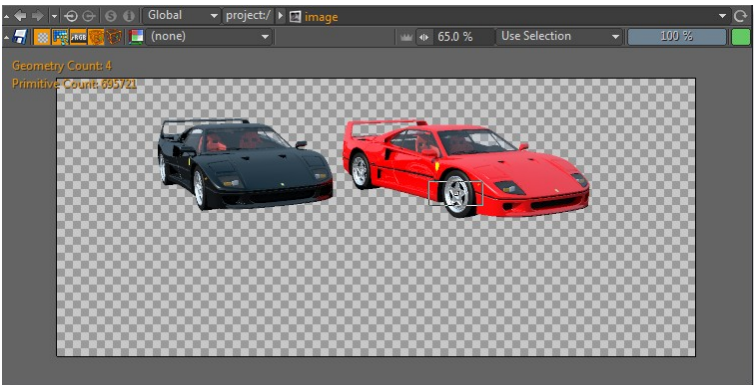
This tool operates on both the Image View and the 3D View.

Pick Fit

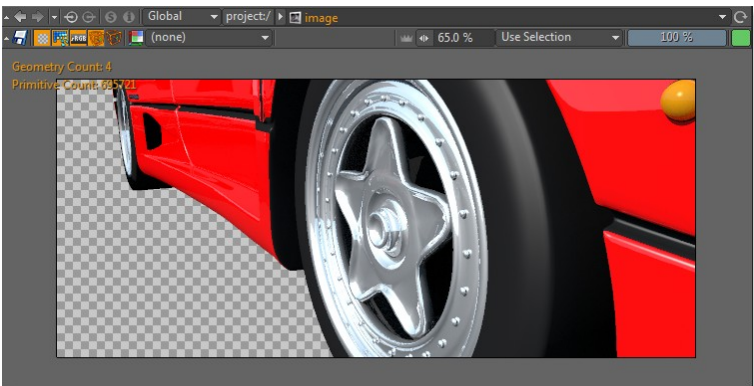
Pick Fit is a very handy tool designed to help you navigate quickly in your 3D Layers. To activate the Pick Fit tool, either select **Tools > Pick Fit** or click on it in the Tools Widget. This tool provides different behaviors that can be set in the using the Tools Options widget. To display the tool options, go to **Window > Tools Options...**

This tool operates on both the Image View and the 3D View.

Mode	Description
Centered Box	Click on an object to pick new center of interest, and drag to set the new frame size. This is the default mode.
Box 2d	Draws a 2d box around two points and fits the view.
Box 3d	Draws a 3d bounding box from two points and fits the view to the 3d bounding box.



Pick fit framing using centered box

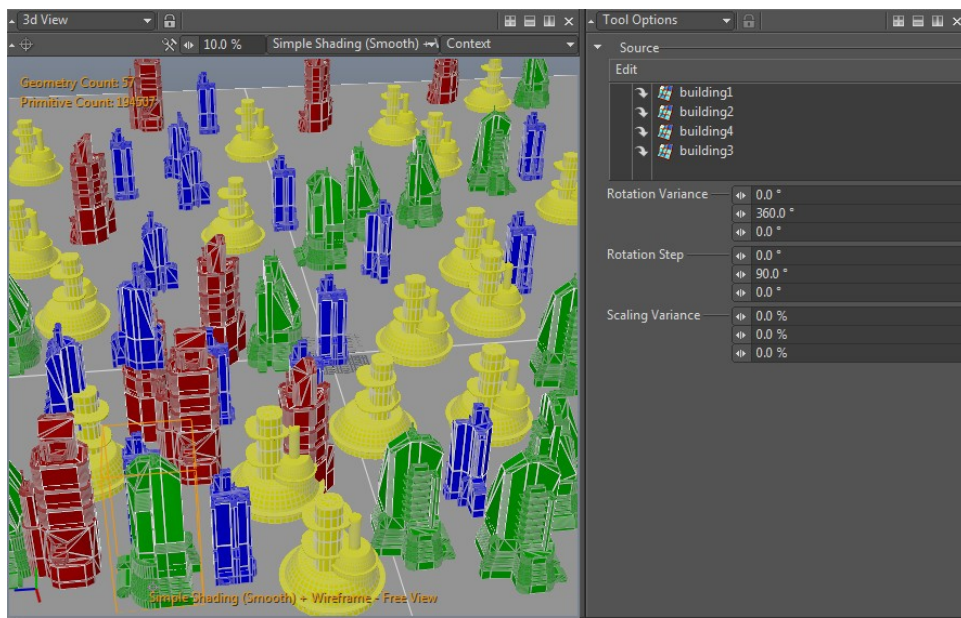


Resulting Pick fit framing

Clone Stamp 3D

This tool is a really handy tool to create complex layouts really quickly. The Clone Stamp 3D tool allows you to stamp Scene Objects on other scene objects. First you need to select one or multiple source objects to stamp. Then directly click on any scene object to stamp your source over. Pasted item pivot points are directly placed on the geometric surface of the picked scene object. If you stamp over an implicit sphere, pasted objects are placed on the surface of the sphere. **Resulting pasted objects are localized instances (Translate, Rotate, Scale)**

of their sources.



To select a source press **Ctrl + Click** over a scene object. To add more items like brushes press **Shift + Ctrl + Click**. You can also use the Tool Options widget to set the list of items and edit scattering parameters.

Attribute	Description
Source	List of items to be used a stamp. The items are randomly chosen from the list.
Rotation Variance	Random rotation value applied to pasted instances.
Rotation Step	Rotation constraint applied to the random rotation. Very useful when pasting buildings for example. If set to 90 degrees, resulting rotation will be a multiple of 90.
Scaling Variance	Random scale value applied to pasted instances.

You can paste any scene objects, including Combiners and Scatterers to any other scene objects.

This tool operates on both the Image View and the 3D View.

Picker

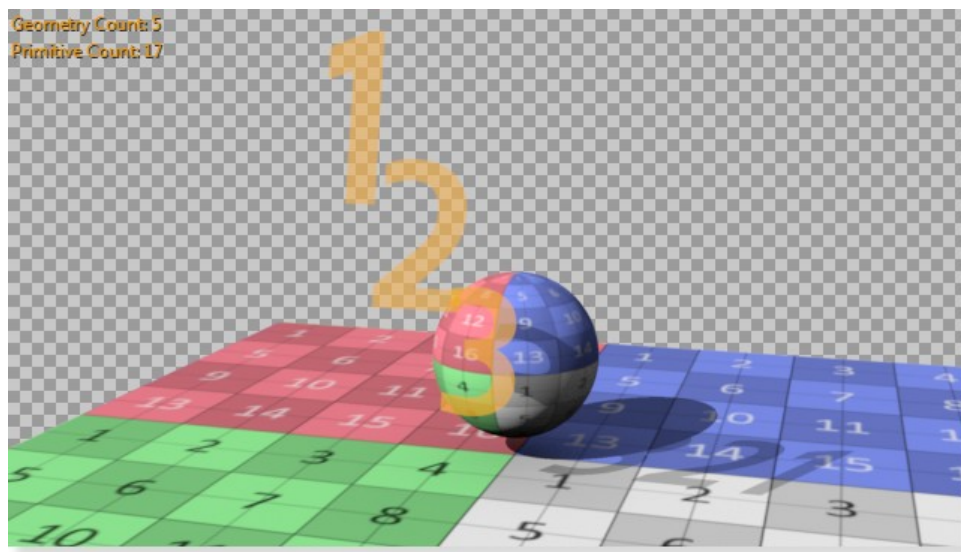
This tool allows you to pick for an underlying color or an underlying item. By default, left click retrieve underlying color while middle click picks the material. You can control picking behavior using the Tool Options:

Mode	Description
Color	Picks underlying rendered color (the selected color can be seen in the Tools widget)
Layer	Selects underlying Layer
Scene Object	Selects underlying Scene Object
Material	Selects underlying Material

Currently, this tool operates on the Image View only.

Tools and Transparency

Depending on what you want to do, you may either want to pick transparent items or only opaque ones.



A typical example where 3 transparent quads are in front of the sphere.

Most of the times, Tools dedicated to selection and picking have a special attribute to deal with transparency. This attribute, called *Opacity Threshold*, is set to 50% by default. By default, Tools consider items that are over than 50% transparent as invisible.

To change the opacity threshold of the currently selected tools, go to **Window > Tools Options...** and change the value of *Opacity Threshold*. When set to 100%, transparent items are invisible to selection whereas set to 0% transparent items are always selectable.

Using the Resource View

Clarisse is built upon a smart memory manager that detects and eliminates, automatically and on the fly, redundant data. These data are not only limited to geometries or images. They can be of any kind.

In Clarisse, these data are called Resources. A resource is any arbitrary data that can be potentially shared by other project items. Resources are automatically created or loaded during evaluation.

As there can be several thousand of resources in a typical production project, Clarisse provides a dedicated widget to manage those resources.

Overview

The Resource View allows to manage, sort, track and clear resources that are currently in Clarisse memory. It's divided into two parts: the resource toolbar and the resource list.

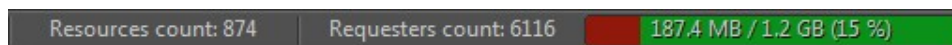
	Path	Class	Tag	Size	Requester Count
1	project://Hidden_Sources/batmod_01bot	GeometryPolyfile	geometry_smoothed	10.9 KB	9
2	project://city_build/main_structures/detailpad01	GeometryPolyfile	geometry_smoothed	5.2 KB	33
3	project://city_build/instances/towerside	GeometryPolyfile	gas	105.8 KB	2
4	project://Scatter_Supports/scatter_towerguide	GeometryPolyfile	tessellation_uv_maps	20.8 KB	1
5	project://city_build/bigtower/mainbuild_foot	GeometryPolyfile	surface_geometry	5.0 MB	8
6	project://Hidden_Sources/road	GeometryPolyfile	geometry	99.0 KB	11
7	project://city_build/fog/foggrid	GeometryPolygrid	geometry	1.1 KB	5
8	project://city_build/bigtower/mainbuild_body	GeometryPolyfile	gas	901.7 KB	1
9	project://city_build/main_structures/building1	GeometryPolyfile	gas	44.3 KB	1
10	project://city_build/main_structures/platform_turn01	GeometryPolyfile	geometry_smoothed	7.6 KB	16
11	project://city_build/instances/ground_m2	GeometryPolyfile	surface_topology	32.8 KB	1
12	project://Hidden_Sources/rocks/rock04	GeometryPolyfile	gas	2.4 MB	1
13	project://Hidden_Sources/batmod_01top	GeometryPolyfile	geometry	1.3 MB	4
14	project://city_build/main_structures/miniplate	GeometryPolyfile	geometry	32.1 KB	5
15	project://city_build/main_structures/building5	GeometryPolyfile	geometry	114.4 KB	9
16	project://city_build/main_structures/roundbat	GeometryPolyfile	geometry	663.5 KB	62
17	project://Hidden_Sources/rocks/rock02	GeometryPolyfile	gas	2.1 MB	1
18	project://city_build/bigtower/mainbuild_base	GeometryPolyfile	geometry	115.9 KB	3
19	project://city_build/main_structures/domebat	GeometryPolyfile	gas	4.3 MB	1
20	project://Shading/roads/map_file	TextureMapFile		3.0 MB	1

The Resource View

(1) Empty Resource Visibility Toggle (2) Display File Resource only Toggle (3) Show Shared Resources (4) Clear Resource (5) Reload Resource (6) Resource Information (7) Resource Memory Statistic (8) Resource List

Memory Statistics

The Resource View displays memory statistics related to what is currently loaded in memory. These statistics only cover resource memory consumption. To have more complete statistics please use **Help > Log Statistics...** or press **CTRL + Shift + F1**.



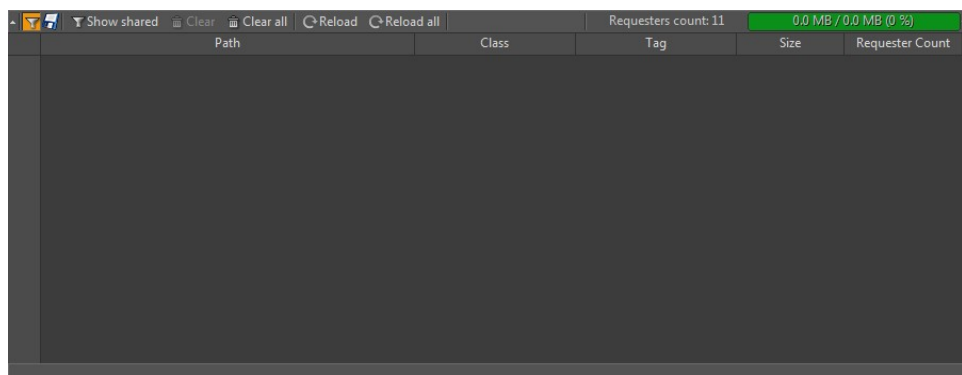
Resource count is the total number of resources declared in the memory (empty or not).

The **Requester count** displays the number of items requesting resources. An item can request one or more different resources. For example, geometries can request over 20 different resources.

At the right is located memory usage statistics. The first digit displays a good approximation of the physical memory used by all resources. The second digit displays the amount of RAM Clarisse would virtually need if there were no de-duplication. Finally the last digit, in percentage, is the ratio between real memory usage over the total virtual memory usage. **Lower values mean high level of de-duplication.**

Empty Resources

When an item is created without being evaluated (not rendered or seen...), its resources are likely to be empty: Clarisse allocates data only when necessary. The Hide Empty Resources (1) toggle button allows you to filter the display of declared empty resources in the Resource List (8).



In this scene we've created only an empty implicit Box. It has only empty resources as no widgets required any of its data.

	Path	Class	Tag	Size	Requester Count
1	project://box	GeometryBox		0.0 MB	8
2	project://box	GeometryBox		0.0 MB	1
3	project://box	GeometryBox		0.0 MB	1
4	project://box	GeometryBox		0.0 MB	1

Same scene with Hide Empty Resource off. We see all its resources are empty.

	Path	Class	Tag	Size	Requester Count
1	project://box	GeometryBox	geometry	8.0 B	8
2	project://box	GeometryBox	gas	1.3 KB	1
3	project://box	GeometryBox		0.0 MB	1
4	project://box	GeometryBox		0.0 MB	1

After displaying a 3D View showing the Box, 2 of its resources have been automatically created.

	Path	Class	Tag	Size	Requester Count
1	project://box	GeometryBox	geometry	8.0 B	8
2	project://box	GeometryBox	gas	1.3 KB	1

Renabling Hide empty resource displays only 2 resources.

File Resources

There are two kind of resources: generated resources or resources originating from a file. Using the Resource View you can filter the resource list to display only file resources by clicking on the disk icon (2).

	Path	Class	Tag	Size	Requester Count
1	project://Hidden_Sources/road	GeometryPolyfile	geometry	99.0 KB	11
2	project://Hidden_Sources/batmod_01top	GeometryPolyfile	geometry	1.3 MB	4
3	project://city_build/main_structures/miniplate	GeometryPolyfile	geometry	32.1 KB	5
4	project://city_build/main_structures/building5	GeometryPolyfile	geometry	114.4 KB	9
5	project://city_build/main_structures/roundbat	GeometryPolyfile	geometry	663.5 KB	62
6	project://city_build/bigtower/mainbuild_base	GeometryPolyfile	geometry	115.9 KB	3
7	project://Shading/roads/map_file	TextureMapFile		3.0 MB	1
8	project://Hidden_Sources/rocks/rock03	GeometryPolyfile	geometry	1.2 MB	3
9	project://Shading/rocks/citytile/map_file1	TextureMapFile		8.2 MB	1
10	project://Hidden_Sources/rocks/rock02	GeometryPolyfile	geometry	2.2 MB	3
11	project://city_build/main_structures/gateway	GeometryPolyfile	geometry	3.0 MB	5
12	project://city_build/bigtower/mainbuild_body	GeometryPolyfile	geometry	2.2 MB	3
13	project://Shading/generic_concrete/cleanconcrete/logo	TextureMapFile		1.0 MB	11
14	project://Hidden_Sources/turnL	GeometryPolyfile	geometry	259.9 KB	7
15	project://city_build/spaceships/pointcloud_ships	GeometryPolyfile	geometry	61.2 KB	3
16	project://Shading/rocks/bgrocks/map_file	TextureMapFile		827.6 KB	1
17	project://city_build/main_structures/needle01	GeometryPolyfile	geometry	12.1 KB	120
18	project://city_build/main_structures/building4	GeometryPolyfile	geometry	191.4 KB	21
19	project://Shading/roads/roadsides/lightdots_mask	TextureMapFile		207.2 KB	6
20	project://city_build/main_structures/passerelle	GeometryPolyfile	geometry	787.5 KB	24

File filter enabled, the Resource View displays only file resources

If a file has been modified, you can reload file resources at any time and even during an evaluation.

To reload one or more resources, just select them in the resource list and press **Reload** (5). Clarisse reloads files only if they are modified.

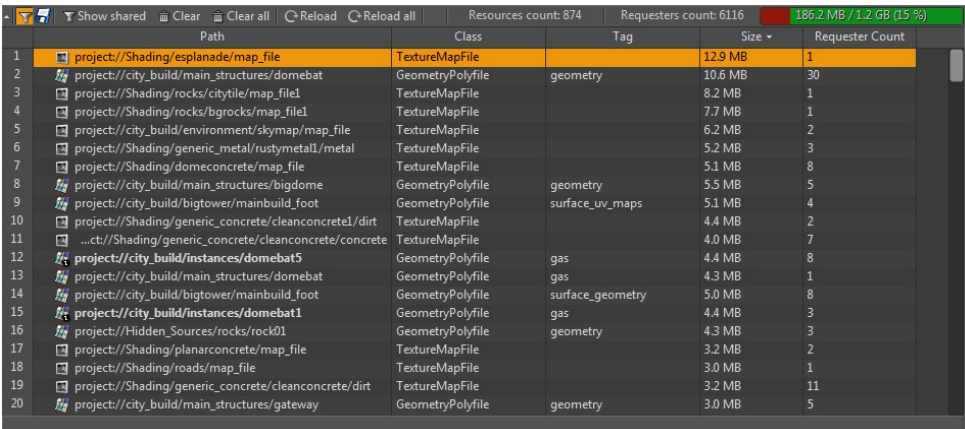
Alternatively, you can use **Reload All** (5) to reload any modified file resources or use **File > Reload Resources** or **CTRL + R**.

Clearing Resources

You can clear any existing resources at any time and even during an evaluation. To clear a resource, either select it in the resource list (8) or select an object in Clarisse using the browser for example. Then press **Clear** (3).

An easy way to clear all resources used by an item, is to select this item in Clarisse. When you select an object in Clarisse, all its resources are selected in the Resource View.

If you wish to free memory, you can also sort the resource list (8) by memory usage. Click on the Size column to sort items in ascending or descending order.



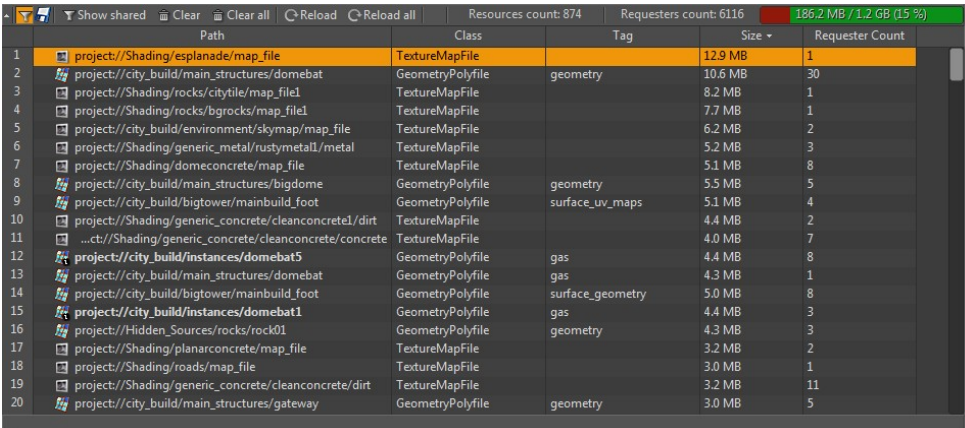
	Path	Class	Tag	Size	Requester Count
1	project://Shading/esplanade/map_file	TextureMapFile		12.9 MB	1
2	project://city_build/main_structures/domebat	GeometryPolyfile	geometry	10.6 MB	30
3	project://Shading/rocks/citytile/map_file1	TextureMapFile		8.2 MB	1
4	project://Shading/rocks/bgrocks/map_file1	TextureMapFile		7.7 MB	1
5	project://city_build/environment/skymap/map_file	TextureMapFile		6.2 MB	2
6	project://Shading/generic_metal/rustymetal/metal	TextureMapFile		5.2 MB	3
7	project://Shading/domeconcrete/map_file	TextureMapFile		5.1 MB	8
8	project://city_build/main_structures/bigdome	GeometryPolyfile	geometry	5.5 MB	5
9	project://city_build/bigtower/mainbuild_foot	GeometryPolyfile	surface_uv_maps	5.1 MB	4
10	project://Shading/generic_concrete/cleanconcrete1/dirt	TextureMapFile		4.4 MB	2
11	...ct://Shading/generic_concrete/cleanconcrete/concrete	TextureMapFile		4.0 MB	7
12	project://city_build/instances/domebat5	GeometryPolyfile	gas	4.4 MB	8
13	project://city_build/main_structures/domebat	GeometryPolyfile	gas	4.3 MB	1
14	project://city_build/bigtower/mainbuild_foot	GeometryPolyfile	surface_geometry	5.0 MB	8
15	project://city_build/instances/domebat1	GeometryPolyfile	gas	4.4 MB	3
16	project://Hidden_Sources/rocks/rock01	GeometryPolyfile	geometry	4.3 MB	3
17	project://Shading/planarconcrete/map_file	TextureMapFile		3.2 MB	2
18	project://Shading/roads/map_file	TextureMapFile		3.0 MB	1
19	project://Shading/generic_concrete/cleanconcrete/dirt	TextureMapFile		3.2 MB	11
20	project://city_build/main_structures/gateway	GeometryPolyfile	geometry	3.0 MB	5

Resources sorted by memory usage

You can also clear all resources that are in memory by pressing **Clear All** (3).

Resource List

The resource list (8) is where resources and their information are displayed. Here, you can sort items either by Path, Class, Tag, Size and Requester Count.



	Path	Class	Tag	Size	Requester Count
1	project://Shading/esplanade/map_file	TextureMapFile		12.9 MB	1
2	project://city_build/main_structures/domebat	GeometryPolyfile	geometry	10.6 MB	30
3	project://Shading/rocks/citytile/map_file1	TextureMapFile		8.2 MB	1
4	project://Shading/rocks/bgrocks/map_file1	TextureMapFile		7.7 MB	1
5	project://city_build/environment/skymap/map_file	TextureMapFile		6.2 MB	2
6	project://Shading/generic_metal/rustymetal/metal	TextureMapFile		5.2 MB	3
7	project://Shading/domeconcrete/map_file	TextureMapFile		5.1 MB	8
8	project://city_build/main_structures/bigdome	GeometryPolyfile	geometry	5.5 MB	5
9	project://city_build/bigtower/mainbuild_foot	GeometryPolyfile	surface_uv_maps	5.1 MB	4
10	project://Shading/generic_concrete/cleanconcrete1/dirt	TextureMapFile		4.4 MB	2
11	...ct://Shading/generic_concrete/cleanconcrete/concrete	TextureMapFile		4.0 MB	7
12	project://city_build/instances/domebat5	GeometryPolyfile	gas	4.4 MB	8
13	project://city_build/main_structures/domebat	GeometryPolyfile	gas	4.3 MB	1
14	project://city_build/bigtower/mainbuild_foot	GeometryPolyfile	surface_geometry	5.0 MB	8
15	project://city_build/instances/domebat1	GeometryPolyfile	gas	4.4 MB	3
16	project://Hidden_Sources/rocks/rock01	GeometryPolyfile	geometry	4.3 MB	3
17	project://Shading/planarconcrete/map_file	TextureMapFile		3.2 MB	2
18	project://Shading/roads/map_file	TextureMapFile		3.0 MB	1
19	project://Shading/generic_concrete/cleanconcrete/dirt	TextureMapFile		3.2 MB	11
20	project://city_build/main_structures/gateway	GeometryPolyfile	geometry	3.0 MB	5

Resources sorted by memory usage

Column	Description
Path	As Resources have no name, it displays the path of the first item declaring the resource.
Class	Object class declaring the resource.
Tag	Optional description of the resource.
Size	Memory size used by the resource.
Requester Count	Number of requester using this resource. The higher the number, the higher the de-duplication.

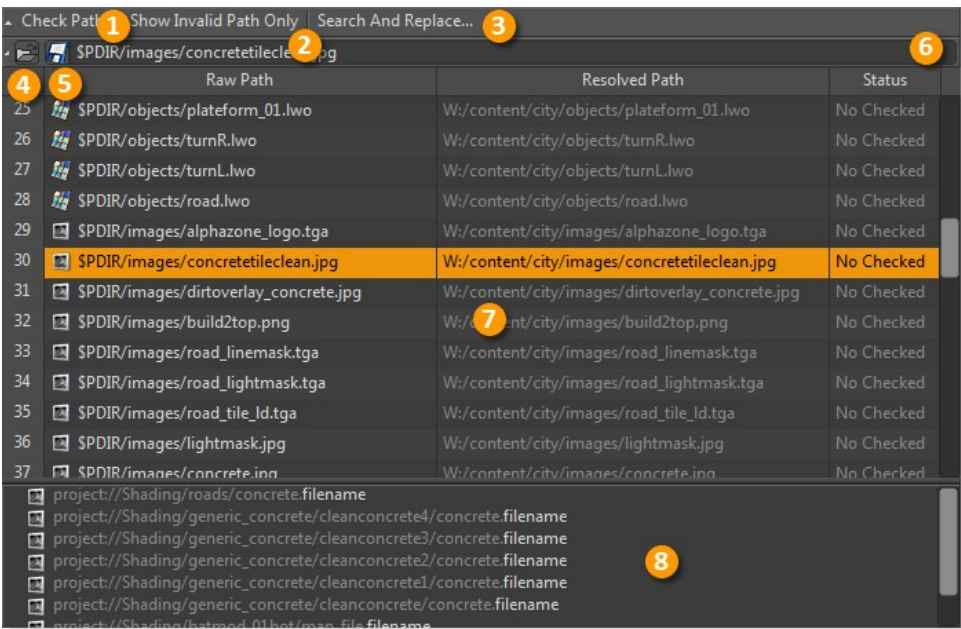
When you select one or multiple resources, the Resource View selects all project items sharing those resources.

Using the Path Manager

As an assembly tool, Clarisse can work with many external assets at the same time within a single project. External assets are always defined by a file path. Clarisse provides with the Path Manager a widget dedicated to path management.

Overview

The Path Manager is divided in 4 sections: the toolbar, the edit field, the path list and the attribute list.



The Path Manager

(1) Check Paths (2) Show Invalid Path Only (3) Search And Replace Tool (4) Path Folder Replacement (5) File Browser (6) Path Edit Field (7) Path List (8) Object Attribute List

Path List

The Path List (7) is split into 3 columns. The first column displays raw paths, the second one displays resolved paths and the last one displays the state of the file. You can sort paths in ascending or descending order.

Column	Description
Raw Path	Displays the path as it is stored in the attribute. If the path has variables, they will be displayed
Resolved Path	Displays the path as it is interpreted by Clarisse (variables or expressions are expanded and evaluated).
Status	Displays the path status if it is valid or invalid.

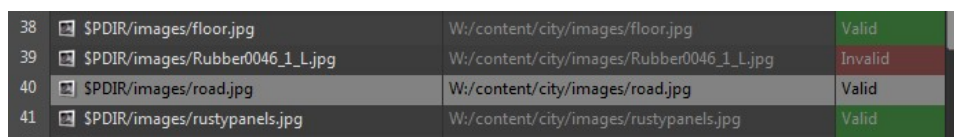
Changing Selected Path

To change the path of the selection you must use the path edit field (6). There are 3 ways to change the path of your selection:

- typing the new path in the text field.
- Browsing to a new folder to replace the path prior to the file name by pressing the folder icon (4).
- Browsing to the new file by pressing the disk icon (5).

Checking Paths

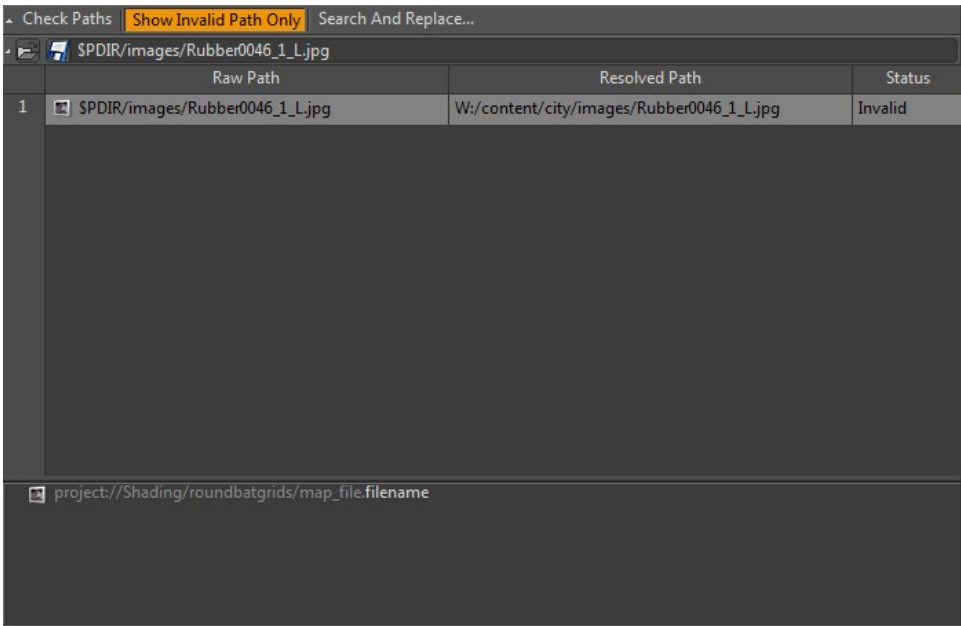
You can check paths for a selection or for all items (when nothing is selected) by pressing **Check Paths** (1). Basically, the path manager will check if the specified paths exist. Once completed, the last column of the path list (7) will switch from *Unknown* to either *Valid* or *Invalid*.



38	SPDIR/images/floor.jpg	W:/content/city/images/floor.jpg	Valid
39	SPDIR/images/Rubber0046_1_L.jpg	W:/content/city/images/Rubber0046_1_L.jpg	Invalid
40	SPDIR/images/road.jpg	W:/content/city/images/road.jpg	Valid
41	SPDIR/images/rustypanels.jpg	W:/content/city/images/rustypanels.jpg	Valid

Path Manager displaying valid and invalid path

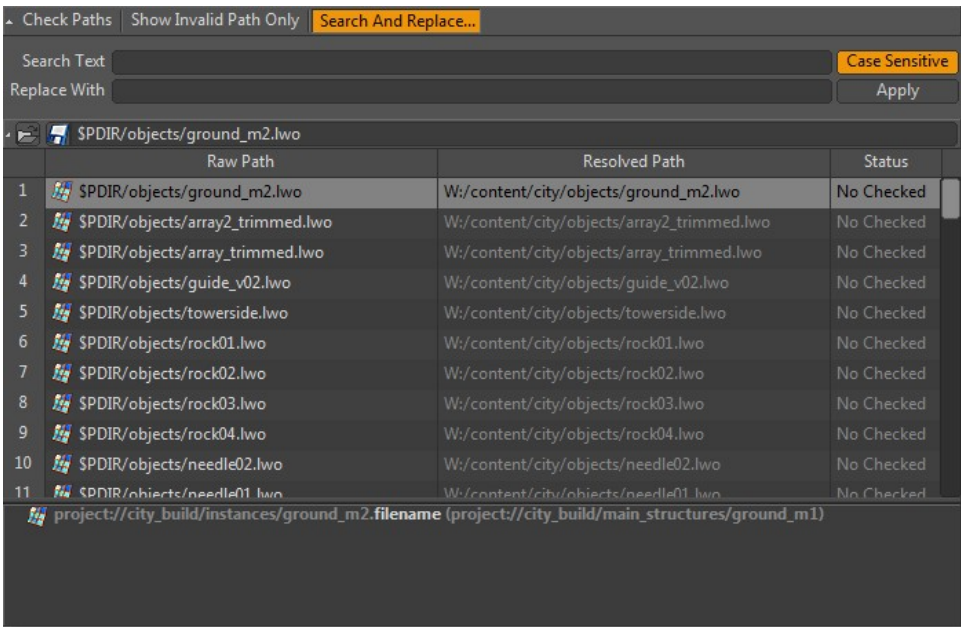
You can also filter the path list to display only path flagged as invalid. To do so, press **Show Invalid Path Only** (2).



Path Manager in Show Invalid Path Only

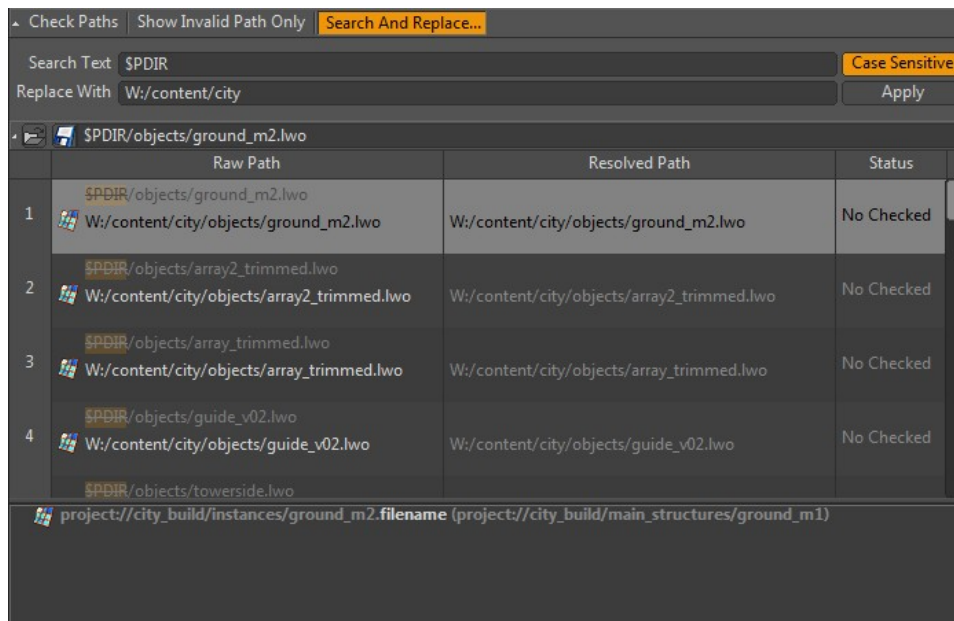
Search and Replace

The Path Manager provides a handy search and replace tool allowing you to replace paths in batch. To bring the search and replace tool, press **Search and Replace...** (3)



Search and Replace tool of the Path Manager.

Type what you want to replace in the Search Text field and the replacement in Replace With text field. While you are typing your text, a preview of the resulting replacement is updated in real time. Hit **Apply** when you are done or hit again **Search And Replace** to cancel. By default the search is case sensitive. You can disable the case sensitivity by pressing **Case Sensitive**.

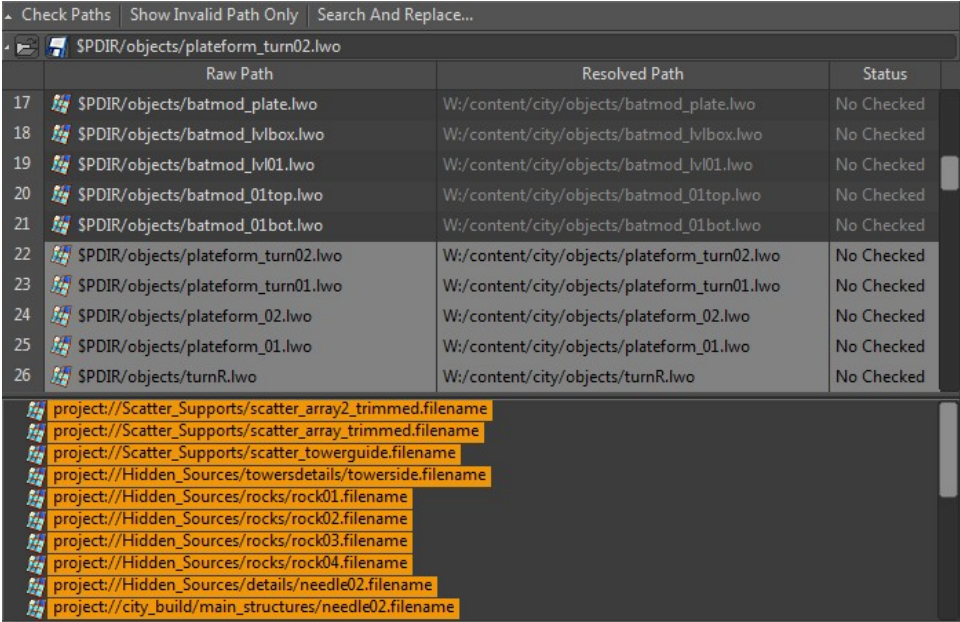


Replacing \$DIR by a custom path

You can also press **Check Paths** before applying your changes. This is very useful if you want to check file existence prior applying your replacement.

Object Attribute List

The Object Attribute List (8) is located at the bottom of the path manager. It displays all objects and attributes that reference selected paths. If you select any of the items located in the object attribute list, they will be selected in the application. Nothing prevents you from modifying their attributes in the Attribute Editor.



The screenshot shows the Path Manager window with the following tabs: Check Paths, Show Invalid Path Only, and Search And Replace... The window title is 'SPDIR/objects/plateform_turn02.lwo'. The table below lists the paths and their resolved paths.

	Raw Path	Resolved Path	Status
17	SPDIR/objects/batmod_plate.lwo	W:/content/city/objects/batmod_plate.lwo	No Checked
18	SPDIR/objects/batmod_lvbox.lwo	W:/content/city/objects/batmod_lvbox.lwo	No Checked
19	SPDIR/objects/batmod_lv01.lwo	W:/content/city/objects/batmod_lv01.lwo	No Checked
20	SPDIR/objects/batmod_01top.lwo	W:/content/city/objects/batmod_01top.lwo	No Checked
21	SPDIR/objects/batmod_01bot.lwo	W:/content/city/objects/batmod_01bot.lwo	No Checked
22	SPDIR/objects/plateform_turn02.lwo	W:/content/city/objects/plateform_turn02.lwo	No Checked
23	SPDIR/objects/plateform_turn01.lwo	W:/content/city/objects/plateform_turn01.lwo	No Checked
24	SPDIR/objects/plateform_02.lwo	W:/content/city/objects/plateform_02.lwo	No Checked
25	SPDIR/objects/plateform_01.lwo	W:/content/city/objects/plateform_01.lwo	No Checked
26	SPDIR/objects/turnR.lwo	W:/content/city/objects/turnR.lwo	No Checked
	project://Scatter_Supports/scatter_array2_trimmed.filename		
	project://Scatter_Supports/scatter_array_trimmed.filename		
	project://Scatter_Supports/scatter_towerguide.filename		
	project://Hidden_Sources/towersdetails/towerside.filename		
	project://Hidden_Sources/rocks/rock01.filename		
	project://Hidden_Sources/rocks/rock02.filename		
	project://Hidden_Sources/rocks/rock03.filename		
	project://Hidden_Sources/rocks/rock04.filename		
	project://Hidden_Sources/details/needle02.filename		
	project://city_build/main_structures/needle02.filename		

Object attribute list displaying the full object path with its file attribute name

Working with Variables

Clarisse supports custom, built-in and system variables. Variables can be used in most attribute text fields and they are very useful if you wish to constraint attribute path resolution according to a global variable for example. They also can be used to load sequence of geometries. Please refer to Using Time Variable section for more information on this subject. Finally, variables are managed using the *Variable Editor* widget.

Basic Usage

To use a variable in an attribute value, simply insert its name in the string of the text field. The expression system uses separator characters to read the variable names from the expression. Any character that is not a separator is considered valid to be used to name a variable. The separator characters are listed below:

```
/ ? , . < > ' " ; : [ ] { } \ | = ( ) * & ^ % $ # @ ! ` ~
```

Variable Naming Limitation

Variables have the following naming limitations:

- they can't start with a digit
- they support only underscore and alphanumeric characters

For example `FIRST_JOB` is a valid variable name whereas `1ST_JOB` is invalid.

Variable Types

Clarisse provides 3 kinds of variables:

- System (defined by the shell)
- Built-in (defined by Clarisse)
- Custom (user-defined)

In the event of a naming collision, for example a custom variable sharing the same name with a system one, variable lookup is performed like this. First Clarisse checks if the custom variable exists, if not then it looks for a built-in then for the system one. This allows users to easily override system or built-in variables.

System

System variables are read-only variables imported upon Clarisse launch. System variables are variables defined in the shell environment or in the `clarisse.env` file. Please note Clarisse skips variables with invalid names.

Built-in

Built-in variables are read-only variables automatically declared by Clarisse. These variables can't be deleted and their values are automatically set by Clarisse during execution. Typical built-in variables are `F`, `FPS`, `T`, `PDIR`. For a complete list of built-in variables please check the *Variable Editor* widget.

Built-in Variable	Description
F	Clarisse current frame.
T	Clarisse current time.
PDIR	Current project directory.
PNAME	Current project name. (with no extension)
FPS	Frame per second value. This value can be changed in the Preferences Panel.
CTEMP	Clarisse temp directory.
CDIR	Clarisse content directory.

Using Time Variables

Clarisse provides 3 time-dependent built-in variables that can be used to write basic expressions in filename type attribute fields.

Variable	Description
\$F	Returns the current frame
\$T	Returns the current time (in seconds)
\$FPS	Returns the current frames per second

For example you can use `F` variable to load a sequence of Wavefront OBJ file. Just set the filename field of your polyfile to:

```
/data/seq_0089/sh_009/geometry/collapse_$4F.obj
```

When time changes, the filename will be automatically expanded with current frame number formatted with a padding of 4.

Padding

If you want to pad a variable value, you can insert a number between the \$ character and the name of the variable. The number represents the padding of the variable value. Padding applies only if the length of the variable value string is smaller than the given padding. By default, padding value is 0.

Example	Expanded Version with \$F = 25
c:\my_image_\$F.exr	c:\my_image_25.exr
c:\my_images_\$4F.exr	c:\my_images_0025.exr
c:\my_image_\$F\$F.exr	c:\my_image_2525.exr
c:\my_image_\$F_final.exr	c:\my_image_25_final.exr
c:\my_images\frame_\$Fimage.exr	c:\my_images\frame_25image.exr

Custom

Custom variables are user-defined variables that are saved in the project. When reloading a project, variables and their values are automatically recreated.

Variables and Evaluation

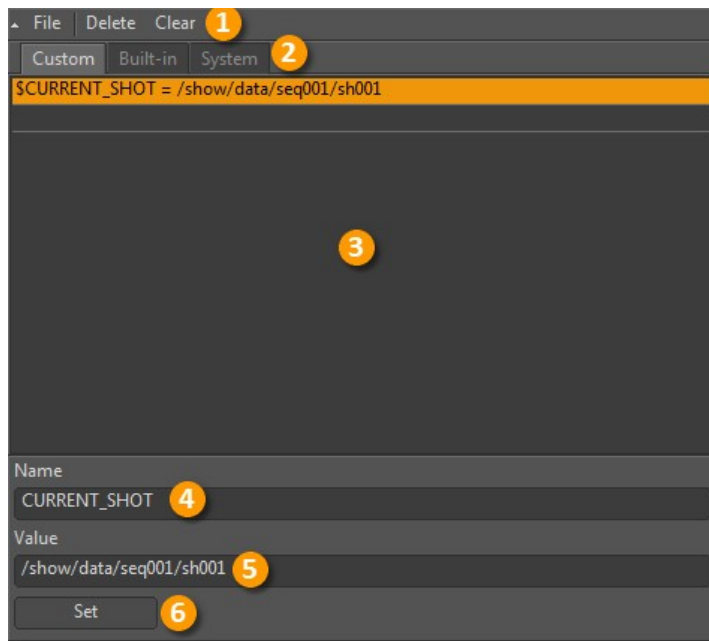
Variable values are tracked in real-time. If the value of a variable is changed somehow during an evaluation, and the running evaluation somehow depends on the variable value, the evaluation will be automatically restarted.

For example, let's say you've defined the directory of your texture maps in a variable: MY_MAPS_PATH. Now, during a rendering, you decide to change MY_MAPS_PATH value. All textures that were using this variable get notified so they can update their new path. Then rendering gets interrupted and restarts automatically.

Using the Variable Editor

The Variable Editor is a simple dedicated widget that allows you to create, inspect or manage variables.

Overview



The Variable Editor

(1) Menu Bar (2) Variable Tabs (3) Variable List (4) Variable Name (5) Variable Value (6) Set or Add a variable.

Loading

You can load a set of predefined custom variables by clicking on **File > Load...** in the menu bar section (1). When you load a variable set, new variables are appended to the current ones. Existing variables are set to the new value defined in the file. If you wish to replace the current set of variables, press **Clear** before loading your variable set.

Saving

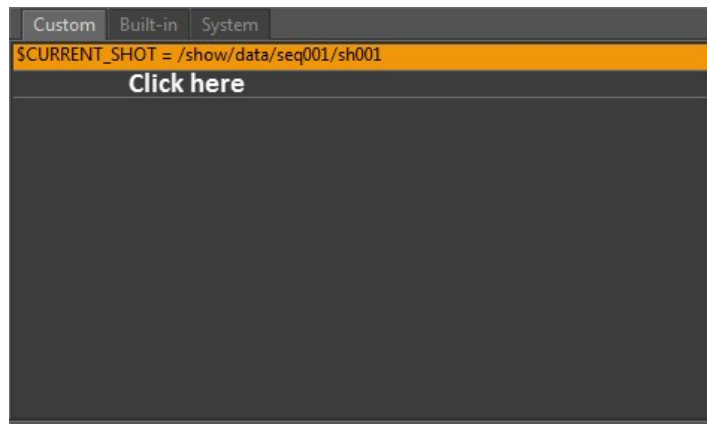
You can save your set of custom variables in a file by clicking on **File > Save...** in the menu bar section (1). The file format is pretty straight forward, there is one variable per line and the syntax is defined as:

```
VARIABLE_NAME1 "a value"  
VARIABLE_NAME2 "another value"  
VARIABLE_NAME3 "a third value"
```

Adding

To add a new variable, make sure to select the empty area in the variable list (3)

. To select the empty area please click here:



Then type the name of your variable in the variable name field (4) and its value in the value field (5) and press **Add** button to create your new variable.

Setting

You can set the value or rename an existing variable by selecting a variable in the Variable list (3) and changing either its name (4) and/or its value (5). Press **Set** button (6) to set your variable.

Deleting

To delete a variable, select the variable in the Variable list (3) and press **Delete** in the menu bar (1). This is not a undo-able action, once a variable is deleted, you will have to recreate it.

Clearing

To delete all custom variables, press **Clear** in the menu bar (1). This is not a undo-able action, once variables are deleted, you will have to recreate all of them.

Understanding Clarisse Projects

Clarisse projects are based on an easy to read ASCII object description format. Each element created in Clarisse is serialized in a file and the sum of this serialization becomes the project file.

There are a few key concepts in Clarisse project: entries, comments, and directives. Please, note the following documentation is for informational purpose only. It does not replace the SDK documentation.

Serialization Basics

Entries

In Clarisse, an entry is a name separated by a space and followed by one or multiple values or a group. Here is an example of entries:

```
parent "project://canyon/renderCam"  
aim_distance 33.38  
scale 1 1 1
```

Values

A value can be of any of following types: numeric, binary (yes/no), string or group. For example:

```
parent "project://canyon/renderCam"
```

Entries may also have an array of values such as:

```
scale 1 1 1
```

Groups

A value can be of group type. This is actually how entries declare a group of entries. A group will always start and end by curly brackets {}. For example:

```
CameraPerspective {name "renderCam"}
```

A group can be empty and in that case you would have:

```
CameraPerspective {}
```

Comments

A comment should always start with a double slash `//` or a hash `#` character, for example:

```
// this is a comment  
# this is another comment
```

Currently, Clarisse only supports one liner comments. Please also note, comments are ignored during project loading. If you save a project that has comments in it, they'll be lost.

Directives

Project directives are powerful tools for assembling scenes without even the need of opening Clarisse. This way, you can assemble shots from any other application and send them directly to CNODE for rendering. Directives are keywords and function calls placed inside project files. They are executed/evaluated during project loading.

Important

Directives are not saved back in projects. If you open a project having directives and you want to overwrite your project file, using File > Save for example, your directives will be lost and your project will be saved as an expanded project without directives.

There are 2 kinds of directives:

- pre-processor directives that are executed just before project loading starts (like pre-processor directives in C/C++).
- command directives that are executed just after project loading. They acts like post-process operations in your project when loaded in Clarisse.

You can find a full working example inside directive directory in Clarisse contents.

Path Resolution

Some directives, such as `#include` preprocess directive or `import_abc` command directive, require a file path as argument. When loading a project, Clarisse resolves file paths in the following order:

1. relatively to the path of the project being loaded
2. relatively to the search paths defined in Clarisse or CNODE (specified using `-search_path` command line argument)
3. relatively to the execution directory of Clarisse or CNODE

Preprocessor

As its name suggests, Preprocessor directives preprocess the project file just before Clarisse starts building the project (creating objects, setting attributes...). More and more preprocessor directives will be available in the future.

#include

This directive is used to copy the content of a specified file and paste it in the exact location where the directive stands. It basically acts the same way as the C/C++ #include preprocessor directive.

```
#include "filename"
```

Unlike in C/C++, this directive only accepts string like file names and not the #include <filename> syntax.

If the path of the specified file is not resolved, the loading of the project stops and Clarisse returns an error. This can happen when the input file is not found. Remember, preprocessor directives specifying missing files are not ignored and project loading fails.

Usage

Simplest #include usage example:

Let's imagine we have a project file named `simple_include.project` which contains:

```
#include "my_included_project.project"
```

Loading this file will give the exact same result as loading directly `my_included_project` file.

Another example would be to split your project into elements:

```
#include "my_image.inc"  
#include "my_renderer.inc"  
#include "my_shaders.inc"  
...  
#include "my_geometries.inc"
```

During project loading, Clarisse assembles all the project's elements into a single one. Please note that the file extension ".inc" is not mandatory. Any extension (even no extension at all) is valid for file inclusion and you can name your project's elements the way you want.

Command

Command directives are executed just after project loading and can be seen as postprocess operations that are applied to your project. More and more command directives will be available in the future. Command directives use a function call syntax as following:

```
directive_name(arg1, arg2, ... ,argN)
```

Context Sensitivity

Directives are context sensitive. This means that the context in which they are called may have an impact on the directive result as the holding context is passed to the directive. For example:

```
#Isotropix_Clarisse_Project_Version 0.91  
directive(arguments)
```

Execute the dummy directive directive in the root context of the project (project:/). However, if we had this:

```
#Isotropix_Clarisse_Project_Version 0.91  
Context "my_context" {  
    directive(arguments)  
}
```

Execute the dummy directive directive inside the context "my_context" (project://my_context)

Execution Order

Command directives are executed following their call order in the project file. For example, if we have:

```
directive1(arguments)  
directive2(arguments)  
directive3(arguments)
```

Clarisse executes directive1 then directive2 and finally directive3.

Argument Types

Command directives may optionally make use of arguments. Like in C/C++, directive arguments have types.

Type	Syntax	Examples
string	C style character string	"my_value"
boolean	yes or no	yes no
integer	a positive or negative number.	10 -10
floating point	a number with floating point precision.	3.14 -3.14
any	any supported types.	any of the previous types.

Their syntax uses the same syntax as the values of the properties written in project files. The *any* type means that any types can be used as an argument.

import_abc

This directive import an Alembic archive inside the context from which the directive is called. It acts the same way using **File > Import > Scene**.

```
import_abc(string filename)
```

Where filename is the filename of the Alembic file to import.

Note

The archive file path resolution follows the same rule as directive path resolution.

Usage

The following example will import the Alembic file my_assets.abc inside the context "my_scene".

```
#Isotropix_Clarisse_Project_Version 0.91  
  
Context "my_scene" {  
    import_abc("my_assets.abc")  
}
```

set_value

This directive sets the value of an object's attribute.

```
set_value(string attribute_value_path, any new_value)
```

Where:

- `attribute_value_path` is the relative path to the attribute which value will be modified. Please refer to Attribute Path section for more information.
- `new_value` is the new value that will be assigned to the specified attribute.

Depending on the attribute type specified in the Technical tab of the Reference chapter in Clarisse User Guide, you must type your value argument according to the following rules:

Attribute Type	Argument	Example	Notes
bool	boolean	yes	
long	integer	-10	
double	floating	3.14	
string	string	"my_value"	used to represent a name or a file path
reference	string	"my_object"	relative path to the object from the context where the object owning the attribute lies. Can also be an absolute path to the object.
object	N/A		not supported
curve	N/A		not supported

Usage

The following example will set the parent attribute of locator1 to locator2.

```
#Isotropix_Clarisse_Project_Version 0.91
Context "my_scene" {
    Locator {
        name "locator1"
    }
    Locator {
        name "locator2"
    }
    set_value("locator1.parent", "locator2")
}
```

Copying and Pasting

When you copy one or multiple items in Clarisse, Clarisse basically serializes its current selection to the clipboard. On the contrary, when you paste in Clarisse, Clarisse looks in the clipboard and build items from the serialization if found. This is how it gets easy to copy and paste between different Clarisse processes.

If you wish to see the serialization of your clipboard coming from Clarisse, copy your selection and paste it in any text editor. For example:

```
#Isotropix_Clarisse_Clipboard_Serialization 0.91
CameraPerspective {
  name "renderCam"
  #version 0.9
  copy_from "project://canyon/renderCam"
  translate 0.0 0.0 0.0
  rotate 0.0 0.0 0.0
  scale 1 1 1
  parent ""
  constraints ".mot_player"
  aim_distance 33.3820041068195
  field_of_view 50.9822839085102
  embedded_objects {
    ConstraintMotPlayer {
      name "mot_player"
      #version 0.9
      copy_from "project://canyon/renderCam.mot_player"
      active yes
      filename "$PDIR/motion/mn_camera.mot"
    }
  }
}
```

When Clarisse looks in the clipboard for an available serialization, it first check for `#Isotropix_Clarisse_Clipboard_Serialization`. This line is mandatory. If it's not present then Clarisse doesn't paste anything.

Using CNode/CRender

CNode and CRender are standalone command line applications. They can be used to render Clarisse images. With a generic external render controller application, it becomes really easy to render Clarisse's images on a render farm, using either CNode or CRender.

CNode also provides an interactive mode (console like interface). The interactive mode allows you to use Clarisse without the user interface using python commands.

Differences between CNode and CRender

The main difference between the two applications is that CRender can only read binary render archives whereas CNode can directly read Clarisse projects. Basically CNode is Clarisse as a command line application and CRender is just the renderer.

Feature	CNode	CRender
Licensing	Floating	Floating/Node Locked
Read Render Archives	•	•
Read Clarisse Projects	•	
Resolve Referencing	•	
Evaluate Directives	•	
Command Port	•	
Interactive Mode	•	
Scripting	•	

Exporting a Render Archive

Render Archives can only be exported from a running Clarisse application. You can't export Render Archives from CNode or from a script executed by Clarisse command line. To export a project as a render archive, go to **File > Export > Render Archive**.

Important

Currently, Render Archives don't package all file dependencies (textures, geometries). Render Archives still reference those files. In future versions we will introduce Render Archives that will package all file dependencies.

Setup on a render farm

First, make sure each render node (with the exception of the graphics card) meets Clarisse's minimum system requirements. CRender/CNode also need a special licensing (one per node), so please make sure you have enough licenses available. Please refer to Installation and Licensing for more information.

Important

As any distributed command line application, make sure each render node is able to resolve network paths. Otherwise, CNode or CRender will unlikely be able to load assets or save renders.

Installation Folder

Clarisse binaries (including CNode and CRender) don't require any special privileges to run and don't use any system registry key. In other words, installing Clarisse on each render node is not necessary: all you have to do is to make sure each render node has access to the Clarisse installation folder. However, you'll have to make sure to install optional third parties if needed. Please refer to the Installing Clarisse section.

We highly recommend you to use a shared Clarisse installation folder:

- You will be sure that every render node uses the same application version.
- It simplifies the installation process when deploying new Clarisse builds.

Configuration File

By default, CNode and CRender both use Clarisse configuration file. Once again, we recommend to use a shared configuration file location so that render nodes share the same configuration file. By using the argument `-config_file` you can specify the location of the configuration file CNode/CRender will be using.

For example:

```
cnode /sq001/sh001/beauty.project -config_file /  
clarisse_installation/clarisse.cfg
```

CNode Usage

```
cnode <input_file> [-image <image_path>] [-start_frame  
<start_frame>] [-end_frame <end_frame>] [-frame_step <frame_step>]  
[-output <output_file>] [-format <output_file_format>]
```

Use -help argument to display command line help.

CRender Usage

```
crender <input_file> [-image <image_path>] [-start_frame  
<start_frame>] [-end_frame <end_frame>] [-frame_step <frame_step>]  
[-output <output_file>] [-format <output_file_format>]
```

Use -help argument to display command line help.

Arguments

Name	Description
input_file	Path to the render archive (CRender/CNode) or Clarisse project file (CNode only) to be loaded.

Optional Arguments

Info

-info

Output in the console a XML summary of the project. The output XML describes file dependencies, images, layers and project settings from an input project. This argument is extremely useful if you wish to integrate Clarisse in an asset management system or a render farm controller. With the XML output you have a quick and parse-easy description of any Clarisse project or Render Archive. Here is the resulting XML from an input project file:

```

<project>
  <filename>/opt/isotropix/Clarisse-1.5/content/directives/
my_shot.project</filename>
  <version>1.0</version>
  <dependencies>
    <file>/opt/isotropix/Clarisse-1.5/content/directives/
my_assets.abc</file>
    <file>/opt/isotropix/Clarisse-1.5/content/directives/textures/
DIFF_texture_HD.tga</file>
  </dependencies>
  <image>
    <name>project://scene/image</name>
    <output></output>
    <format>exr16</format>
    <save_to_disk>0</save_to_disk>
    <start_frame>0</start_frame>
    <end_frame>50</end_frame>
    <frame_step>1</frame_step>
    <gamma>0</gamma>
    <layer>
      <name>project://scene/image.background</name>
      <visible>1</visible>
      <output></output>
      <format>exr16</format>
      <save_to_disk>0</save_to_disk>
      <start_frame>0</start_frame>
      <end_frame>50</end_frame>
      <frame_step>1</frame_step>
      <gamma>0</gamma>
    </layer>
  </image>
  <settings>
    <fps>24</fps>
    <motion_blur_sample>3</motion_blur_sample>
    <motion_blur_direction>centered</motion_blur_direction>
    <motion_blur_length>50</motion_blur_length>
    <texture_cache>512</texture_cache>
  </settings>
</project>

```

Important

Except for search_path all other arguments are ignored when using info

Reference

Tag	Description
<project>	Defines a project or a render archive.
<filename>	Defines the input filename used to generate the xml.
<version>	Defines the clarisse xml output file version.
<dependencies>	Lists project external dependencies.
<file>	Defines a path to file. Inside <dependencies> this defines an external file dependency.
<image>	Defines a Clarisse image.

<name>	Defines, inside a block, the absolute name of the item in Clarisse project.
<output>	Defines the image output path.
<format>	Defines the saved image format.
<save_to_disk>	If 1, the image is flagged to be saved to disk, 0 otherwise.
<start_frame>	Defines the sequence start frame.
<end_frame>	Defines the sequence end frame.
<frame_step>	Defines the sequence frame step.
<gamma>	If 1 a sRGB LUT is applied prior saving the image to disk.
<layer>	Defines a Clarisse image layer. Layers are always embedded within an <image> block.
<visible>	Defines layer visibility. If 0, the layer is hidden and is unlikely evaluated. Note: a hidden layer still can be explicitly evaluated by the using of override arguments.
<settings>	Defines project settings.
<fps>	Sets the animation frame per second.
<motion_blur_sample>	Sets the number of time samples used for motion blur. Please refer to Motion Blur Sample section.
<motion_blur_direction>	Sets the direction of the motion blur. Please refer to Motion Blur Direction section.
<motion_blur_length>	Sets the length of the motion blur. Please refer to Motion Blur Length section.
<texture_cache>	Sets the size on the texture cache in MB (Megabytes)

Interactive*

-interactive

Start CNode in interactive mode. For more information please refer to Using the Interactive Mode section.

Important

Except for search_path all other arguments are ignored when using interactive

*** This feature is only available for CNode.**

Script*

-script *script_path*

Execute the specified python script.

*** This feature is only available for CNode.**

Search Path

`-search_path ...`

Specify an ordered list of path used when looking for includes.

Print Expanded*

`-print_expanded`

Output resulting input project after evaluating all its directives. Evaluating directives may be slow specially if using import directives. This flag is extremely useful to output a cached version of a project containing many directives.

Important

Except for `search_path` all other arguments are ignored when using `print_expanded`

*** This feature is only available for CNode.**

Frame

`-frame first_frame [last_frame] [frame_step]`

Specify globally which frames have to be rendered. Note `last_frame` and `frame_step` are optional values. If you have multiple images with different frame ranges in your project, CNODE will only render what's needed. For example, in your project named `my_project.project`, you have two images (`image1` and `image2`) ranging respectively `[10, 50]` and `[80, 90]`. Now, let's say you've specified the following command line arguments:

```
cnode my_project.project -frame 1 100
```

In this case, CNODE only renders the `image1` as an image sequence starting from frame 10 to 50 and `image2` starting from 80 to 90.

Now, let's see what happens if you've specified the following command line arguments:

```
cnode my_project.project -frame 50 60
```

In this case, CNODE only renders `image1` at frame 50.

Threads

`-threads number`

Specify the number of threads used by CNode/CRender. By default, it uses the number of threads specified in the configuration file (defaulting to *All Available Cores*)

Configuration File

`-config_file path`

Specify the configuration file used by CNode/CRender. By default they both look for `clarisse.cfg` in the user home directory.

Module Path

`-module_path path ...`

Specify the path of Clarisse's module. This argument is really helpful if you wish to specify custom or third party modules. **This argument supports an argument list.** Path order defines look-up priority.

License Server

`-license_server server:port`

Specify the ILISE license server network location.

For example: `ILISE-SERVER:40500`

Stats

`-stats`

Display complete project and memory usage statistics after each render.


```

00:00:01      61/65MB Project statistics:
                        Memory
                        -----
                        application = 26 MB
                        scene       = 2 MB
                        resources   = 481 KB
(...)
                        memory size      = 481 KB
                        geometries       = 0 B
                        accel structures = 277 KB
                        misc             = 204 KB
                        -----
                        resources count   = 199
                        geometry          = 0
                        accel structures = 1
                        misc              = 198

```

Verbose

-verbose

Display detailed information about what the application is doing.

```

00:00:08      408/408MB Building mesh object 'tunnel2' in 1.11 s.
00:00:08      408/408MB Building smoothed geometry of object
'project://canyon/envset/ground' in 0.004 s.
00:00:08      410/410MB Building surface topology of object
'project://canyon/spaceships/spaceship_R' in 0.01 s.
00:00:08      411/411MB Building surface geometry of object
'project://canyon/spaceships/spaceship_R' in 0.016 s.
00:00:08      412/412MB Building surface uv maps of object
'project://canyon/spaceships/spaceship_R' in 0.035 s.
00:00:08      415/415MB Building tessellation topology of object
'project://canyon/spaceships/spaceship_R' in 0.003 s.
00:00:08      417/417MB WARNING: Building mesh object
'point_trimmed': the triangulation has detected 27257 degenerated
triangles

```

Override Arguments

Image

-image *itempath* ...

Specify Clarisse's project item path to the image(s) and/or layer(s) to be rendered. **This argument supports an argument list.**

```
-image f40/myimage will render the image myimage
-image f40/myimage.layer_3d will render the layer myimage.layer_3d
```

You can also specify a list of images and/or layers to be rendered

```
-image f40/myimage golf/myotherimage tower/mythirdimage.layer_3d will
both render myimage and myotherimage images and
my_thirdimage.layer_3d
```

Output

```
-output filepath ...
```

Specify images/layers output path. **This argument supports an argument list.** Please note extensions are automatically appended to output filenames according to the chosen image file format. You also can specify image output frame number padding by using the # character. By default the padding is set to five digits (#####).

Output Value	Result
c:\folder\my_image_	c:\folder\my_image_00002
c:\folder\my_image_##	c:\folder\my_image_002
c:\folder\my_image.##	c:\folder\my_image.0002
c:\folder\####_my_image	c:\folder\0002_my_image
c:\folder\my_image_#	c:\folder\my_image_2

Format

```
-format type ...
```

Specify or override images/layers output file format. **This argument supports an argument list.** By default, images are saved in exr16 format.

Format type values	Description
exr16	Open EXR 16 bit (half float) per channel file format
exr32	Open EXR 32 bit (float) per channel file format
jpg	Jpeg file format
bmp	Windows Bitmap file format

tga	Targa true vision file format
png8	PNG 8 bit per channel file format
png16	PNG 16 bit per channel file format
tiff8	TIFF 8 bit per channel file format
tiff16	TIFF 16 bit per channel file format
tiff32	TIFF 32 bit per channel file format

Start Frame

`-start_frame number...`

Override the first frame to be rendered for an image/layer. **This argument supports an argument list.**

End Frame

`-end_frame number...`

Override the last frame to be rendered for an image/layer. **This argument supports an argument list.**

Frame Step

`-frame_step number...`

Override the frame step used when rendering an image/layer. **This argument supports an argument list.**

Gamma Correction

`-gamma_correction number ...`

Specify the gamma correction applied to output image(s).

Value	Description
0	No gamma correction
1	Apply SRGB gamma correction

FPS

`-fps number`

Specify the project frames per second. Please note this also affects the motion blur.

Motion Blur Sample

`-motion_blur_sample number`

Specify the number of time samples used for motion blur. The more samples there are, the more accurate the item motion decomposition is. **Beware large numbers increase memory usage.** Minimum value is 2 (begin motion, end motion).

Motion Blur Direction

`-motion_blur_direction type`

Specify the direction of the motion blur.

Type	Shutter time
backward	[render_frame-1, render_frame]
centered	[render_frame-0.5, render_frame+0.5]
forward	[render_frame, render_frame+1]

Motion Blur Length

`-motion_blur_length value`

Specify motion blur length. The value is between 0% (no motion blur) to 100% (shutter fully opened for an entire frame)

Texture Cache

`-texture_cache size`

Specify the size of image texture cache in MB (Megabytes).

Examples

Basic

```
crender my_project.render
```

In this simplistic example, CRender will load `my_project.render` and will automatically render what was setup in the project. It basically renders all images/layers using paths and frame ranges that were specified in the project.

Specifying an Image

```
crender my_project.render -image context/my_image -start_frame 1 -  
end_frame 100 -frame_step 1 -output my_folder/my_image -format tga
```

In this example, CRender will load the archive `my_project.render` and will only render the image `context/my_image` starting from frame 1 to frame 100 and saving each frame as a TGA image inside `my_folder/my_image` folder.

Rendering Multiple Images

```
crender my_project.render -image context/my_image1 context/my_image2  
-start_frame 1 50 -end_frame 100 60 -frame_step 1 5 -output  
my_folder1/my_image1 my_folder2/my_image2 -format tga exr32
```

In this example, CRender, will load the archive `my_project.render` and will render the 2 images `context/my_image1` and `context/my_image2`.

Here, CRender will output `my_image1` from frame 1 to 100 using a frame step of 1:

```
my_folder1/my_image1.0001.tga  
my_folder1/my_image1.0002.tga  
...  
my_folder1/my_image1.0099.tga  
my_folder1/my_image1.0100.tga
```

And it will output `my_image2` from frame 50 to 60 using a frame step of 5:

```
my_folder2/my_image2.0050.exr  
my_folder2/my_image2.0055.exr  
my_folder2/my_image2.0060.exr
```

Important

A Clarisse project can contain 1 to multiple images, themselves composed of 1 to multiple layers. By default, such as in example 1, and assuming it is already set in the project, you don't have to specify anything to render an animation. **However, as soon as you specify an override such as frame range in the command line, you need to explicitly specify each image and/or layer to be rendered.** If you need to specify a global rendering frame range, please use `-frame` argument.

Using the Interactive Mode

The interactive mode gives you the possibility to use Clarisse in pure command line. You can then use Python commands and scripts to create or freely modify projects. To start the interactive mode you must launch CNode with the interactive option:

```
cnode -interactive
```

Here is what you should get:

```
CNODE 1.5 Copyright (c) 2009-2013 Isotropix SAS.  
Build #XXXX MMM DD YYYY  
Isotropix Clarisse command line.  
Clarisse licensing up and running.  
Entering Interactive Mode...  
project:/>>>
```

Important

In order to be able to script with Python, Clarisse and CNode require Python to be installed in the system. Please check Python in Clarisse section for more information.

The Prompt

While running in interactive mode, it's quite similar to using a terminal. You must input commands and press **Return** (or **Enter**). Here, input commands are any built-in or python ones. For more information on the Python API please refer to the SDK/API Documentation section found the web documentation.

CNode prompt always displays the current path in the project. By default, CNODE current context is the root of the project.

```
project:/>>>
```

Built-in Commands

cd

Very much alike a standard shell in which you can change directory, in CNode, you can navigate through your project using the cd command. It's very easy, just type `cd path_to_your_context`. For example:

```
project:/>>> cd default
project://default>>>
```

As displayed by the prompt, you are now located in the default context (project://default). Again similarly to a terminal you can go back up using either:

```
project://default>>> cd ..
project:/>>>
```

Or

```
project://default>>> cd /
project:/>>>
```

Or

```
project://default>>> cd project:/
project:/>>>
```

ls

Within a shell, in which ls command is used to list folder content, in CNode it lists the content of a context. Type `ls context_path`, where `context_path` can be empty to list the current context. For example:

```
project://canyon>>> ls
c--w 10/16/11 15:46 cliffs
c--w 10/16/11 15:46 handplaced_rocks
c--w 10/16/11 15:46 envset
c--w 10/16/11 15:46 spaceships
c--w 10/16/11 15:46 lighting
---w 10/16/11 15:46 raytracer_previz (RendererRaytracer)
---w 10/16/11 15:46 renderCam (CameraPerspective)
---w 10/16/11 15:46 previz (Image)
---w 10/16/11 15:46 raytracer_hd (RendererRaytracer)
---w 10/16/11 15:46 set_geogroup (Group)
---w 10/16/11 15:46 spaceships_geogroup (Group)
---w 10/16/11 15:46 skyenv_geogroup (Group)
---w 10/16/11 15:46 scatter_plants (SceneObjectScatterer)
---w 10/16/11 15:46 scatter_rocks (SceneObjectScatterer)
---w 10/16/11 15:46 vignette (Image)
---w 10/16/11 15:46 vignettemap (TextureMap)
---w 10/16/11 15:46 miniride_layered (Image)
---w 10/16/11 15:46 miniride (Image)
---w 10/16/11 15:46 layout_Cam (CameraPerspective)
---w 10/16/11 15:46 glowsky (Image)
```

As you can see there are a few information on the left of each listed item. Here is the syntax:

```
flags date time item_name (item class) -> source_name
```

flags list the state of the item. c defines if it's a context, p if it's private, s if it's static, w if it's modifiable. item class is the class of the Item. -> source_name is the fullname of the instance source.

Useful Python Commands

Clarisse provides a set of useful commands. They can be found in the Scripting/SDK documentation under Modules clarisse_helper. This helper allows you to quickly load/save projects, render images and so on. As all Clarisse related Python API they are under the namespace ix. For example:

```
project:/>>> ix.load_project("C:/content/canyon/miniride.project")
```

To load a project and,

```
project:/>>> ix.render_image("project://canyon/miniride")
```

to render the image.

Exiting

To exit an interactive session, press **CTRL + D**.

Stopping a Running Script

To stop a running script press **CTRL + C**

Tutorials

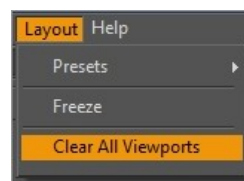
In this section we will go through tutorials to help you understand Clarisse basics to reduce learning curve.

Clarisse Basics

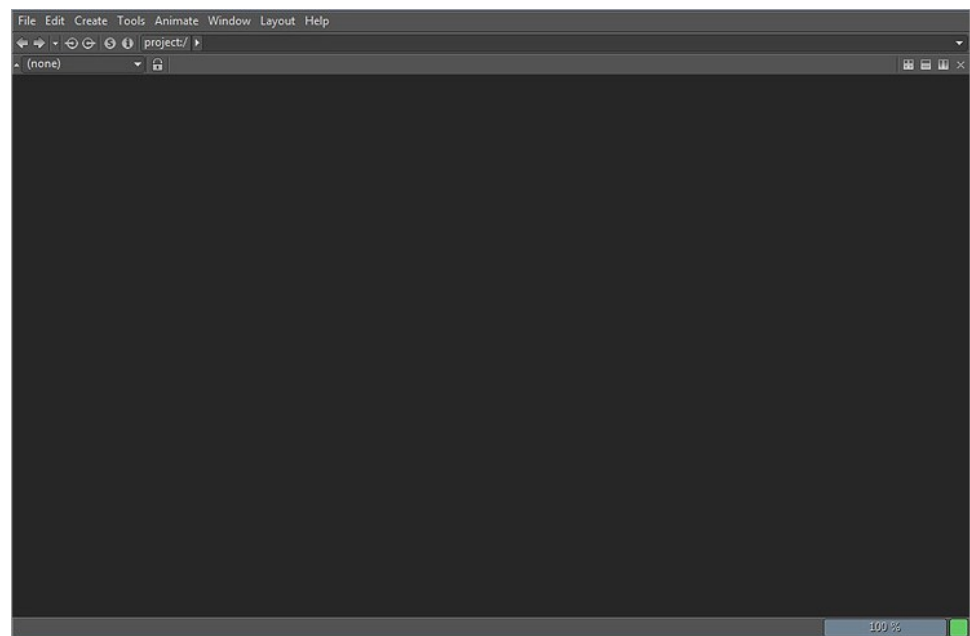
The following set of mini-tutorials go through Clarisse common features. We highly recommend you to go through most of these tutorials before diving into Clarisse by yourself.

User interface Basics

In this section, we'll discover the basics of the user interface. First, go to Layout menu and choose **Clear All Viewports**.



You should get a blank Clarisse window with only the menu and the title bar, like this:



First there's the menu bar:



File Edit Create Tools Animate Window Layout Help

This is a very classic menu, allowing you to perform most of tasks, setting preferences, modifying the user interface... if you don't know how to perform a specific task, there are good chances you will be able to do it from this menu (there is often different ways to do something in Clarisse).

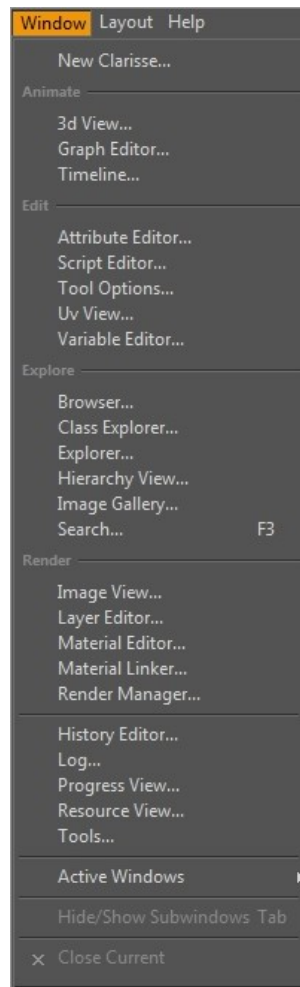
Just below the menu bar is the selection bar



← → ↺ ↻ 🔍 project/ ▶

This one is very close to the one you have in an internet or file browser : you know where you are in the project (you'll discover that Clarisse project's organization is very close to a file system, with hierarchies of folders), and displays what is currently selected. Like a browser, it has back and forward buttons, and a path field. By default, the selection and path is global to the whole application.

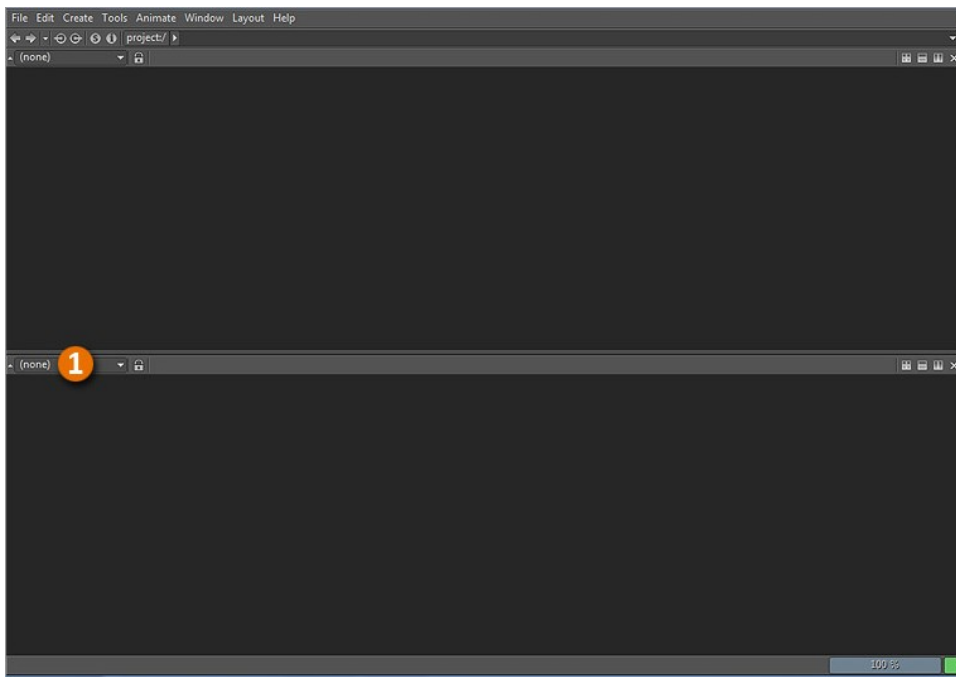
Next, the whole empty space is a viewport : it can be split into multiple viewports, each one containing a widget. A widget is an element of the Clarisse user interface, it can be an editor, a viewer, a mix of the two, anything displaying project data or allowing the user to interact with it. You can list all available widgets in the Window menu.



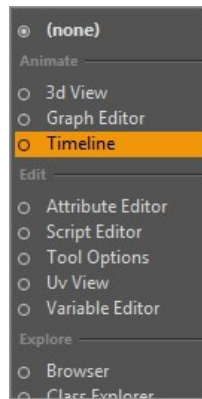
On the top right of the viewport, you have little buttons : they allow you to split the view in four or two, horizontally and vertically.



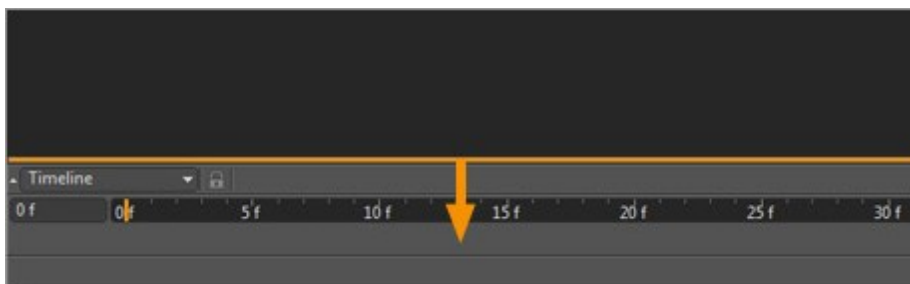
Click on the second icon (Split Horizontal) to divide the viewport into two horizontal parts. Now, click on the menu (1) of the second viewport to display the widget list.



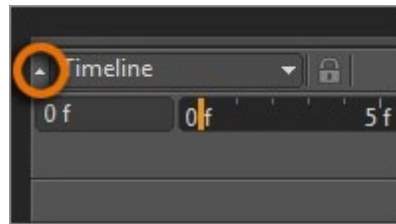
Select "Timeline" to set this viewport to the timeline widget.



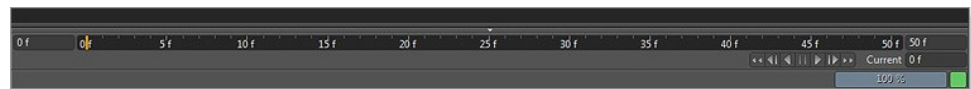
Now, drag down the viewport top bar to resize down the timeline widget, which only needs a few pixels height.



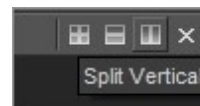
You can get a little more place, by clicking on the little arrow on the top left of the widget, to fold the viewport control.



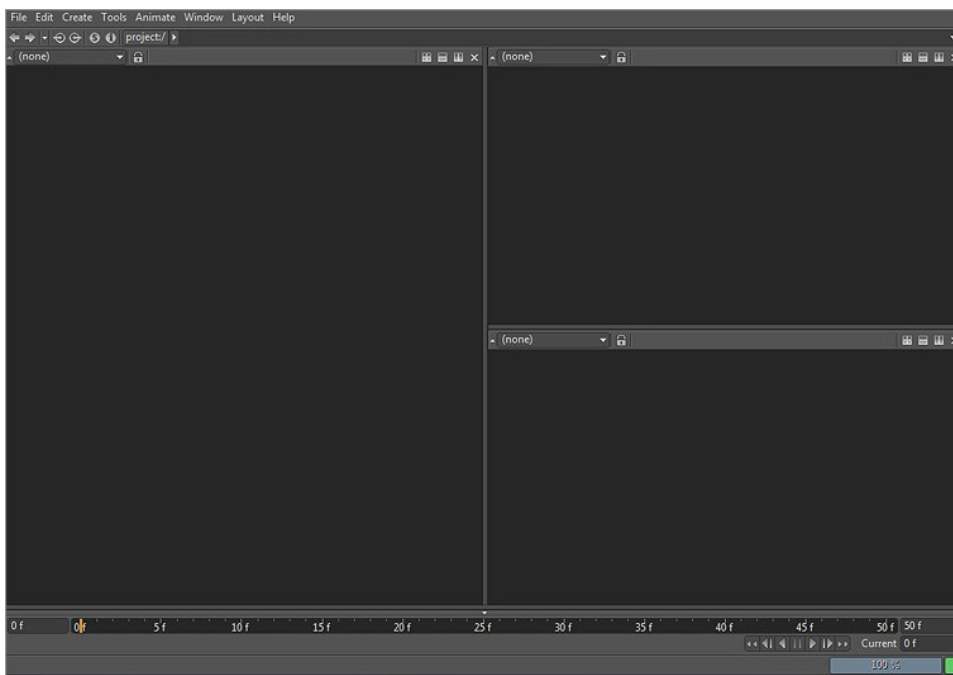
You have now a fully functional timeline at the bottom of your interface.



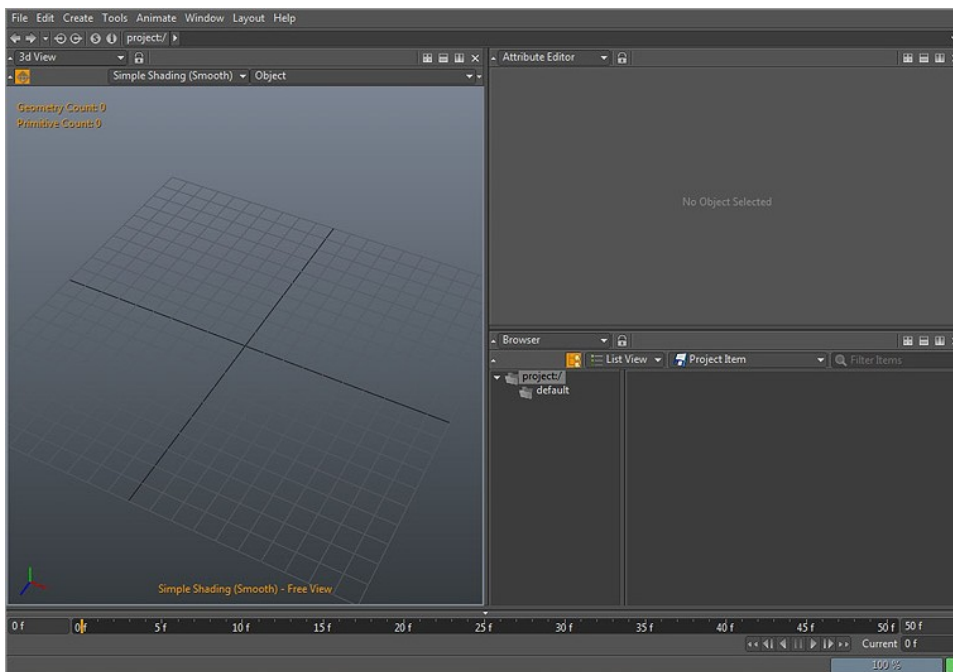
Now, click on the split vertical icon of the first, empty viewport.



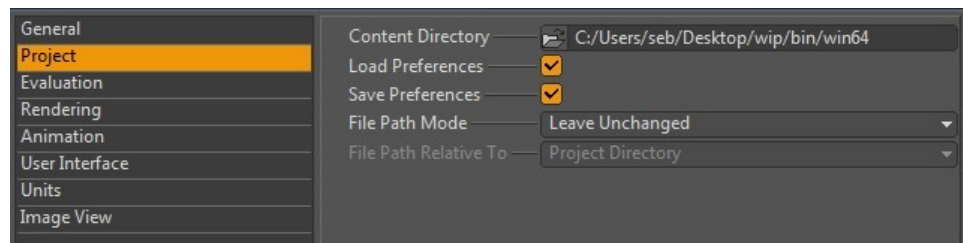
Then click again, but on the split horizontal button... Drag the horizontal divider to leave more space to the top viewport. You should get a layout similar to this:



Now, set the left viewport to *3D view*, the top right to *Attribute Editor*, and the bottom left to *Browser*. You should get this:



You can also move widgets to independent windows. Go to the **Window > Tools...** You can arrange the windows the way you want by dragging their corners. By default, the user interface layout is saved in the project, and loaded back when you reopen it. You can override this by opening the *Preferences Panel* using **Edit > Preferences...**, in the Project section, by unchecking *Load Preferences* and *Save Preferences*, depending on your needs.



To conclude this tutorial, you can also set pre-built layouts in the **Layout > Presets** menu by choosing: *Animation Workshop*, *Shading Workshop*, etc...

Combiner Basics

There are several ways to combine items.

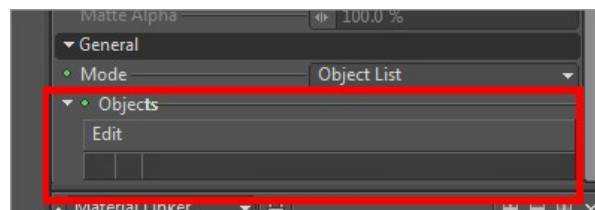
Using The Browser

In the Browser, select your geometries, right click and select **Combine** in the popup menu. This will create a new combiner including your selection. Combiners allow you to combine scatterers, combiners, geometries...

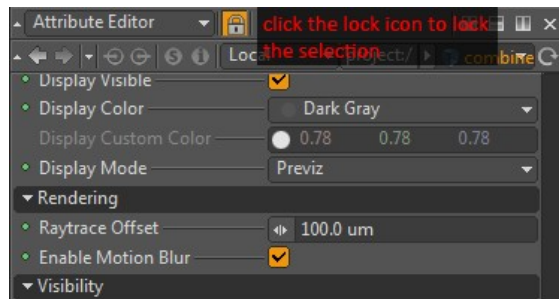
Using The Attribute Editor

First, create a new empty combiner. To create a new combiner you can either go to the main window menu bar and use **Create > Combiner**, or right click in the right Browser pane and select **New... > Combiner**.

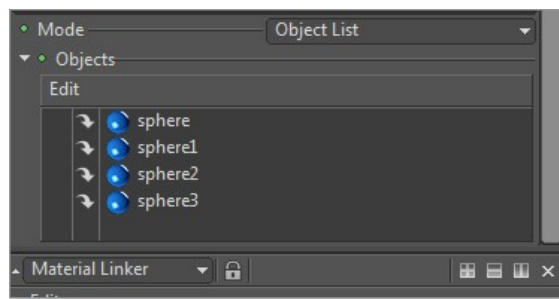
Once created, it should be displayed in the Attribute Editor. Now scroll down the Attribute Editor until you find the attribute *Objects* which should be the last one. As you can see, there's an empty list under *Objects*.



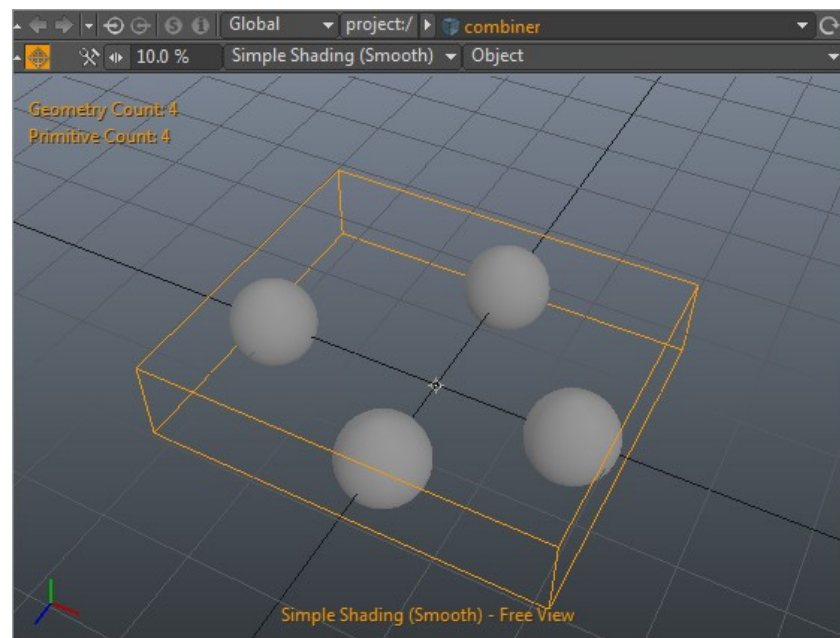
Now Lock the Attribute Editor so that it keeps displaying the combiner even if the application selection changes:



Now the attribute editor is locked to our combiner. Select your geometries in the Browser and drag and drop them in the *Objects* attribute list of our combiner.



They are now combined:



Don't forget to unlock the Attribute Editor by clicking back on the lock icon.

Note

Please note combiners are procedural objects. You can change the referenced item position anytime (they'll updated in the combiner accordingly). And of course, you can combine combiners too.

Group Basics

Groups allow to group items into groups. Groups are used to define visibility and light linking. Before proceeding to this tutorial, please load the project file `content/tutorials/projects/basics/group.project`

There are 3 ways to create groups:

- using the Browser
- using the Attribute Editor
- drag and dropping into an attribute referencing a group

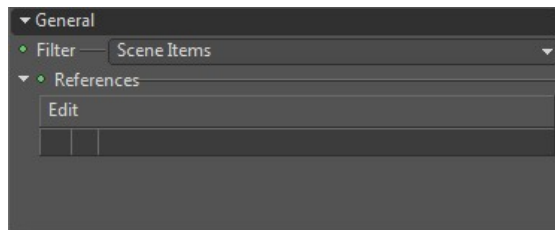
Using the Browser (Quickest Way)

Select your items, right click and select **Group** in the popup menu. This will create a new group with your selection inside.

Using the Attribute Editor

First, create a new empty group (main window menu bar: **Create > Group** or right click in the right Browser pane and select **New... > Group**)

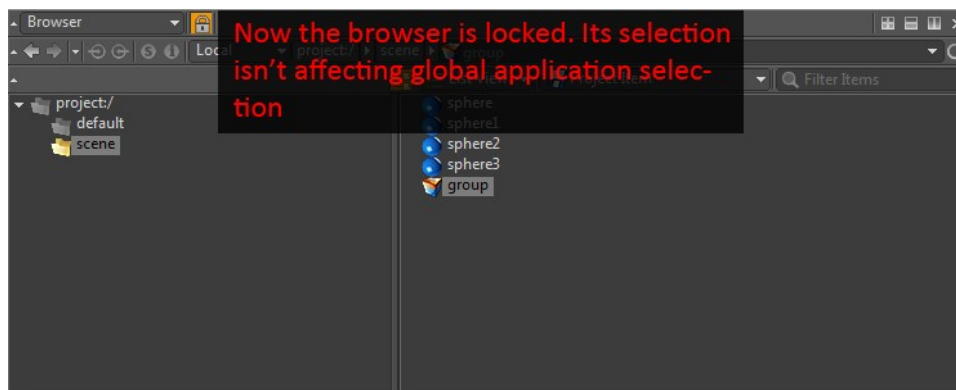
Once created, it should be displayed in the Attribute Editor. Now if you look to the Attribute Editor, you'll see 2 attributes: *Filter* and *References*.



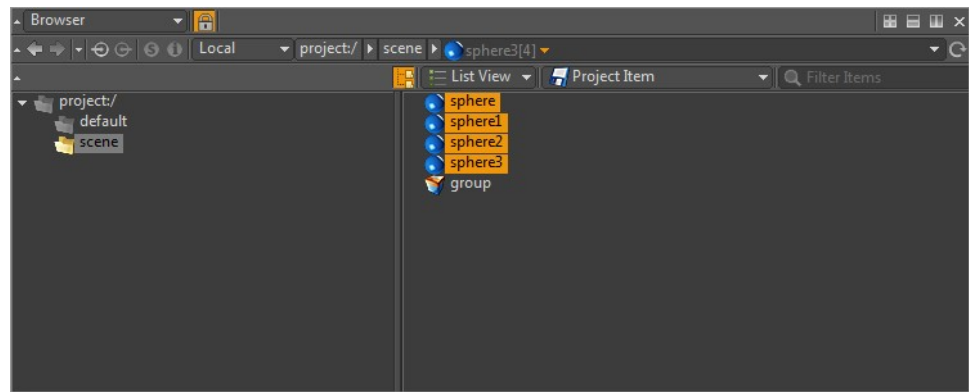
Where:

- *References* is the list of items that are included in your group.
- *Filter* is the class of items you wish to put in your group. By default it's set to Scene Items so you can only put items that inherit from scene items inside (lights, locators, geometries, combiners...).

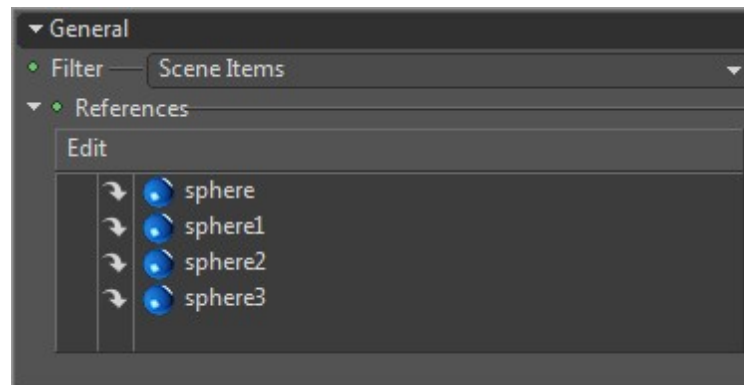
Now lock the Browser so the Attribute Editor keeps displaying the group even if the browser selection changes:



In the Browser, click on sphere hold shift and click on sphere3 to multi-select sphere, sphere1, sphere2, sphere3.

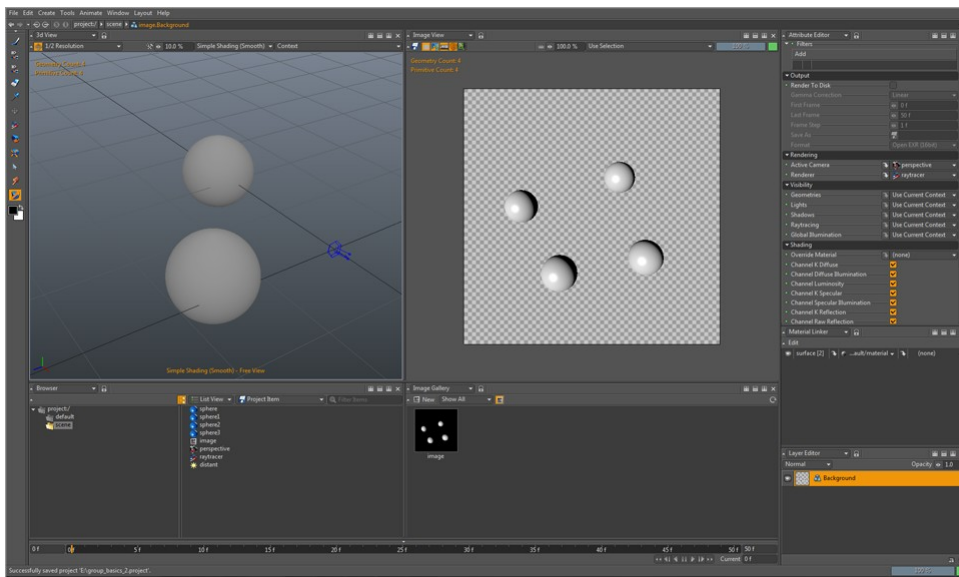


Now drag and drop your selection into *References* attribute in the Attribute Editor:

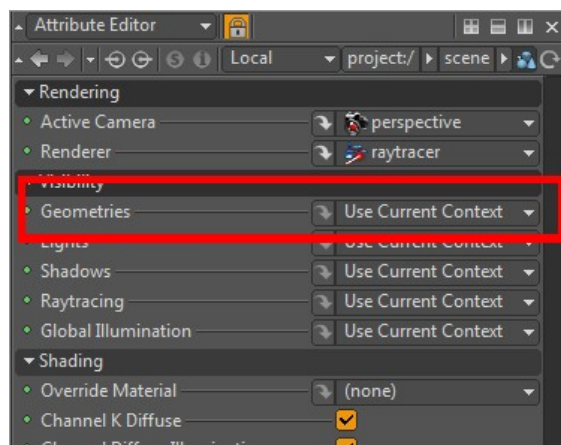


Drag and Drop into a group attribute

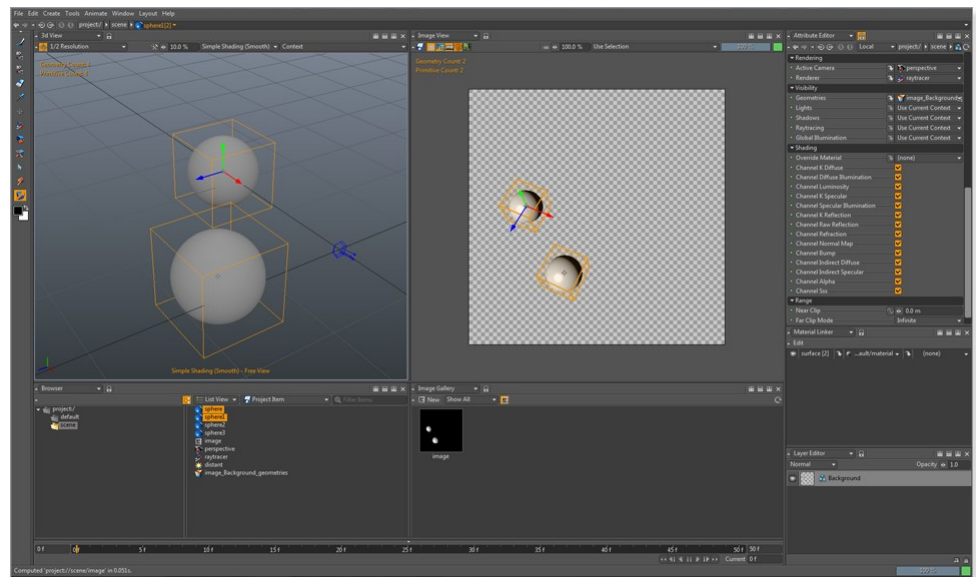
Groups can be automatically created if a selection is dragged and dropped into an attribute referencing a group. Before proceeding to this tutorial, please load the project file content/tutorials/projects/basics/group2.project



Now select the Background layer of the image and lock the Attribute Editor. Look for the attribute *Geometries* that can be found in the *Visibility* horizontal tab.

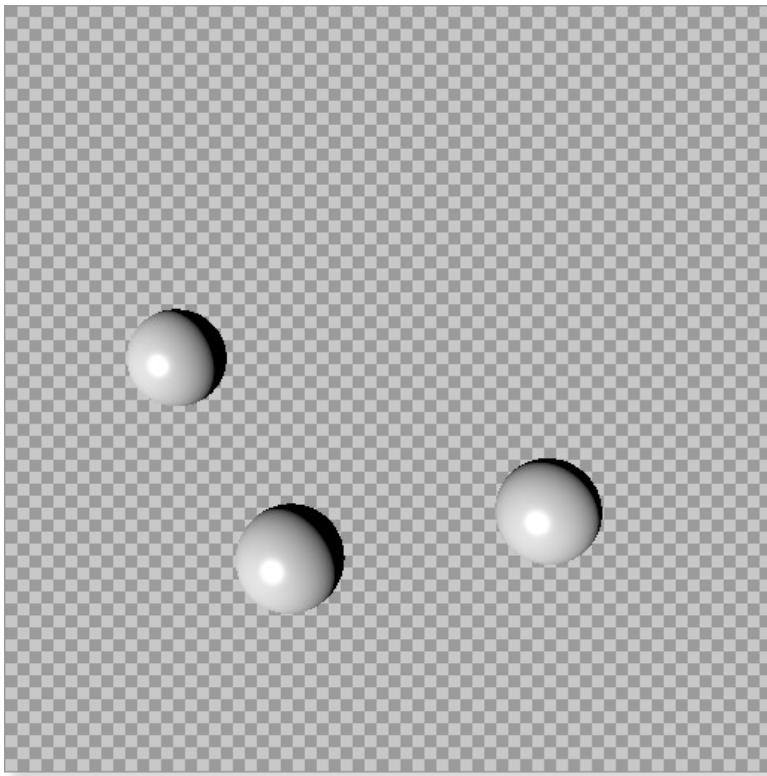


Select both sphere and sphere1 in the Browser and drag and drop them in the *Geometries* attribute of the Attribute Editor. You should now have something like this:



By drag and dropping the 2 spheres, the attribute editor automatically created a new group including sphere and sphere1. You can find this group in the Browser: `image_Background_geometries`

If a group is already referenced (this is the case now), you can still append items in the group by pressing **CTRL** and drag and dropping new items. Select sphere2, hold **CTRL** and drag and drop it to *Geometries*. You should now have:



Scatterer Basics Part 1

In this mini tutorial we will go through Scatterers. Scatterers, scatter geometries, combiners or scatterers used as input particles or geometry vertices. To set the input you need to set the attribute *Geometry Support* to either a point cloud (bunch of particles) or a geometry if you wish to use its vertices.

There are several point cloud generators available: *Point Array*, *Point Cloud*, *Point File*, *Point Volume*.

- *Point Array* fills a box with a number of particles evenly spaced.
- *Point Cloud* samples a geometry and put particles on its surfaces.
- *Point File* imports particles from file (typically from ABCs)
- *Point Volume* fills a volume (box or sphere) with random particles.

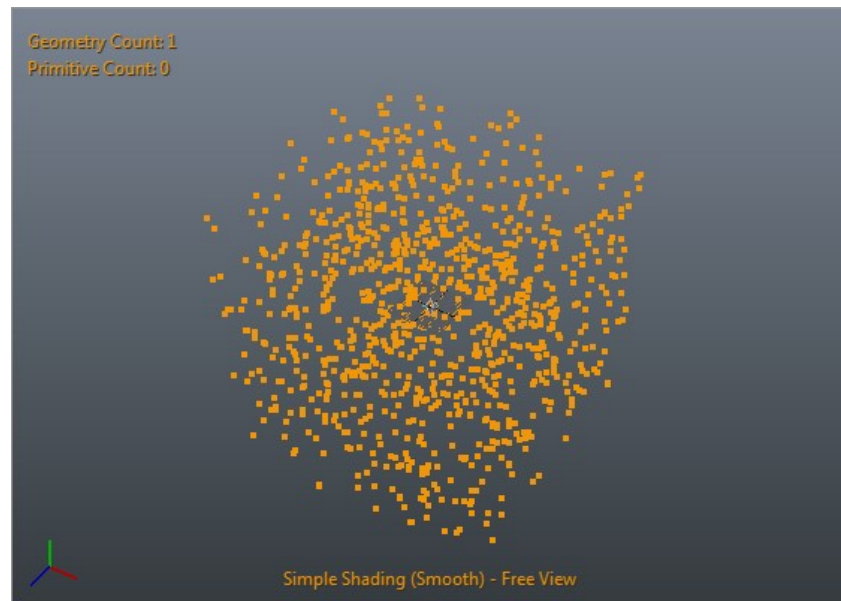
Note

Particles don't show up yet in the Image View, so you'll have to use a 3d View to display them.

Creating a Point Volume

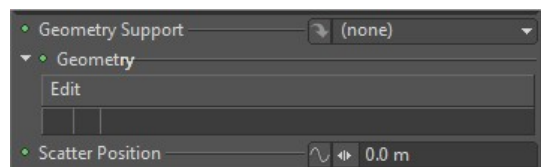
Let's create a *Point Volume*: **Create** > **Geometry** > **Point Volume** or right click in the right pane Browser and select **New...** > **Geometry** > **Point Volume**

Now set its attributes to *Size* (100.0, 100.0, 100.0) and *Count* 1000. You should get something like this:



Creating a Scatterer

Now let's create a *Scatterer* **Create** > **Scatterer** or right click in the browser right pane and **select New...** > **Scatterer**. In the attribute editor look for the attributes *Geometry Support* and *Geometry*. The later being actually a list of geometries similar to the *Objects* attribute of a combiner.



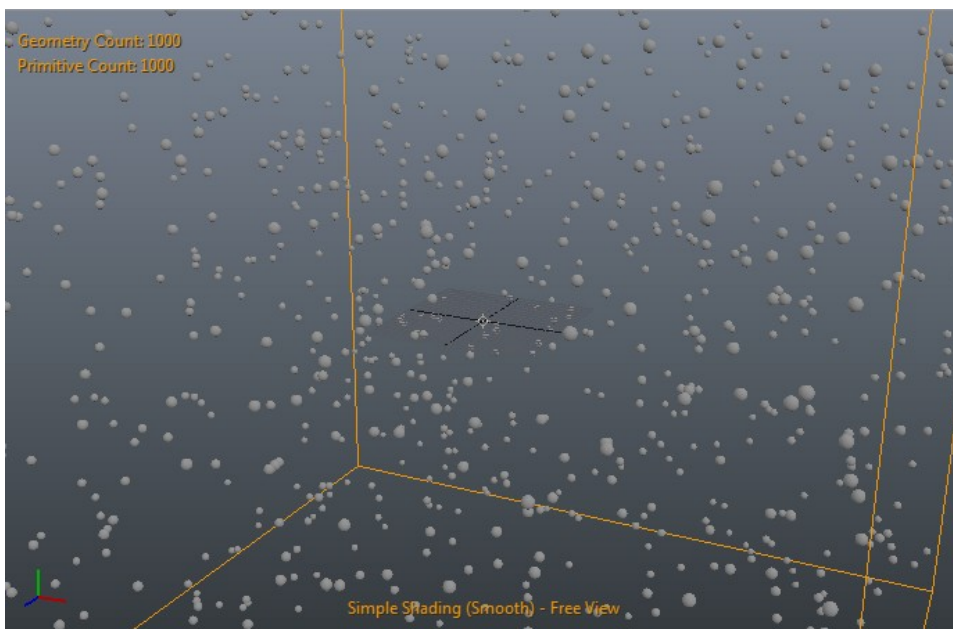
Set the attribute *Geometry Support* to the *point_volume* we've just created. Ok, now we have our scatterer, how do we specify the geometry (geometry, combiner, scatterer) to be scattered on each particle?

Just to do this, we will need a geometry. Let's create a sphere (**Create** > **Geometry** > **Sphere** or right click in the browser right pane and select **New...** > **Geometry** > **Sphere**). Now, select scatterer back scroll down the attribute editor until *Geometry* attribute is visible to drag and drop sphere.

Note

When you click and drag in the Browser without releasing the mouse button, it doesn't select the item, but it initiate a drag and drop.

You have now scattered a thousand spheres:

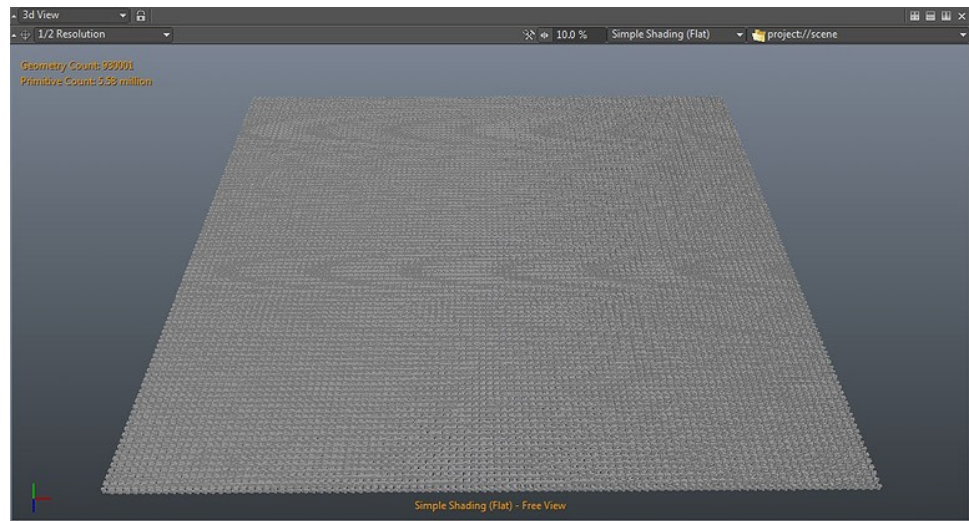
**Note**

If you modify the point cloud, for example, by setting *Count* to 10000, the scatterer will update accordingly.

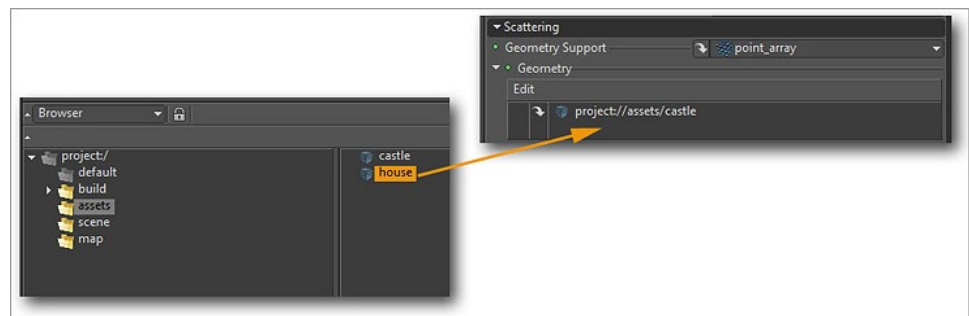
Scatterer Basics Part 2

Before proceeding to this tutorial, please load the project file content/tutorials/projects/basics/scatterers2_start.project

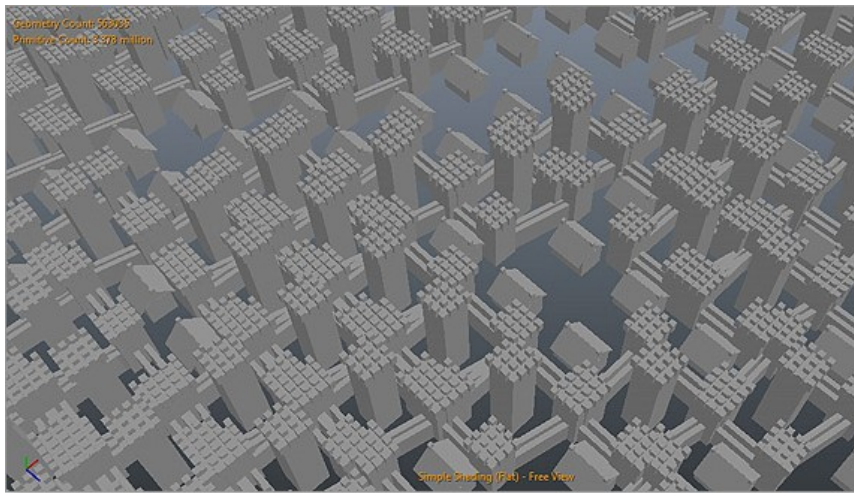
Now, change the *3D View* display filter mode from *Object* to *Browse...* and select the scene context.



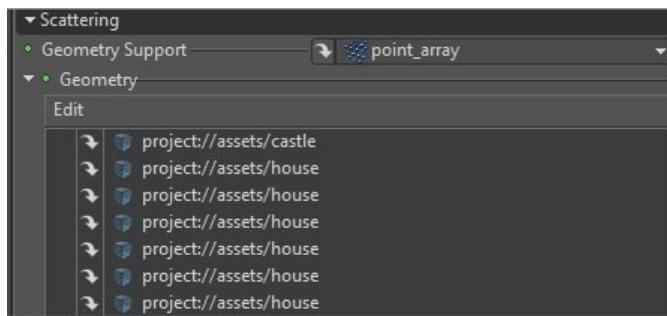
You will see many objects scattered on a *Point Array*. Select `scene\scatterer` and inspect the *Geometry* list attribute. Lock the *Attribute Editor* and go to assets context using the *Browser*. You'll see a castle object (which is already in the scatterer's list), and a house object. Drag and drop house in the scatterer's *Geometry* list.



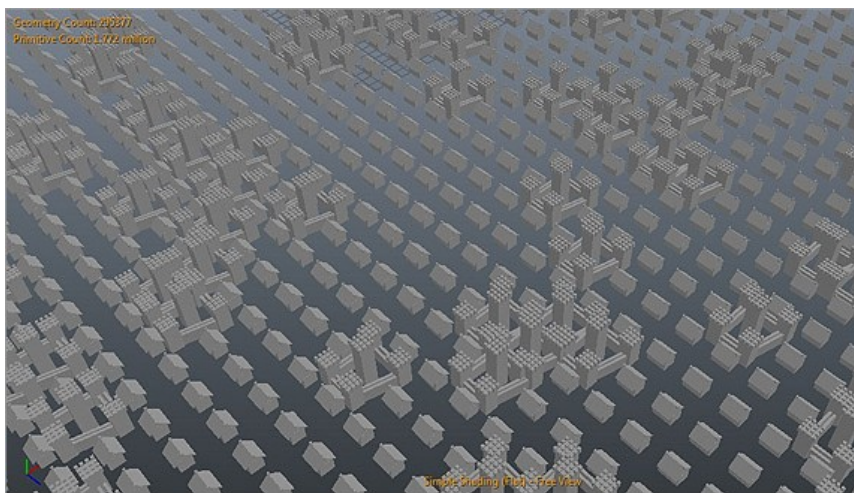
If you get closer in the *3D view*, you'll see a mix of castles and houses.



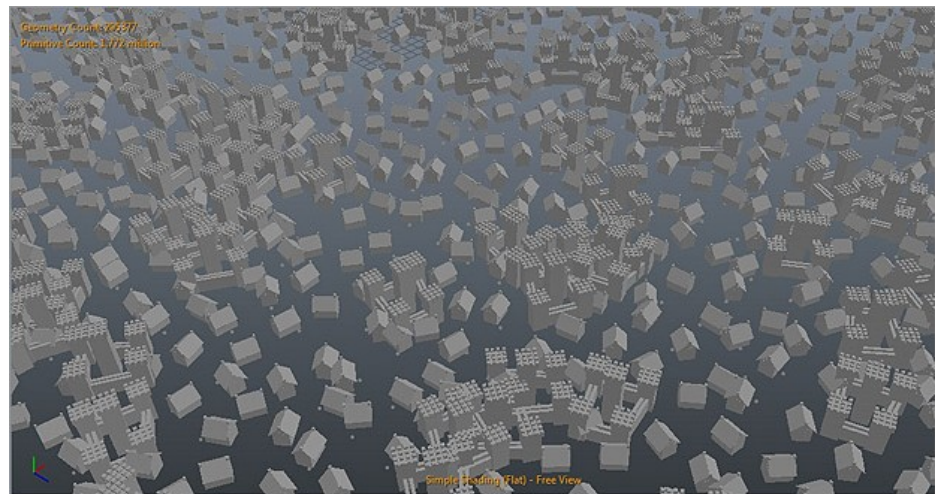
There is way too much castles, we would like more houses. To do that, simply add house again five more times in the *Geometry* list.



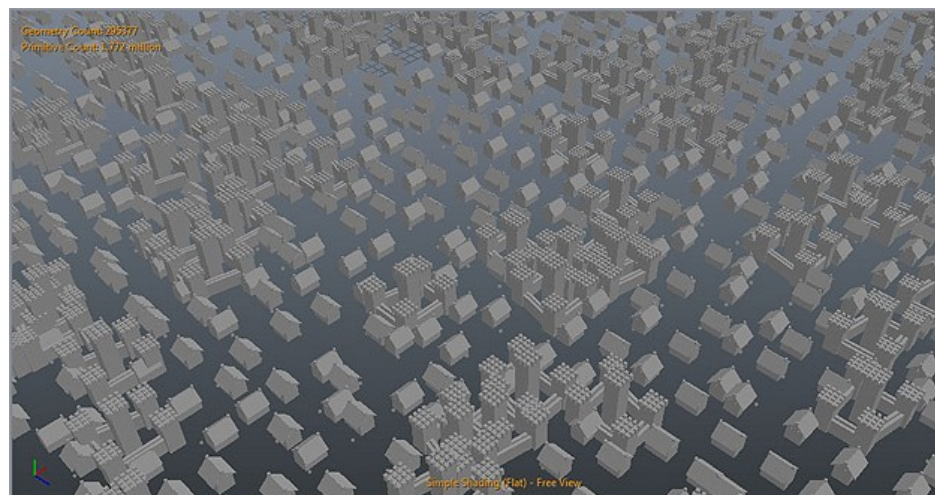
There now is six times more houses than castles in the scatterer.



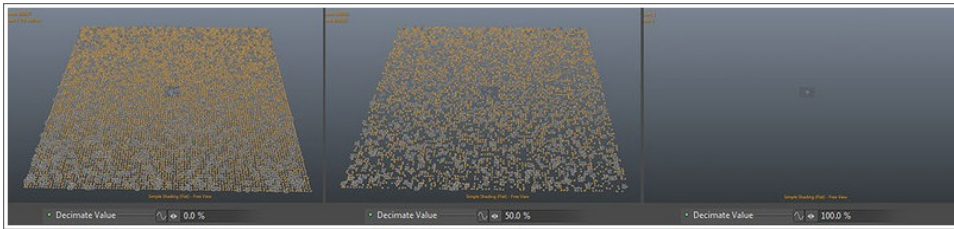
Now let's randomize the position of the buildings. In the scatterer's attribute, set *Position Variance* to 1.0 m , 0.0 m, 1.0 m. We also need to randomize the rotation. In order to randomly rotate each copy around the Y axis, set the rotation variance to 0,360,0, .



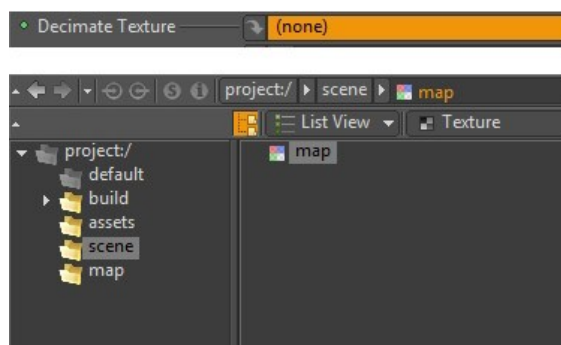
If you want the copies to keep perpendicular angle, you can set the *Scatter Rotation Step* attribute to 90 on Y axis. Here is the result:



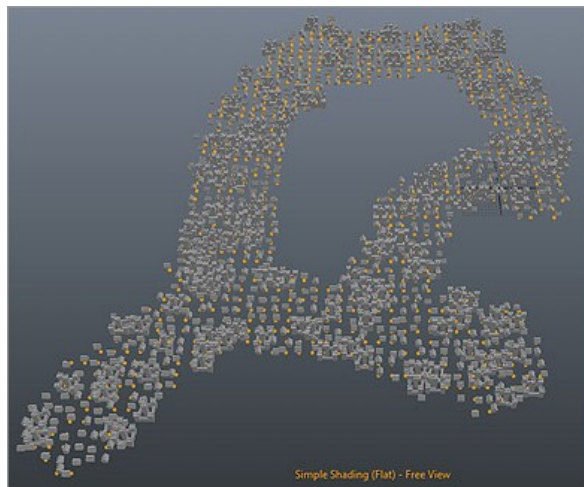
Now, we want to control the density of our houses.. zoom out in the *3d View* to see the whole scene and select the `scene\point_array` object and refresh the Attribute Editor. To refresh the Attribute Editor simply click on the little circular arrow at the right of the attribute editor's selection bar. The *Attribute Editor* selection is now locked to `point_array`. Look for the *Decimate Value* and play with the settings, from 0 to 100%.



Set the *Decimate Value* back to 0% and click on the *Decimate Texture* button. Select scene\map and apply.



Now you should get something like this in the *3D View*:

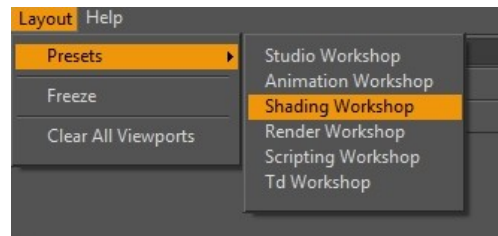


Play in the *Timeline* for a little surprise... If you look at the *Image Gallery*, you will see the animation which now drives the *Decimate*. In fact, this animation is the output of a full Clarisse scene you can find in \map context. This introduces you to the Picture In Picture concept... Basically, any Clarisse image can be dynamically used in any texturable attribute.

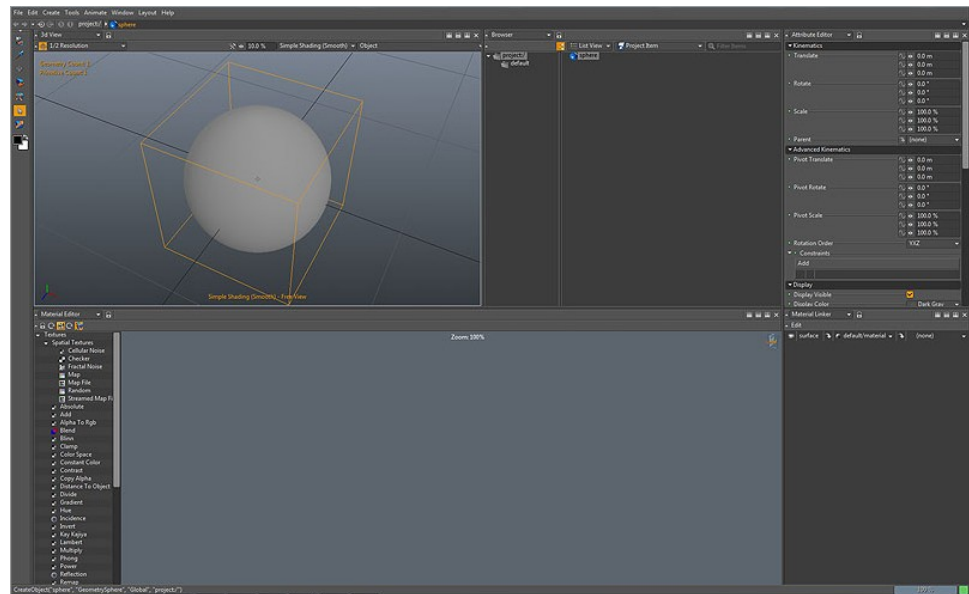
Displacements Basics

Here is a mini tutorial covering displacement shaders basics.

Let's start by setting the layout to *Shading Workshop* (**Layout** > **Presets** > **Shading Workshop**).

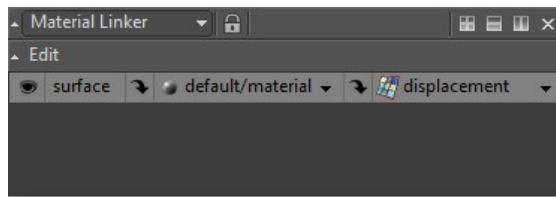


Replace the *Image View* by a *3D View*, then create an implicit sphere (**Create** > **Geometry** > **Sphere**)

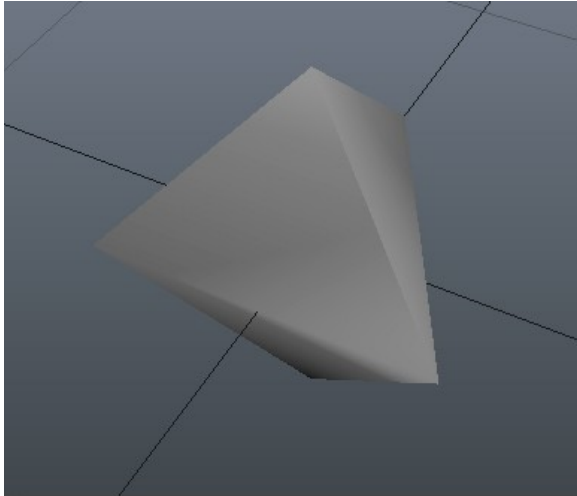


Setting Displacements

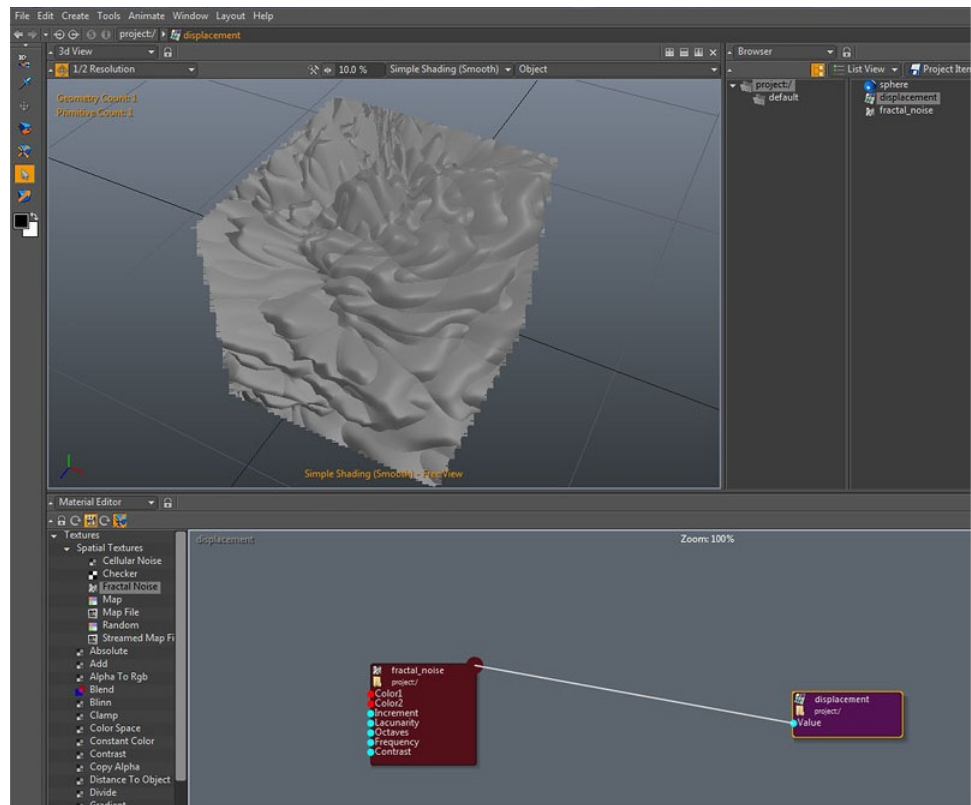
Create a displacement (**Create** > **Displacement**). Then assign it to the sphere shading group by drag and dropping it to the right part of the Material Linker. Material Linker's right part is dedicated to displacement shaders.



You should see this in the 3D View:

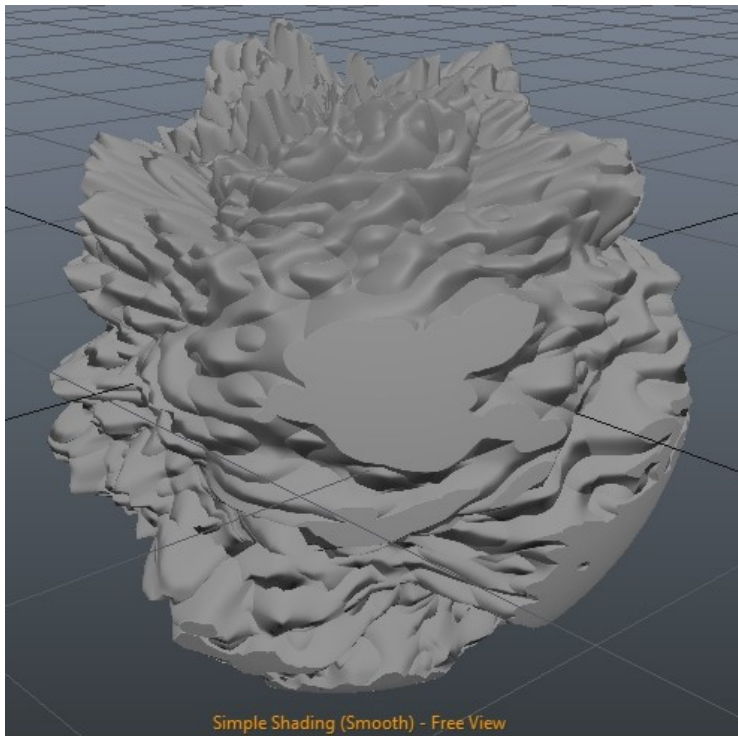


Select the sphere and set *Displacement Adaptive Span Count* to 40. In the Material Editor, now add a fractal noise and connect it to the displacement input *Value*.



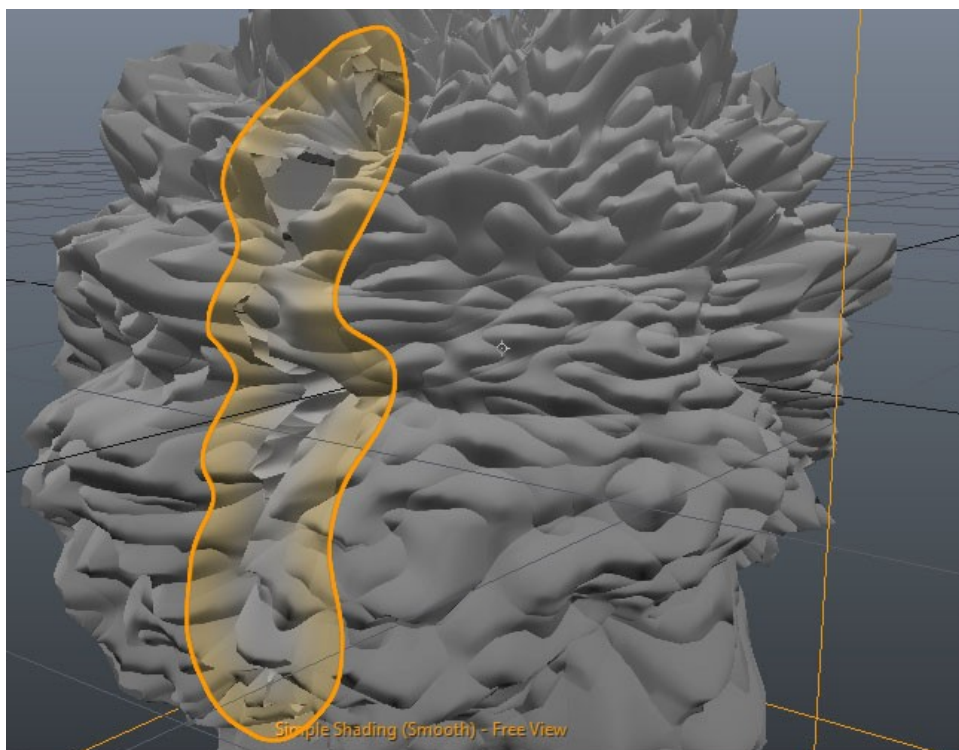
Setting Bounds

You should see a strange result in the 3D View. This is because we didn't setup our displacement bounds. The geometry is then clipped by the bounding box of the geometry. Select displacement and set *Bound* to (1.0,1.0,1.0) .

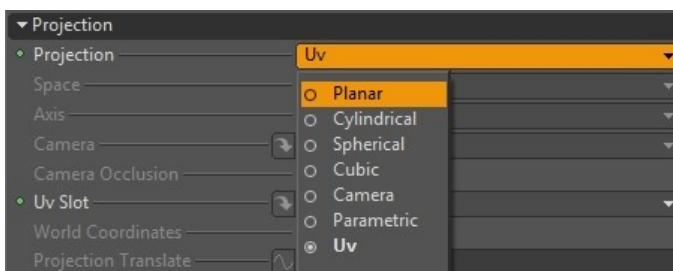


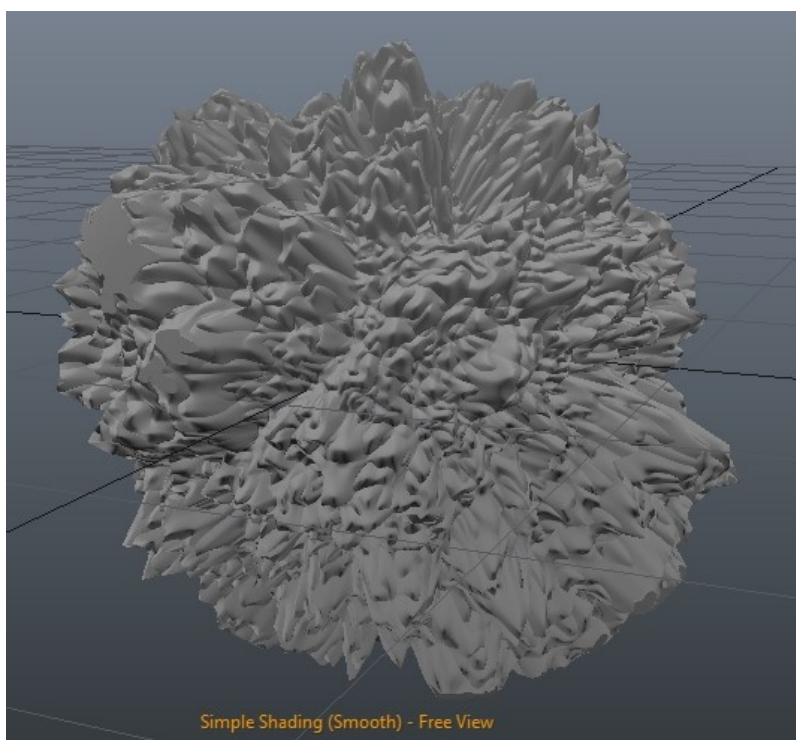
Setting the Texture

If you orbit and look the back of sphere, you will notice a crack. This is caused by the UV projection seam of the Fractal Noise which doesn't tile.



Select the `fractal_noise` and change *Projection* to *Planar* to fix this issue.





In the Material Editor, play with different nodes to tweak the displacement shader while seeing the result interactively.

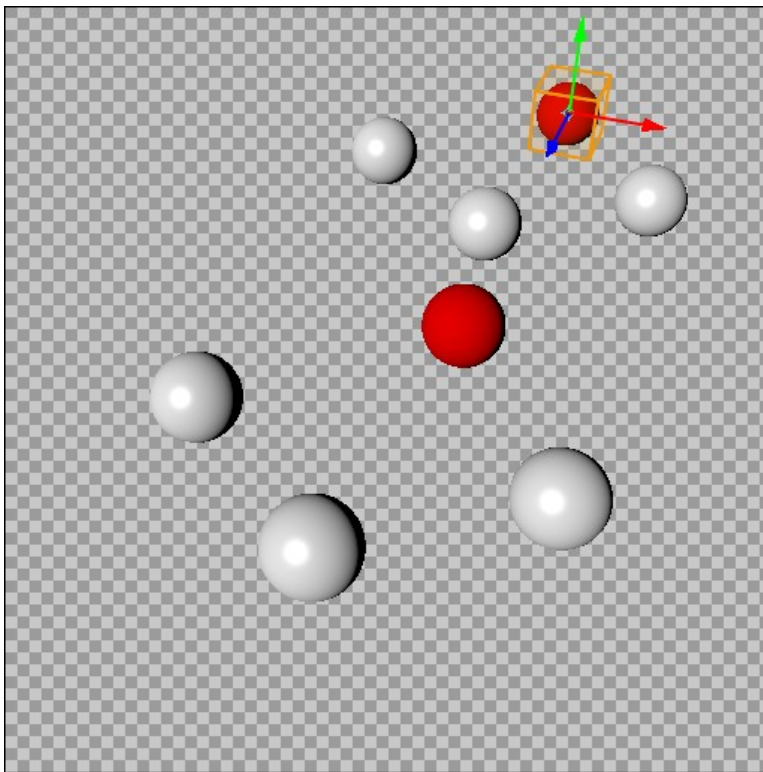
There are several ways to assign materials:

- Drag and dropping in the Image View
- Using the Material Linker
- Using Material Overriding

Drag and Dropping in the Image View

Create a new material: **Create > Material > Standard** or in the right Browser pane right click and select **New... > Material > Standard**

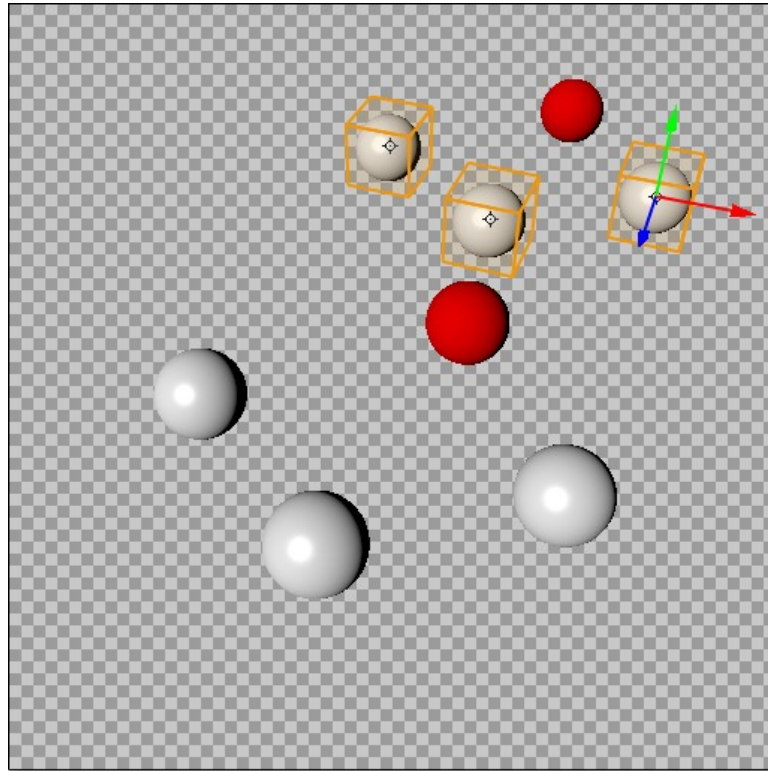
In the Attribute Editor set its *Diffuse* to red (1.0, 0.0, 0.0). To assign the material simply Drag and Drop it from the Browser to one of the sphere in the background displayed in the Image View.



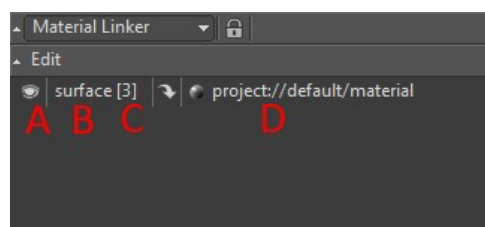
The selected sphere (sphere3) is the sphere on which we performed material drag and drop. As you can see, two spheres turned red. The 4 spheres in the foreground being the combined version of the 4 spheres that are in the background. By default, assigning a material to one of its source affects the combiner.

Using the Material Linker

Multi select sphere, sphere1, sphere2 in the Browser or in the Image View by clicking on them and holding **CTRL** key.



Now, if you look at the Material Linker you can see:

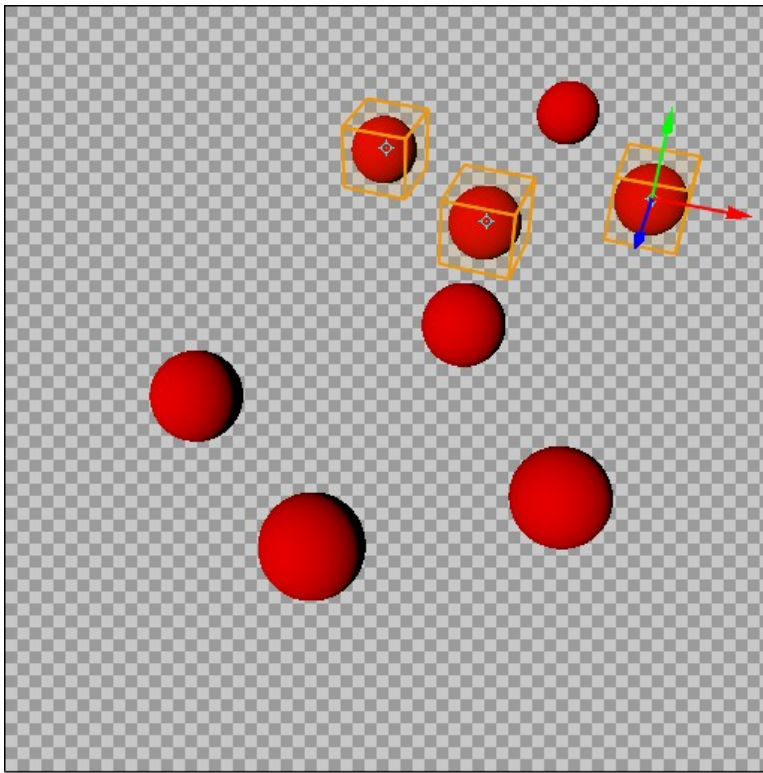


Where:

- A. is the visibility flag of the shading group defined by the geometry.
- B. is the name of the shading group defined by the geometry.
- C. is the number of geometries sharing the same shading group name.
- D. is the name of the material currently referenced by the shading group.

Here it's the default one which is `project://default/material`

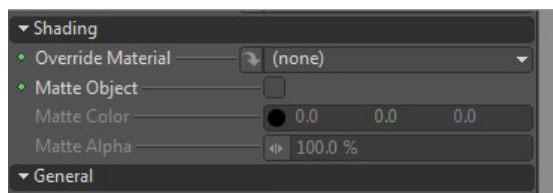
To assign your material simply drag and drop it from the Browser to D. Or click on D. and browse to the material you wish to assign. Here what you should get:



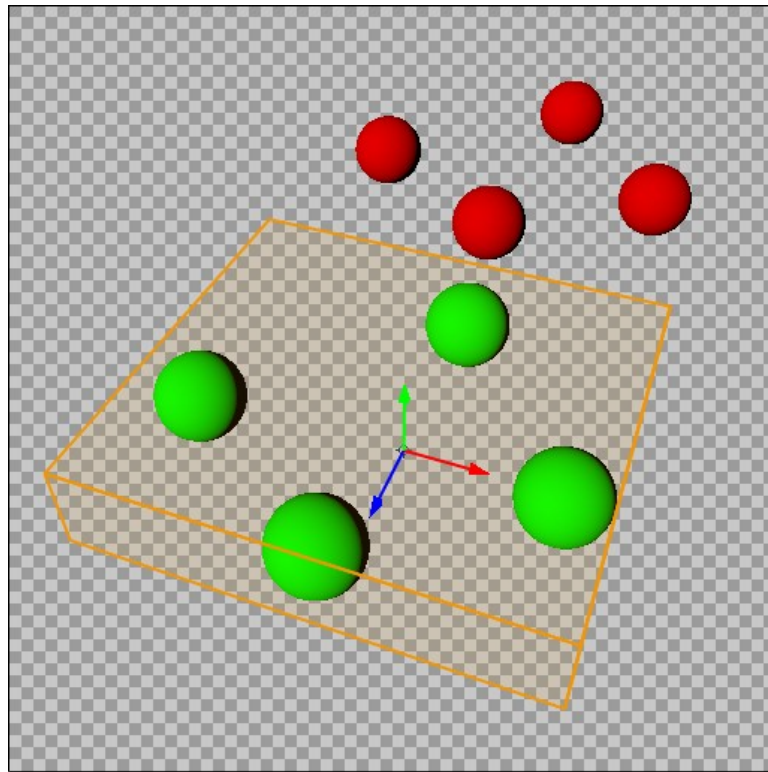
Material Overriding

Every scene objects (Combiners, Scatterers, Geometries) and even 3D Layers share an attribute called *Material Override*. This attribute is very useful if you wish to bypass materials that are assigned to geometry shading groups.

Let's create a new material **Create > Material > Standard** and set it diffuse to green (0.0, 1.0, 0.0). In the browser, select combiner and scroll down the attribute editor until you see *Material Override* attribute:

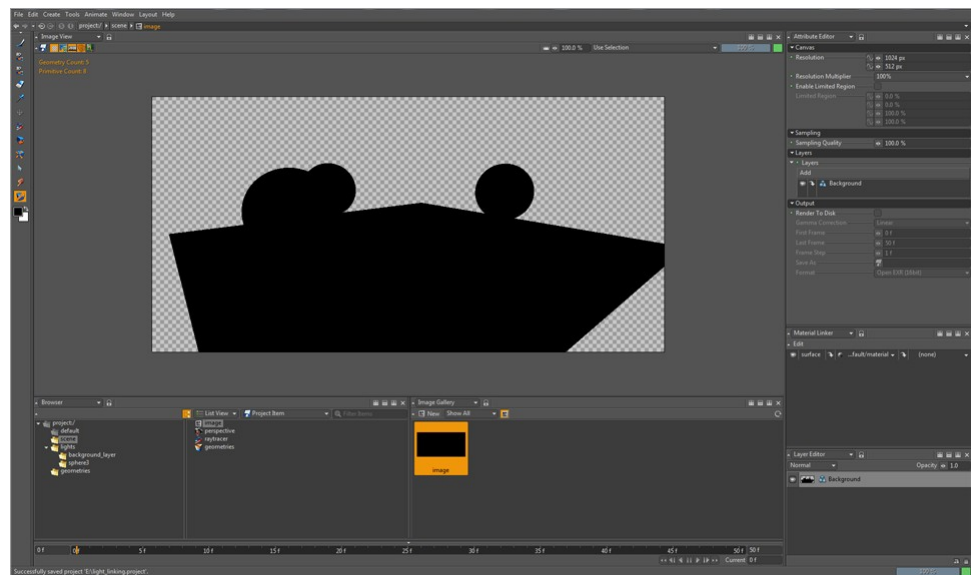


Now drag and drop standard1 (the new material you've created) into the *Material Override* attribute of combiner



Light Linking Basics

Here is a mini tutorial covering light linking. Before proceeding to this tutorial, please load the project file `content/tutorials/projects/basics/light_linking.project`



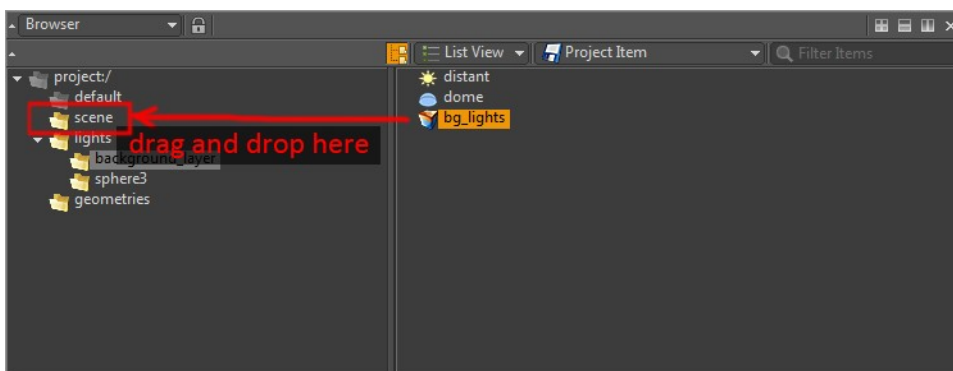
Light linking can be set on 3 different levels:

- 3D Layer level
- Scene Object level
- Light level

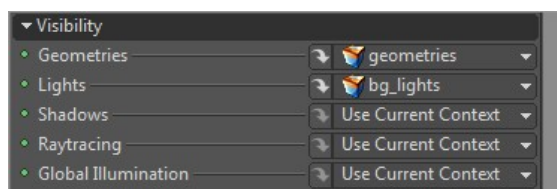
3D Layer

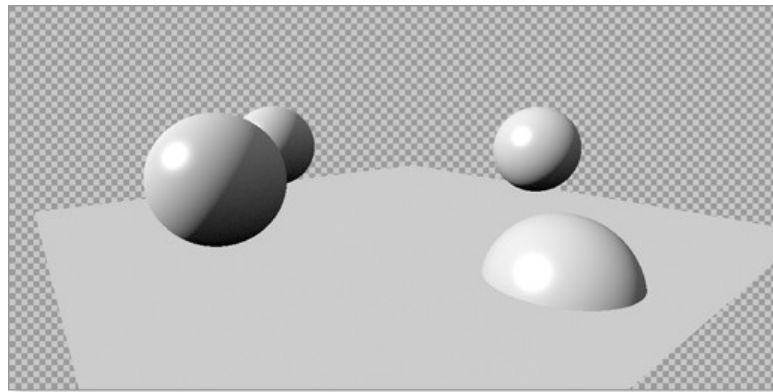
You can change the lighting of a 3D Layer by specifying a group of light to the *Lights* attribute of a 3D Layer. Using the Browser, go to `project://lights/background_layer`. In this context there are two lights: `dome` and `distant`. Select both of them, right click and select **Group** to create a new group referencing these two lights. Press **F2** and rename the group to `bg_lights`.

Now drag and drop `bg_lights` to `project://scene`

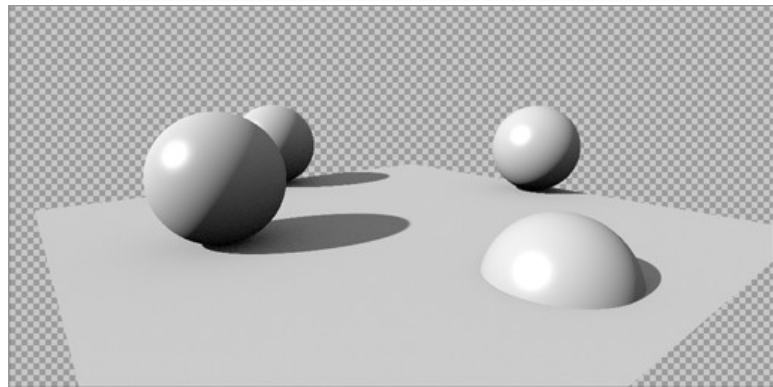
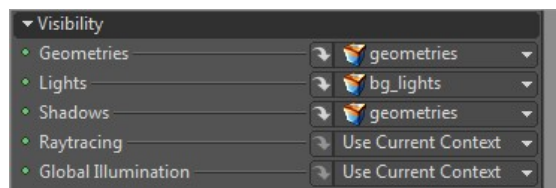


Now select Background in the Layer Editor and drag and drop `bg_lights` into *Lights* attribute.





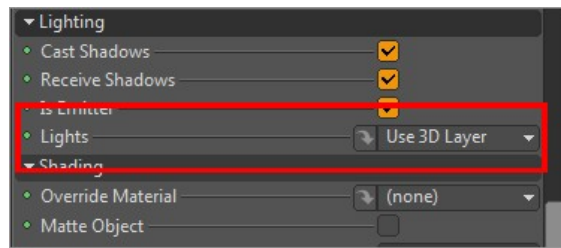
Note there is no shadow in the image. This is normal as *Shadows* attribute of our 3d layer is set to *Use Current Context* and no geometries are in the current context. To get our shadows, drag and drop `project://scene/geometries` group into *Shadows*.



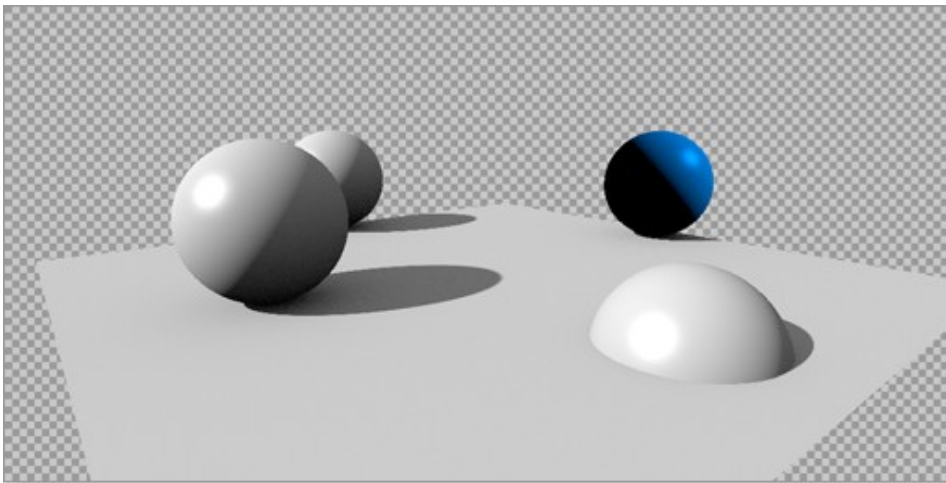
Scene Object

You can also override lighting at scene object level. Every scene object has a *Lights* attribute where you can specify a group of lights. By default, Scene Objects *Lights* attribute are set to *Use Layer 3D*. This means objects use implicitly lights that are referenced in the layer 3d they are rendered in.

Select `sphere3`, lock the Attribute Editor and scroll down to see its *Lights* attribute.



Now using the browser go to `project://lights/sphere3` and drag and drop spot into *Lights* attribute in the Attribute Editor. Unlock the attribute editor.

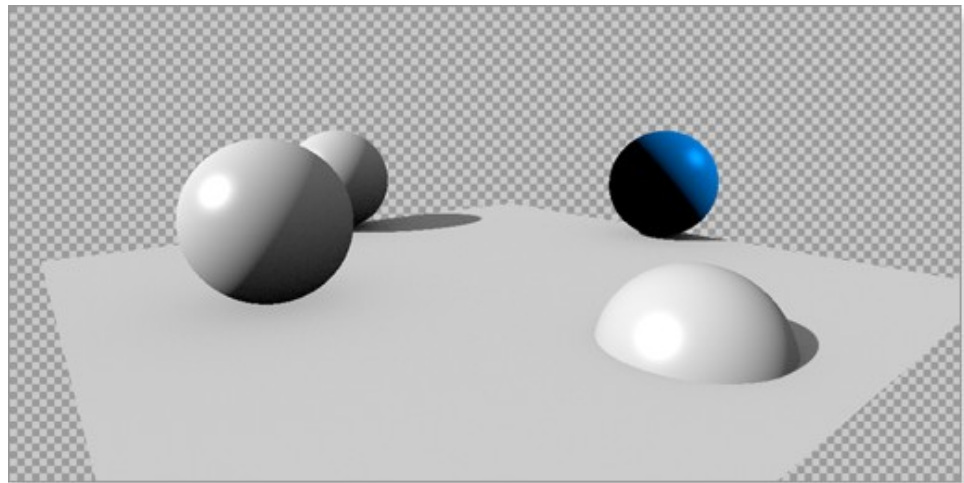


sphere3 is now only illuminated by `project://lights/sphere3/spot`. In the same way, you can exclude an item from all lights by setting *Lights* to the default empty group located in `project://default/empty`

Lights

You can also specify which scene objects are "seen" by the lights by setting *Geometry Group* attribute. By default, light's *Geometry Group* is set to *Use 3D Layer*, which means that lights "see" what's specified in the layer 3d *Shadows* attribute.

Using the browser go to `project://geometries` and select sphere1, sphere2 and sphere3. Right click and select **Group** to group them. Rename the group to `distant_shadows`. Select `project://lights/background_layer/distant`. In the Attribute Editor, set *Geometry Group* to `project://geometries/distant_shadows`



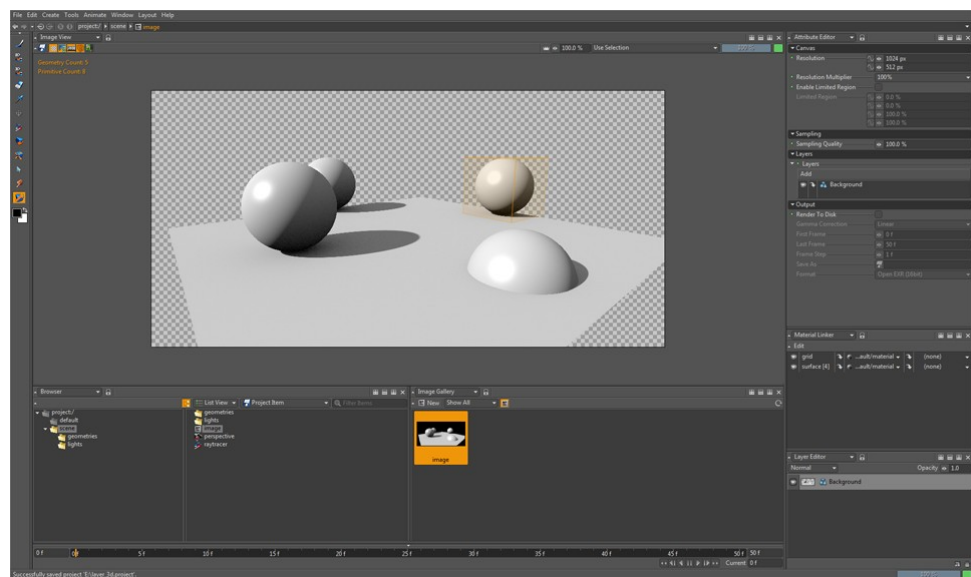
Here, we explicitly set the occluders for the light distant. So sphere isn't casting shadows from distant but still receive its illumination from both distant and dome.

Note

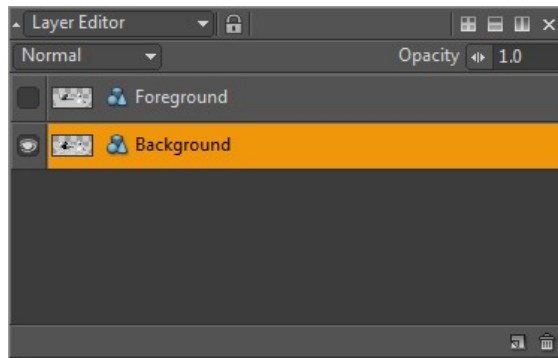
This also applies for global illumination as it's a light too but instead of using what's set in *Shadows* attribute of the 3D layer, it looks for what's set in *Global Illumination* attribute.

Layering Basics

In Clarisse, *images are like onions, they have layers...* Here is a mini tutorial covering image layering. Before proceeding to this tutorial, please load the project file content/tutorials/projects/basics/layer_3d.project

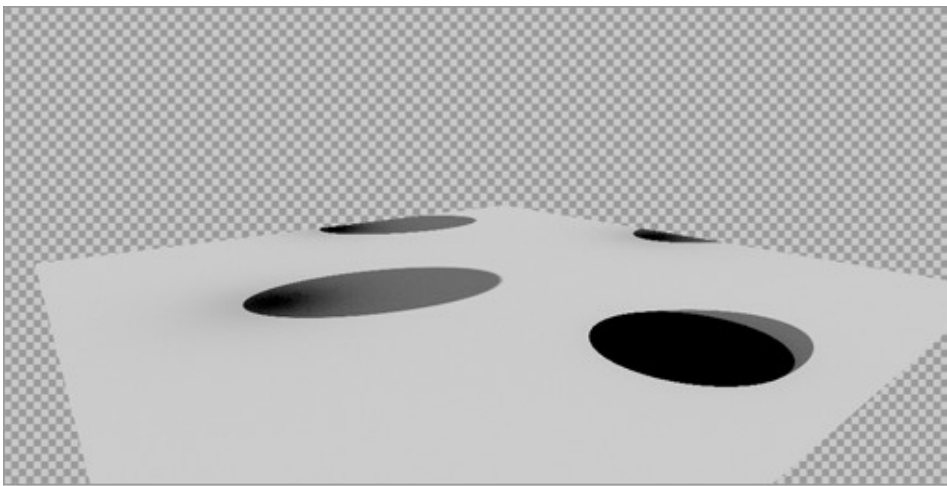


In the Layer editor, right click on the Background layer and select **Duplicate**. You now have two identical layers. Select Background_copy. Press **F2** to rename it Foreground. Click on the eye icon to toggle Foreground layer visibility and select Background layer.



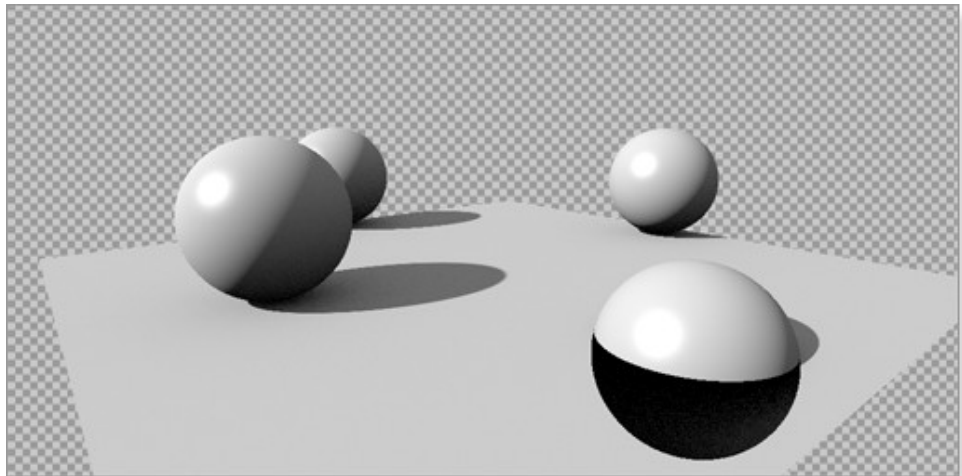
Setting Background Visibility

Now we will set the Background layer visibility. Lock the Attribute Editor and scroll down until you see the attribute named *Geometries*. In the Browser, go to geometries context, drag polygrid and drop it into *Geometries* attribute in the Attribute Editor. You should now see this:



Setting Foreground Visibility

Select Foreground layer in the Layer Editor and toggle its visibility. Unlock and Lock the Attribute Editor to lock its selection to the Foreground layer. Again, scroll down until you see the attribute *Geometries*. Now, in the Browser go to geometries context and select this sphere, sphere1, sphere2 and sphere3. Drag and drop them into *Geometries* attribute in the Attribute Editor. You should now see this:



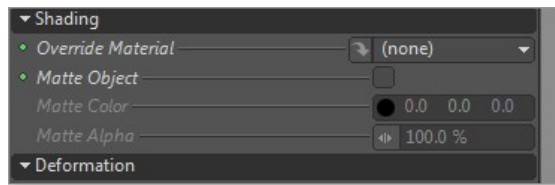
As you can see, something goes wrong with the right sphere. In fact we need to add the polygrid as a black hole to cut out Foreground alpha.

Cutting out Foreground Alpha

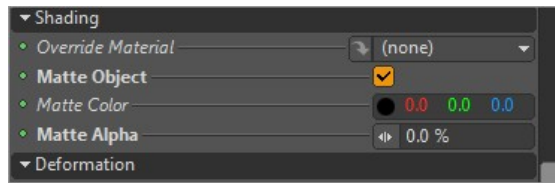
We will now create an instance of the polygrid, localize its matte properties to use it to cutout the alpha of our foreground layer. Select the polygrid in `project://scene/geometries` do either:

- **CTRL + i**
- right click in the Browser and select **Instantiate**
- go to **Edit > Instantiate**

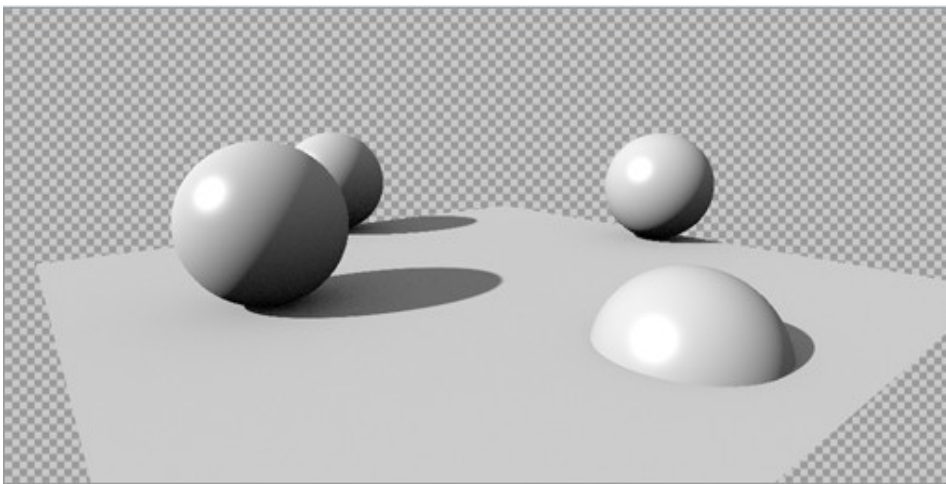
Make sure the Attribute Editor is unlocked to display polygrid instance (that should be named `polygrid1`). Now look for the attribute *Matte Object* which should be displayed in *italic*.



Right click over *Matte Object* attribute and select **Localize** in the popup menu. The attribute name should be now displayed in **bold**. Toggle *Matte Object*. Now, localize *Matte Alpha* and set its value to 0.0%.



Now select back the Foreground layer, lock the editor and scroll down to display *Geometries* attribute. We will now add the polygrid instance to the Foreground *Geometries* group. In the Browser hold **CTRL** and drag and drop `project://scene/geometries/polygrid1` into *Geometries* in the Attribute Editor. You should now see this:



You probably noticed, but each time you're modifying `polygrid1` Clarisse re-renders both layers. In fact, `polygrid1` is implicitly referenced as shadow caster and as affecting raytracing. If you look to *Visibility* attributes of a 3d layer, you can see:

- Geometries (sets what's visible to the camera)

- Lights (sets which lights are used by the layer)
- Shadows (sets which geometries are casting shadows)
- Raytracing (sets what's seen in reflection/refraction)
- Global Illumination (sets what's emitting global illumination)

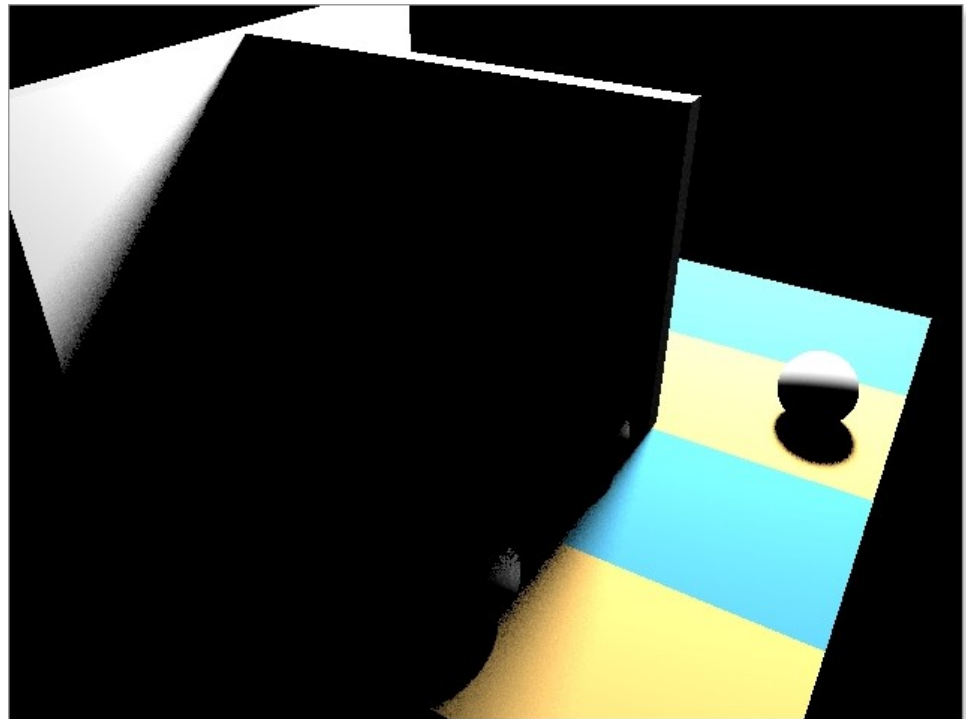
By default, these attributes are set to *Use Current Context*. This means that any item that is in the same context (or in a sub context), where the image is located, will be visible. Here, the image context is `project://scene`. So, any item under `project://scene` is seen by the camera, lights, shadows etc.

As `polygrid1` is located in `project://scene/geometries`. It is then implicitly referenced for *Reflection*, *Shadows*, *Global Illumination*... of our layers. To avoid re-rendering, simply create and reference explicit groups that aren't referencing `polygrid1`.

Global Illumination and Ambient Occlusion Basics

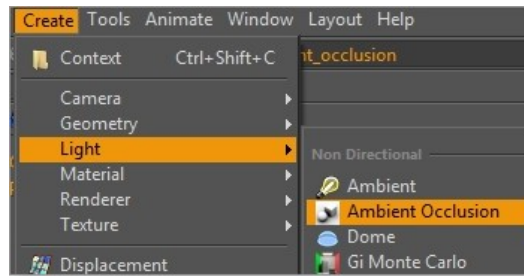
Before proceeding to this tutorial, please load the project file content/`tutorials/projects/basics/GI_occlusion.project`

You should get this render if you select the image:

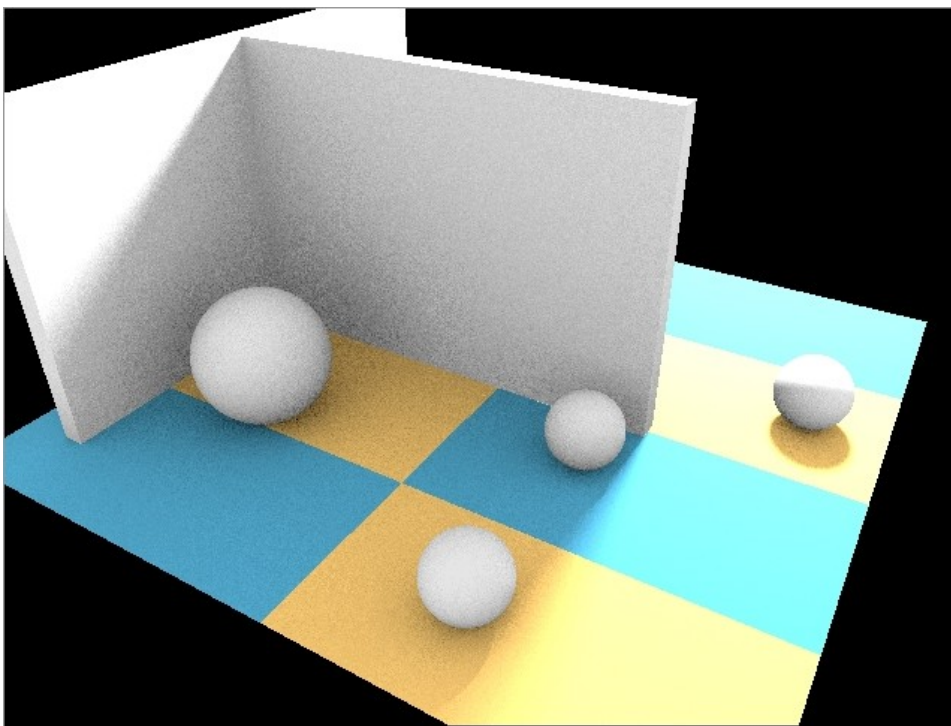


Then, let's see how to add an ambient occlusion lighting. This effect simulates a diffuse overcast day lighting, by shading the objects based on how easy it is for lights to access them (corners and enclosed areas will be darker than open ones). In Clarisse, it is handled by a light.

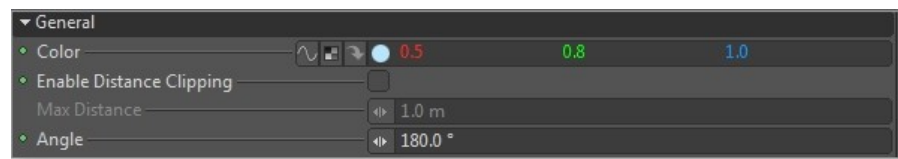
Go to the **Create** menu and choose **Light > Ambient Occlusion**.



The scene is now illuminated by a very soft light, surrounding all objects.



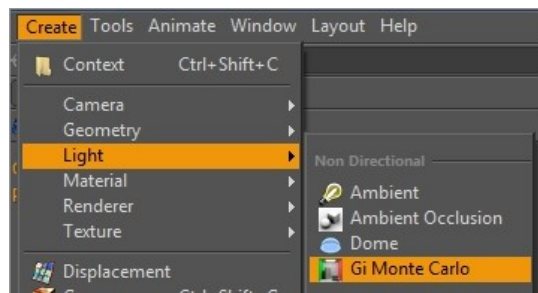
Select the `ambient_occlusion` light to inspect its attributes: click on *Color* and set it to 0.5, 0.8, 1.0



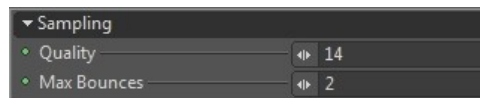
The scene is now lit by a soft blue overcast lighting... this can simulate outdoor's sky lighting. You can go further using a texture in the *Color* attribute, this is explained in the HDRI lighting tutorial.

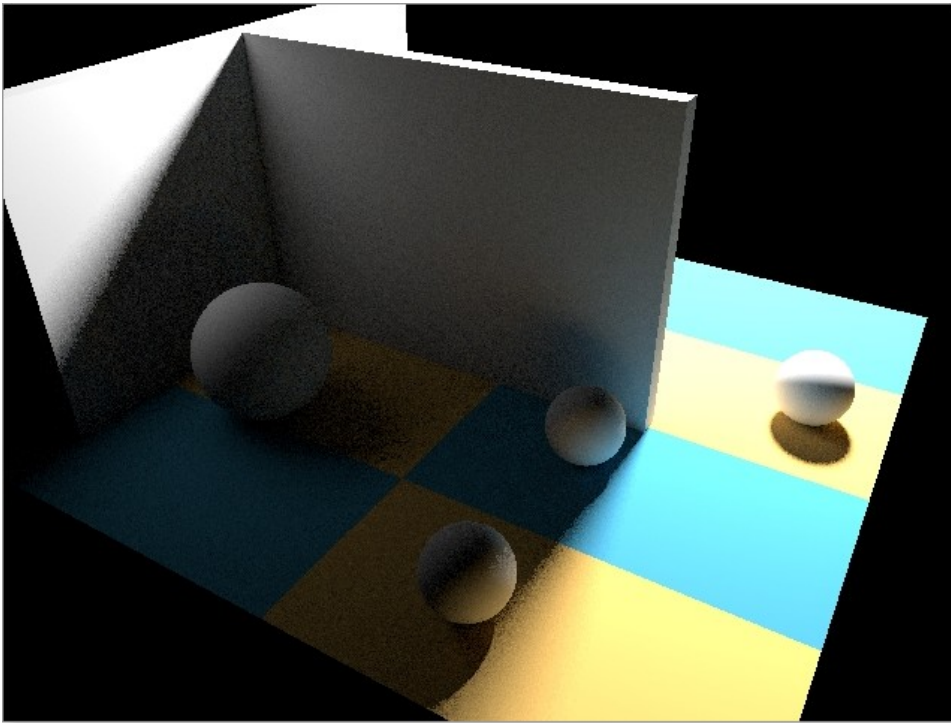
For now, delete `ambient_occlusion` light to revert to the original project.

Go to **Create** menu and choose **Light > GI Monte Carlo**.

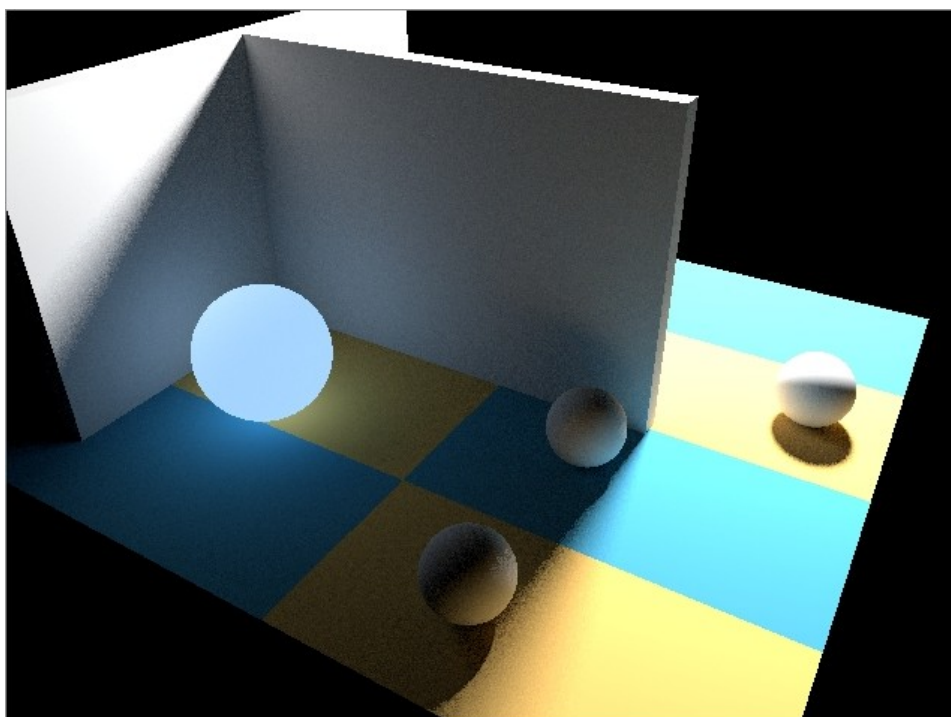


Select `gi_monte_carlo` and set the sampling *Quality* to 14 and *Max Bounces* to 2.. this will be enough for this example.





Now the light is realistically bounced from the floor to other surfaces of the scene. Notice how the blue and orange patches colorize the two center spheres. In the browser, select the `self_illuminated` material and drag and drop it on the left sphere, in the image view. This is another property of global illumination : any self illuminated surface illuminate surrounding objects, like in real life. Combined with textures, you can use this to light your scenes with images (this is explained in the HDRI tutorial).

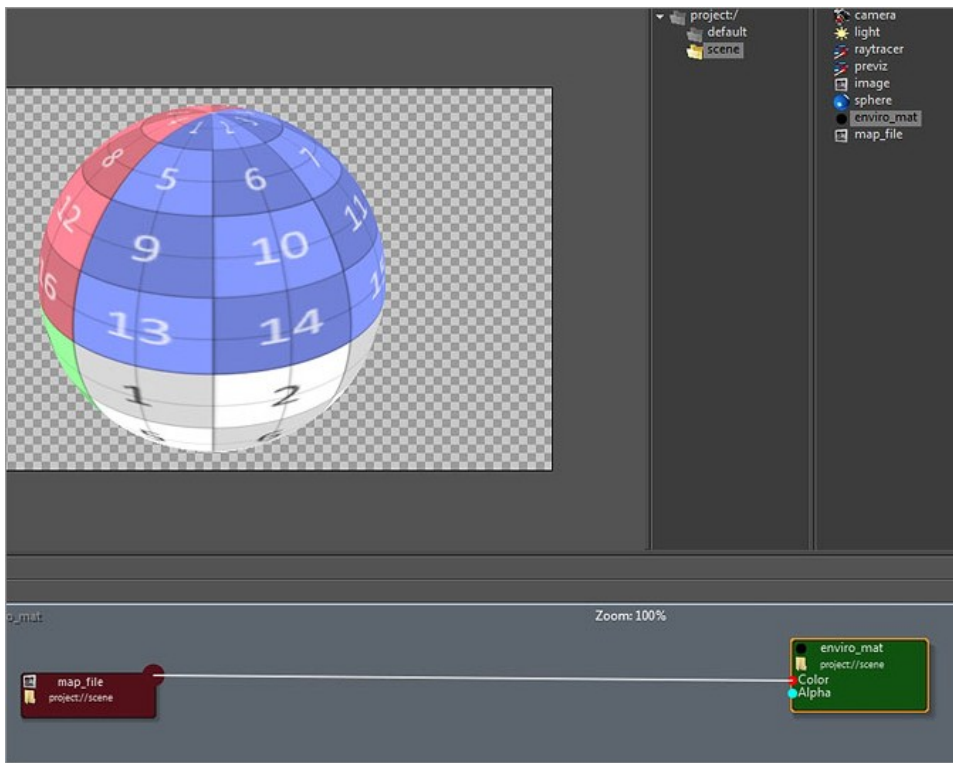
**Note**

As *Ambient Occlusion* and *Global Illumination* are both lights, everything you can do with lights applies too (exclusions, groups etc..)

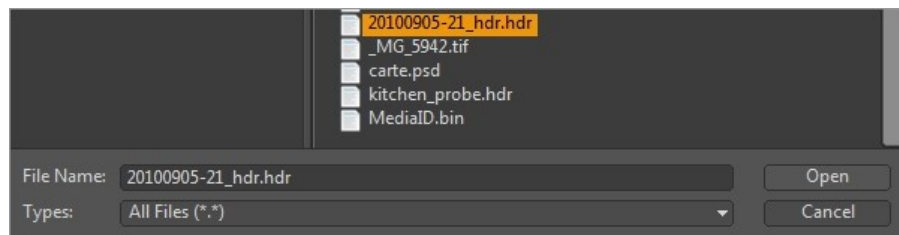
HDRI Lighting Basics

Create a new scene, and a sphere with a radius of 100m . Create a matte material, rename it "enviro_mat" and drag and drop it to the sphere.

Add a *Map File* and connect it to the *Color* input of enviro_mat.



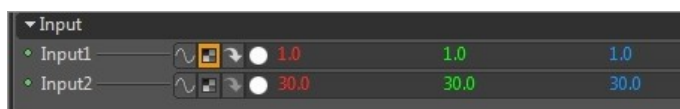
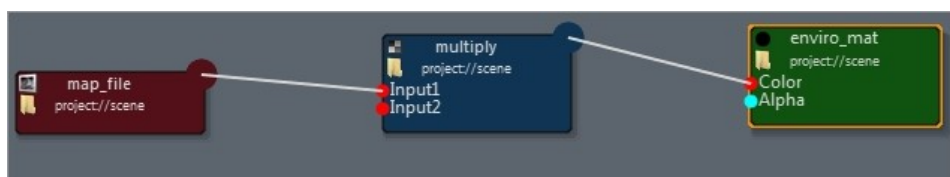
In the `map_file` attributes, set the *Filename* to a HDR map (you can find the one used in this tutorial here: <http://dativ.at/lightprobes/>). You must set the file browser to "all files *.*" to see .hdr format. The hdr map must be in latitude/longitude projection.



Next, uncheck *Linearize* attribute (HDR map are already in linear color space).



The image is too dark : insert a *Multiply* node between map_file and enviro_mat, and set the second input to 30.



Load an object using **File > Import > Geometry** and fit it in the image view by pressing **F** key.



Remove the default light and create a *GI Monte Carlo* light. Your object is now lit by the HDR image.

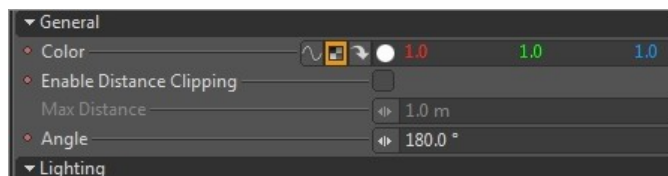
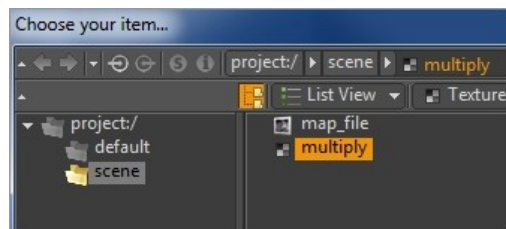


Create a *Standard Material* and drag and drop it on the dragon. Set *Reflection* to 0.1, *Enable Blurry Reflection*, and set the *Glossiness* to 70%, to see nice highlights from the HDR map. Of course you can set a higher sample quality both in the GI light and raytracer. Here is the result:

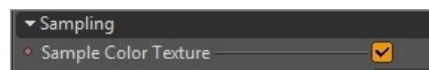


You can also use HDRI lighting without the cost of monte carlo radiosity, using *Ambient Occlusion* light. Here is how:

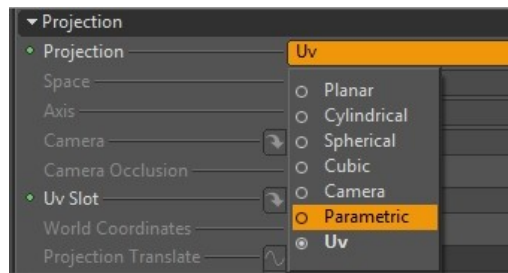
Remove `gi_monte_carlo` light. Select sphere and uncheck the *Cast Shadows* attribute. Create a new *Ambient Occlusion* light and connect `multiply` to its *Color* attribute.



In the Sampling tab, check *Sample Color Texture*.



Next, select `map_file` and set *Projection* from *UV* to *Parametric*.

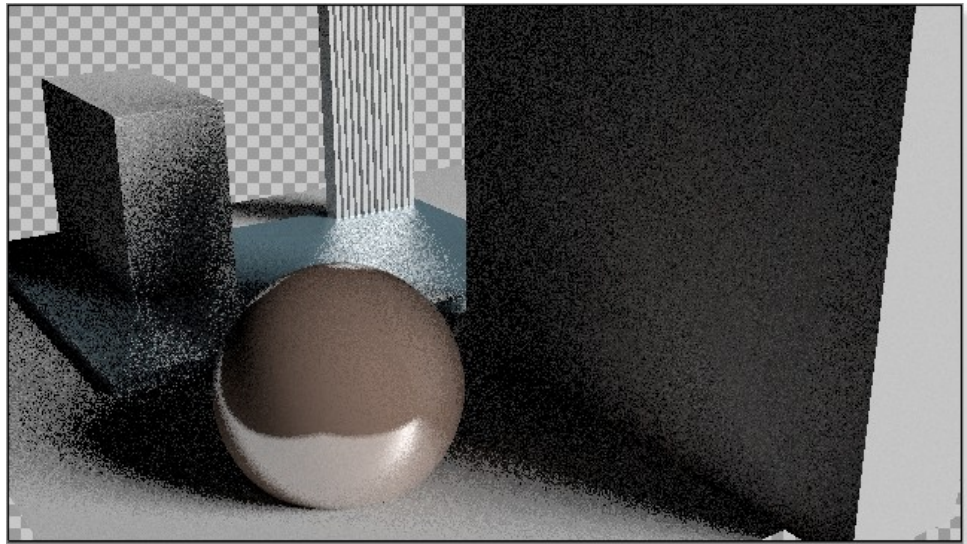


Set the ambient_occlusion *Quality* to something like 16 to get a finer result and you're done. The dragon is now illuminated by the HDR image, without using global illumination.



Sampling Quality Basics

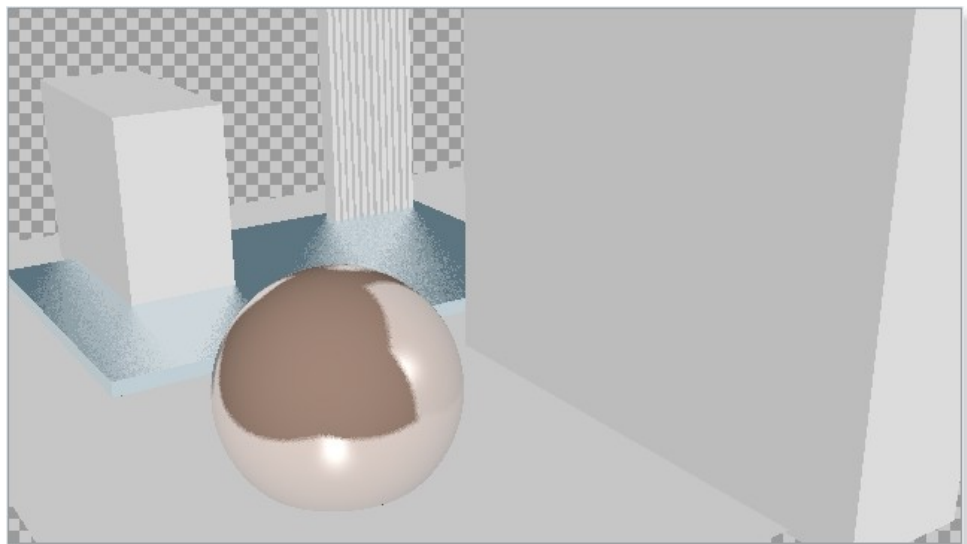
Before proceeding to this tutorial, please load the project file content/
tutorials/projects/basics/sampling.project



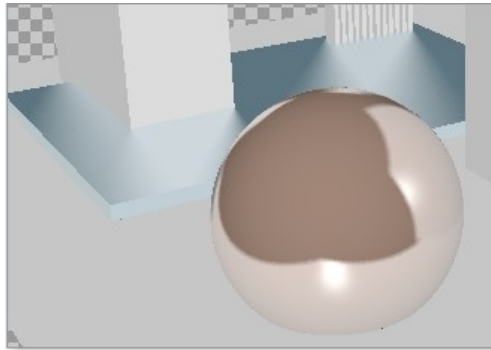
You will notice lots of noise and aliasing in this image...We'll discover how to remove them, keeping rendering times to a minimum.

In Clarisse, shading , lighting and anti-aliasing are independent. This is very important to keep your assets independent from shots. In many renderers, shading quality is multiplied by anti-aliasing quality. The user is thus forced to tweak all the shaders to keep render time low if a particular shot need more anti-aliasing.

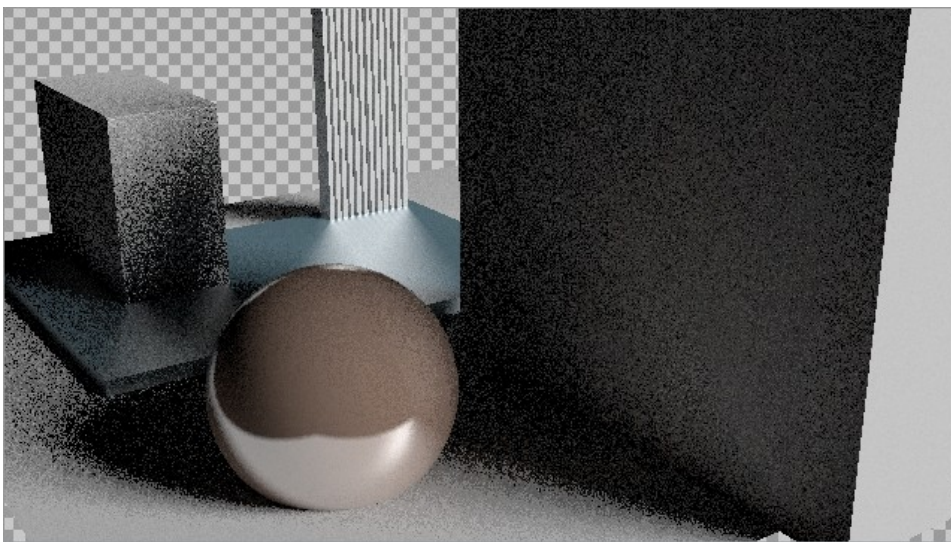
First, we need to set the materials quality. Select raytracer and enable *Previz Mode*.



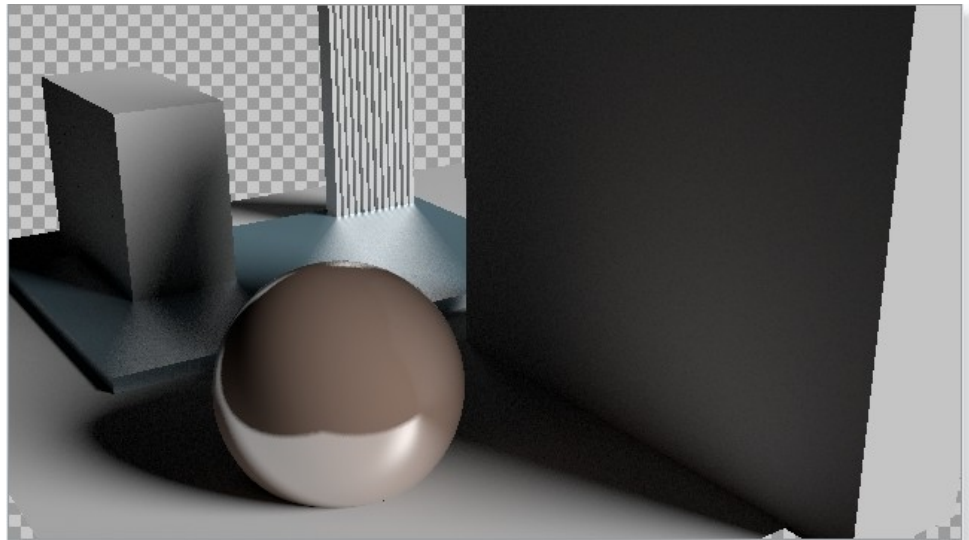
It overrides all the lighting by a very simple one, without shadows, allowing us to concentrate on the shading. We can see some noise in the soft reflections of the colored materials. Select `reflective_mat1` and raise *Reflection Quality* until the noise disappears. A value of 10 works well in this case. Now, select `reflective_mat2` and do the same. A *Reflection Quality* of 4 should be enough for this one.



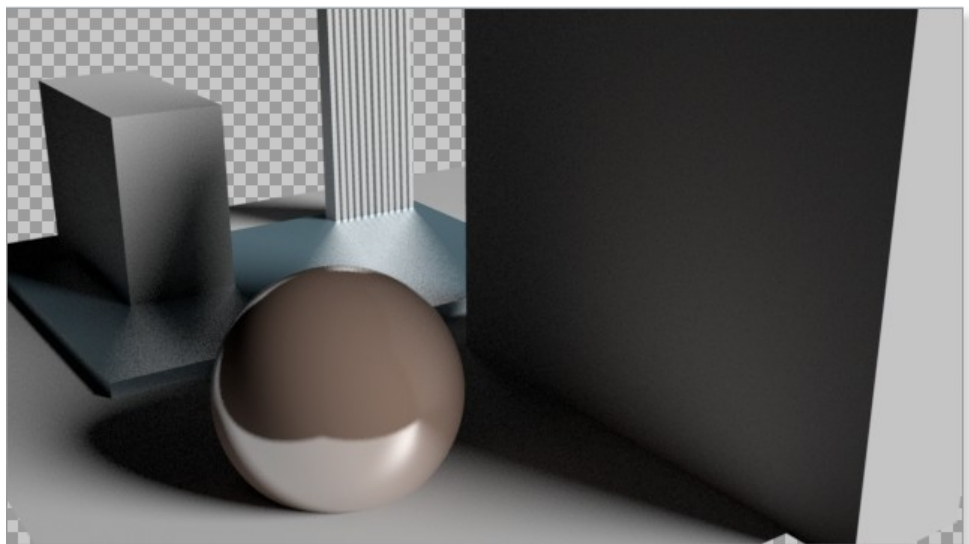
Now, our shading is good, we must set the lighting quality. Select back raytracer and disable the *Previz Mode*. We still see a strong noise, but now we know it only comes from lighting.



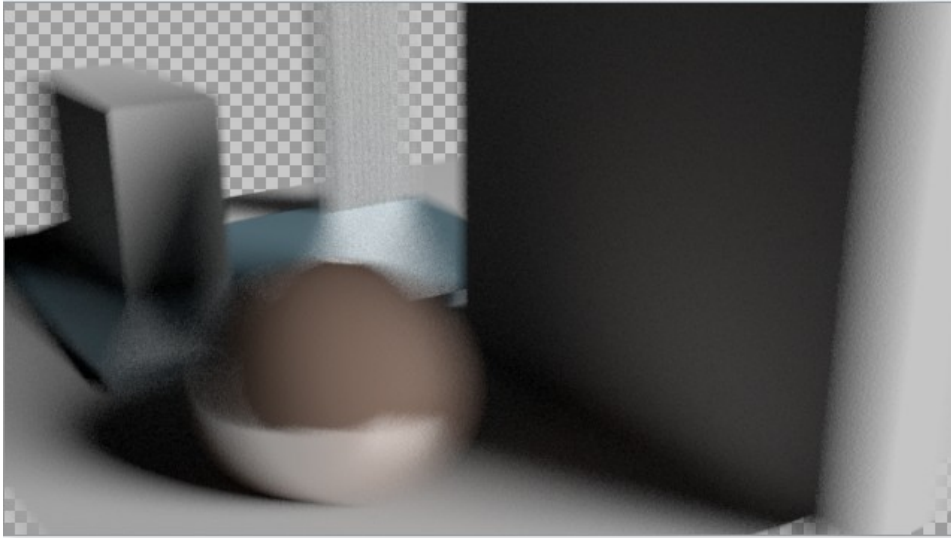
Select `gi_monte_carlo` and disable *Affect Diffuse* to temporary disable global illumination. It will be easier to set the direct light quality. Select `direct_light` and raise the quality until the noise disappear. A value of 14 should be fine. Select now `gi_monte_carlo`, enable *Affect Diffuse*. Raise *Quality* to 24 to get rid of the noise.



We still have lots of jaggies because we didn't set up the geometric anti-aliasing. The good thing is, it will almost not change the rendering time. Select raytracer and set *AntiAliasing Samples* to 4. The render time goes from 23 to 25 sec and the image is much smoother. You can get a perfect result using a quality of 6 and a *Blackman Harris* filter.

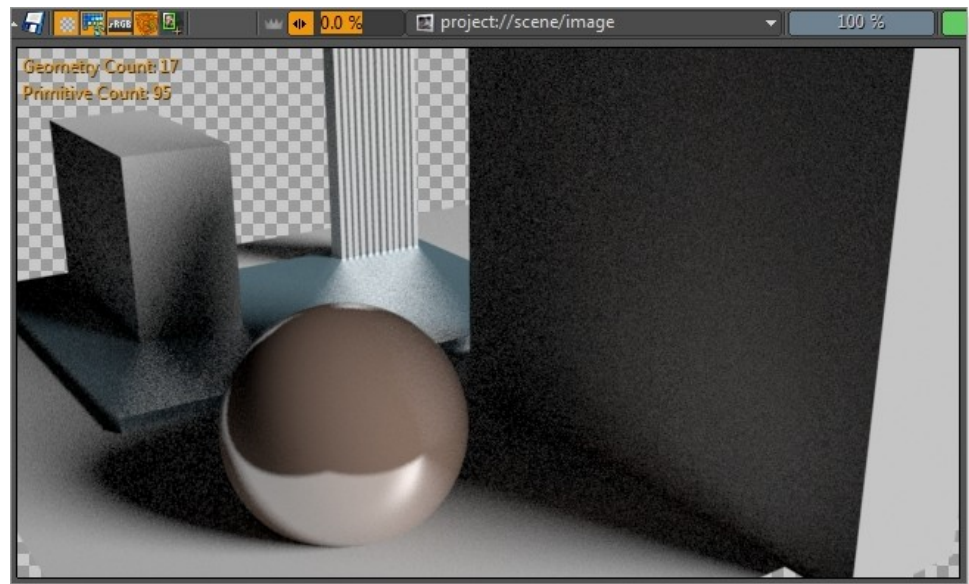


Now, if you want motion blur, just turn on *Enable Motion Blur* in raytracer. Again, you should get almost the same render time.



There is a last way to globally refine the quality of your render. Select raytracer, look for the *Shading Oversampling* attribute. If you set it to 100%, it multiplies all the shading rays by the number of anti-aliasing samples. For example, for one pixel with a soft reflection set to 10, and an antialiasing set to 6, you will get $10 \times 10 \times 6 \times 6 = 3600$ rays (like in most raytracers). Intermediate values will multiply a to fraction of them. Usually you should not use this attribute, keeping it down to 0 and set the appropriate shading quality in materials and lights instead.

Now, how to quickly adjust your lighting without waiting for the final quality, and without breaking all your settings? You just have to use the quality attribute on top of the *Image View*. Turn it down to 0% for example, and you will get a very fast noisy render. In fact, this attribute is the same found in the image object (*Sampling Quality* attribute).



You can go further with a lower anti-aliasing in the raytracer, if you need more speed.

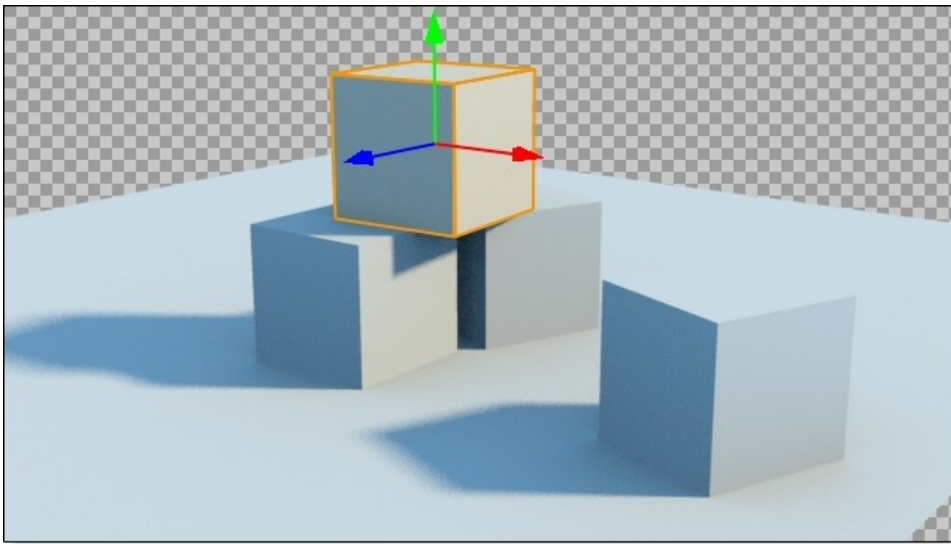
Note

To get the actual number of samples, just multiply the quality by itself. For example, 8 is 64 samples (8x8)

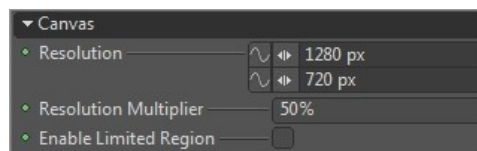
Camera Mapping Basics

Here is a mini tutorial covering camera mapping basics. **This tutorial requires the use of an external 2D Paint Application that can be any of your favorite application.** Before proceeding to this tutorial, please load the project file `content/tutorials/projects/basics/camera_mapping_basics.project`

Create a *Perspective Advanced* camera (**Create > Camera > Perspective Advanced**) for the 3D layer of the image. For camera mapping, it is very important not to use the basic *Perspective* camera). Rename camera to `render_camera`, and frame boxes in the image view like this:



Select the image and set *Resolution* to 1280x720. Our image ratio is then $1280/720 = 1.777$



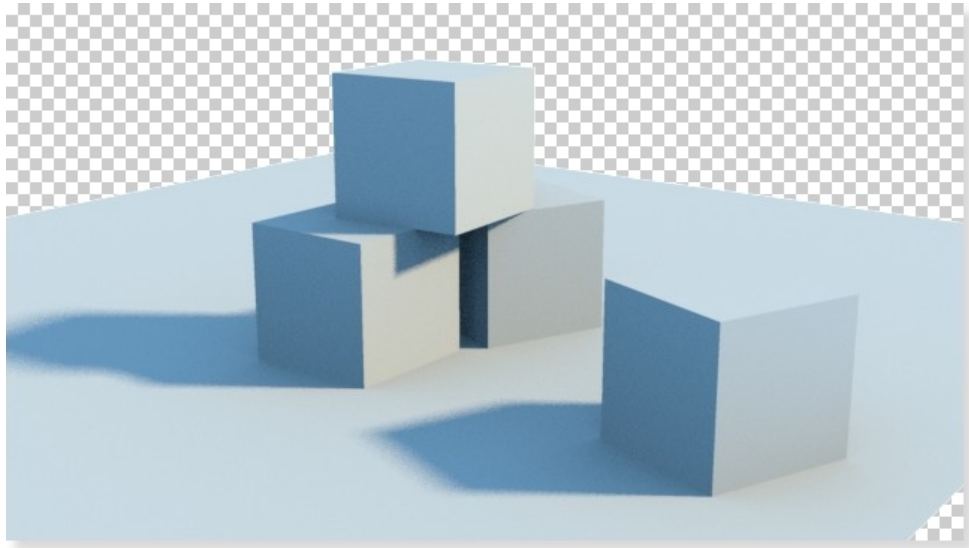
Now, select *render_camera*. In the Attribute Editor you will see an *Horizontal* and *Vertical Aperture*. They define the projection plane of the camera and they must match our image ratio. Depending on the *Fit* mode, it will distort or crop our render if camera and image ratio are different.

Divide *Horizontal Aperture* by your image ratio ($3.6 \text{ cm} / 1.777 = 2.02 \text{ cm}$) and enter the value (2.02 cm) in *Vertical Aperture*.

Note

For camera mapping, it is critical to define image ratio in the camera used for the projection, not in the image itself. If not, all your camera maps would be screwed, if you modify your rendered image ratio. This is why you should always use Advanced Camera for camera projections. You will notice a change in the camera projection plane in the 3d view.

Save the render (using the little disk icon of the image view) to `camap_renderguide.png`, and load it on your favorite image paint/editing program.



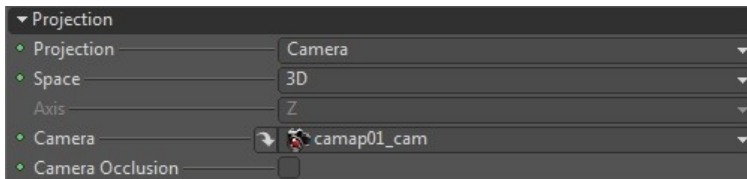
Once loaded, do all your paint work on your render. Typically, create an overlay layer set to multiply. When finished, keep only your painted layer, flatten your image and save it to `camap_paint.png`



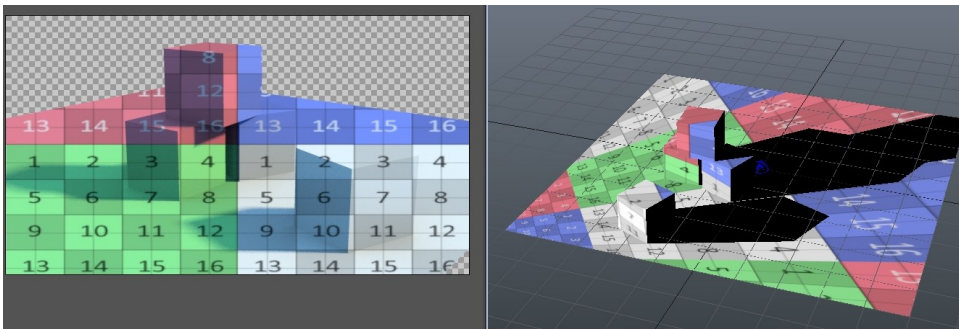
Resulting painted image

Copy/paste `render_camera` and rename the copy `camap01_cam`.
Select the `camap_mat` material (it is already applied to all objects in the current scene), and create a *Map File* (**Create > Texture > Map File**) and plug it in the *Diffuse* slot of our material.

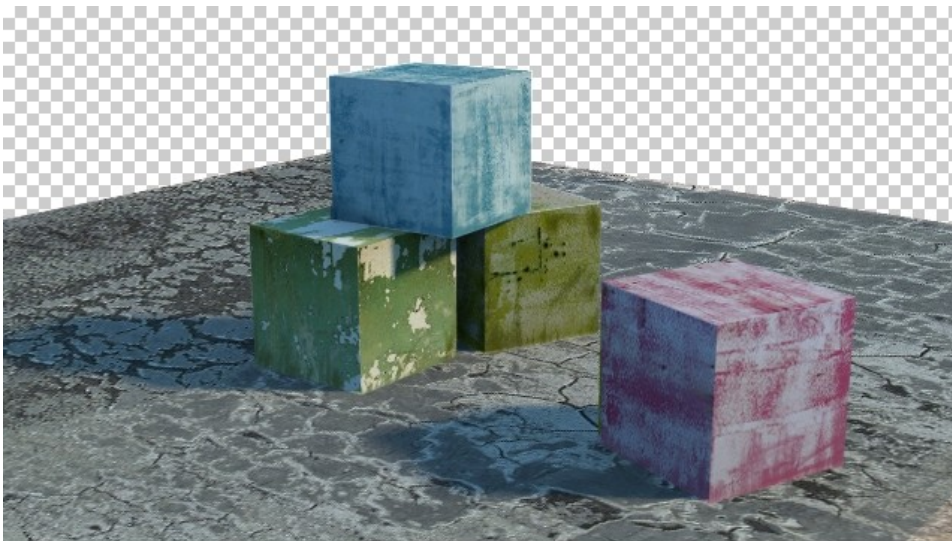
Rename it to `camap01_diffuse` and set its *Projection* attribute to *Camera* and *Camera* to `camap01_cam`. Set the 3d View to *Previz* mode and orbit around your objects... you will see the camera mapping passing through your objects.



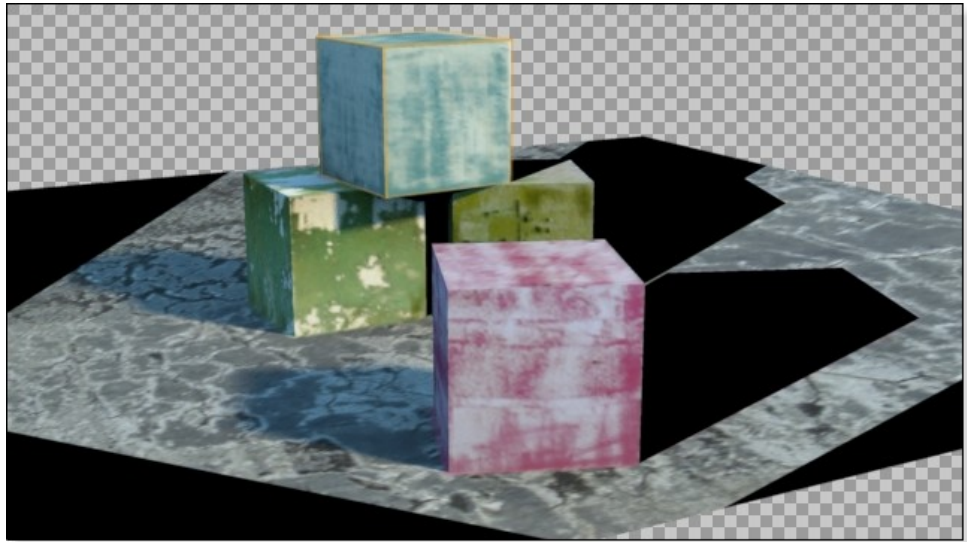
Activate the *Camera Occlusion* attribute and now, all occluded parts becomes black.



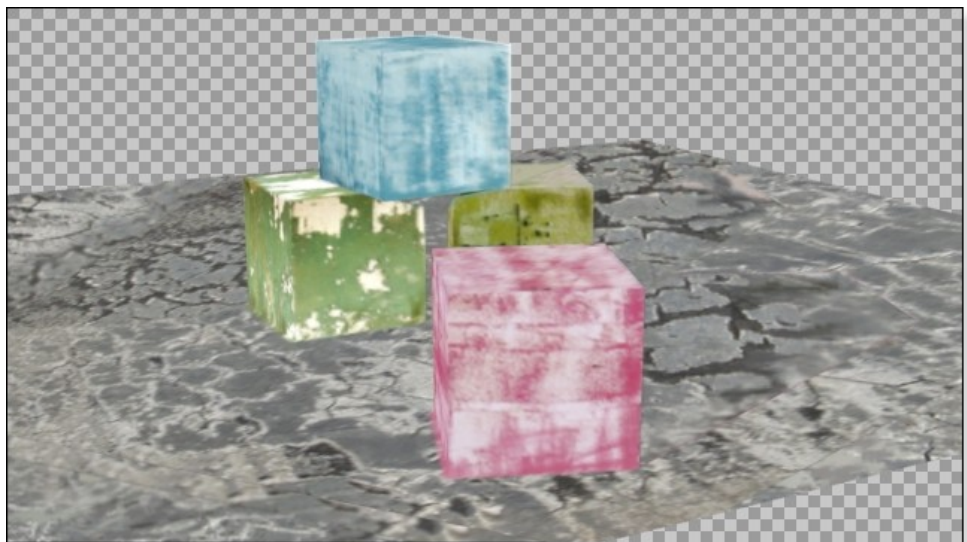
Now set *Filename* attribute to `camap_paint.png` file path. Here is what you should see in the Image View:



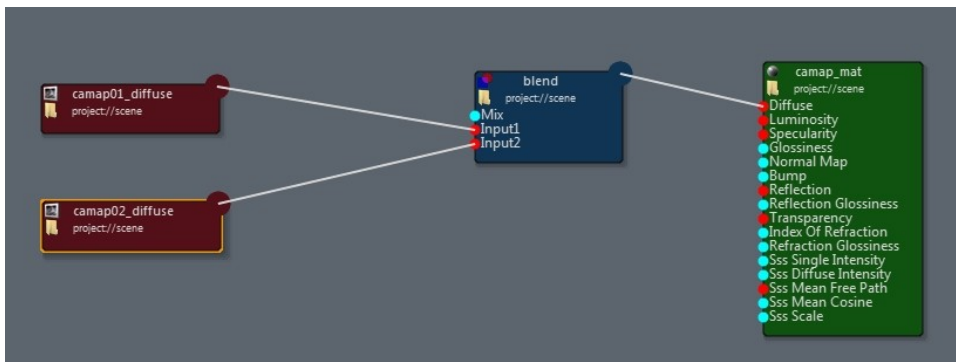
Now, select `render_camera` and create a keyframe (hit enter twice). Move to frame 50 and orbit a little in your Image View. Create another keyframe.



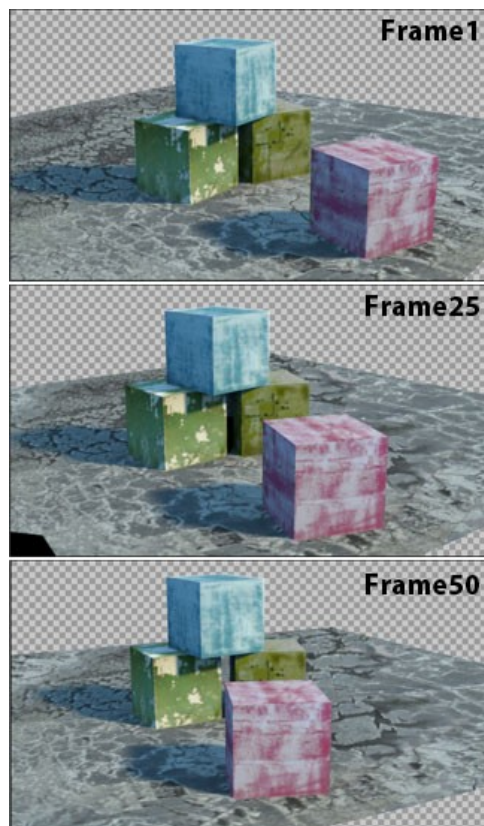
Save the render to `camap02_renderguide.png` and open it in your paint program. In your paint program, create a new layer and fill the black areas. Once you've finished your work, save the flattened result to `camap02_paint.png`.



Back to Clarisse, select `render_camera`, copy/paste it and rename the copy to `camap02_cam`. Remove all animation to keep only frame 50, by right clicking in the *Translate/Rotate/Scale* attributes and selecting **Remove F-curve** in the popup menu. Select `camap_mat` and open up the material editor. Insert a *Blend* texture in-between `camap01_diffuse` and the *Diffuse* input. Copy/paste `camap01_diffuse`, rename it to `camap02_diffuse` and connect it to `blend input2`.



Select `camap02_diffuse`, set its *Camera* to `camap02_cam` and its *Filename* to `camap02_paint.png`. Now, if you scrub the timeline, you should see a complete mapping with almost no blacks zones.. (you can correct the last ones using a third camera, or any other mapping techniques... (generic tiled maps work pretty well for small zones and zones that are difficult to access)).



UDIM Texture Basics

Clarisse provides a UDIM texture specially designed to handle The Foundry - Mari's UDIM texture system. To use it, simply fill the *Filename* attribute, using

the \$UDIM keyword to specify where the texture index is to be found.

Let's say you've created for example 6 texture maps:

```
paint.1001.tx  
paint.1002.tx  
paint.1003.tx  
paint.1011.tx  
paint.1012.tx  
paint.1013.tx
```

In order to use those textures, you'll need to set UDIM texture *Filename* attribute to: `paint.$UDIM.tx`

UDIM texture is optimized to work with tiled files. Using large files that are untiled may greatly slow down rendering times.

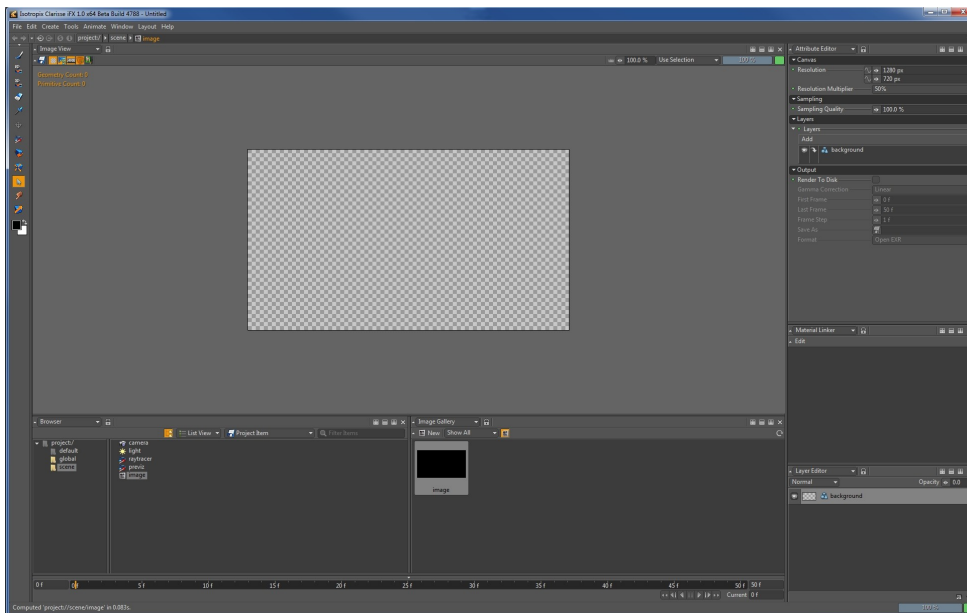
Please also note that we use Mari's file indexing convention. The index of the file is then equal to $\text{index} = 1000 + 10 * V + U + 1$ where U and V are floored values of the texture coordinates.

Clarisse 101

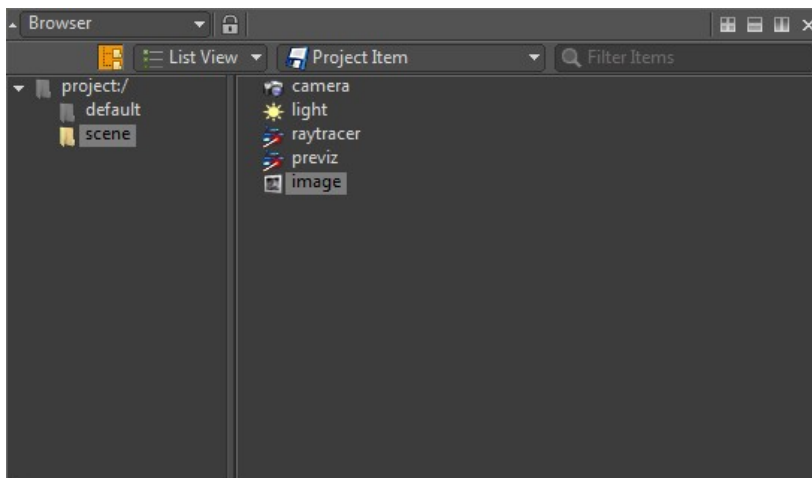
In this tutorial we will discover how to import data from other applications, shade and light a simple scene.

Discovering the User Interface

Launch a new Clarisse. In the center of the user interface, you can see a checkerboard which is the empty background of the default image coming with each new project.



In the left bottom of the interface, you'll find a widget very similar to a file browser. The Browser is actually a project browser designed to explore and select the content of your project.

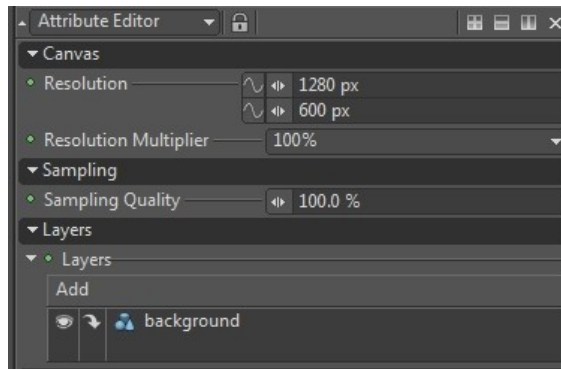


It shows folders like icons in the left part. Those item are called Context in Clarisse and used to organize your items. Items are displayed in the right part.

Editing your first object

Our image is one of these items. You need to know that in Clarisse everything is an object, items, contexts to even user interface elements. Any objects can have attributes and attributes are displayed in the Attribute Editor.

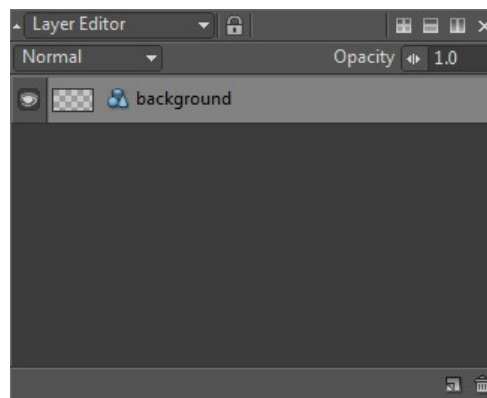
Now let's select the image in the Browser. As the image is now selected, we can see all of its attributes in the Attribute Editor widget, in the top right of the user interface. Let's modify its resolution to 1200x600, and change the resolution multiplier to 100%.



You should see the result of the image object attribute modifications in the image view: the image canvas is now bigger.

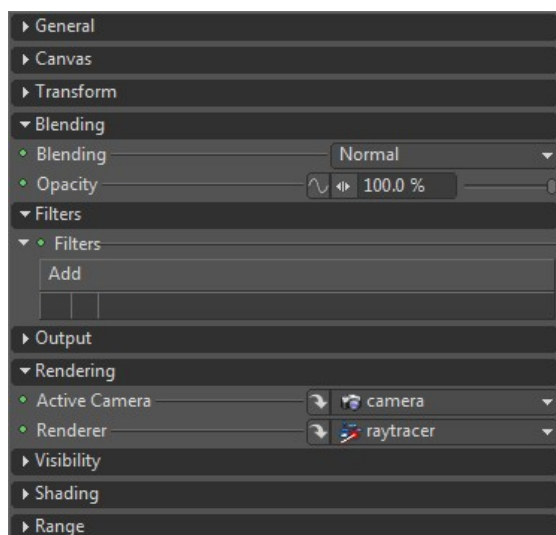
The Layer Editor

Now we have defined our basics images attributes, let's define some content... You see in the bottom right of the user interface, a widget called Layer Editor.



In Clarisse, images are made of layers. We will see later on that we can use different kinds of layers. But for now, we will focus on one of the most important ones: the 3D layer. Inside the default image that comes with the default project, there is one that is already setup. This layer, named background is what we call an embedded object. You can't see it directly in the browser, because it is embedded inside the image object. However embedded objects are still like any other objects and are editable in the Attribute Editor.

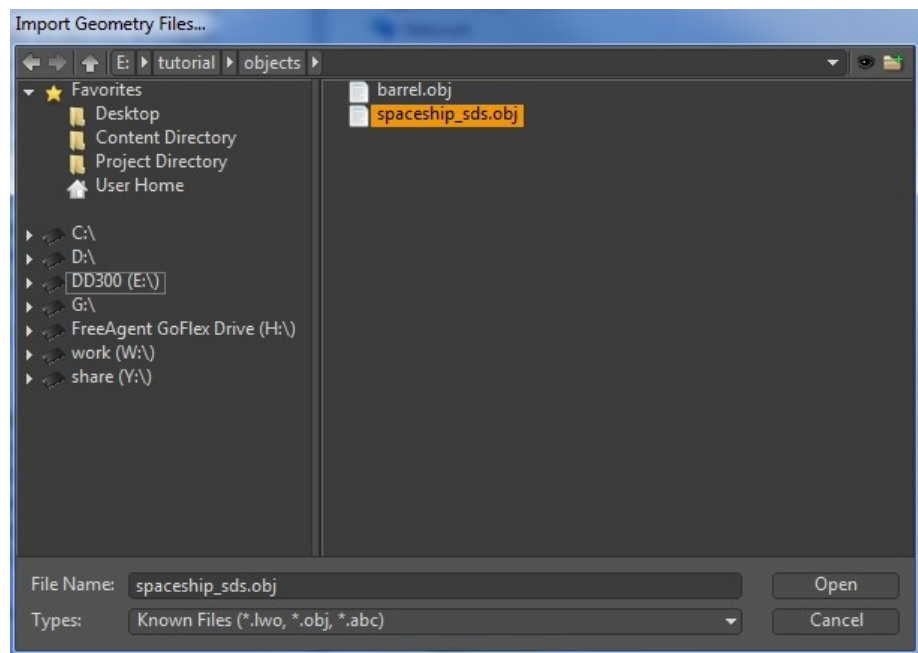
Click on the background layer in the Layer editor to select it. Layer attributes are now displayed in the Attribute Editor: blending mode and opacity, like you would find in a 2D application, a filter list to apply 2D filters... a camera, and a renderer. Notice how attributes are sorted into categories. These ones are displayed through horizontal tabs and can be minimized.



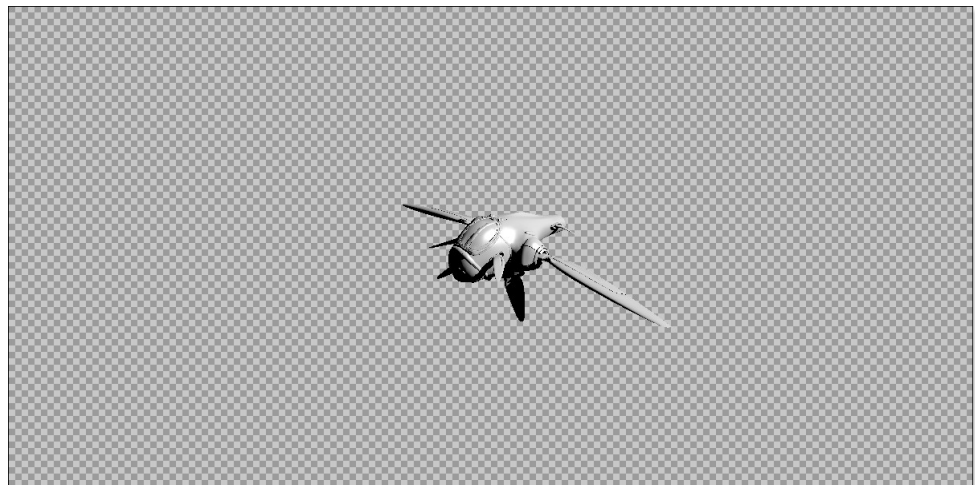
The 3D layer allows you to define what is called in other packages a 3D scene. However in Clarisse, the 3D scene is one of many different kind of layers composing an image. Please note that any images including the ones that reference 3D layer can be used as texture maps.

Importing geometry

We are now going to import a geometry in our project. Go to **File > Import > Geometry...** and select `content/tutorials/objects/spaceship_sds.obj` in the file browser.

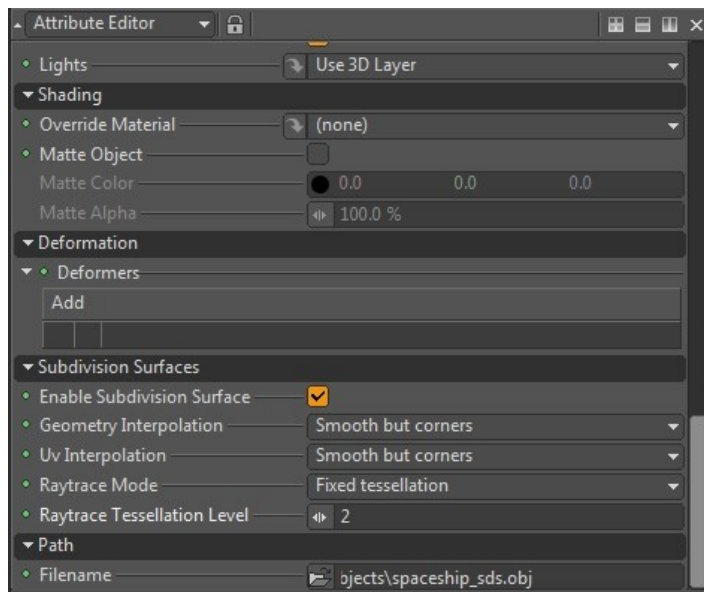


As you can see our geometry is now rendered in the Image View.

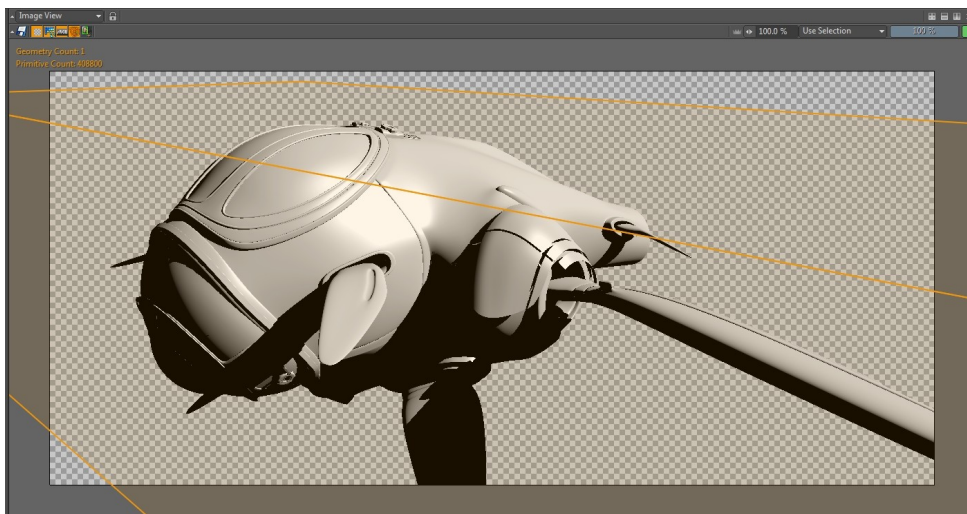


Look at *Filename* attribute, under Path category in the Attribute Editor. You'll see the file path to the Wavefront OBJ file we've just imported. In Clarisse, imported geometries are only defined by a path referencing external files. This allows you to replace your geometries with another one, or edit it in an external application without losing your work. By default path is absolute, you can make it relative to your project or to a content directory using environment variables: `$PDIR` to make it relative to the project, or the content directory using `$CDIR`.

We are now going to turn on Subdivision surfaces. Go to Subdivision Surfaces category and check *Enable Subdivision Surface* attribute. Set *Raytrace Tessellation Level* to 2 to increase the mesh smoothness.

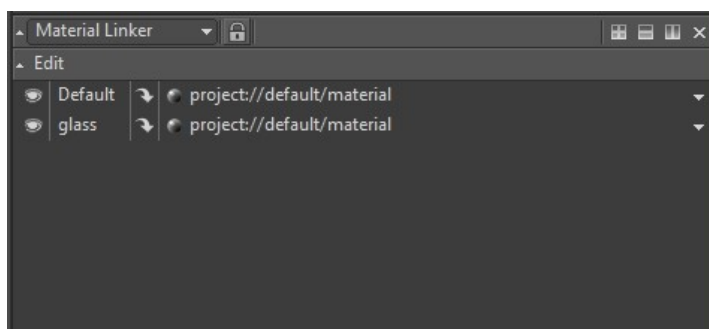


To orbit the viewpoint, press Alt and Click in the 3D View. Drag in any direction to orbit around the center of interest. To move the viewpoint, press Alt + Middle click in the 3D View. Drag in any direction to move both the viewpoint and its center of interest in the view plane. To move forward or backward, press Alt + Right click in the 3D View. Drag right or down to move forward, and left or up to move backward.



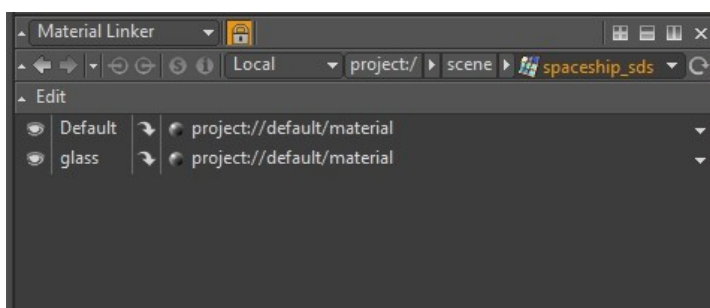
Assigning materials

Under the Attribute Editor, there's a widget called Material Linker. The Material Linker allows you to link shading groups to materials. Material links are kept even when you replace your geometry by a new one sharing shading group names.



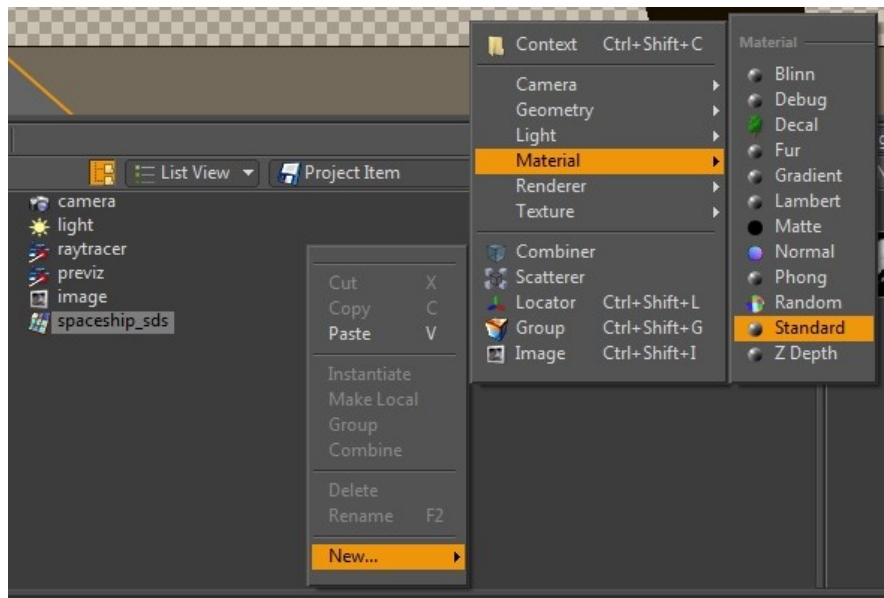
The Material Linker

By default, the Material Linker displays all the shading groups of the current selection. You can lock the Material Linker with the little padlock icon to desynchronize it from the global selection (by default every widget tracks the global selection).

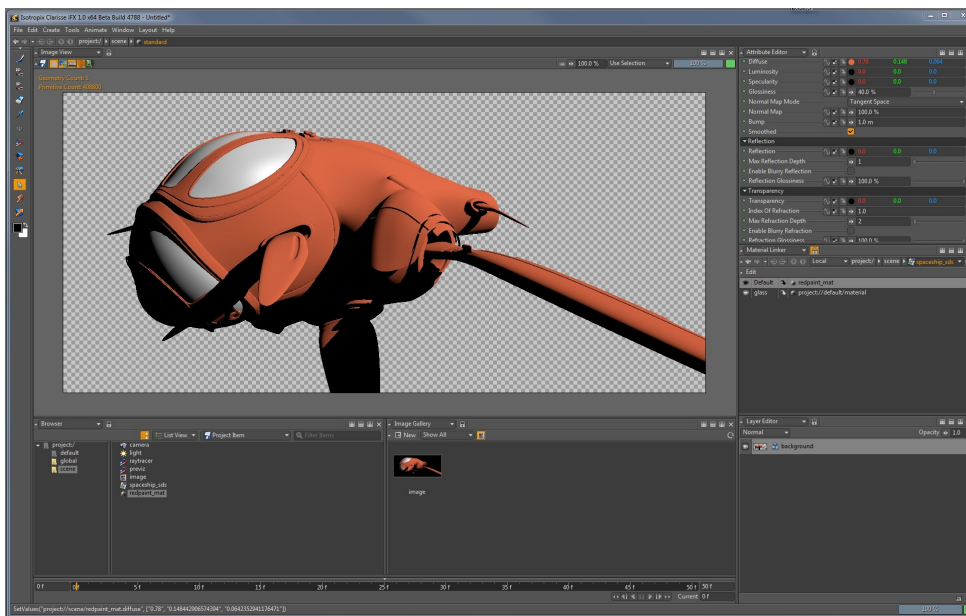


The locked Material Linker

Now, right click in the right part of the browser and select **New > Material > Standard**.



Press **F2** and rename it `redpaint_mat`. Drag it to the shading group named `Default` in the Material Linker. Your material is now assigned to the shading group. Set the color to red (0.78,0.148,0.064) in the Attribute Editor.

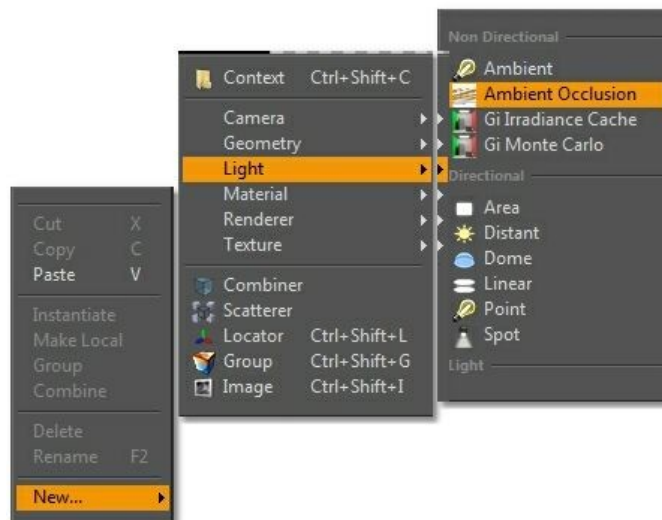


The spaceship assigned with our material

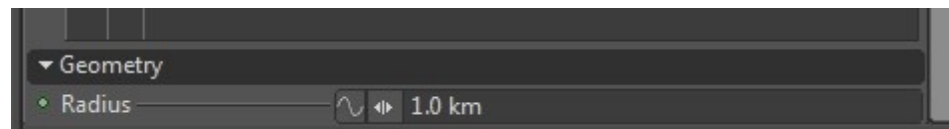
Create a second material following the same steps, and name it to `glass_mat`. This time we'll use a different method to assign the material : drag it in the image view and drop it on the spaceship cockpit. The material is now assigned. Set its diffuse to greyish color.

Setting an environment

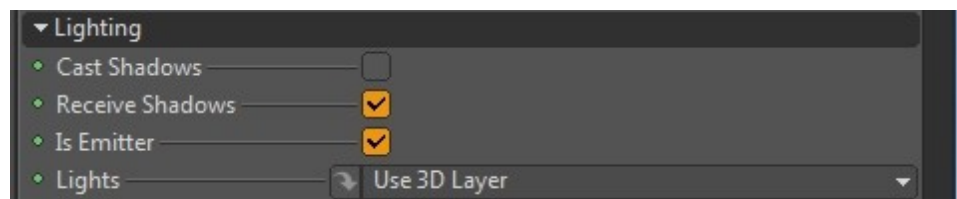
Right click in the Browser and select **New > Light > Ambient Occlusion** to create an ambient occlusion light in our scene. In the Attribute Editor, lower its intensity by setting its color to 0.3, 0.3, 0.3.



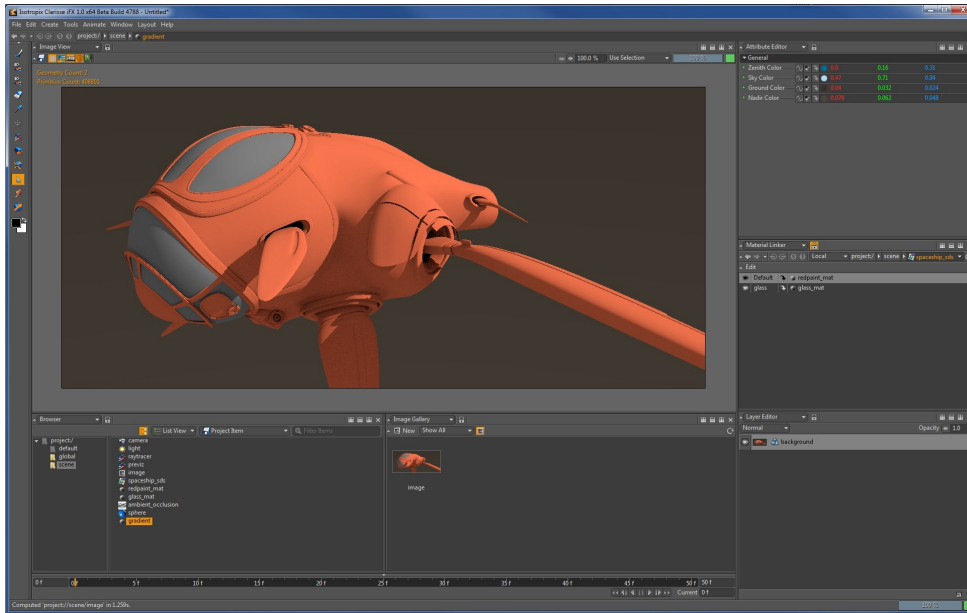
We will now setup a simple environment to setup reflection... Right click in the browser and create **New > Geometry > Sphere**. Under the geometry *Radius* attribute type 1km.



Now the whole scene is surrounded by a giant sphere occluding lights, which is why everything is now black in our image. To avoid having our sphere casting shadows go to its Lighting category and uncheck *Cast Shadows* attribute.

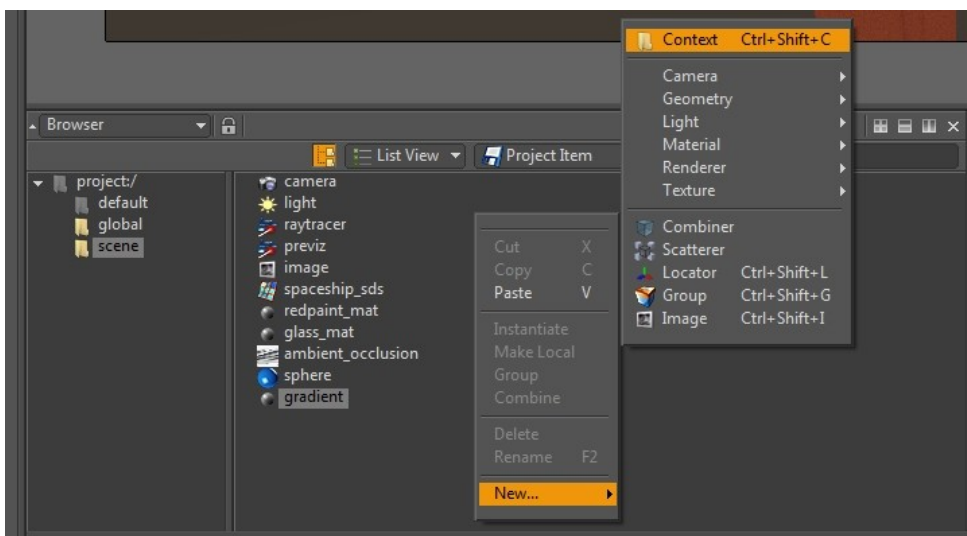


Now in the browser create a **New > Material > Gradient**, and drag and drop it on the sphere in the Image View. We have now our environment set and we're ready to continue tweaking our materials.



Organizing items

If you look at the Browser, things are a little messy. We have many different objects (lights, renderers, geometries, materials...), all around the place. We will now see how Contexts can be used to sort and organize our project the way we want. For now, just imagine Contexts as folders.

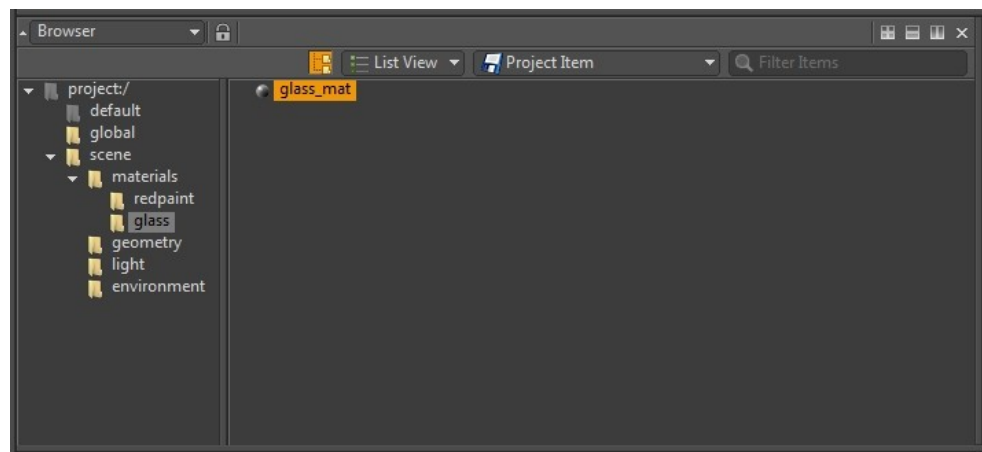


Creating our first context

Right click in the browser and create a **New... > Context**. Rename it to materials, and create another one named geometry.

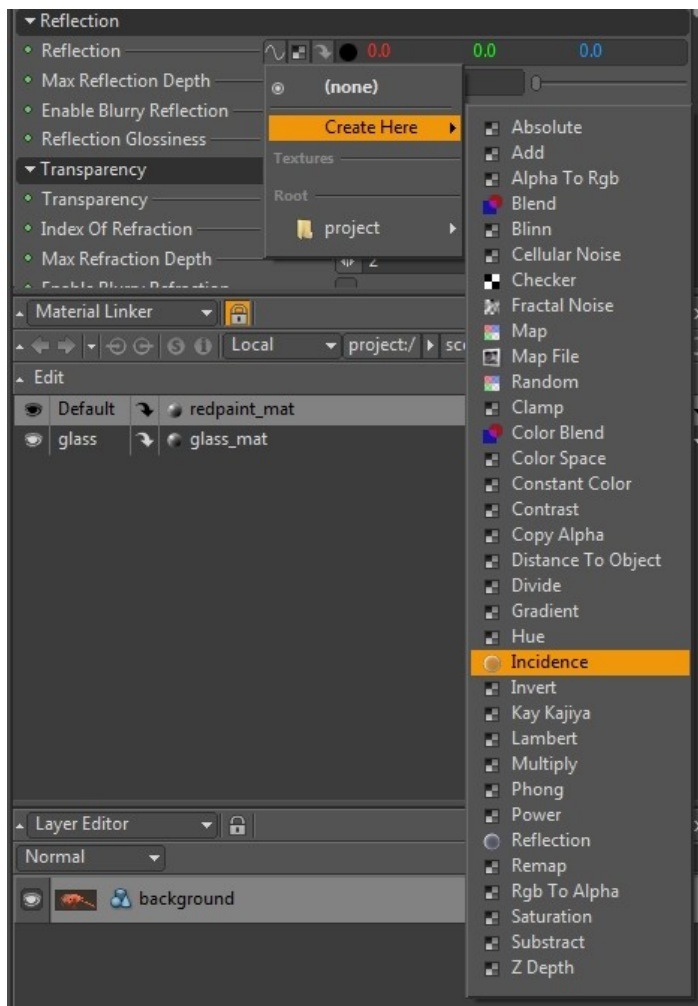
We will now move our items into our new contexts. To move items just select and drag and drop them in the context. Now move both redpaint_mat and glass_mat in materials and spaceship_sds to geometry. We can also add a light context and move our two lights inside, as well as an environment context and move sphere and gradient inside.

Inside the context material, let's create two new contexts that we will name redpaint and glass, and move corresponding materials inside.



The final context structure with the glass material

Select the glass context and select glass_mat. In the Attribute Editor, click on the checker next to *Reflection* and **Create Here > Incidence**. This popup creation menu appears only if you have set *Object Selection Mode* to *Use Popup Menu* under *User Interface* tab from Preferences panel.

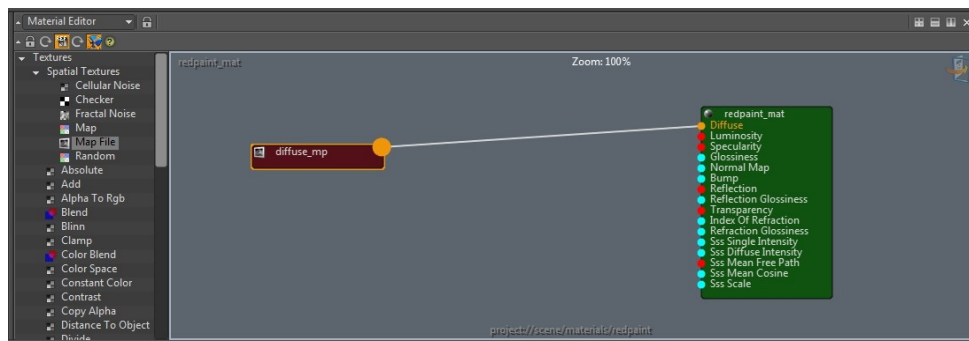


We've just created an incidence texture (two colors varying along the angle of incidence) which is connected to the *Reflection*. In our image we can now see our environment sphere reflected on the cockpit. To adjust the incidence, you can select the incidence texture by clicking the little arrow next to the checker or select it in the browser.

The Material Editor

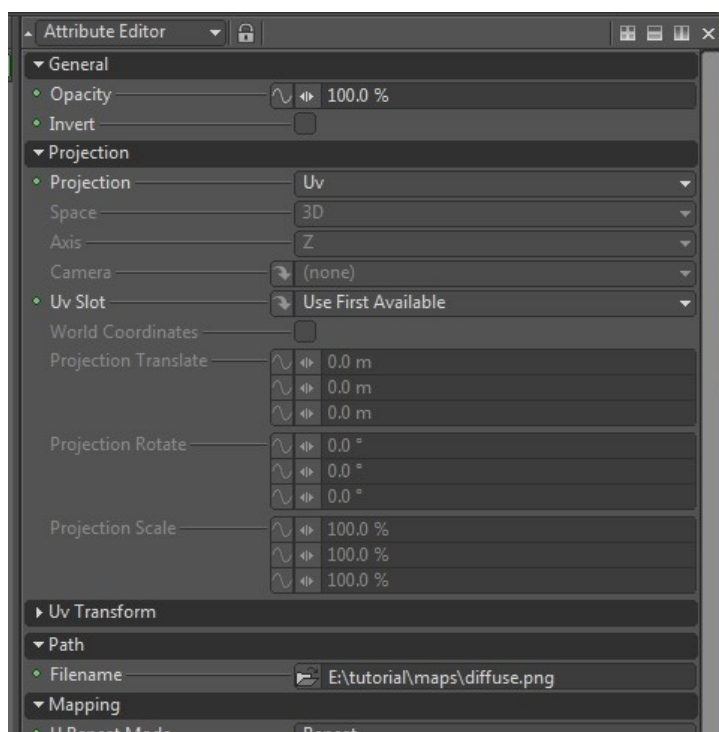
Next to the Browser widget is another called Image Gallery. This widget allows you to view thumbnails of different images that are in your project. We won't need this widget for the moment. Click on the Image Gallery title and select Material Editor instead. Drag the vertical splitbar to the left to resize the viewport. In redpaint context, and select the redpaint_mat to display the shading network in the Material Editor.

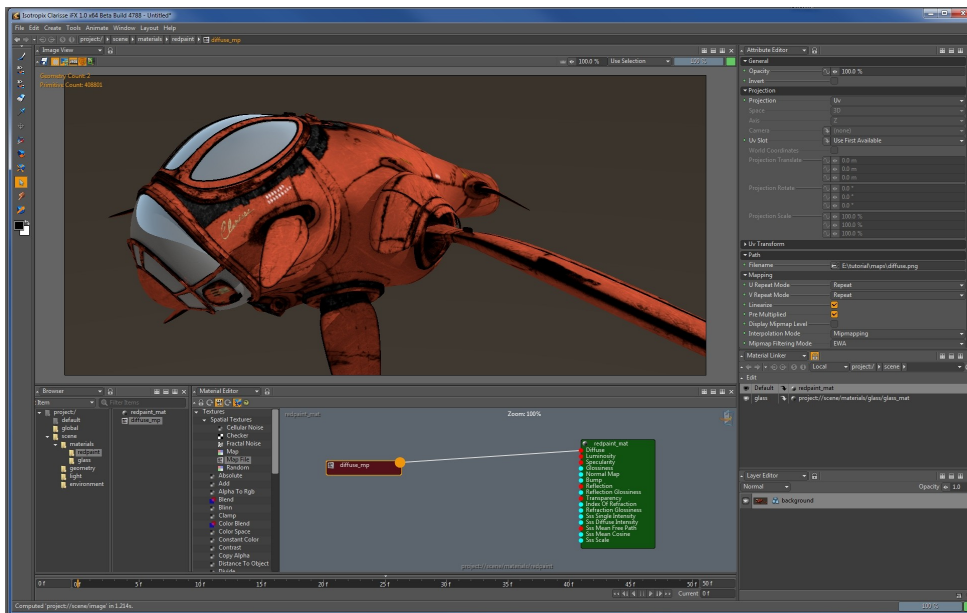
In the left tree, drag a **Texture > Spatial Textures > Map File** and drop it into the blue area. Select the newly created node and rename it `diffuse_mp`. Now connect its output (big circle shape) to the material Diffuse input.



The material editor

Make sure `diffuse_mp` is still selected to edit its *Projection* attribute to *UV*. You can notice a *Uv Slot* attribute. It allows you to select explicitly a UV Map. Set the *Filename* attribute to `content/tutorials/images/diffuse.png`.

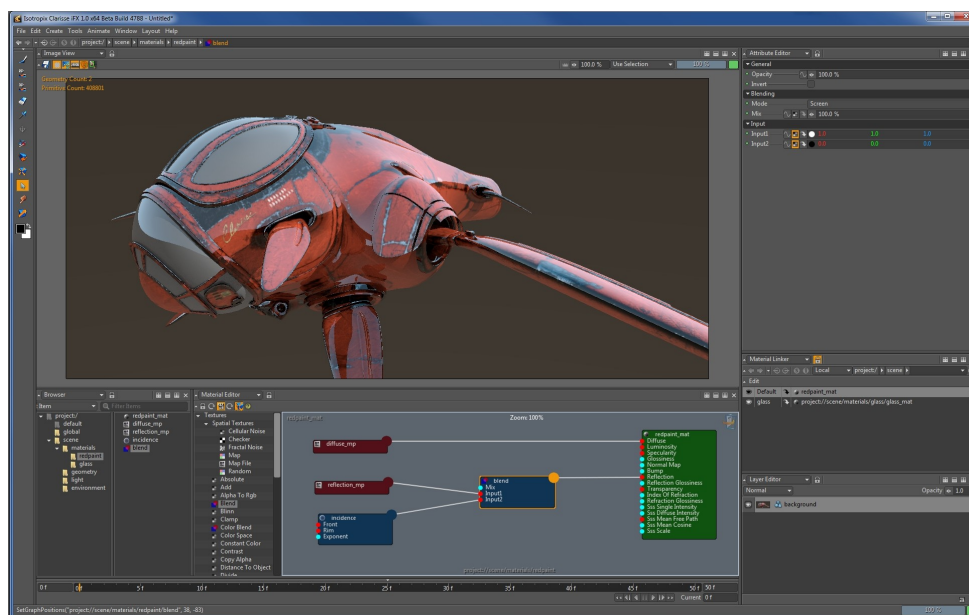




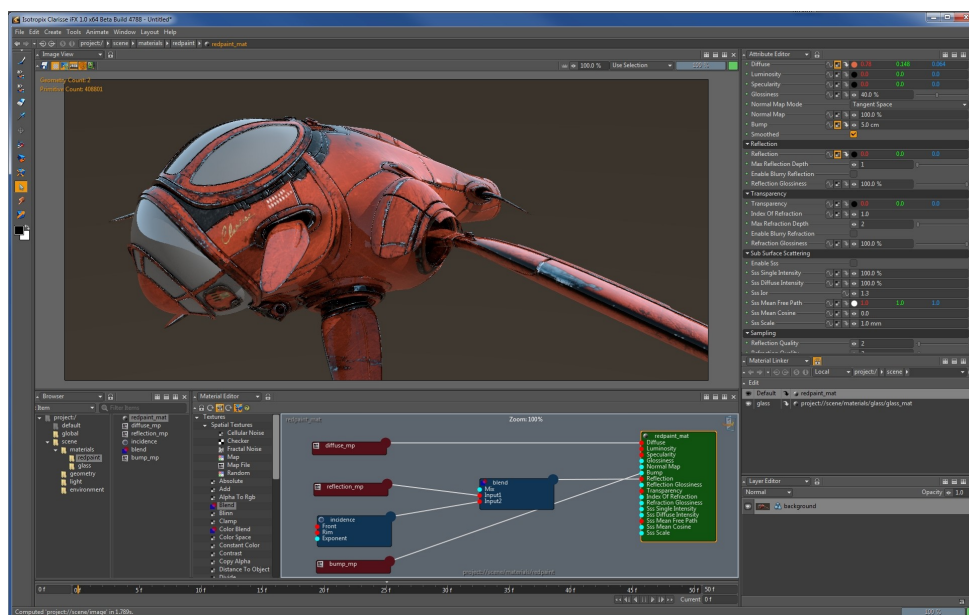
Working on materials

Now, we want to add reflection in the scratched areas to mimic as if there was metal under the paint. We do have a dedicated texture map mask to control this. Copy/Paste `diffuse_mp` and rename the copy `reflection_mp`. Change its *Filename* to `content/tutorials/images/reflection.png` and connect the output to *Reflection* input.

Now we want to add some reflective coat on top of the paint... Add an Incidence and a Blend texture. Connect `reflection_mp` and incidence to the two blend inputs (*Input1* and *Input2*) and set *Mode* to *Screen*. Connect the blend output to `redpaint_mat` *Reflection*. You now have a glossy look on your paint. You can select the incidence texture, set the *Front* color to 0.02 , 0.02, 0.02 and *Exponent* to 200% to improve the result.



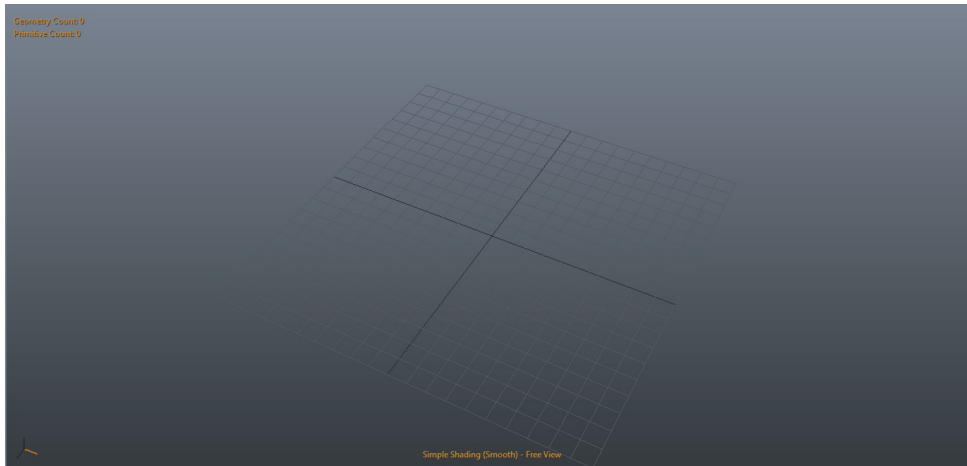
Now let's add bump mapping. Copy again `diffuse_mp` and rename the copy to `bump_mp`. Change its *Filename* to `content/tutorials/images/bump.png` and uncheck *Linearize* attribute. Linearize converts sRGB textures to Linear (but our bump texture is already in Linear color space). Select `redpaint_mat` and set *Bump* to 5 cm. Ok, we are now done with our material.



Building our ground

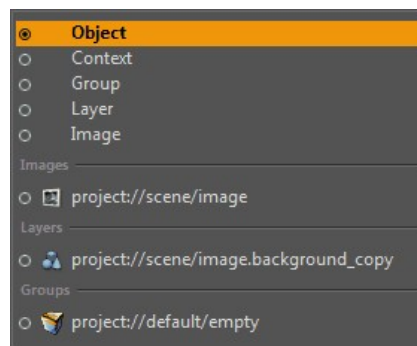
We will now add a procedural environment that we will directly model in Clarisse. We will see we are not restricted to import geometries, and how we

can build things like terrains and rocks very easily. We will also discover the 3D View, a new widget designed to edit, view and manipulate your scenes. Replace the Image View by the 3D View.



The empty 3D view

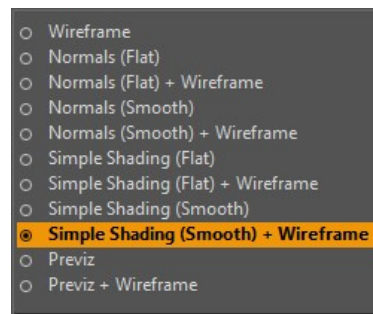
By default, the 3D view displays and fits the view to currently selected object(s). If you look at the widget top right, there is a dropdown button named *Use Selection* that controls the visibility of this widget.



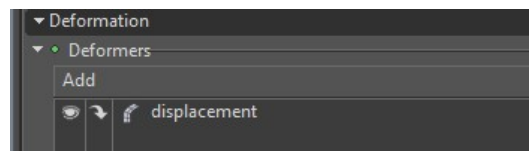
3D view visibility

You can set it to Context, to show everything in the current context and its subcontexts ; to Group to display the content of a group ; to Layer for layers and Image for images. All these modes track the current selection. Just below, you'll find a list of all existing images, layers and groups to lock the view to a particular one, without worrying about the selection. Play with the different modes and modify accordingly your selection to see how it works, then leave it on context.

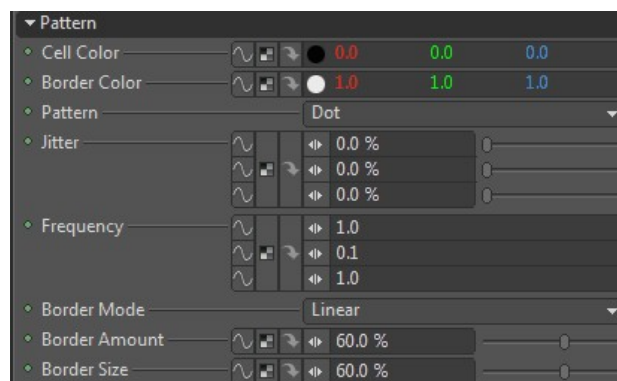
Next to this menu, you will find Control the display rendering mode, from *Wireframe* to *Previz* the latter allowing you to see the full shading (including raytraced reflections, sub surface scattering etc..). For now, set it to *Simple Shading + Wireframe*.

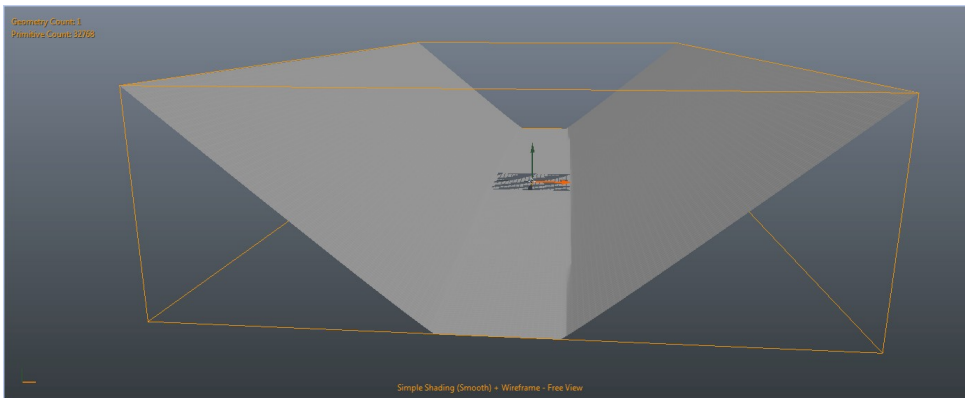


Now, inside scene context, create a new context and rename it *set*. Inside, create *New... > Geometry > Polygrid*, rename it *ground*, and set its *Scale* attribute to 10000%, 100%, 20000%. Set its *Spans* attributes to 128 and 256 (you'll need to set 256 by typing using your keyboard as the mini slider max value is 128). Our polygon grid has now 32768 quads. This will give us quite a few polygons to displace. Now in the Deformer tab, add a Displacement, and jump inside using the little arrow.



Set *Displacement Axis* to *Y*, *Scale* to 5000%, and create a Cellular Noise texture. Set its *Projection* to *Planar*, its *Axis* to *Y* and the pattern like this:

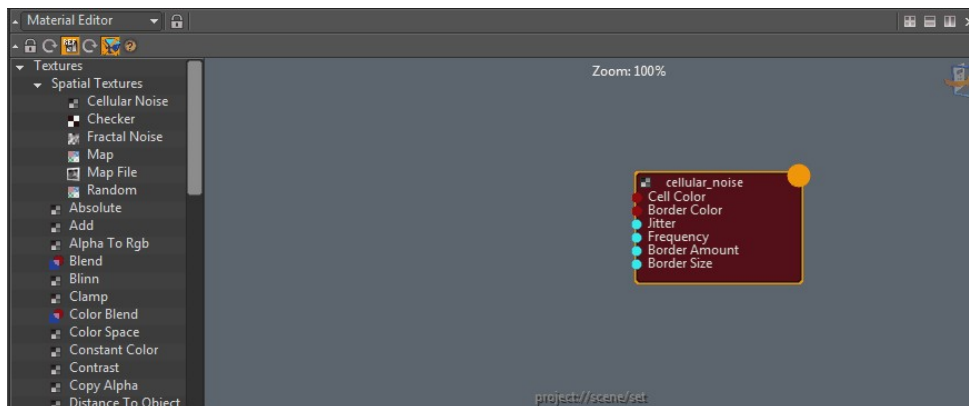




The displaced grid

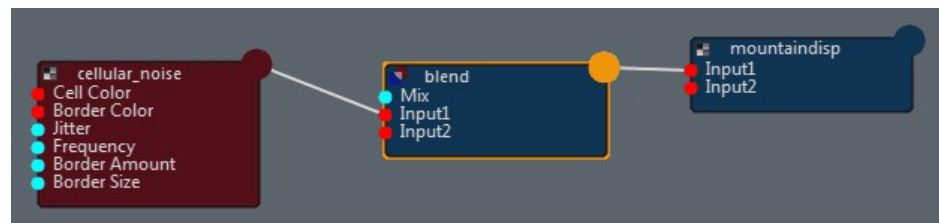
Now if you look at the Material Editor, it should still be displaying our redpaint material, or the last material you edited. Select all the nodes and drag and drop them on the little exit door icon (top left of the widget). This will not delete anything, but instead will remove our nodes from the current workspace. Remember, the editor can display any texture node, not only the material ones.

Go back to the set context. Drag and drop previously created `cellular_noise` texture, from the Browser to the Material Editor. This doesn't create a new node, it just displays it in material editor's workspace. It's really important to understand the fact that all of those nodes come from a unique scene graph. The Material Editor that is dedicated to shading displays only a small subset of your projects.

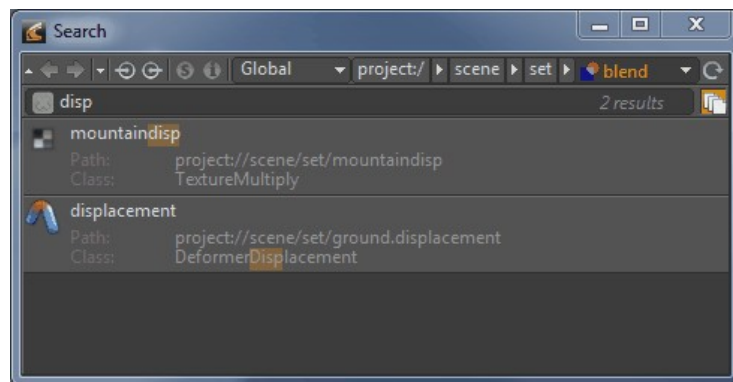


The material editor showing the cellular texture

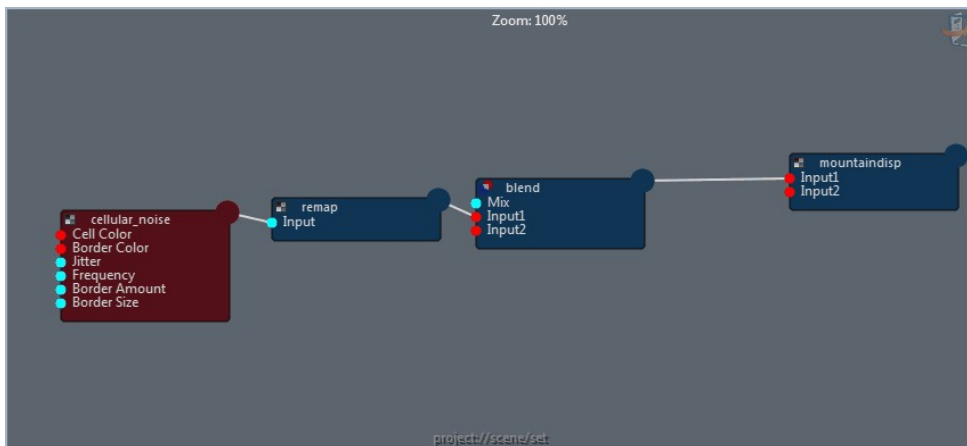
Now we need to mix our cellular texture with other noises to sculpt our procedural terrain. Add a *Multiply* node at the right of the graph, and rename it *mountaindisp*. It will be used to control our displacement amplitude. Between *cellular_noise* and *mountaindisp*, insert a *Blend* node. Set its blending *Mode* to *Add*, and connect all of the nodes like this:



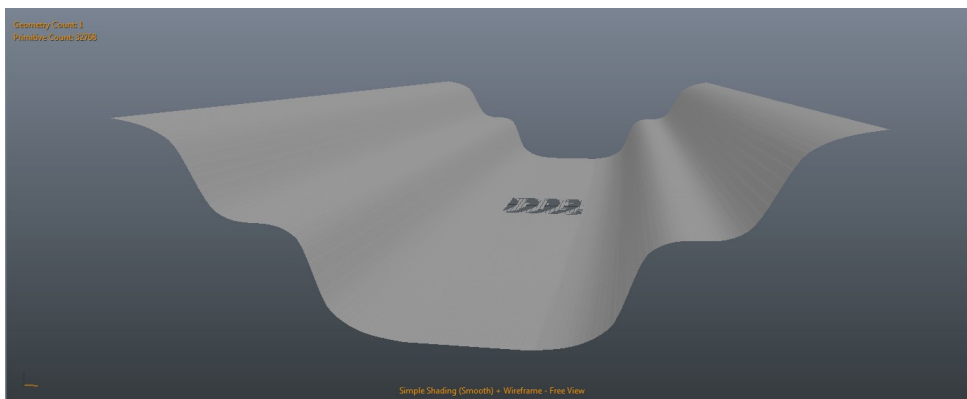
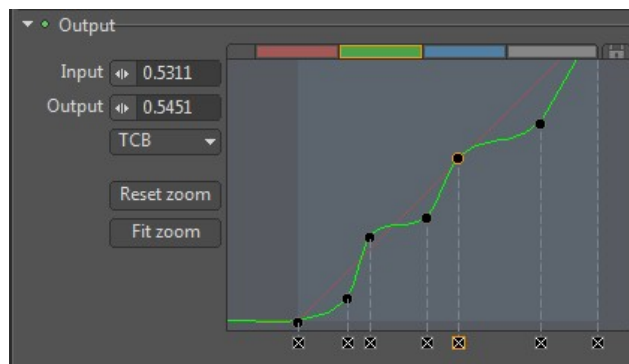
If you modify *blend* or *mountaindisp* attributes, no change occurs to *polygrid*. In fact, our displacement deformer still uses *cellular_noise* as input. To find quickly the displacement deformer, press F3 and type *disp*, then Enter.



It's now really easy to select our displacement deformer using the search widget. Select it and set its *Texture* attribute to reference *mountaindisp*. Now all the nodes connected to *mountaindisp* affect our *polygrid*. Let's play for example with *Input2* attribute of the *blend* node. This offsets vertices along the Y axis. Of course, it can be textured. For now, let's insert a *Remap* node between *cellular_noise* and *blend*. To do this, create a *Remap*, drag it over the link and release mouse button when the link turns orange. The node is automatically be inserted in between.



Select the green remap curve in the Attribute Editor, and insert a few keys using middle mouse button. Move them to a valley shape looking like this:

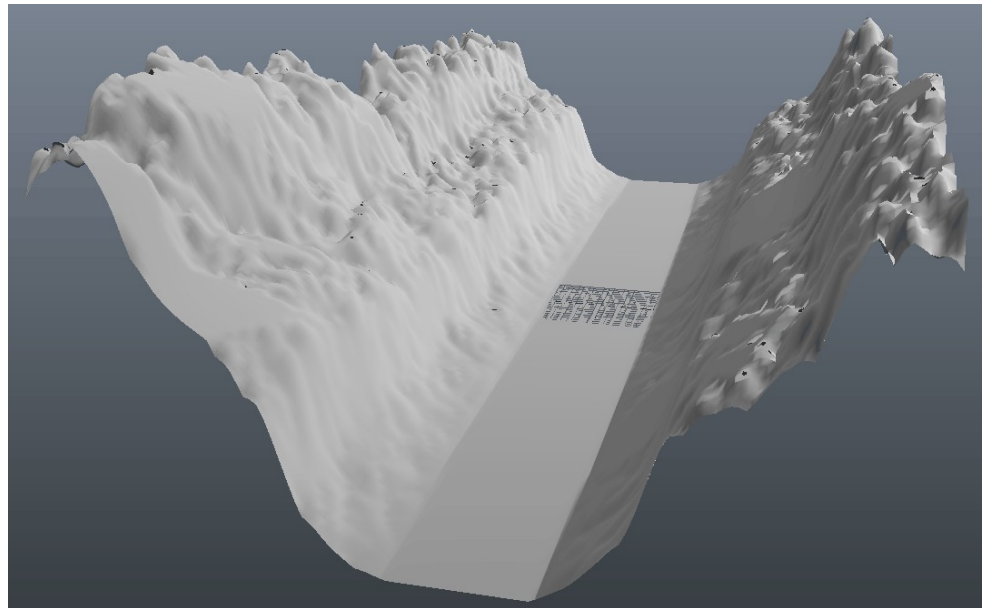


Resulting remap

Now, let's add a *Fractal Noise* texture and connect it to the blend's *Input2*. Set its *Projection* to *Planar* on Y axis. You should get some displacements on the valley now, but we would like to have it only on our hills.

To achieve this, we'll use our cellular noise to mask the center of the grid. Create and insert a multiply between the fractal noise and the blend, then connect cellular noise output to the multiply's *Input2*.

Now the noise is fading gradually. Feel free to adjust your displacement to add more detail. You can mix texture map displacement for example, using a *Map File* texture.

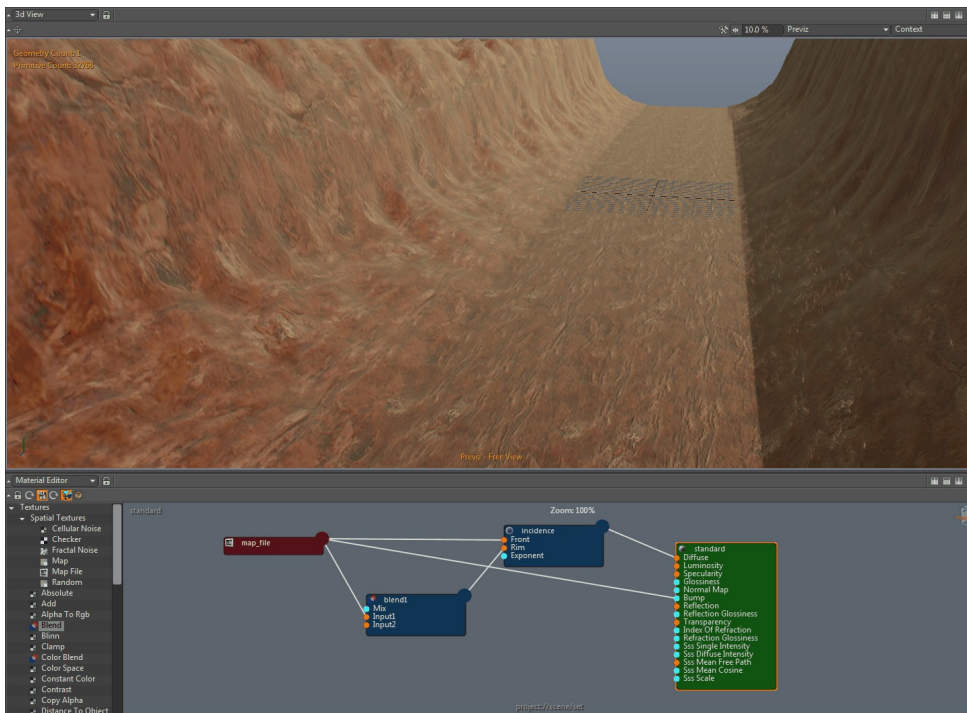


Our displaced Valley

The ground material

Set the 3D view to *Previz* and create a new standard material and rename it to *ground_mat*. Assign it to the ground shading group. In the Material Editor, add a incidence texture, set its *Exponent* to 200% and connect the node to the *Diffuse* of our material.

Create a *Map File* texture and connect it to our material *Bump* and to *Front* and *Rim* of our incidence texture. Set the projection *Axis* to *Y*, change the *Scale* to 10%, 100%, 10%, and set the *Filename* to *content/tutorials/images/rock.jpg*. Insert a blend node between the mapfile and the *Rim* incidence color. Set *Mix* to 50% and *Input2* color to 0.71, 0.622, 0.354 . This tints our rock texture with a warm tone when looking from glazing angles. Go back in the material set its *Bump* to 0.5m.



The valley with a rock material

Setting the layout

Replace the 3D View by the Image View and lock the view to project://scene/image



Now, select the Translate Item Tool (press **W** or **Tools > Translate Item**)



Translate Item Tool

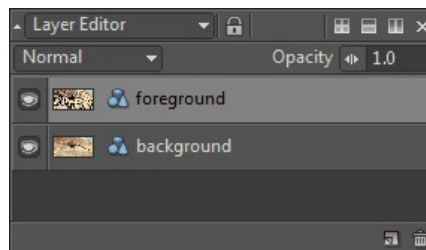
Now select `spaceship_sds` and move it a little above the ground using the gizmos displayed in the Image View. Rotate it a little using the Rotate Item Tool (press **E** or **Tools > Rotate Item**). Now select the ground, copy paste it and move it in the distance.



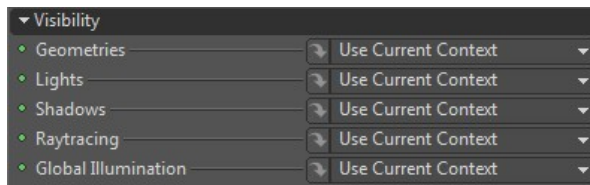
Splitting into Layers

We are now going to see how we can split our image into layers and then how to perform 2D operations on them.

In the layer editor, right click on the background layer and select **Duplicate**. Rename the newly created layer to foreground. To rename a selected layer, just press **F2**. We have now two identical 3D layers, each one having the same geometries and lights.



We're now going to differentiate background from foreground by hiding unwanted elements from each layers. First, let's hide the foreground one by unchecking its eye shaped icon. Now, select the background layer and lock the Attribute Editor. Scroll down the Attribute Editor to find the visibility category.



Each of those attributes can reference a group to define what's visible to the renderer: *Geometries* defines what's visible to the camera, *Lights* specifies which lights are used, *Shadows* which objects cast shadows, *Raytracing* what's reflected and refracted and finally *Global Illumination* what objects emitting GI rays are.

By default, each one is set to *Use Current Context*. When in this mode, attributes act as if they were referencing a hidden group that is automatically referencing the items located in current's image context and sub contexts. In our case, here, every single objects and lights inside the Scene context and sub contexts are used for each visibility attributes. For example, if you moved `project://scene/geometry/spaceship_sds` to the project root `project://`, it would be totally invisible from all of our layers if they had their visibility attributes set to *Use Current Context*. Try it if you want, you can always undo the changes before continuing this tutorial. This behavior is very important to understand as it's one of the most fundamental aspect of Clarisse workflow.

Now, select the two ground objects (ground and ground1) in the browser. Drag and drop them on *Geometries* attribute. This will automatically create and reference a new group referencing the objects we've dragged.

As you can see, the spaceship is now invisible to the camera. Indeed, the group is only referencing ground and ground1. However, if you look just bellow the spaceship's position, we can still see its shadow. Indeed, *Shadows* attribute is still set to *Use Current Context* and all our objects are still casting shadows accordingly. Let's take a good habit and rename our new group to `background_geometries`.

Let's hide the background layer and select and unhide now the foreground one. In this layer, we will need to hide everything but the spaceship. This time, let's see another way of creating our group. Right click in the browser and create **New... > Group**. Rename it to `foreground_geometries`. Resync the Attribute Editor by clicking of the refresh button to display attributes of our new group. Now drag and drop `spaceship_sds` in *References* list.

Select the foreground layer, resync the Attribute Editor and click on the *Geometries* attribute to select foreground_geometries. That's it, unhide the background layer to see the composite.



Rendering our work

Now that we've finished our work in Clarisse, there are chances you want to render your work, to be viewed or used in other applications. You can export the whole composite result or each individual layers. To quickly save your image, you can click on the little disk icon in the image view. This may be fine when working on stills, but definitely not handy when you work on sequences. A better way is to setup output attributes directly in layers and images. Once they are defined, you won't have to worry about tracking images and layers output paths.

Setup Image Output

Select your image or a layer. In Output category in the Attribute Editor check *Render To Disk*. Set *Gamma Correction* to Linear if you use OpenEXR as *Format*, otherwise choose sRGB. Set the *First Frame* and Last Frame to 0. Finally set your *Filename* without extension or padding as they'll be added automatically.

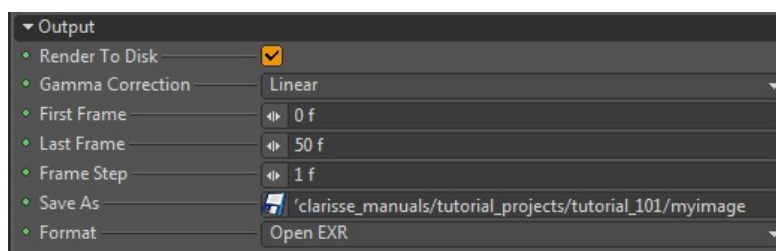
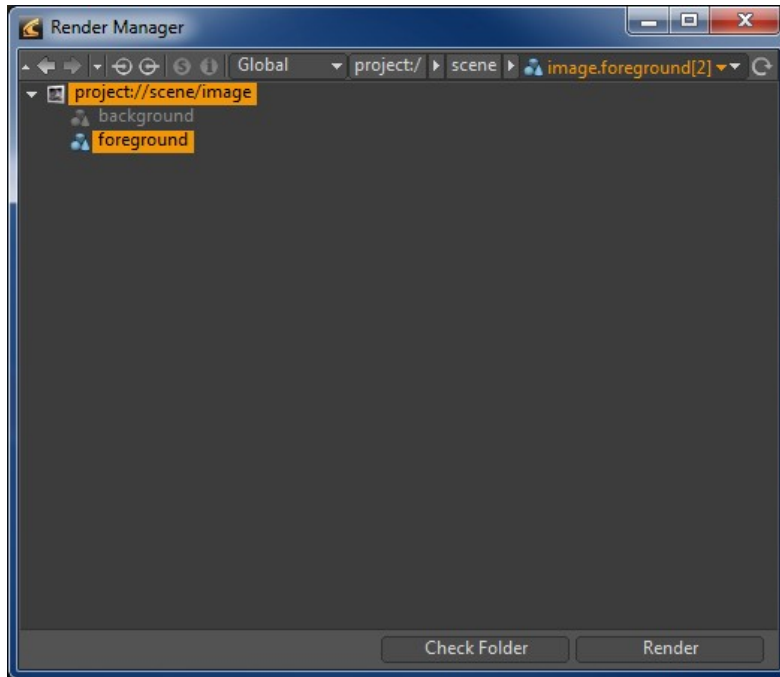


Image output attributes

Render Images

To render final images, open **Window > Render Manager** and press the Render button. For more information please refer to Using the Render Manager.



Clarisse 103: Advanced Shadows and Reflections

This tutorial covers one of the unique features only available in Clarisse: the ability of specifying exactly what reflections or lights "see". It assumes you have already familiarized yourself with Clarisse Basics and with Clarisse object instancing and localization. The resulting project of this tutorial is found in your project directory under the name `content/tutorials/projects/clarisse_103_reflections.project`.

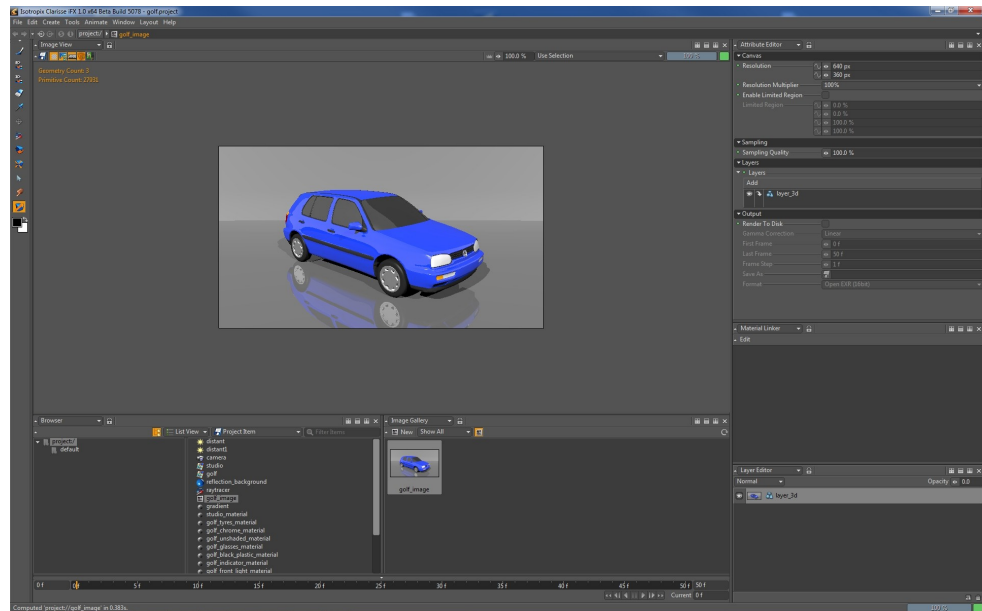
Specifying what is reflected on a per-material-basis, what casts shadows on a per-light-basis or what affects global illumination can be a nightmare with 3D Animation packages. Clarisse makes it so easy that it's like a walk in the park.

Creating group

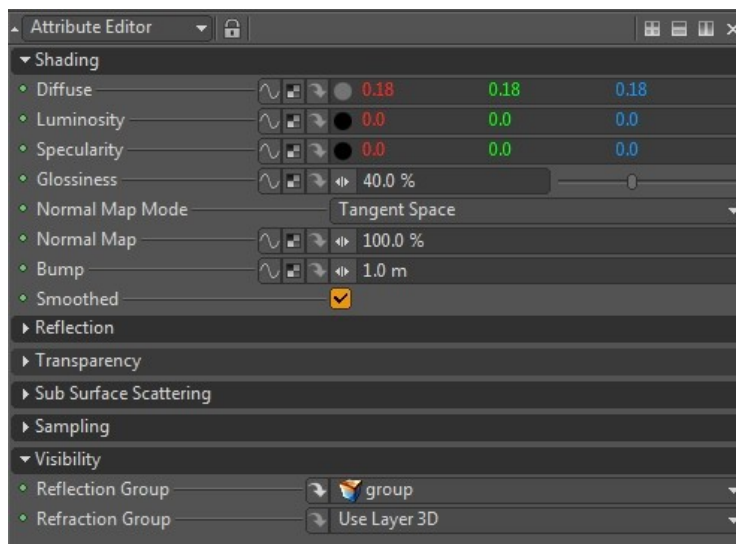
You may have noticed lights and standard materials have, respectively, attributes named *Geometry Group* and *Reflection Group / Refraction Group*. By default, those attributes are set to *Use Layer 3D* which means they use geometries referenced in the 3D Layer and seen by its camera.

These two attributes look for a group of Scene Objects as input. When you explicitly specify a group to a material or a light, the material or the light use specified geometries instead of using ones referenced in the 3D layer.

Open the project content/tutorials/projects/golf.project. Once the project is loaded, you should get something like this:



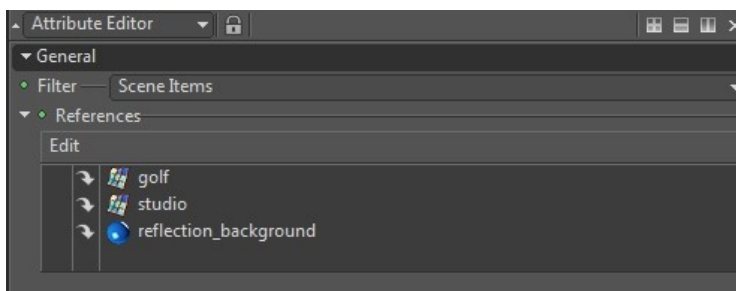
What we are going to do is explicitly specify what reflects in the `studio_material`. In order to explicitly set the reflection, we first need to create a group. To create a group, you can go to the main menu **Create > Group**, use **Ctrl + Shift + G**, or click on the *Reflection Group* attribute drop down of the `studio_material` and select **Create > Group**. Make sure *Reflection Group* attribute of `studio_material` references the newly created group.



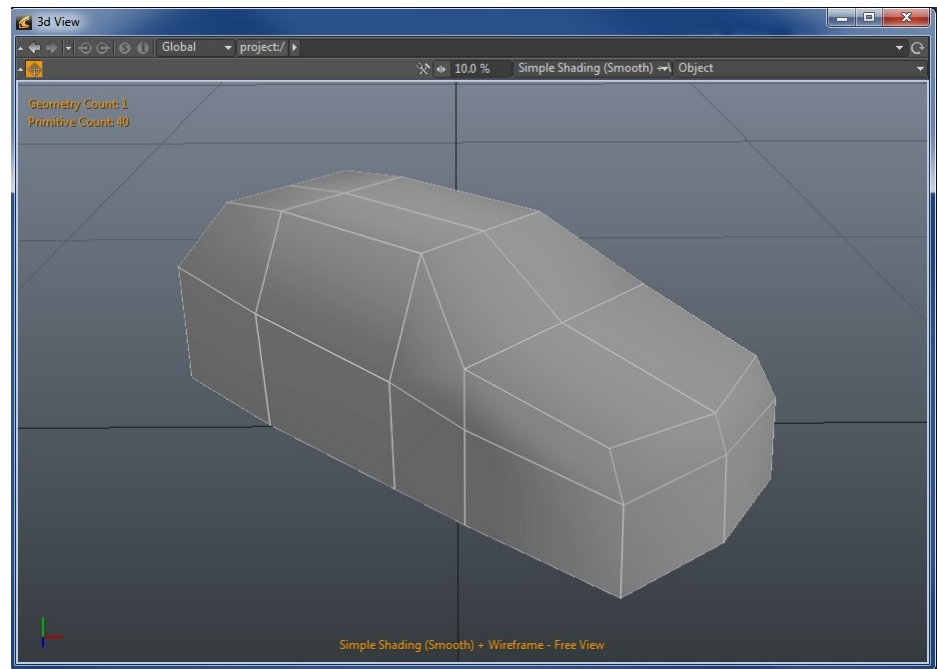
Setting your groups

Select the newly created group in the Attribute Editor and rename it to `studio_reflection`. To rename a selected object, press F2 to display the renaming window. Next step is setting the group with objects we wish to "see" in our reflection. To do so, we simply need to add the object we wish to see in the group. To add an object in a group, you simply have to drag and drop it on its references list.

Add `studio`, `golf` and `reflection_background` in the group.

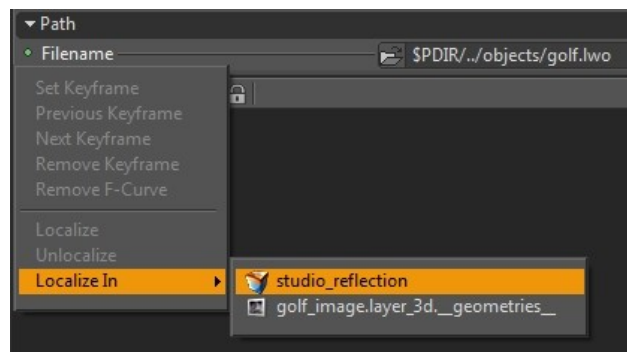


So far, there should be no change in the reflection as the reflection group basically references same objects that are visible to the camera. What we are going to do now is to localize within `studio_reflection` context, the *Filename* of the `golf` object to reference our new geometry: `content/tutorials/objects/golf_proxy.lwo`.



The golf_proxy object

Jump using the link button to golf in the *Attribute Editor* and then look for the *Filename Attribute* and Right Click on the *attribute label* and select in the menu **Localize in > studio_reflection**.

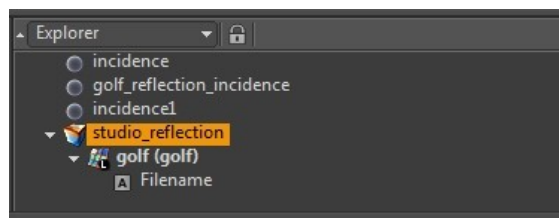


We just created a localize version of the golf within the group studio_reflection.

Replace `content/tutorials/objects/golf.lwo` to `content/tutorials/objects/golf_proxy.lwo` and you should now see this:



If you open the *Explorer Widget* and unfold the `studio_reflection` group, you will see the contextual specialization exposed:



Any changes you might do to the car such as transformation, light linking etc... will now automatically affect the localized object : this is the beauty of localization.

Clarisse 104: Layers and Mattes

This tutorial covers the use of multiple 3D layers and matte objects to compose an image. It assumes you have already familiarized yourself with Clarisse object instancing and localization. The resulting project of this tutorial is found in your project directory under the name `content/tutorials/projects/clarisse_104.project`.

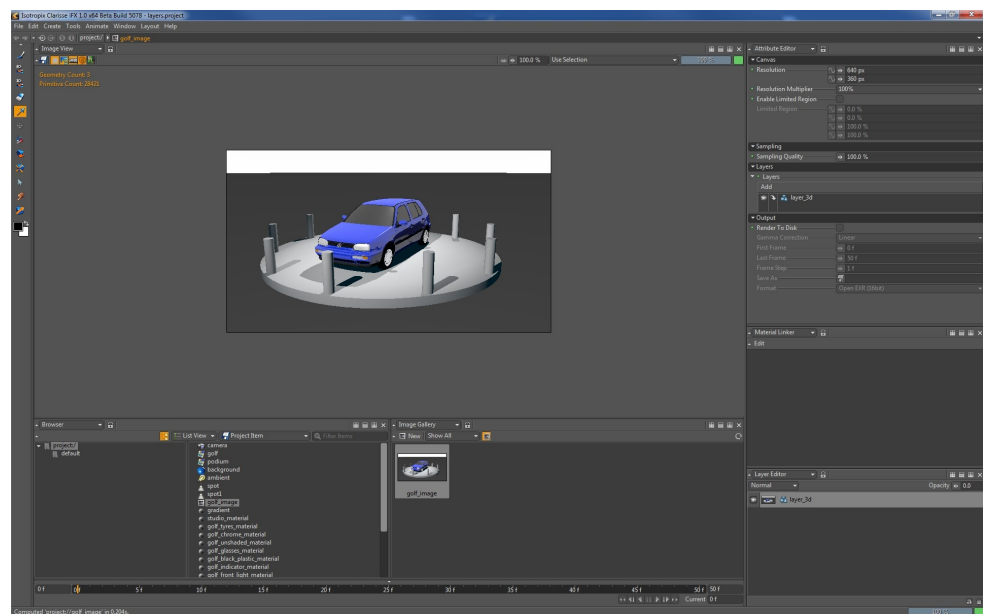
Create your first layer

Getting correct layer composition needed to compose an image usually requires the use of different packages. Using different packages means a lot of back and forth, manual file synchronization and lead to time wasting. Indeed, the job is split into several tasks: texturing, layering, lighting and shading, masks

creation, compositing and filtering.

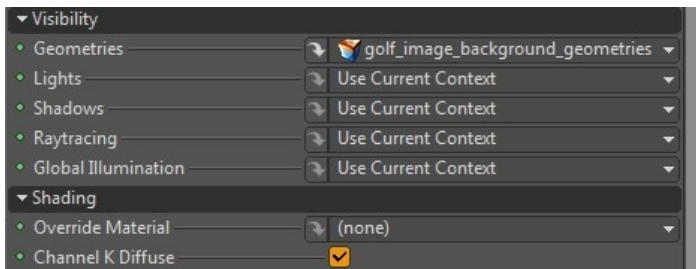
Offering more than a 3D animation package, Clarisse has been specifically designed to put layering and compositing at the heart of its unique workflow. Instead of working on segmented and sequential tasks, thanks to Clarisse, you can work on everything at the same time in any order, and focus only on your job to create the final image.

Start Clarisse and open the project `content/tutorials/projects/layers.project`.



Now select the `layer_3d` via the layer editor rename it to `background`. To rename a layer you can either use the menu **Edit > Rename**, the shortcut **F2** or double click on the layer name in the layer editor.

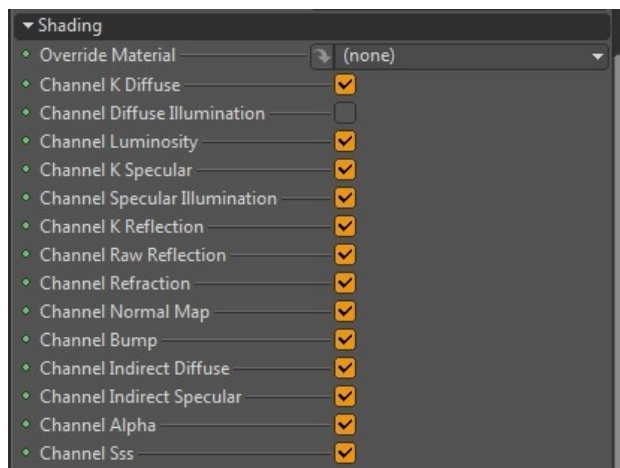
Locate the geometries group in the visibility category and drag and drop the `background` object. It will create a group referencing the background sphere, used to control geometries visibility in our 3D layer.



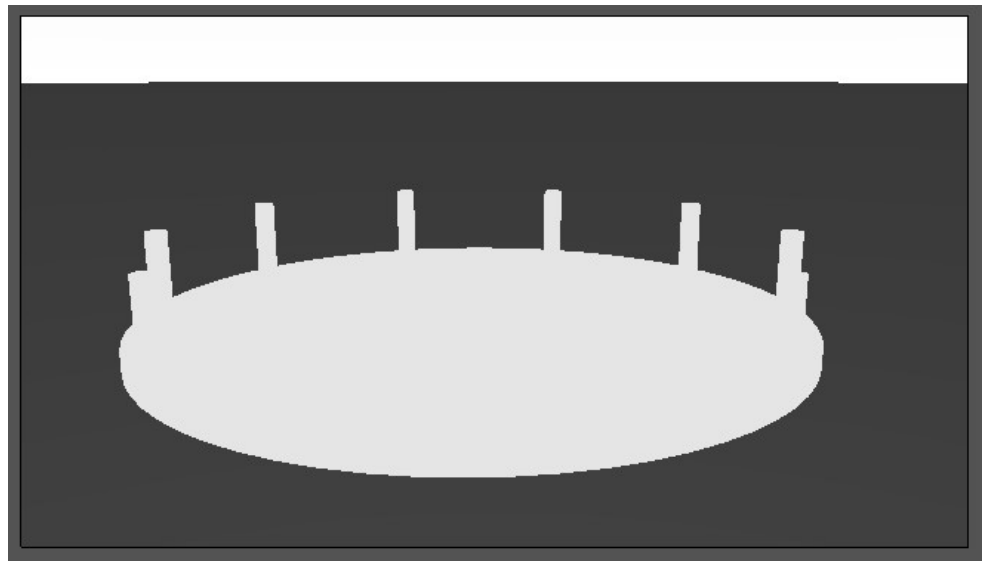
Select `golf_image_background_geometries` in the browser and rename it to `group_bg`.

Now duplicate the background layer by right clicking on it and selecting **Duplicate** in the popup menu. Rename the layer to `podium_shading`. Like you did for the background layer with the background sphere, drag and drop the podium object to geometries visibility group. It will create and reference a new group containing the podium. Rename the newly created group to `group_podium`.

In the `podium_shading` layer, locate the Shading category and uncheck *Channel Diffuse Illumination*.

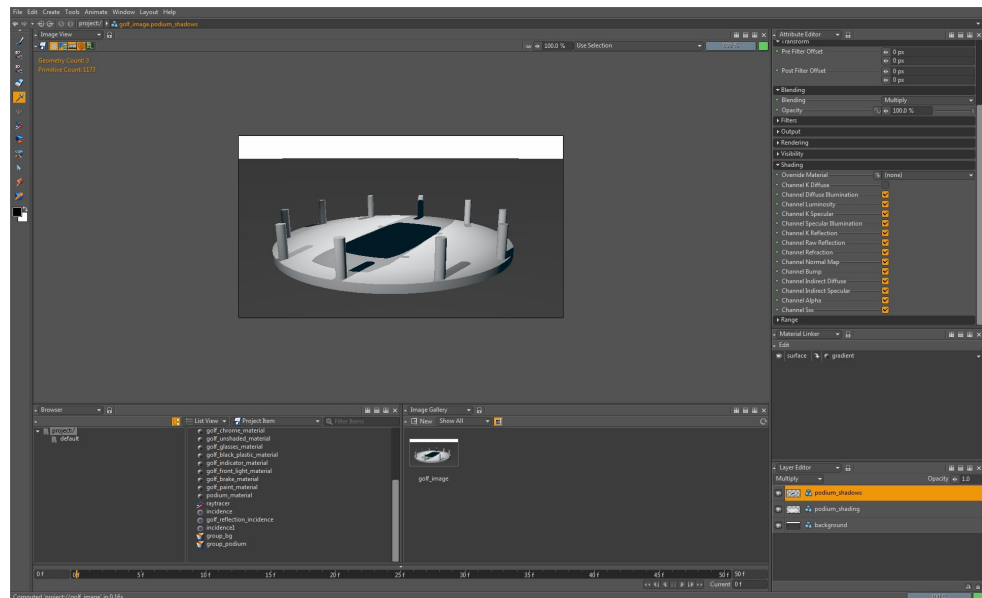


You should get this result in the Image view:



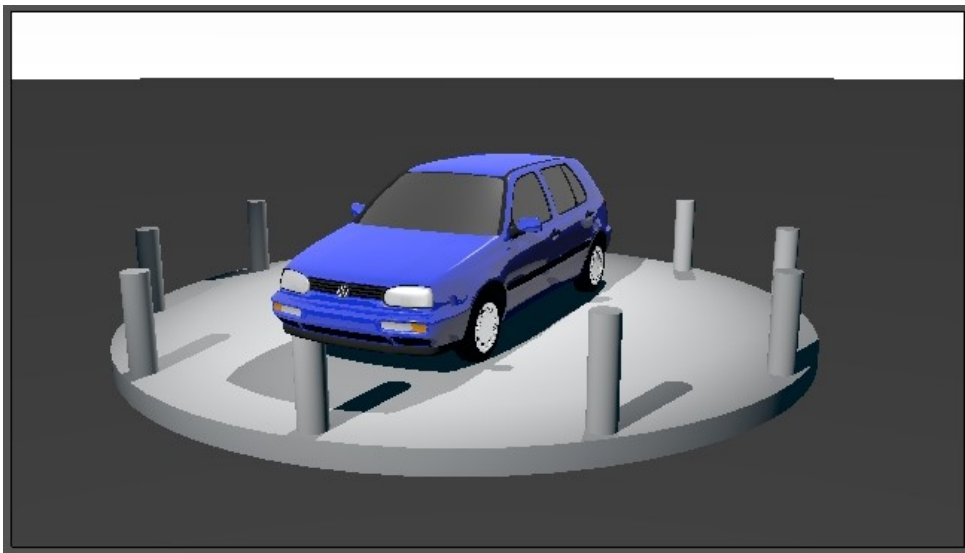
Shadows Layer

Next, we will add a layer of shadows on top of our `podium_shading` layer. Duplicate `podium_shading` layer and rename it to `podium_shadows`. Re-activate the *Channel Diffuse Illumination* of our new layer, uncheck its *Channel K diffuse*, and set its blending mode to *Multiply*.

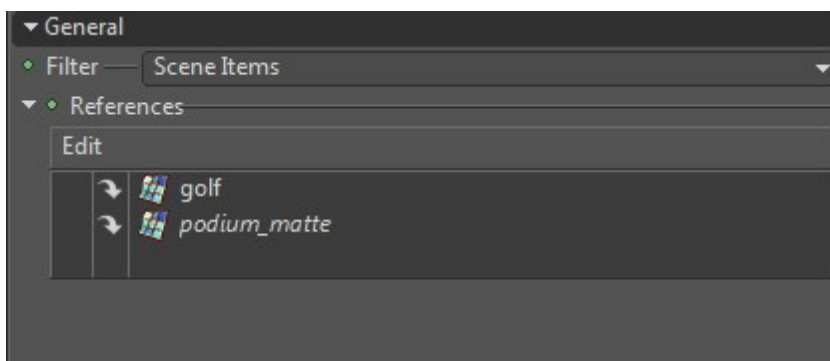


Setup Mattes

Finally we would like to add the golf layer in our image. Duplicate podium_shadows layer, rename it to golf, change the blending mode to Normal, and enable Channel K Diffuse. Drag and drop the golf object to Geometries to create a new group, and rename it group_golf.



As you can see, we have a typical compositing issue with the golf layer as it is in front and in back of our podium. To fix this, we will add a podium instance as matte object to cut off the alpha of the golf layer. Instantiate podium (Ctrl + I) and rename the instance to podium_matte. Localize both *Matte Object* and *Matte Alpha*. Check *Matte Object* and set *Matte Alpha* to 0.0. Now go back to the golf layer and add podium_matte object to group_golf (select the group in the browser and drag and drop the object in the list).





Now our compositing problem is fixed. Obviously the whole construction is procedural : for example, if you move the camera everything will be transparently recomputed with the full compositing on.



Clarisse 105: Using Scatterers

This tutorial covers geometry scattering and combining. Starting with a simple example, we will then create a procedural forest by combining and scattering scatterers. The tutorial assumes you have already familiarized yourself with Clarisse basics. The resulting projects of this tutorial are found in your project directory under the names `content/tutorials/clarisse_105_golf.project` and `content/tutorials/clarisse_105_forest.project`.

What is a scatterer?

The scatterer is a really powerful tool designed to massively populate an input geometry by using points of an input geometry. In other words, it's the tool to use if you want to populate a landscape with trees, rocks, flowers, characters, hair cuts on characters etc...

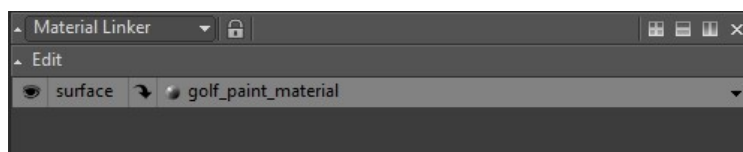
One of the scatterer advantages is the memory footprint: only one instance of the cloned geometry is in memory. With scatterers, you can scatter scatterers too and this is a very useful feature as you can quickly create a forest. First, you will need to scatter leaves on tree branches and then scatter the tree and its leaves using an object set to create a forest. Using the scatterer, any scene object (geometries, particles, volumetrics, furs...) specified in the *Geometry* attribute list will automatically be positioned using the points or vertices available in the referenced *Geometry Support*.

Scatterer in action

Let's see the scatterer in action in a simple example:

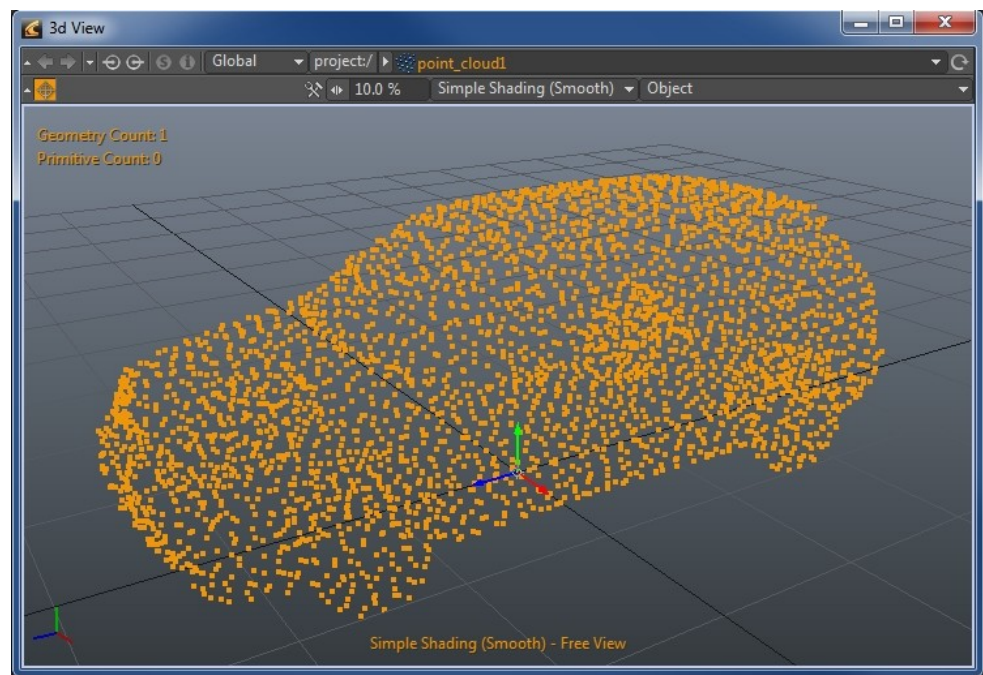
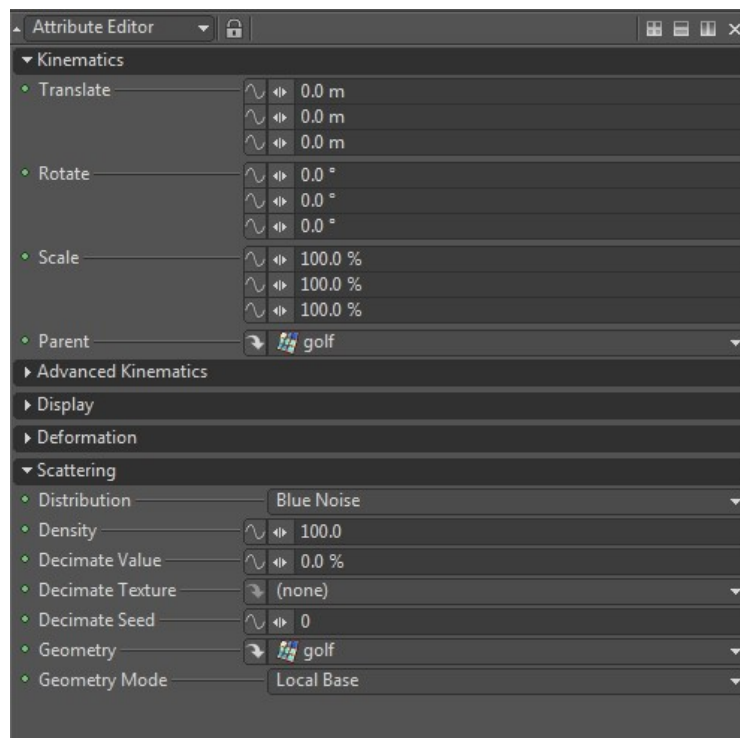
Launch Clarisse, open the project content/tutorials/projects/golf.project and create a scatterer via **Create > Scatterer**.

Create a sphere via **Create > Geometry > Sphere** and set its scale to 5% 5% 5% and assign the `golf_paint_material` to the surface shading group, in the material linker.



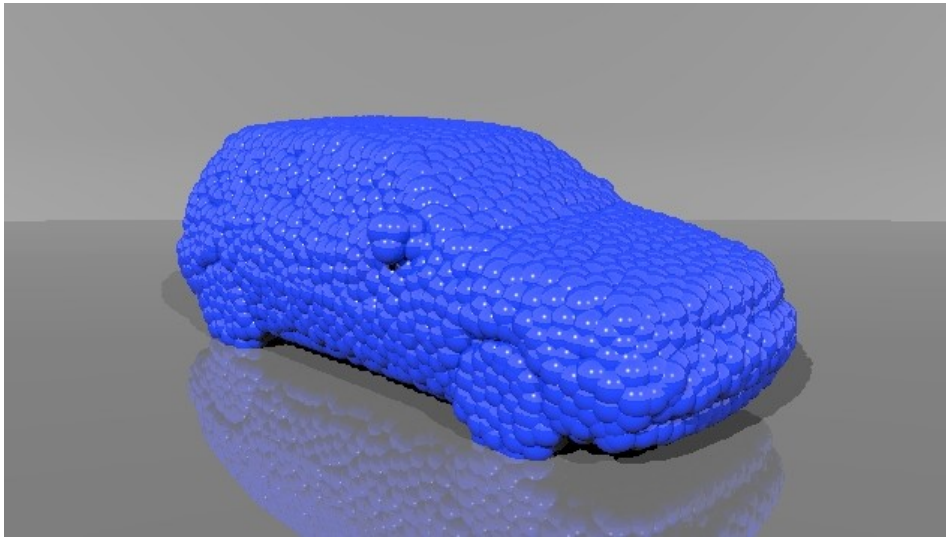
Creating a Point Cloud

Open a 3D View via **Window > 3DView**, and create a point cloud using **Create > Geometry > Point Cloud**. Set *Density* to 100, *Geometry* to `golf`, and parent it to `golf`. Change *Distribution* to *Blue Noise*. For more information on Distribution please refer to Point Cloud section.



Referencing the Point Cloud

Now, select the scatterer, set *Geometry Support* to *point_cloud*, and drag and drop the sphere to the geometry list. You should see this in the Image View:



How to create a forest

We will create a procedural forest by scattering leaves over the branches of a tree first. Secondly, we will combine every geometry related to the tree into a single scene object we will scatter over a ground to create a forest.

In this tutorial we assume Clarisse's Object Selection Mode is set to Use Popup Menu. To change the object selection mode, go to **Edit > Preferences...** under the User Interface tab change *Object Selection Mode* to *Use Popup Menu*.

Open the project `content/tutorials/projects/tree.project`. The project contains 3 geometries: trunk, branches and leaf. The image `img_tree` shows a tree without leaves:

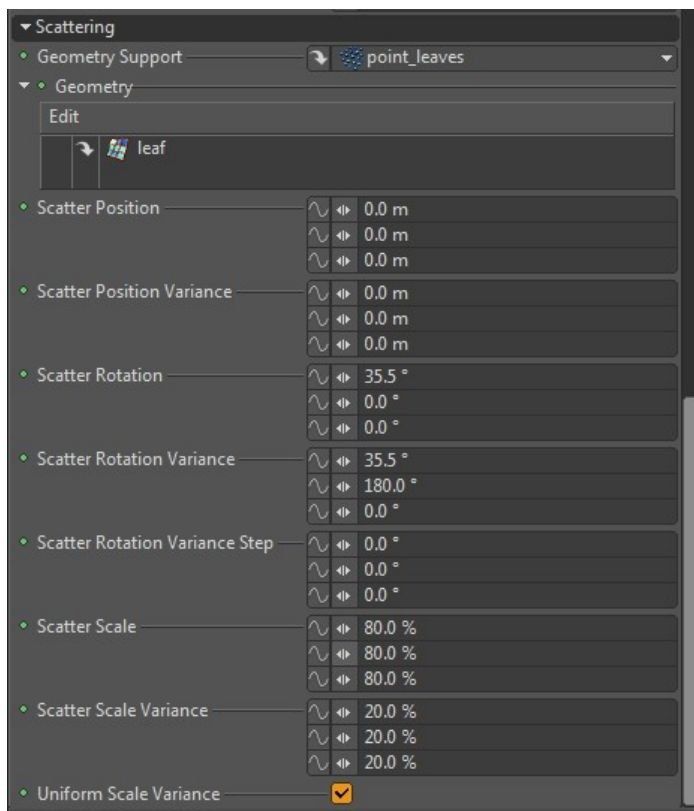


We will first populate the leaves over the branches:

Create a Point Cloud as we did in the previous example and rename it `point_leaves`, then create a scatterer and rename it `scatter_leaves`.

Select the `point_leaves`. In the Attribute Editor, set *Geometry Support* to `branches`. Change its *Distribution* to *Blue Noise* and its *Density* to 200. Now select `scatter_leaves`, set the *Geometry Support* to `point_leaves` and drag and drop `leaf` object in the geometry list.

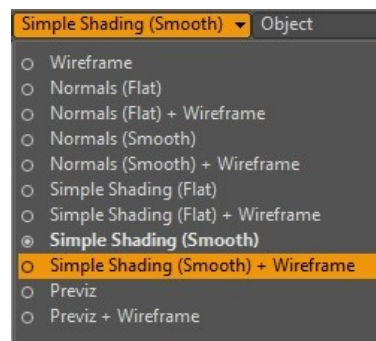
Finally we will set some randomness in the rotation and scaling of each leaf: select `scatter_leaves` and set *Scatter Rotation* to 32.5 0.0 0.0, *Scatter Rotation Variance* to 33.5 180.0 0.0, *Scatter Scale* to 80% 80% 80% and *Scatter Scale Variance* to 20% 20% 20%.



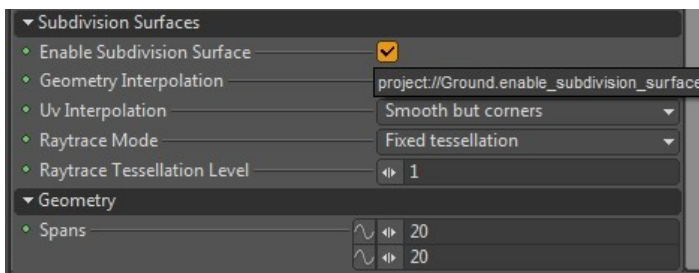
In the Image View, select the image `img_tree`. You should see something like this:



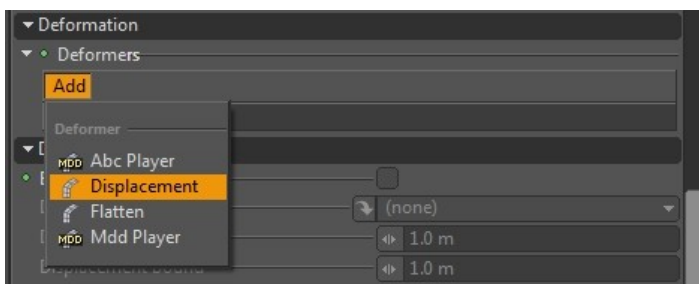
We will now create the ground and deform it using a displacement texture. Replace the image view by a 3D view. Set the display mode to *Simple Shading (smooth) + Wireframe*.



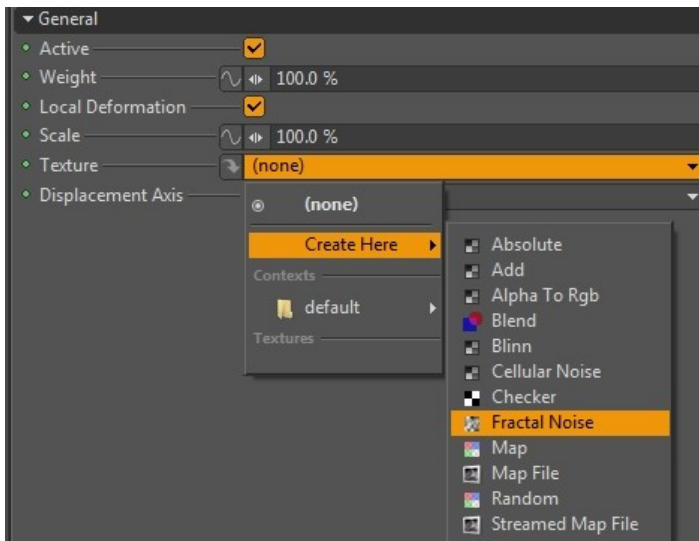
Create a grid via **Create > Geometry > Polygrid**, rename it ground and set the *Scale* attribute to 40000% 40000% 40000%. Now refine the grid geometry by setting *Spans* attribute to 20 20, checking *Enable Subdivision Surface* and set *Raytrace Tessellation Level* to 2.



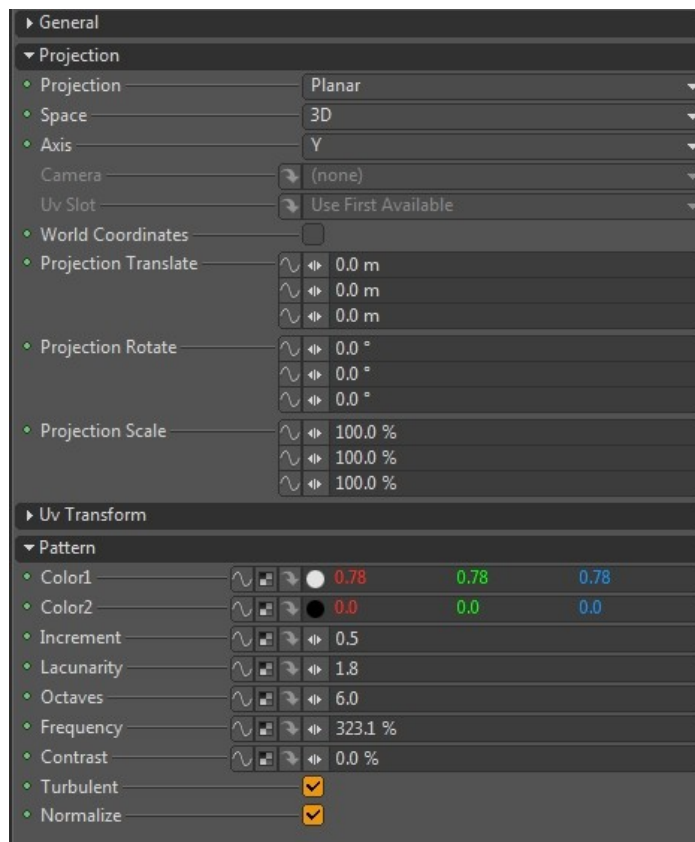
In the Attribute Editor, open *Deformers* tab and add a Displacement deformer via Add Deformer.



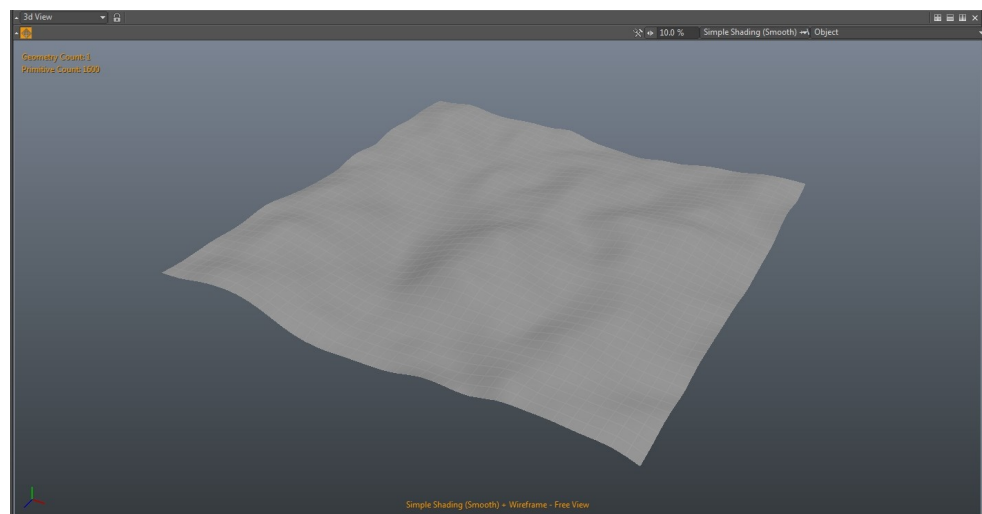
Select the newly created deformer and create a fractal noise texture via the contextual menu of the *Texture* attribute.



Set the Scale attribute to 20% and select the fractal_noise, then set the texture attribute as shown in the following picture:

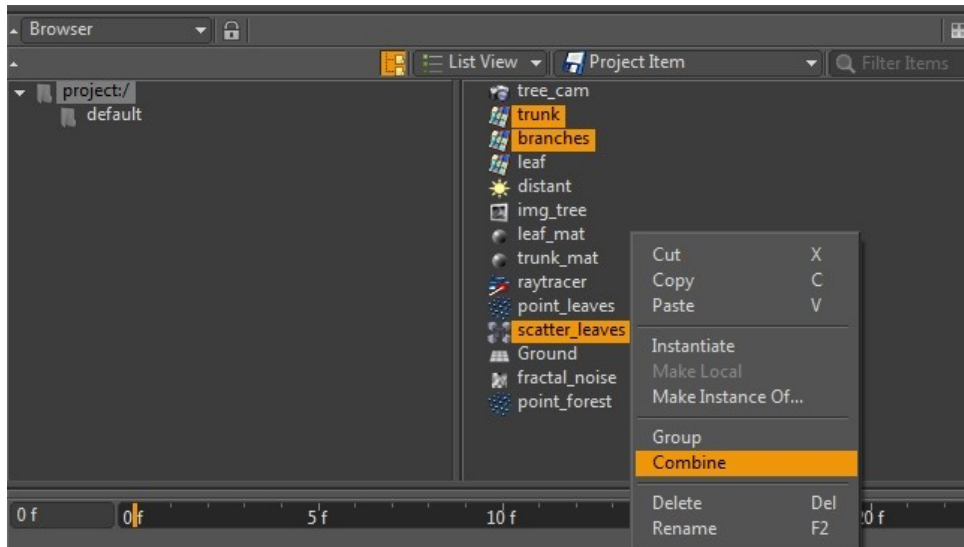


You should see something like the following picture in your 3D View:

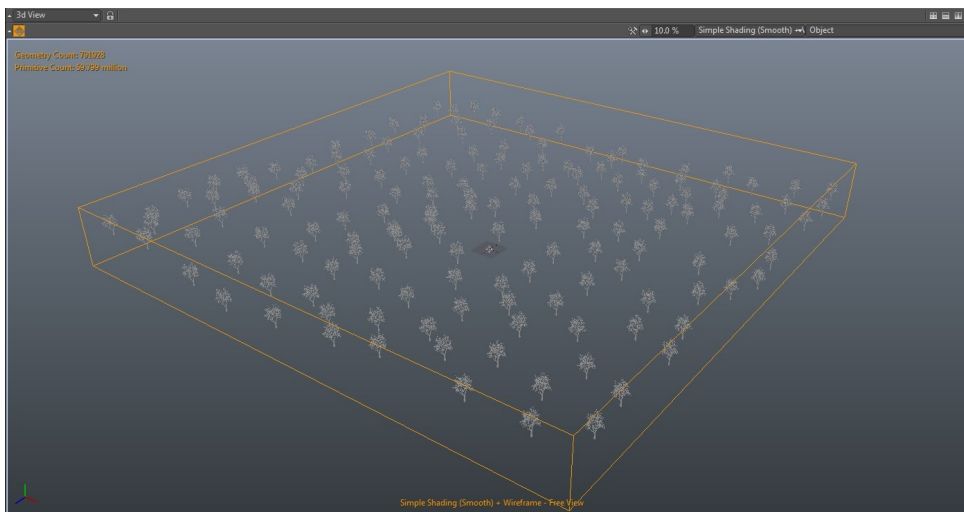


Create a point cloud, rename it `point_forest` and plug *ground* into the *Geometry* attribute and set its parent to *ground*. Set its *Distribution* to *Blue Noise* and its *Density* to 200 in order to scatter about 85 points.

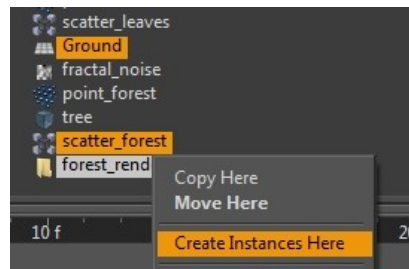
In order to properly scatter the tree, we must first of all, combine all its geometries into a single scene object that will be plugged into the scatterer *Geometry* list. In the browser, select trunk, branches and scatter_leaves, then right click and choose **Combine** to create a new combiner referencing the tree's elements. Rename it tree.



From the contextual menu of the *Geometry* attribute, create a combiner and select it in the Attribute Editor. Rename it tree. Now, create a new scatterer, rename it scatter_forest. Set the *Geometry Support* to point_forest and drag and drop the tree object in the geometry list. You should get this :

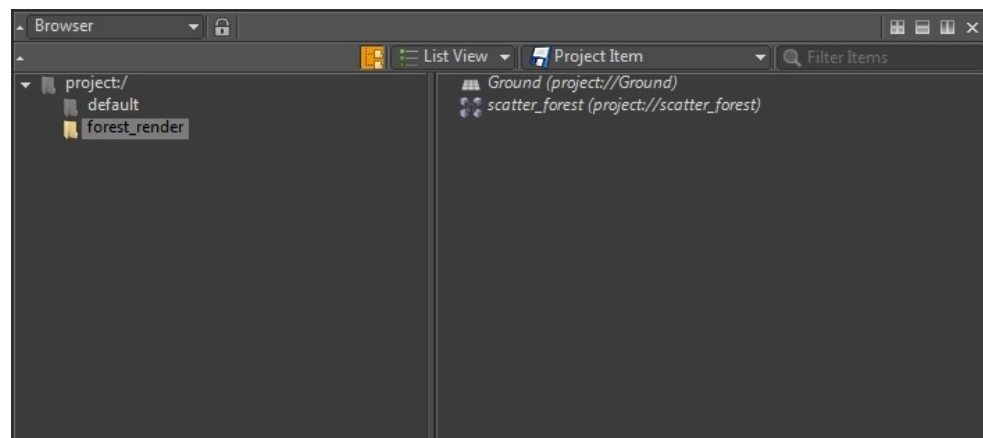


Now, it's time to create an image using this forest and to introduce a new object: the context. Replace the 3D view by an Image View, then create a new context using **Create > Context**. Rename it `forest_render`. Select `scatter_forest` and `ground` in the browser, and drag and drop them in the `forest_render` context, using the right click mouse button. Choose **Create Instances Here**.

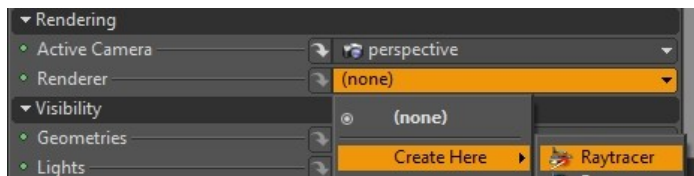
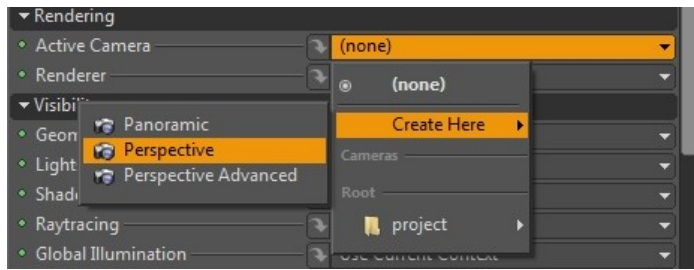


Contexts are folder like objects that let you organize your objects and isolate parts of your project. In this example, we only want to render the forest and the ground, not all individual objects like leaves, branches or the trunk. We could achieve this, manually setting visibility groups or using attributes like `unseen by camera`, but contexts are way more powerful to do this.

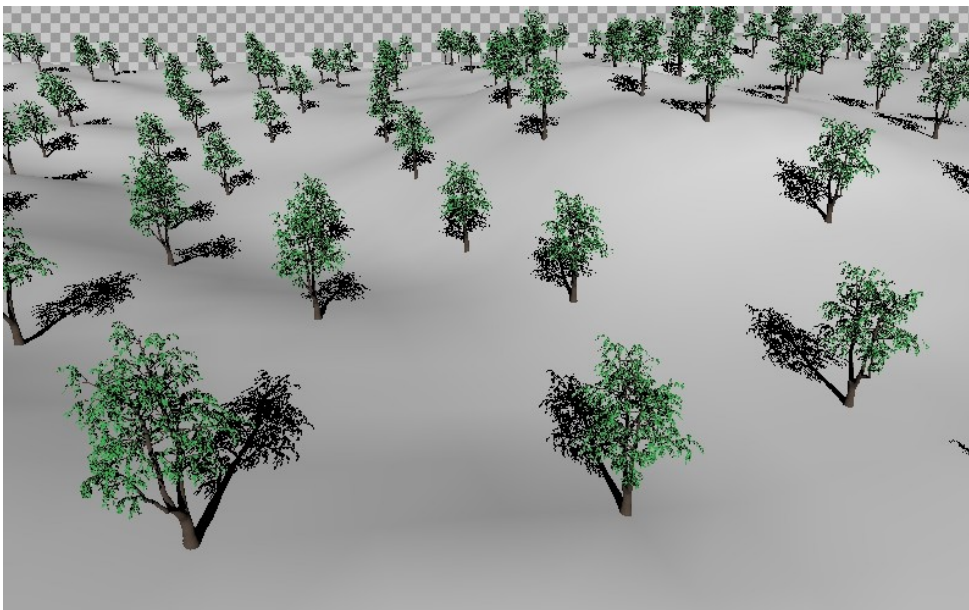
Go to `forest_render` context. You should see your instances inside. Instances are like shortcuts in a filesystem (plus you can localize any of their attributes, creating variants linked to the source object).



Now, create a image using the **New > 3D Render Image** of the Image Gallery, then rename it `Forest_image`. Set the *Resolution* to 800x500px. Select the background 3D layer in the layer editor and create a perspective camera and a renderer, directly in the attribute editor.



Because of default settings, you should see a black thing in the image view because a 3D layer only sees objects in the same context (including lights). Create a distant light via **Create > Light > Distant**, select the ground and hit F over the Image View to fit the object. Orbit the camera a little with **Alt+Left Mouse Button** and move forward using **Alt+Right Mouse Button**. You should see something close to the following picture in your Image View:



The Island

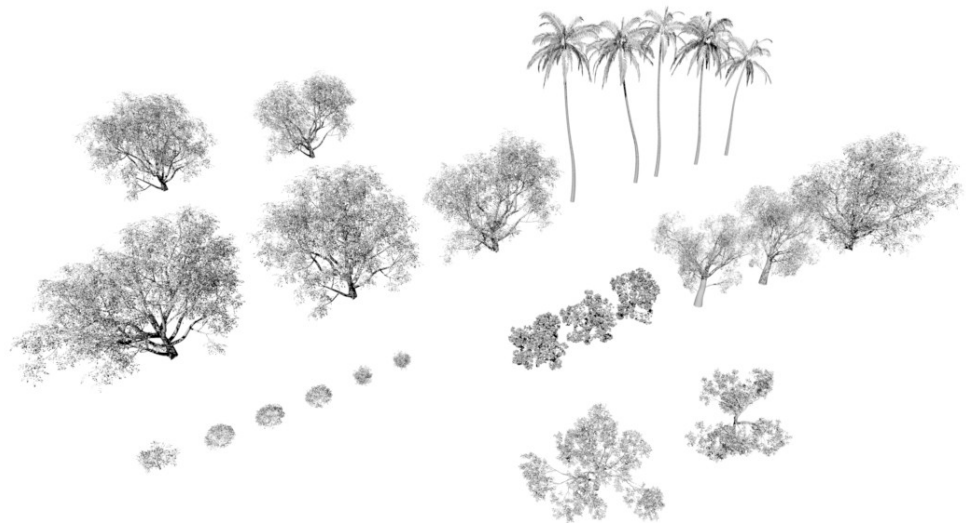
In this tutorial, we'll create a typical tropical island landscape using various techniques particularly suited for general 3D matte painting work. Many

exterior scenes are covered with vegetation and the complexity of this kind of structures can be very challenging. We'll see how Clarisse tools and its ability to easily handle complexity can help the matte painter to achieve such a challenge.

Creating a plant library

First, you have to identify the needed species for your landscape. If you're not sure about them, you can search the internet and spot the plants growing in the chosen location. In this example, we chose a typical Polynesian landscape and identified a few species (mainly ficus, palm trees, and some various small plants). There are three very important aspects for each plant: size, leave density and leave scale. Fortunately, in Clarisse, we can adjust each of these properties at any time, but if you want to get a realistic result, it is important to start with the right values. So, make sure you have good references with right scale information.

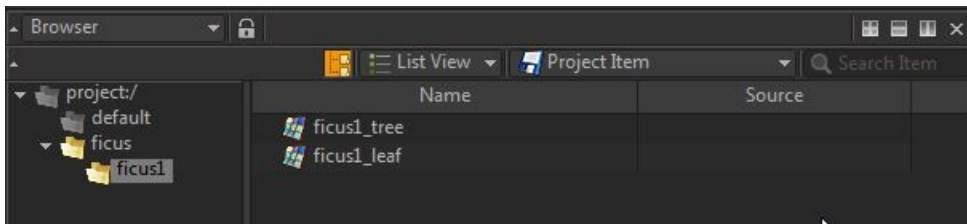
You can use any modeling application to build your species but, procedural and specialized plants modelers are more suited. Pre-build libraries are very good too and can save a lot of your precious time. I recommend starting with this solution, and go to custom modeling only for very specific plants. Indeed, for this tutorial, you can use the supplied ones.



The tutorial plant library

Preparing plants for scattering

Launch Clarisse and replace the Image View by a 3DView. Create a context named `figus`. This will be our first species. In this context, create a new one named `figus1`. In Clarisse, it is very important to keep things organized and place your objects in named contexts.



The Browser with the context structure

Import your first object in `figus1` context using **File > Import Geometry**, and choose `Objects/Ficus/figus1.lwo` file. This will create two objects: `figus1_tree` and `figus1_leaf`, because a `.lwo` file contains two layers.

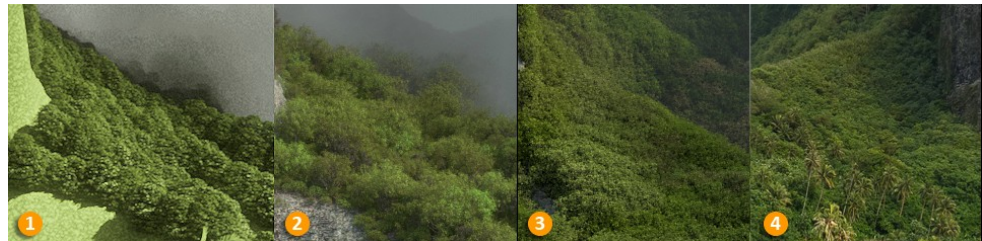
Note

Note : The various File paths used in this tutorial are all relative to the Island tutorial directory. All intermediate Clarisse projects should be saved in here.



The imported tree

The look of plants scattered on a landscape is extremely sensitive to leave size and density: this is why it is better to control this in Clarisse. In this series of thumbnails, you can see how density affects the look of the render:



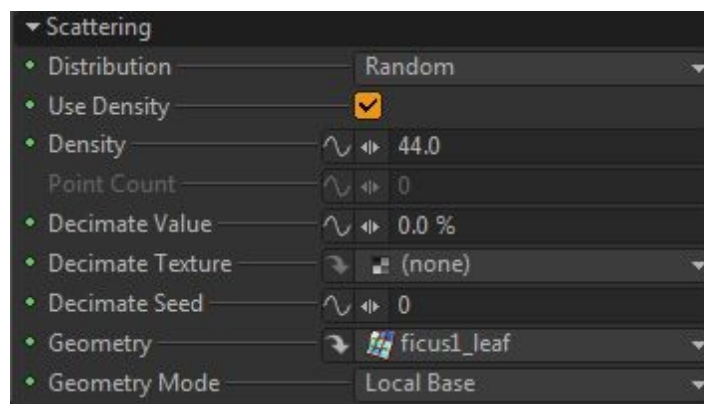
How leaves density affect the look of scattered trees

(1) Too low density, "hard" result (2) Too high density, "fuzzy" result (3) Right density, realistic result (4) Photo reference

We've just seen why density control is important for the quality of our future forest. Unfortunately, our tree has a pre-defined number of leaves. Let's see how we can replace this by a procedural foliage system that will allow you to change the number and the size of the leaves.

Create a *Point Cloud* object : **Create... > Geometry > Point Cloud**, rename it `leaves_pc`, and set the following attributes:

```
density = 44  
Geometry = ficus1_leaf
```

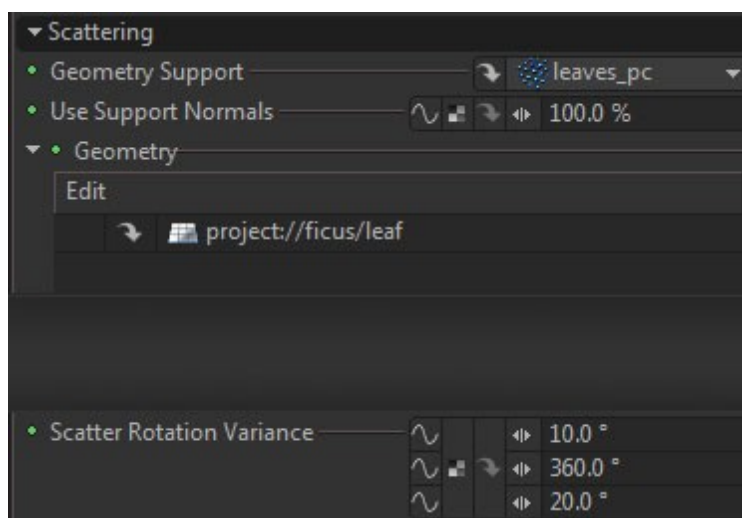


The leaf_pc point cloud attributes

The *Density* attribute will allow us to control the leaves density.

Let's now create the leaf. As we will use the same for all our ficus trees, go from your current place to `project://ficus`. Create a grid (**Create > Geometry > Polygrid**), rename it leaf, and set the *Scale* attribute to 15%, 100%, 20%, and the *Span* to 1 and 1.

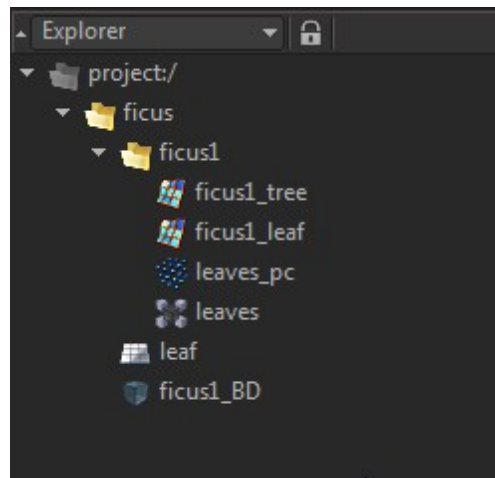
Now, go back to `project://ficus/ficus1`, create a *Scatterer* object (**Create > Scatterer**) and name it leaves. Drag and drop the previously created leaf (`project://ficus/leaf`) into the scatterer's geometry list (you must lock the *Attribute Editor* on the scatterer to be able to do that), and set its attributes like this:



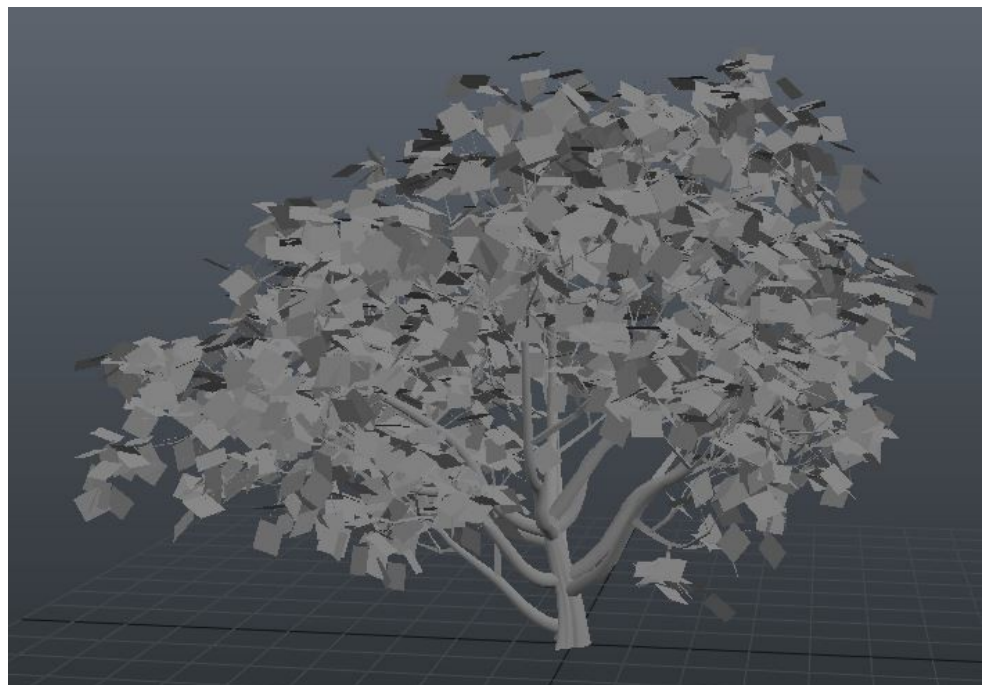
leaves scatterer attributes

All we have to do now is to "package" our tree in a more convenient way to be used in the project. Select the leaves object and the `ficus1_tree` object in the *Browser*, and right click **Combine**. This creates a combiner with the two selected objects inside. Then, you will have only one object to manage. Rename the combiner to `ficu1_BD` and move it up to `project://ficus/` for future use.

If you open an *Explorer* widget and unfold the contexts, you should have this:



You can now play with the *Point Cloud Density* and *Scatterer Scale* attributes to vary the density and size of the leaves.



The same tree with less density and bigger leaves

Using contexts to re-use your work

A real good thing about contexts is they act as namespace: you can't have two objects with the same name inside one context, but you can have the same name many times as long as it is in different contexts. This allows to duplicate full setups using copy and paste on contexts.

Set your browser to `project:/ficus` and select the context `ficus1` in the right part. Copy/paste it twice using **CTRL+C** and **CTRL+V**. You have now three copies of your original setup. Go to context `ficus2` and multiselect `ficus1_tree` and `ficus1_leaf`. In the Attribute Editor, look for the *path* attribute and replace `ficus1.lwo` by `ficus2.lwo`. Once done, Clarisse loads the second object and the second plant is ready to be tweaked. For the demonstration, select `leaves_pc` and play with the *Density* attribute, then, revert it back to 44. Rename `ficus1_tree` and `ficus1_leaf` objects to `ficus2_tree` and `ficus2_leaf`. **Combine** the two objects to `ficus2_BD` combiner and move it to `project:/ficus`. Do the same for `ficus3` context, but this time pointing to `ficus3.lwo`.

This approach is very important to get the full potential of Clarisse. You can use this method to recycle complex materials, lighting sets, compositing setups etc...Combined with referencing, this feature allows very powerful collaborative workflows.

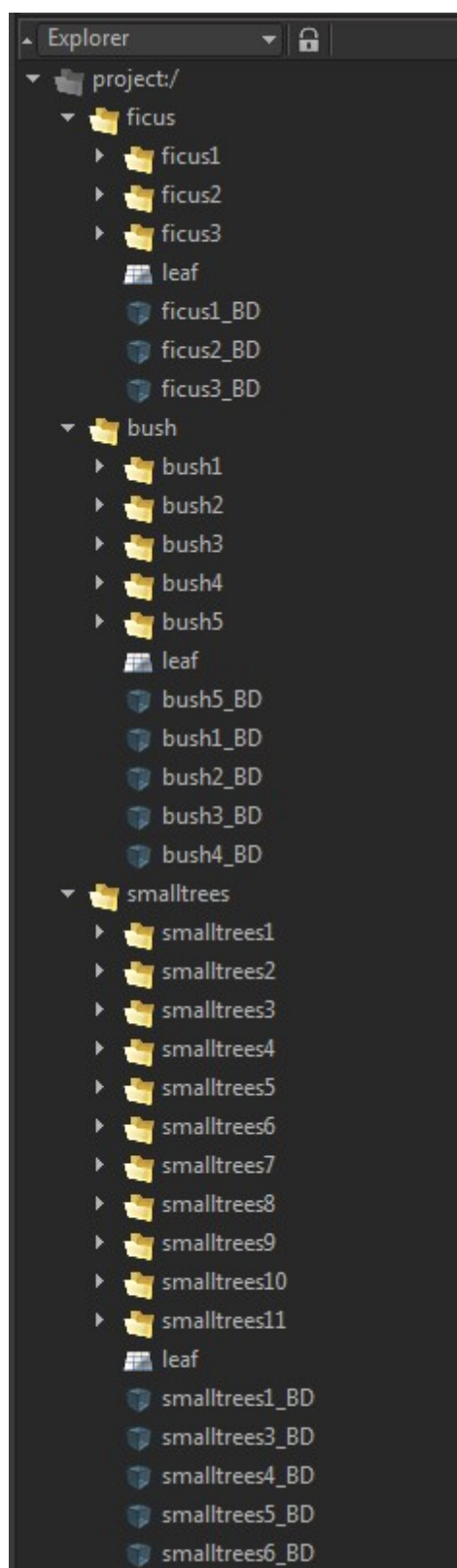
You can now create `bush` and `smalltrees` contexts next to the `ficus` context, and do the same for the corresponding objects directory found in the objects tutorial folder.

Note

It is very important to make the distinction between Clarisse's contexts and filesystem's folders or directories. Every time you read "context", that means Clarisse project's contexts. If you see "directory" or "folder", it refers to the file system. Thus, don't search the corresponding items in your Clarisse project. It is important to respect this convention when using public forums, internal support etc...because lots of misunderstandings occur when the two concepts are mixed.

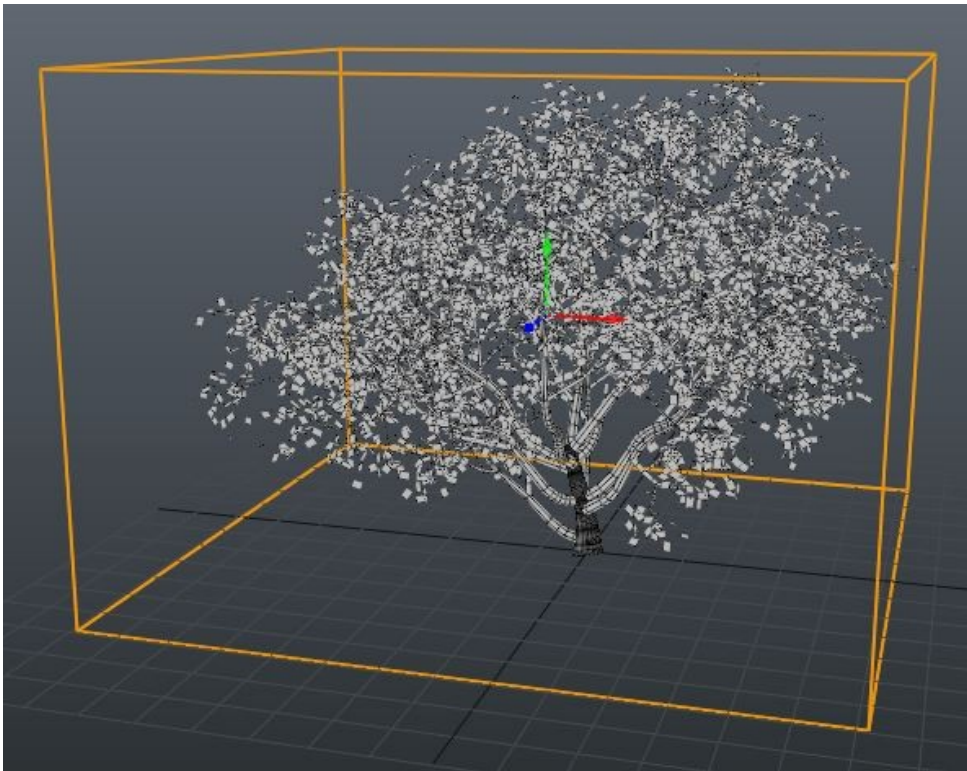
You'll need to adjust the leaves size and density for each species. The bush, for example, need a *Scatter Scale* of 20% with a *Variance* of 10%, and a point cloud *Density* of 844 to get good results.

If you open an Explorer, you should have something like this:



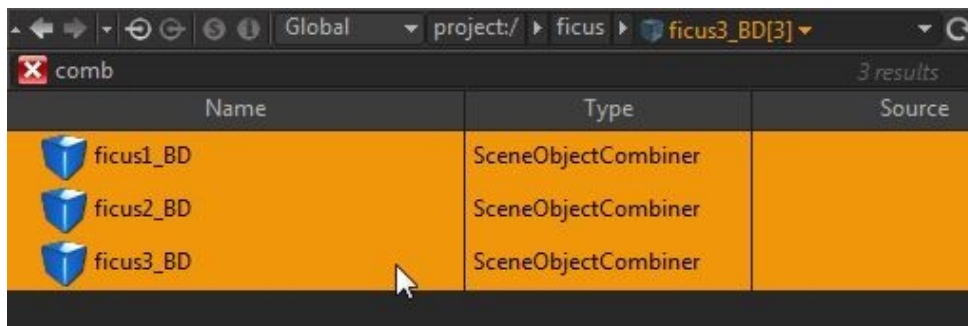
The completed context structure

Now, if you take a look at your combiners, you may notice a little problem. Pick the move tool (W) and select `project:/ficus/ficus1_BD`. The pivot should be at the base of the tree and not in the center. By default, combiners automatically set the pivot at the center, which is usually fine, but will cause problems in this very case.



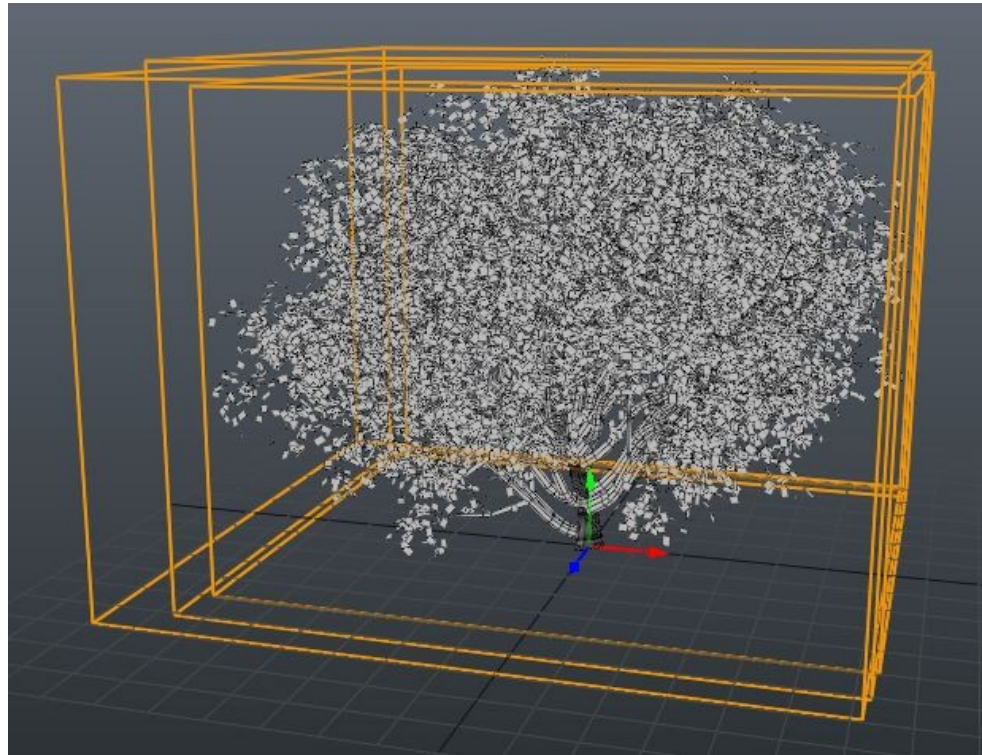
Pivot centered

Hit the **F3** key to open the search widget. This will allow to select all combiners in the project. Type "comb", **Enter**, and select all objects in the list.



The Search widget

In the Attribute Editor, set all the *Translate* and *Pivot Translate* attributes to 0. Now, all our combiners have the pivot correctly centered at the base.

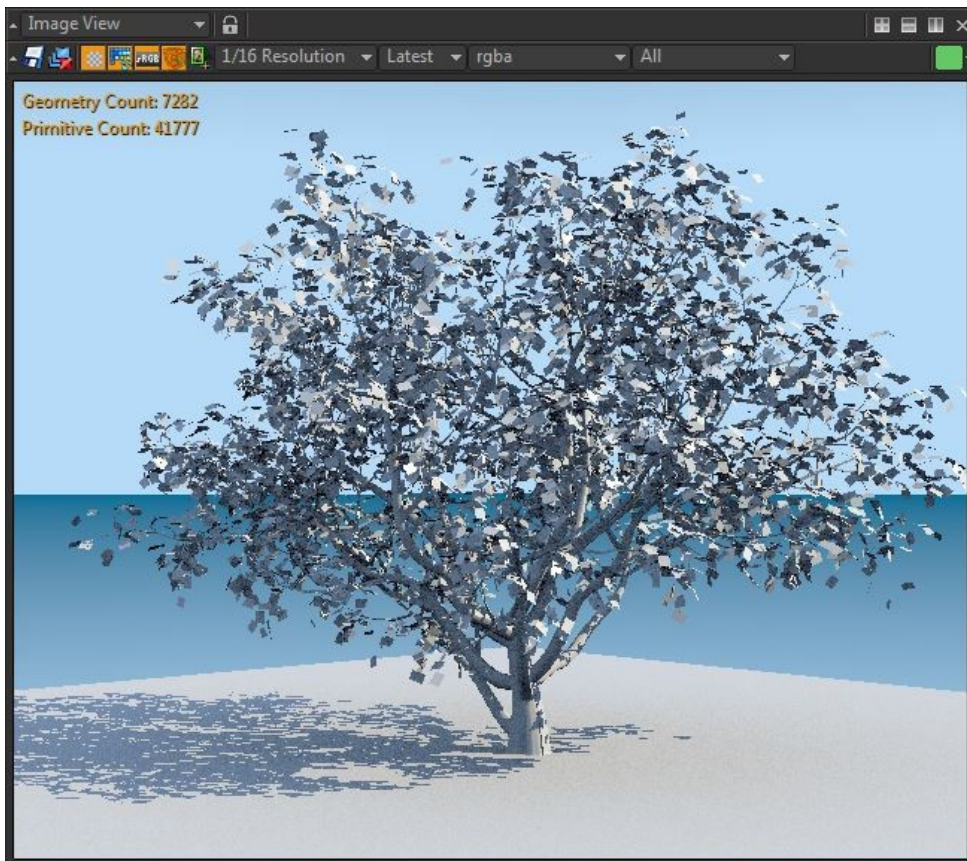


Pivot at the base

Shading the plants

We will now create materials for our plants. In order to view our materials, it will be easier if we use a small render environment. Go to **File > Import > Project...** and choose `enviro.project`. Replace the *3D View* by an *Image View* and select the image inside the *enviro* context. You should get a grey plane with a blue gradient.

Now, select `ficus1_BD` and **drag and drop** it on the ground plane in the *Image View*. This automatically creates an instance of the tree inside the *enviro* context.

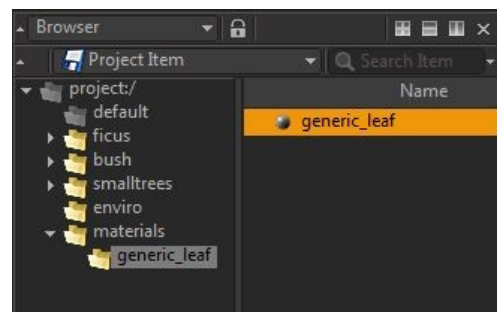


The instanced tree in the imported environment

If you look at the `ficus1_BD` object, you will notice a little "s" next to the object icon because the object is now the source of one or more instances.

Replace the *Image Gallery* widget by a *3D View* set to *Object* mode, *Previz* mode, with *Fit on select* on. This will allow us to inspect in detail our individual objects while seeing the render on the *Image View* at the same time (you can of course use a different layout, the important thing is to have at least an *Image View* and a *3D View* opened).

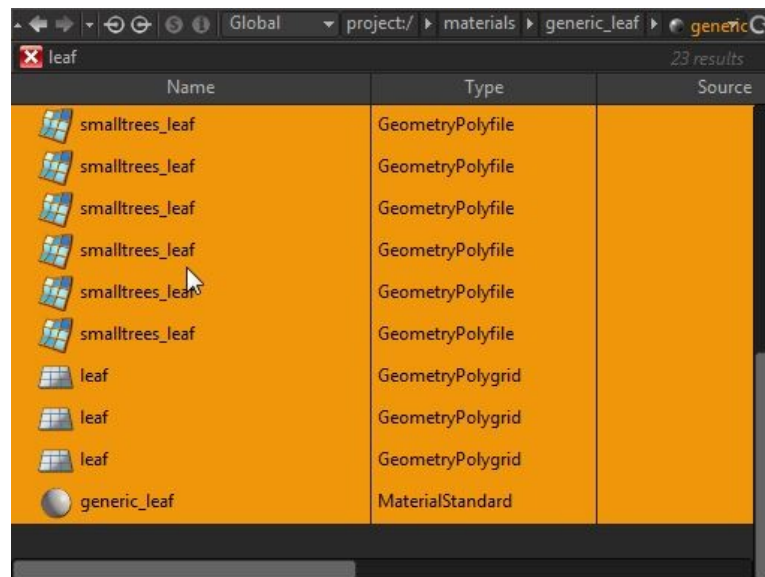
At the root of the project, create a context named `materials`. Then, inside `materials`, create another one named `generic_leaf`. Inside this last one, create a *Standard Material* and rename it `generic_leaf`.



The renamed standard material

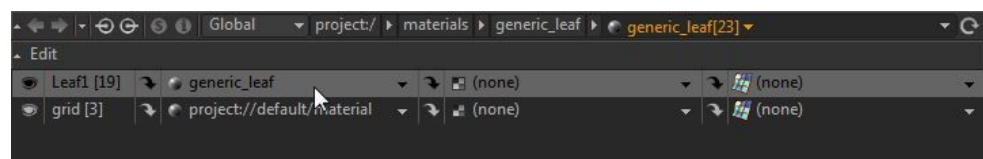
For now, just modify the *Diffuse* attribute of the generic_leaf material and set it to a green color.

Search "leaf" word using the **F3** shortcut, and select all objects in the list.




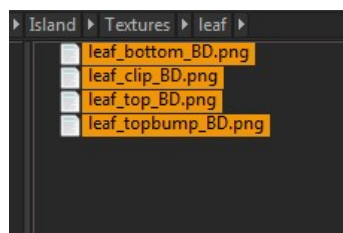
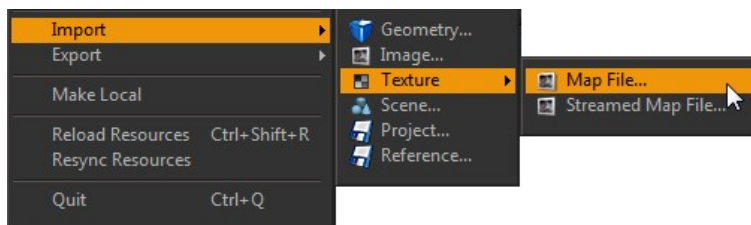
Select all objects containing "leaf"

In the material linker, replace the default material by the generic_leaf material for the Leaf1 shading group. Do the same for the grid shading group.



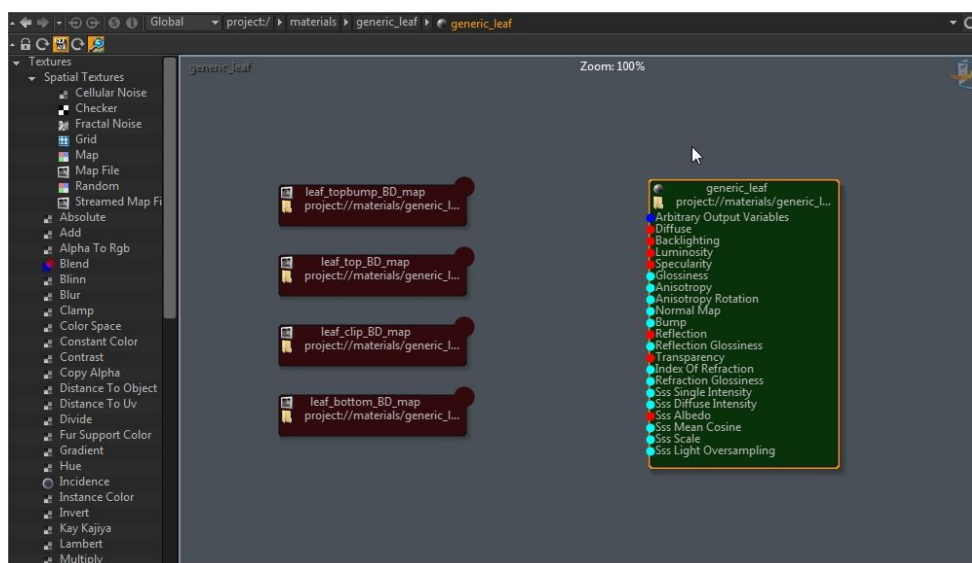
The Material Linker

Use the *Pickfit* tool  in Image View to zoom closely to a leaf. Use **File > Import > Texture > Map File...** and select all images in textures/leaf tutorial directory.

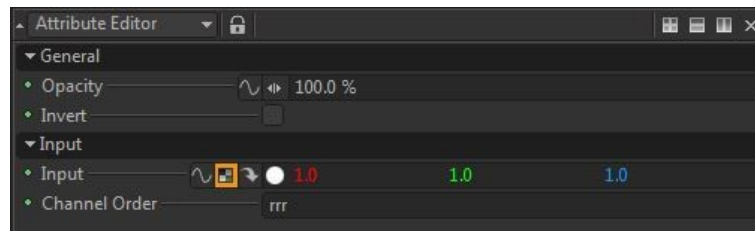


Select all the leaf textures

Open the Material Editor (make sure the generic_leaf material is still selected) and drag all *map file* objects in the Material Editor workspace. Hit **L** to auto layout all textures.



Create a *Utility Texture*, with *Output attribute* set to *is facing?*. Add a *Reorder Texture*, type *rrr* in the *Channel Order* attribute, and plug the *Utility Output* to the *Reorder Input*.



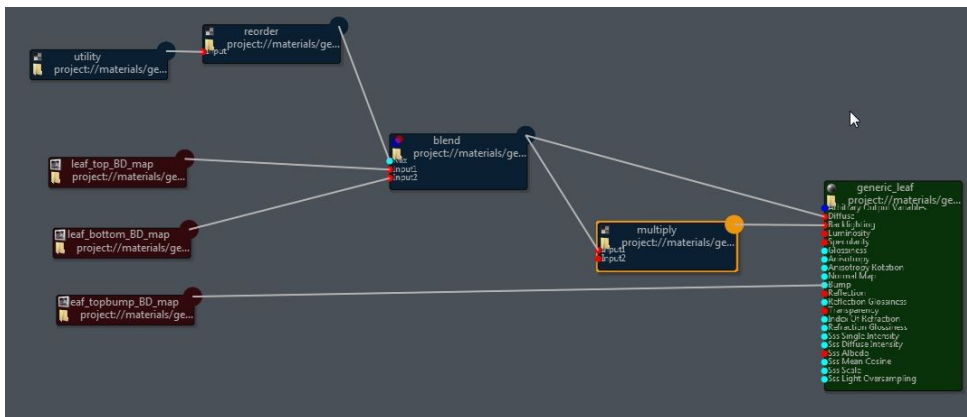
The reorder texture attributes

Create a *Blend Texture* and plug the *Reorder Output* to *Blend Mix* attribute. Plug *leaf_top_BD_map* in *blend Input1* and *leaf_bottom_BD_map* on *blend Input2*. Connect *blend output* to *generic_leaf diffuse*. Each side of the leaves now receive a different texture.

Add a *Multiply Texture* and plug its *Input1* attribute to the *blend output*. Connect *multiply output* to *generic_leaf backlighting* attribute. Now you can control the *backlighting* of the leaves using the *multiply texture*. For now, set *multiply Input2* attribute to 0.5 .

Connect *leaf_topbump_BD_map* to *generic_leaf Bump* attribute. Go to *leaf_topbump_BD_map* attributes and look for *Single channel file behavior*. Set it to *force luminance* (this map is a grayscale/single channel one, so you must set it as described). Select *generic_leaf* material and set the *bump* to 5mm. Set the *specularity* to 1. If you want more realism, you can also add some raytraced blurred reflection, but in this example, using the good old specular model is a fine option.

To complete the material, select the four *map files* and set their *Uvscale* attribute to 80% in the first axis.



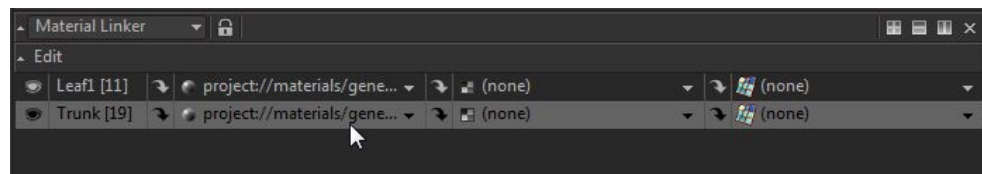
The finished generic leaf material

Select all objects containing "leaf" in their name using **F3**, and open the material linker. Next to the grid shading group, click on the little checker icon, followed by a (none), and browse to `project://materials/generic_leaf/leaf_clip_BD_map`. This *Map File* will be used as a *Clip Map*: every white element in the texture will be clipped in render, transforming the grid into a leaf shape.



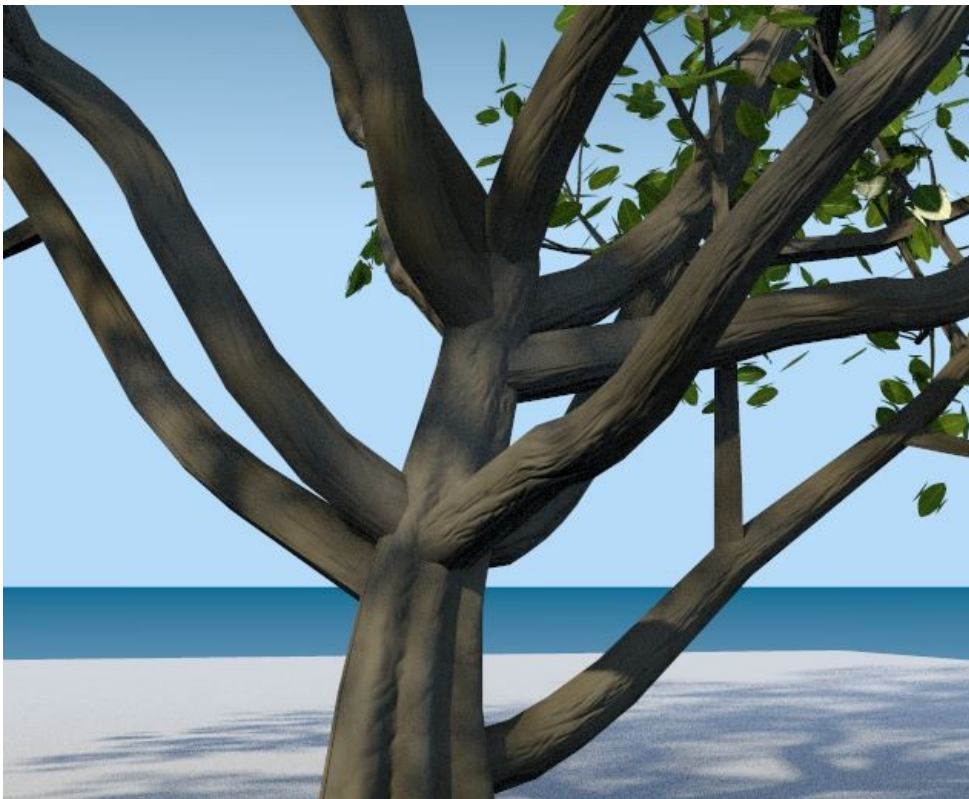
The shaded leaves

Return to the material context and create a new context named `generic_trunk`. In there, create a new *Standard Material* renamed `generic_trunk`. Hit **F3** and search for `tree`. Select all objects in the list (including leaves, which is not a problem). Now, look at the Material Linker: you can see the `leaf1` shading group (don't touch this one), and the Trunk shading group. Click on the material section for the trunk and pick your `generic_trunk` material.



The `generic_trunk` material assigned

Now all trunk shading groups share this material. Select the `generic_trunk` material and connect a new *Fractal Noise Texture* to the *Diffuse* attribute. Set *Color1* and *Color2* to two different brown colors (0.25 0.23 0.17 and 0.14 0.1 0.05 in this example). Set the *projection* to *planar*, the *projection scale* to 800% on all axis, and the *contrast* to 50%. Select again the `generic_trunk` material and connect a new *Fractal Noise Texture* to the *bump*. Leave the texture settings to default value and set, in the material, the *bump* attribute to 5 cm. This completes our trunk material.

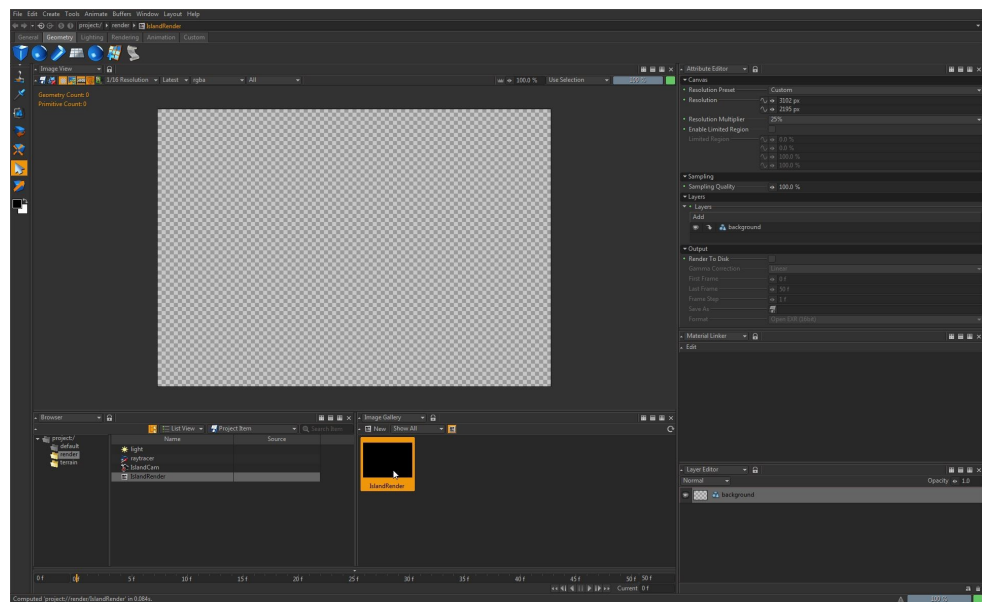


The rendered generic_trunk material

Now its time to save your plant library to a project file. For this tutorial, you can name your file `plantlibrary_ref.project`. Using "_ref" in the name, you will always know if the file is meant to be used in other projects.

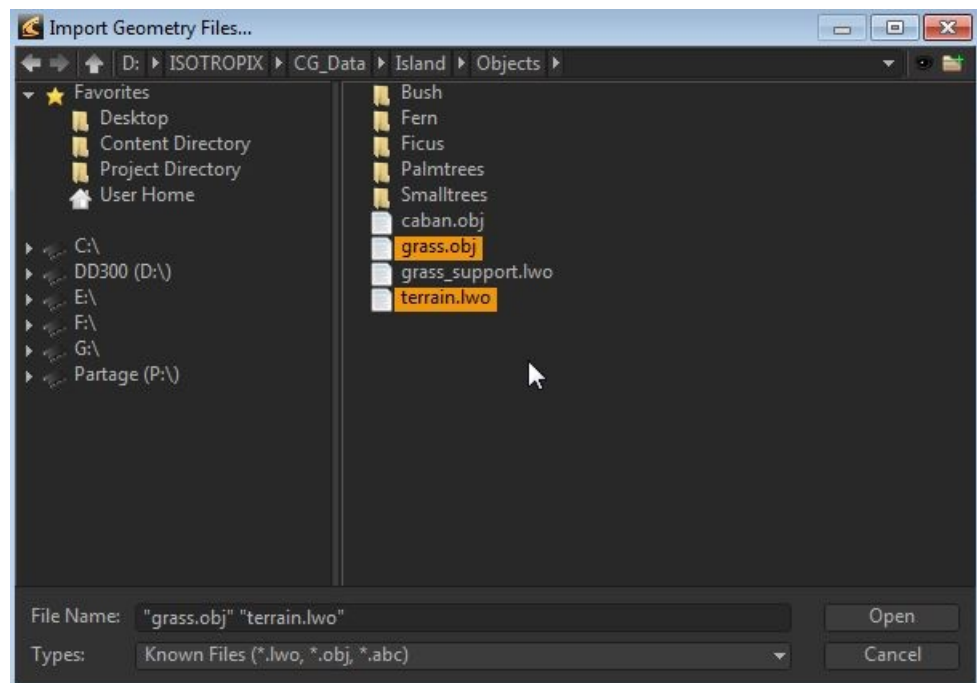
Creating the first scatterer

Load the `islandproject.project`. It contains a minimal setup with an empty image connected to an advanced Camera, all with the correct aspect ratio.

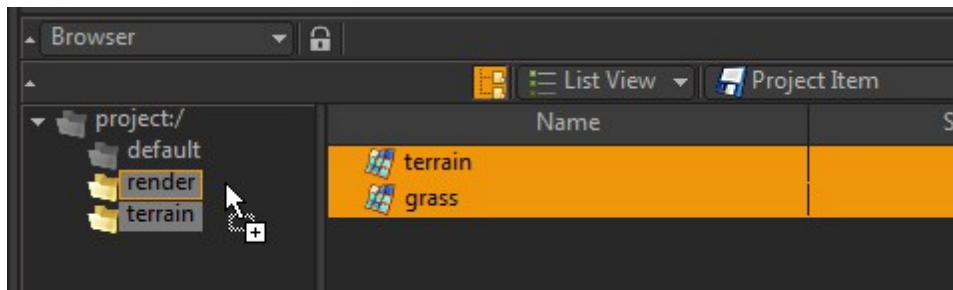


The islandproject start project loaded

Go to the *terrain* context and load `grass_support.lwo` and `terrain.lwo` (both located in the Objects tutorial folder).

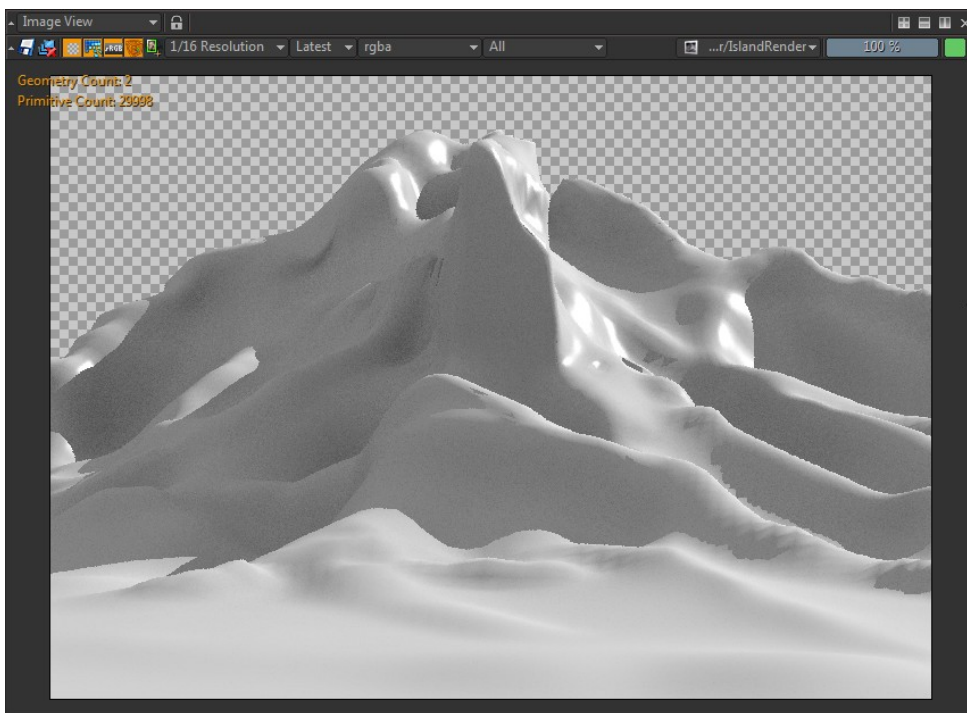


Rename them `grass_support` and `terrain` instead of the double name Clarisse gave them (this is because their object and layer name are the same). Instantiate them in the render context in order to see them in the Image View.



Instantiate using right clic drag and drop

In the render context, create a context named lighting and drag the light inside. Then add an *ambient occlusion* light, with *intensity* set to 0.3.

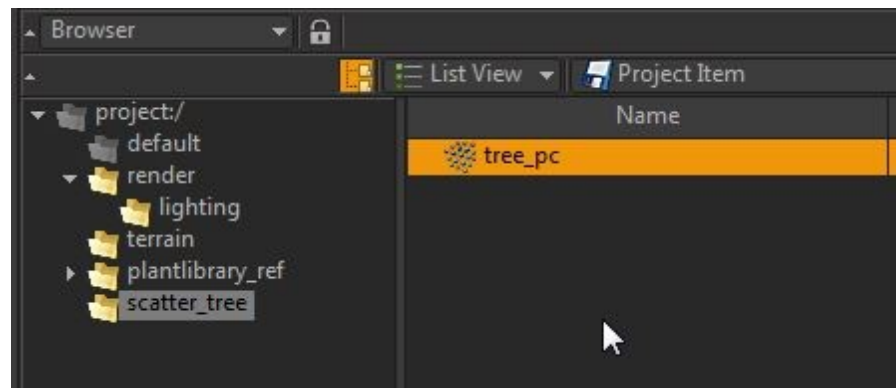


The terrain lighted in Image View

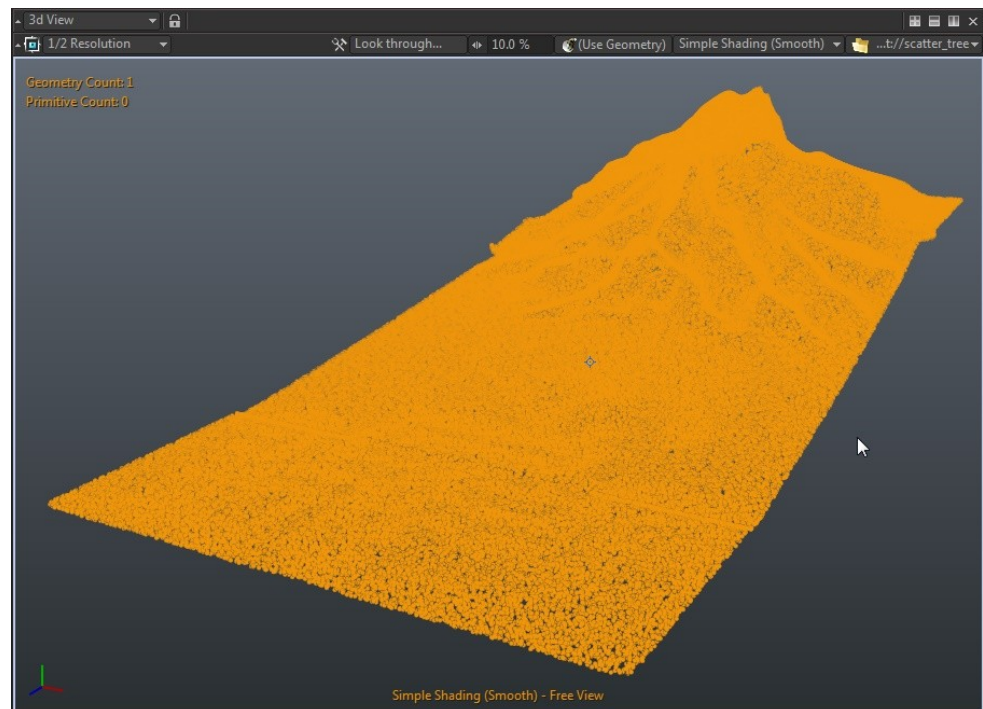
Go to the root of the project, import the plant library using **file>import>project...**, and choose `plantlibrary_ref.project` .

At the root of the project, add a new context named "scatter_tree". We'll build all the tree scattering setup inside this context, isolated from the render. Open a 3DView (you can replace the image gallery by a 3DView), and drag and drop the scatter_tree context on the title bar of the 3DView. This automatically locks the 3DView to this context, avoiding the widget to jump from one context to another when performing selections.

In the `scatter_tree` context, create a point cloud. Rename it `tree_pc` (it is a good convention to use specific extensions for your objects, like `_pc` for point clouds, `_mat` for materials, etc...).

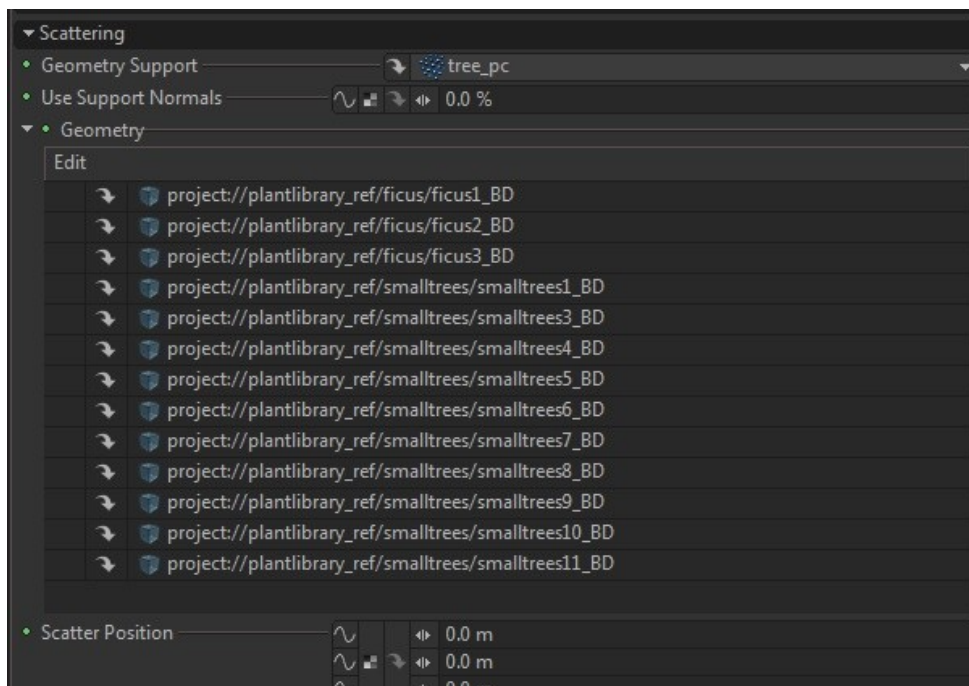


Set the *geometry* attribute to `project://terrain/terrain` and the *density* to 6 (be careful not to go higher...6 is already a very high value for a big object like this). You should see a point cloud in the 3DView, following the terrain shape.



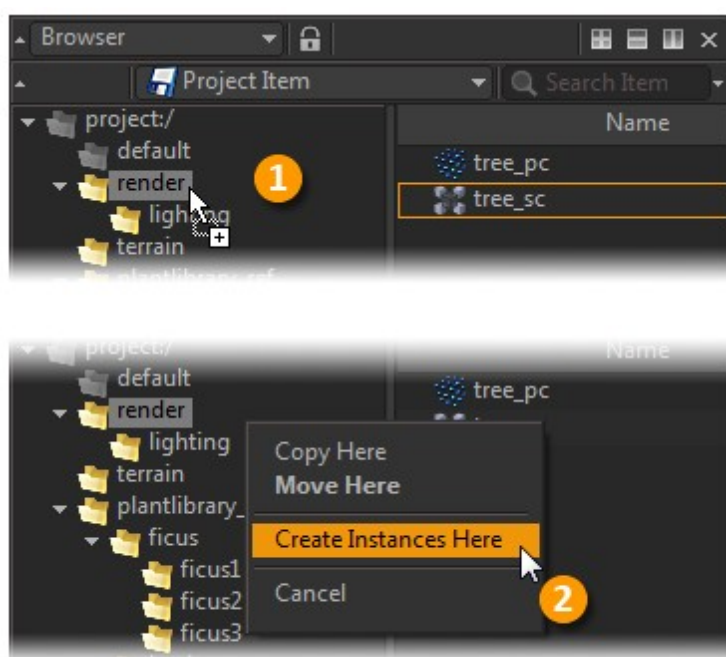
The `tree_pc` point cloud in 3d view

Now, create a scatterer object and rename it `tree_sc`. Set its geometry support to `tree_pc`, lock the Attribute Editor and drag and drop all `ficus` and `smalltrees` combiners in the geometry list (`plantlibrary_ref/ficus/ficus1_BD` , `ficus2_BD`, `ficus3_BD`, and `plantlibrary_ref/smalltrees/smalltrees1_BD`, `smalltrees3_BD` etc...).



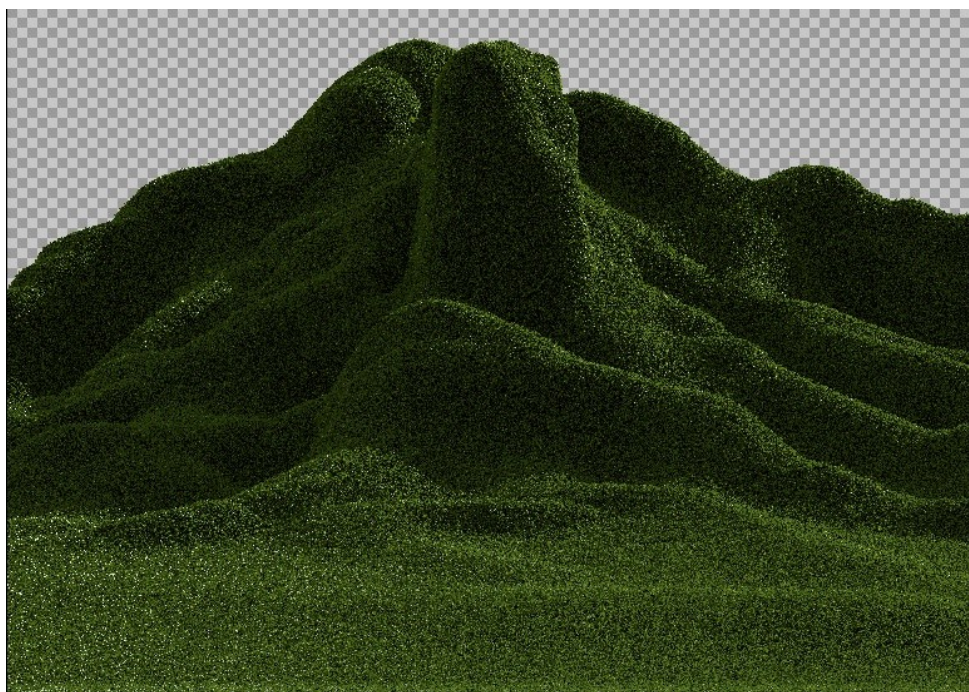
The `tree_sc` Geometry list

Simply instantiate `tree_sc` into the render context, to see the scattering system in the shot.



Right Click drand and drop tree_sc inside render context

You should now see thousands of trees covering the landscape.

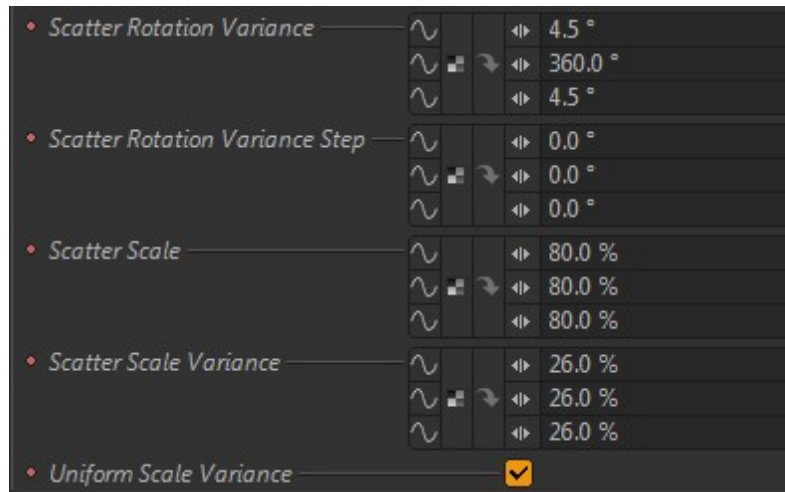


The tree_sc rendered inside the Image View

Preparing your scatterer for texturing

Of course, visually speaking, the tree distribution is too uniform...we'll control this using painted images maps, but first, let's add some automatic variation using some attributes of the scatterer.

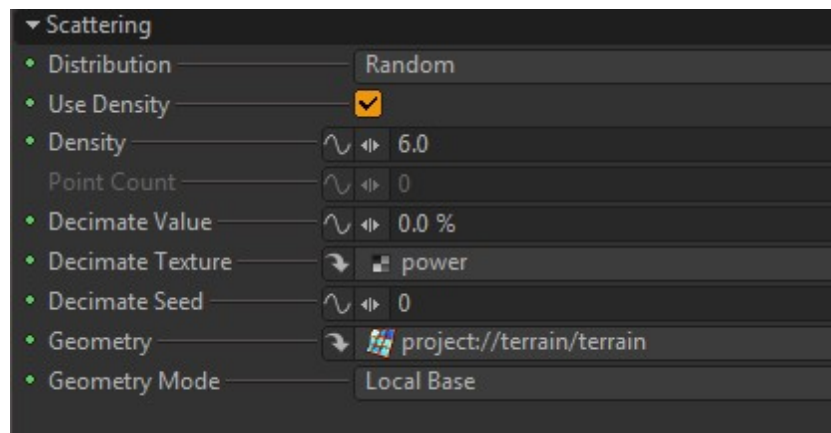
Set it like this:



Note

Note : at this point , let's use at least an antialiasing of 2 in the raytracer, because the look of massively scattered objects is wrong without a minimum of sampling. If you have to match a reference photograph, you should even use a light setup similar to your photo references and a rendering with the same resolution (using limited region to speedup the workflow) .

We need to vary the density of our point cloud. Select `tree_pc` and connect a *Power Texture* on the *Decimate Texture* attribute.

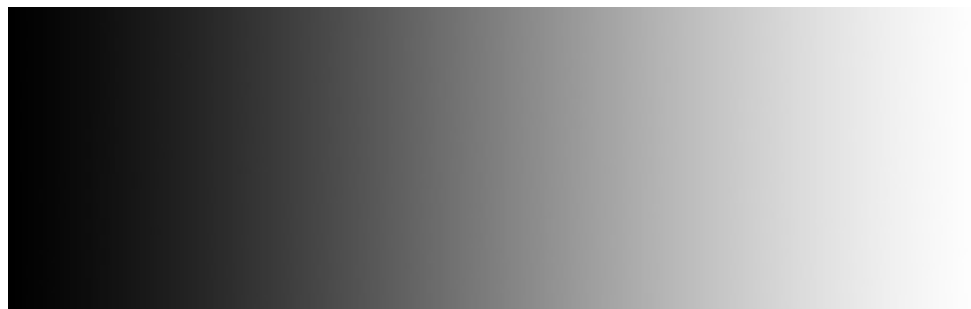


The power texture connected to the decimate point cloud attribute

Rename power to decimate_texture. This texture will be used later as an entry point between our point cloud *density* and the various textures we'll use. The texture changed nothing because, by default, the power output a black color. So it means, 0% decimate.

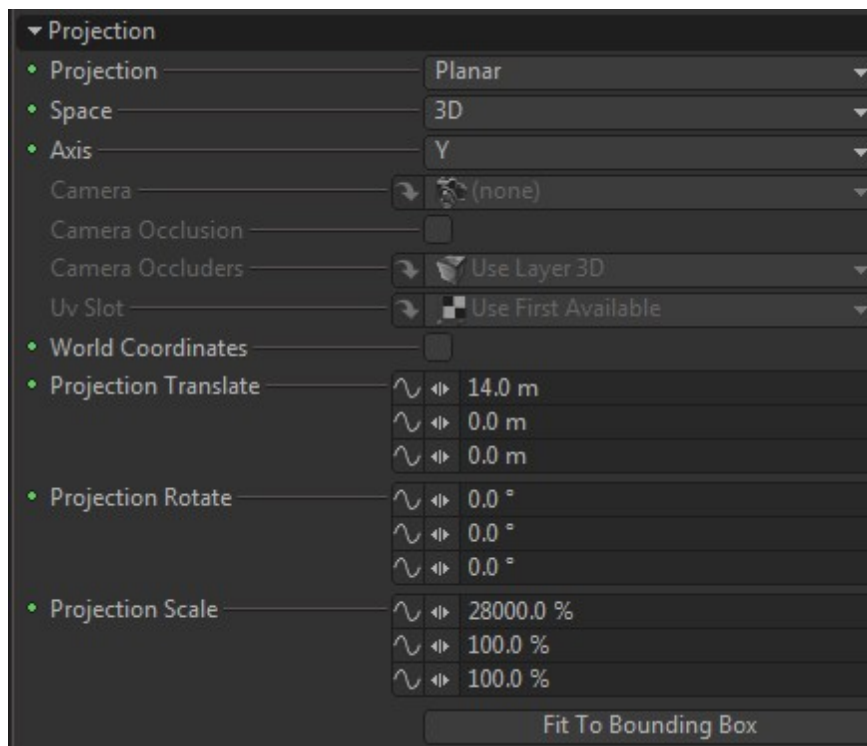
Now, we need to know how each gray value in the texture will affect the render. A very nice trick to achieve this is to use a grayscale ramp on the attribute, and see how it affect the look of our image. This way, we'll have an image reference which can be used in your favorite painting application to know exactly what value can be used in the various areas, depending on the look you want to achieve.

Open your favorite image painting application and create a black to white grayscale ramp (you can also use ramp.png in the texture folder).



The ramp texture

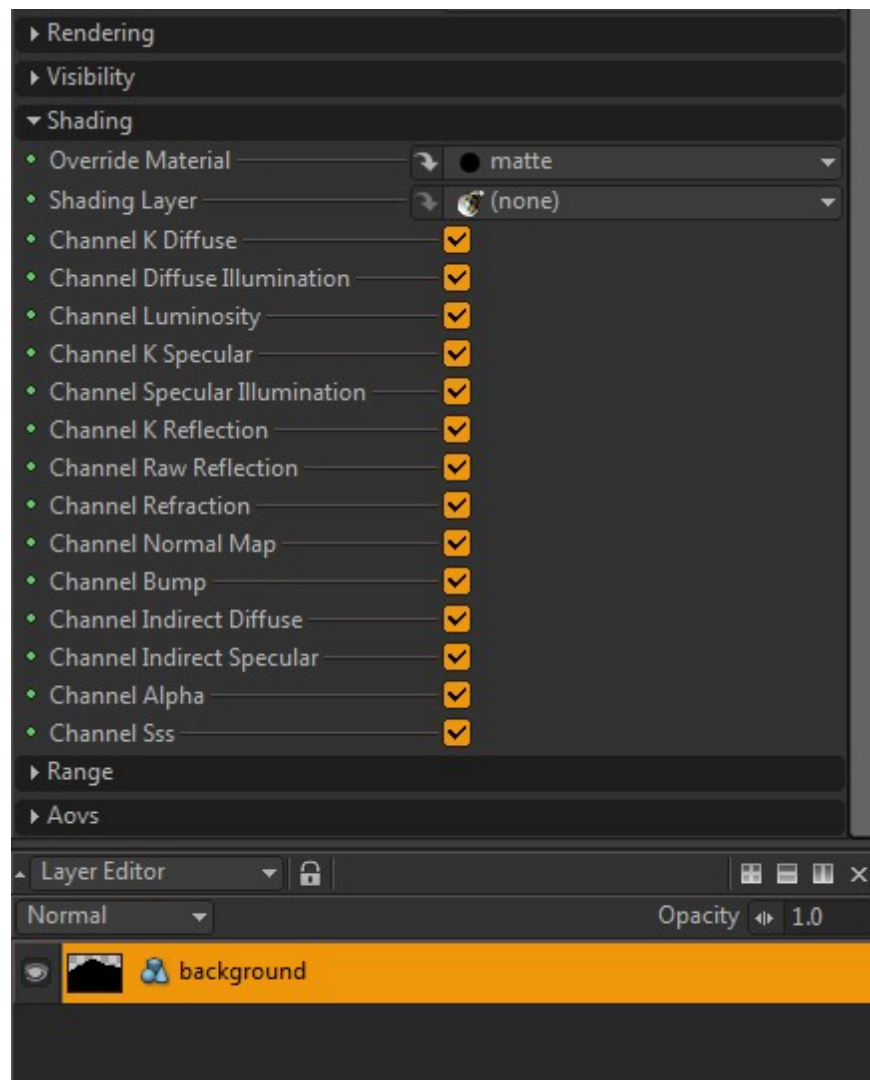
Import the ramp texture (using **File > Import > Texture > Map file...**) in your scatter_tree context and set the *projection* to :



The ramp_map projection attributes

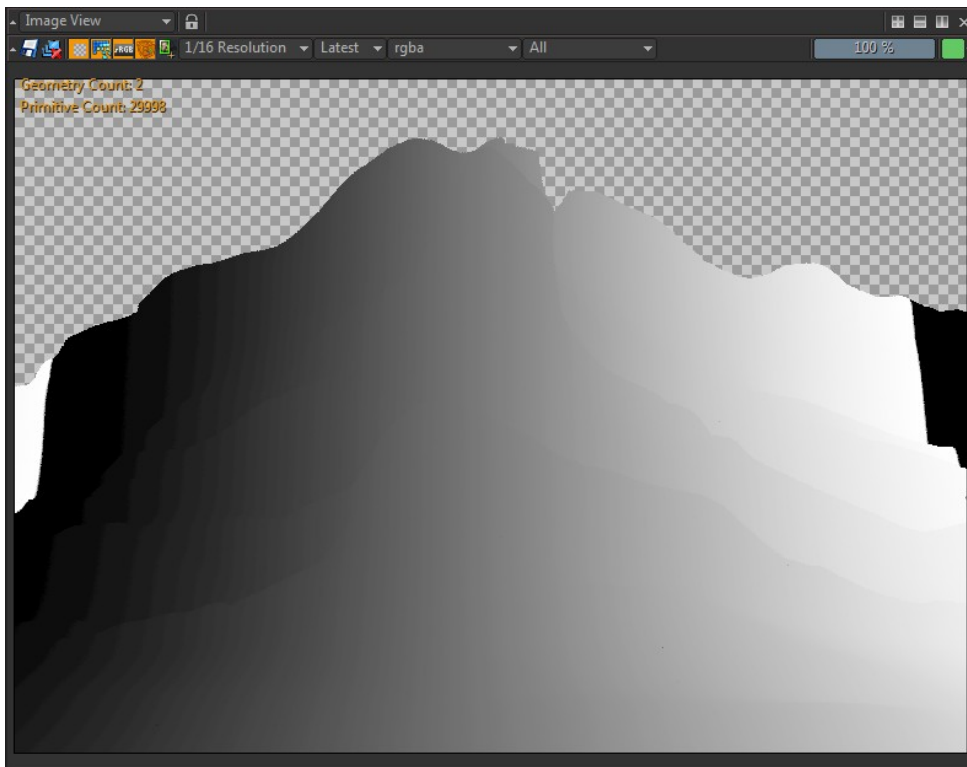
Now, in the render context, select your tree_sc object and move it to the root of the project so the tree are temporary hidden from the render.

In the layer editor widget, select the background layer of your render. Click on the *Override Material* attribute and create a matte material in it.



Matte material connected to the Override Material layer attribute

On the *Color* attribute of the matte material, connect `ramp_map`. You should now see a gray ramp on the landscape (if you used your own landscape geometry, change the `ramp_map` texture coordinates to fit the ramp). Save the image (with the little disk icon on the top of the image view) to `density_guide.png`.



The density guide image

Remove the matte override material of your 3DLayer, and bring back `tree_sc` from the root of the project to the render context.

Select `ramp_map` and uncheck the *linearize* attribute.

Connect the output of `ramp_map` to the *Input* attribute of `decimate_texture` (best way is to open the Material Editor, drag and drop `decimate_texture` and `ramp_map` to the workspace).



You still see a very high tree density on the terrain. This is because the point cloud density doesn't have a visually linear effect: you must use a high decimate value to see little changes in the image. But painting grayscales values between 80% and 100% are not very convenient. It is better to redistribute the map values in a non linear way to be able to paint more intuitively your density control map. This is why we used a *Power Texture* (renamed `decimate_texture`) between the map file and the point cloud *density* attribute.

Select `decimate_texture` and set the *Color* attribute to 0.05. The effect is much stronger now. Set the raytracer to 6 samples with a Blackman-Harris filter, and wait till the render is completed. Save the rendered image to `density_sample.png`.



The result of the density ramp map

Now, if you load `density_sample.png` next to `density_guide.png` in your painting application, you have a very useful visual guide to paint your density maps. You just have to pick the gray value corresponding to the needed density.

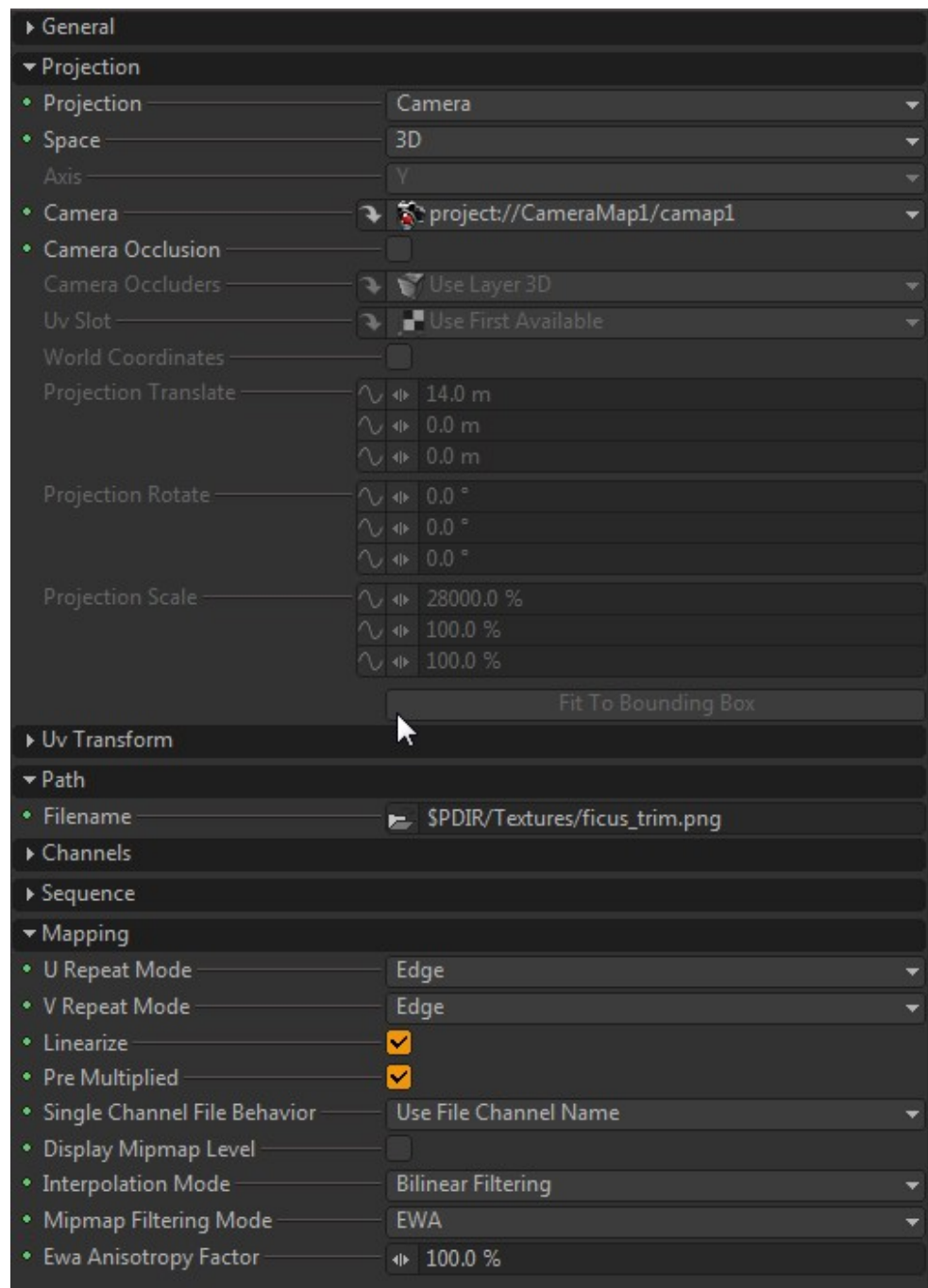
Texturing your scatterer

To paint your control map, using an UV projection, you can use a 3D paint software if you want. In this tutorial, we'll use camera projection mapping instead, so that any painting tool can be used. We could paint through our render

camera, but this is not a very flexible option if you want to change or animate the viewpoint. So let's create a camera covering the whole landscape.

Import the project "CameraMap1.project" in the root of your scene. It contains an advanced camera already setup (if you are not comfortable with camera mapping in Clarisse, this is a good time to read the camera mapping tutorial).

Return to `scatter_tree` context and select `ramp_map`. Rename it `decimate_map`, then set the projection to *camera*, the camera to `project:/CameraMap1/camap1`, U and V repeat mode to *Edge*, and replace the *path* filename by `figus_trim.png`.



The decimate_map attributes

Select `decimate_texture` and set the *Color* attribute to 0.5 0.5 0.5 (remember, it is a renamed *Power Texture*). The 0.05 value was used to better demonstrate the effect but you will get a more subtle result with a 0.5 value. Now, you can see in the Image View a much more realistic distribution of the tree, with no plant on the vertical cliff, more on some other zones etc...



The density map applied to the scatterer

We already set some random size variation using the *Scatter Scale Variance* attribute, but this time we'll use a map to control the size of each tree, using a texture on the *Scatter Scale* attribute. Copy and paste `decimate_map` and rename the copy `size_map`. Select `size_map` and replace the *path* `figus_trim.png` by `figus_size.png`. Select the `tree_sc` scatterer, and connect `size_map` texture to the *Scatter Scale* attribute.



The size map applied on the Scatter Scale attribute

Varying your shading using the instance texture

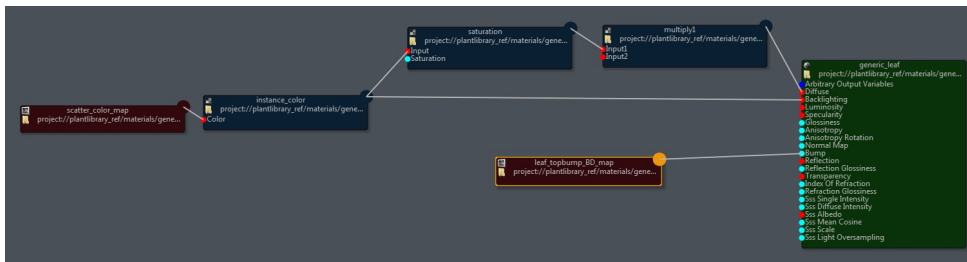
Every tree has the exact same color. We need a way to globally control the color variation of our trees. Select `generic_leaf` material in `project://plant_library_ref/materials/generic_leaf`, and open the *Material Editor*. Create an *Instance Texture* and connect it to the *Diffuse* and to the *Backlighting* attribute.

The *Instance Texture* allows you to individually control a scattered object material, depending on the texture coordinates of the support object (the point cloud, that inherits its coordinates from the terrain). Because a scatterer can be scattered by another scatterer multiple times, recursively, you must specify the hierarchy level to be used by the instance color. In this case, we must use the level 3.

Note

A good trick to find the last level, which is often the needed one, is to set the Color attribute to a red color, then use a high Parent level value, like 10. The object will turn black. Then gradually step down the parent level value until the red color appears.

Go to the `scatter_tree` context, copy `size_map` and paste it in the `generic_leaf` context . Rename the copy `scatter_color_map`. Change the `path` to `figus_color.png` and enable *linearize* (because it is a sRGB color map). Drag and drop `scatter_color_map` in the Material Editor and connect its output to the `Color` attribute of `instance_color`. You should now see nice color variations in the Image View.



The generic leaf material controlled by Instance Texture

To get more control, insert "saturation" and "multiply" nodes between `instance_color` and `diffuse`. Set the saturation to -20% and multiply *Input2* to 2. This concludes the main part of this image.

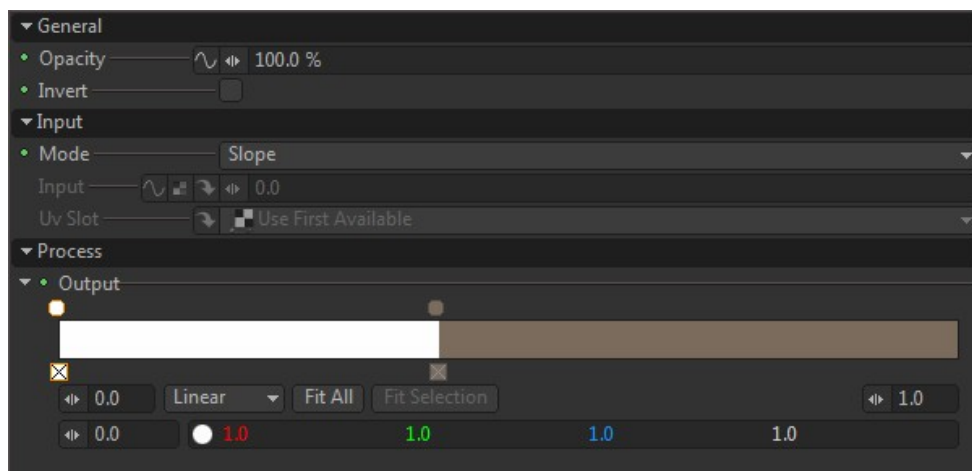


Scattered trees color variations using the Instance Color Texture

Texturing the ground

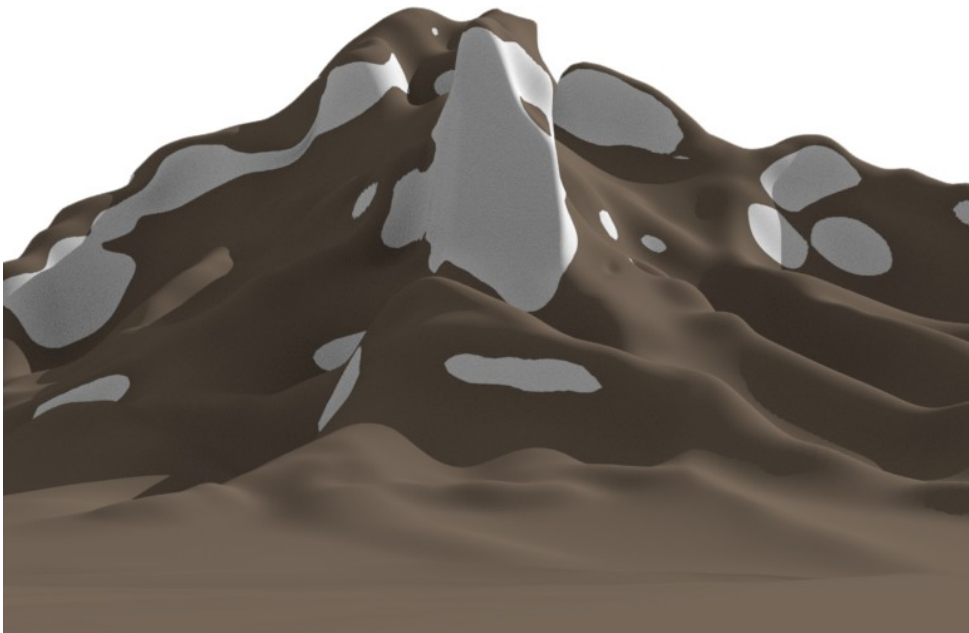
Even if it is mainly covered by vegetation, we need to apply some texture on the terrain itself. Drag and drop `tree_sc` from `render/scatter_tree` at the root of the project in order to temporary remove it from the render.

In the terrain context, create a materials context. In this context, create a *standard material*, renamed `terrain_mat`, drag and drop it on terrain and the `grass_support` in the Image View. In the *Diffuse channel*, create a *Gradient Texture*. Switch the *Mode* attribute from *Incidence* to *Slope*. Set the first key color to 0.6 0.6 0.6, place a second key to 0.65, with a color of 0.2 0.15 0.11, and switch the interpolation mode from linear to Step .



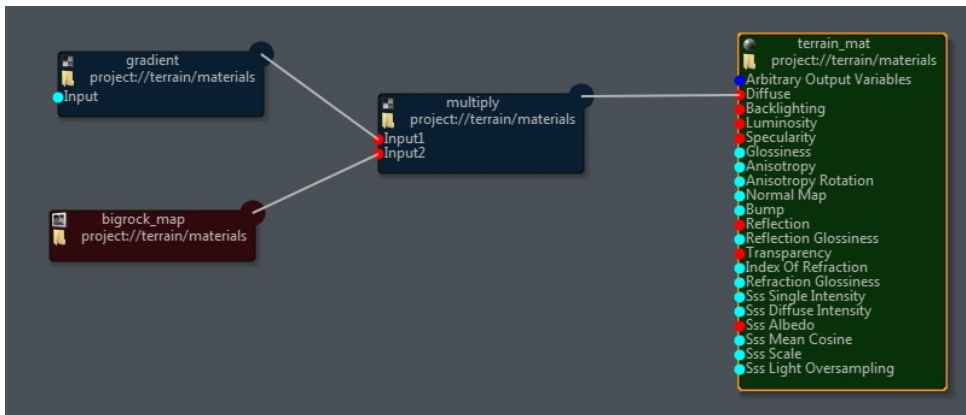
The Gradient texture attributes

You should get a brown terrain with white cliffs.



The terrain and grass support with gradient applied

Import the image Textures/bigrock.png using **File>Import Texture>Map File**, and drag and drop the Map File in the *Material Editor*. Add a *Multiply* Texture, connect the *Input1* attribute to the gradient output, *Input2* to bigrock_map, and the output to the *Diffuse*.

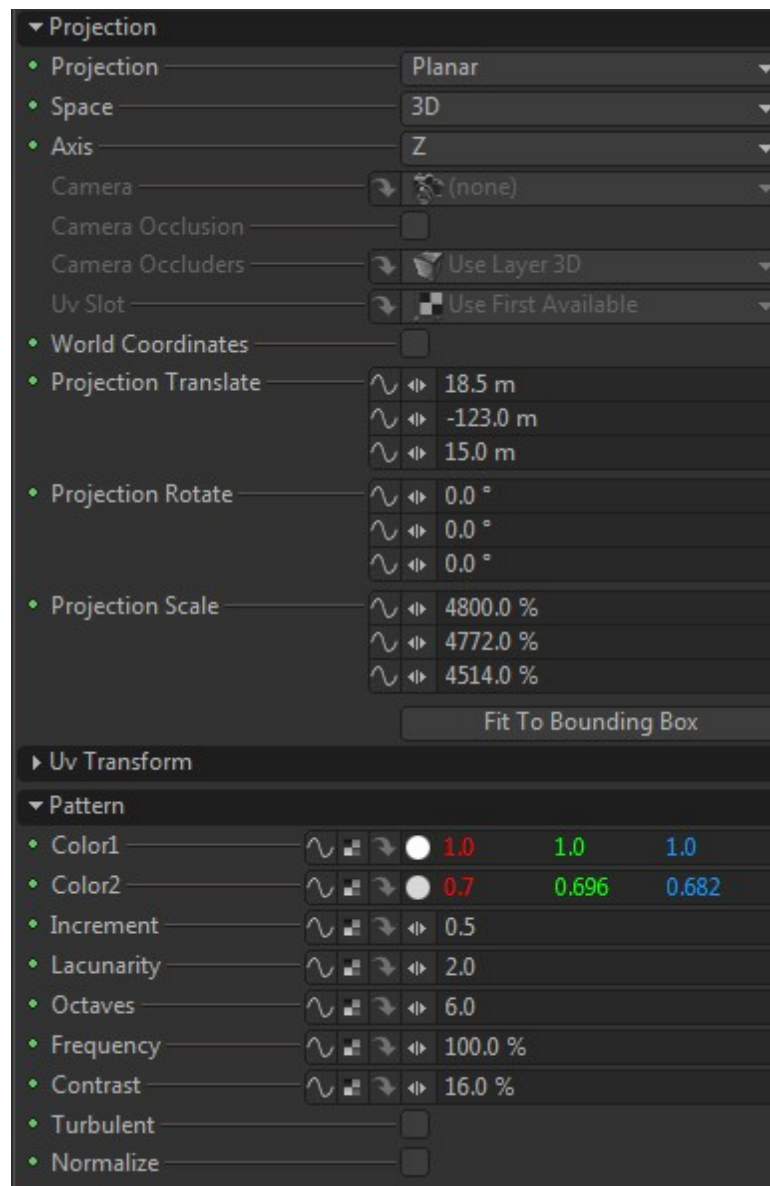


Set bigrock_map attributes as follows:

Projection : Cubic

Projection Scale : 5000% 5000% 5000%

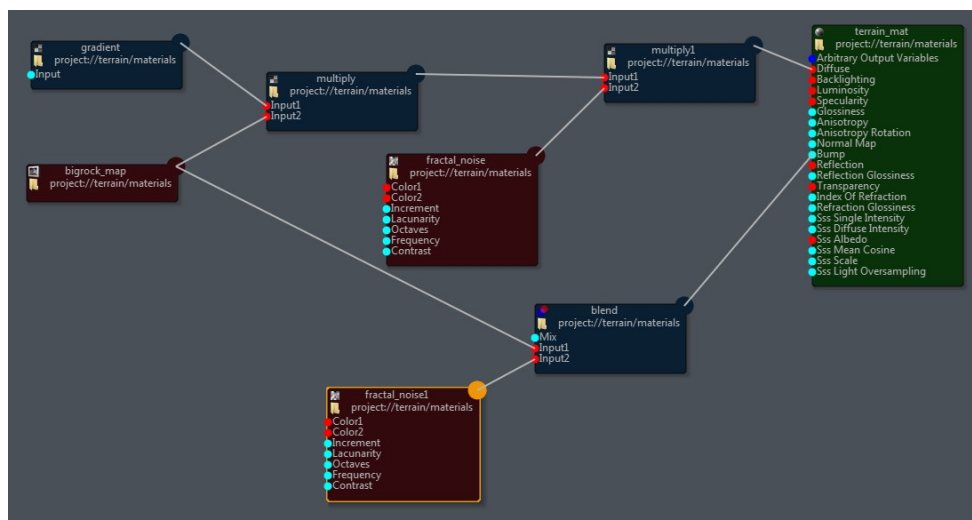
To add detail, create a *Fractal Noise Texture*. Set it like this:



Fractal Noise attributes

Insert a *Multiply Texture* between the already present multiply node and the *Diffuse*. Connect the output of fractal noise to the *Input2* of multiply1.

Connect `bigrock_map` to the *Bump*. In `terrain_mat`, change the *Bump* value to 50cm. Now, insert a *Blend Texture* between `bigrock_map` and the *Bump*. Be careful, by default it connects the *Mix* input. Disconnect this one and connect `bigrock_map` to *Input1* instead. Create a new *Fractal Noise Texture* and connect it to blend *Input2*, and change the *mix* to 50%.



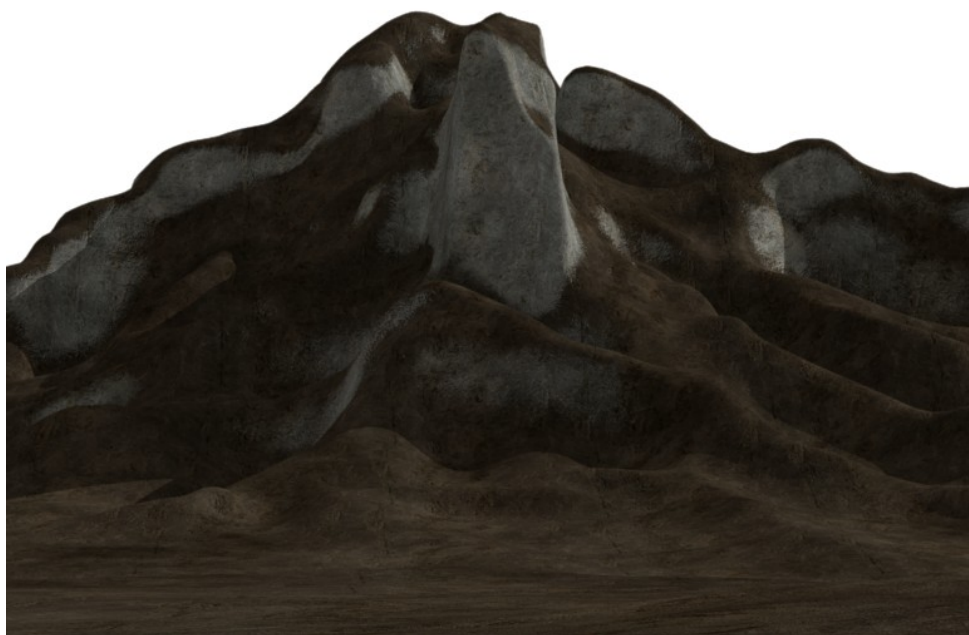
Set the *fractal_noise1* texture like this:

Projection: Planar

Turbulent: off

Normalize: off

Our terrain material is completed.

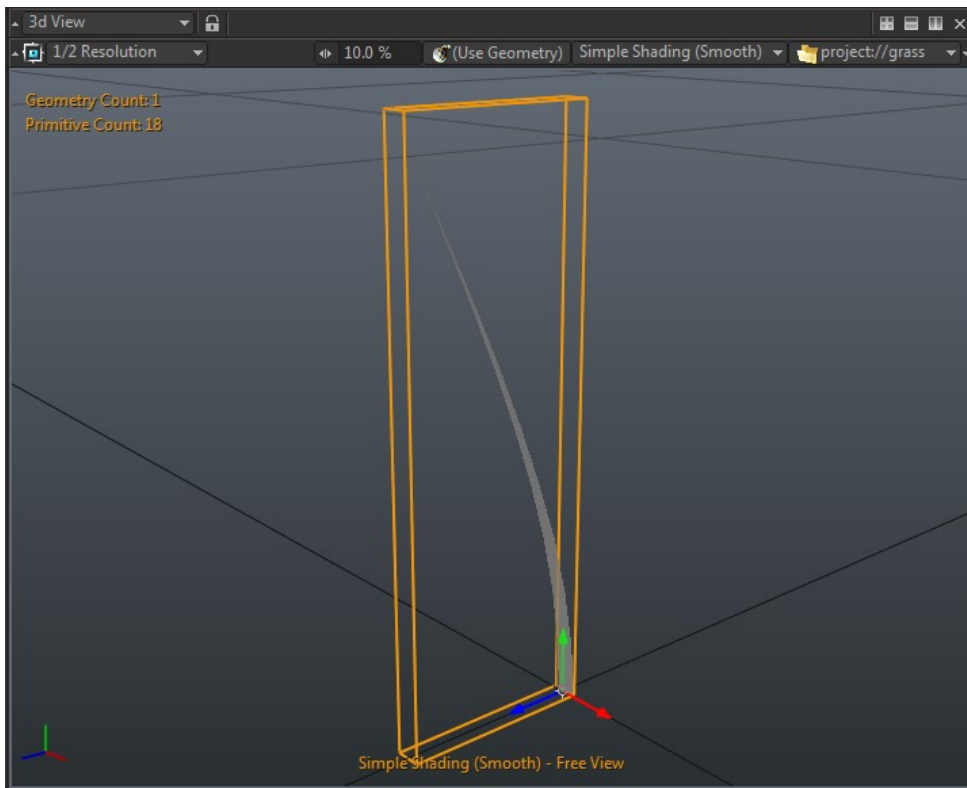


The final terrain material rendered

You can move back `tree_sc` to the render context to see it with the vegetation.

Creating the grass clumps

Now, we need to add some grass on the foreground. Go to the root of the project and create a context, rename it `grass`. Open a 3DView, drag and drop the `grass` context on the 3DView title bar to lock the widget on it. Import the `grass.obj` object and frame it in the 3DView.



The grass object displayed in the 3D View

Create a *Standard Material*, rename it `grass_mat`, and assign it to the grass. Set the *Diffuse* color to 0.23 0.32 0.08.

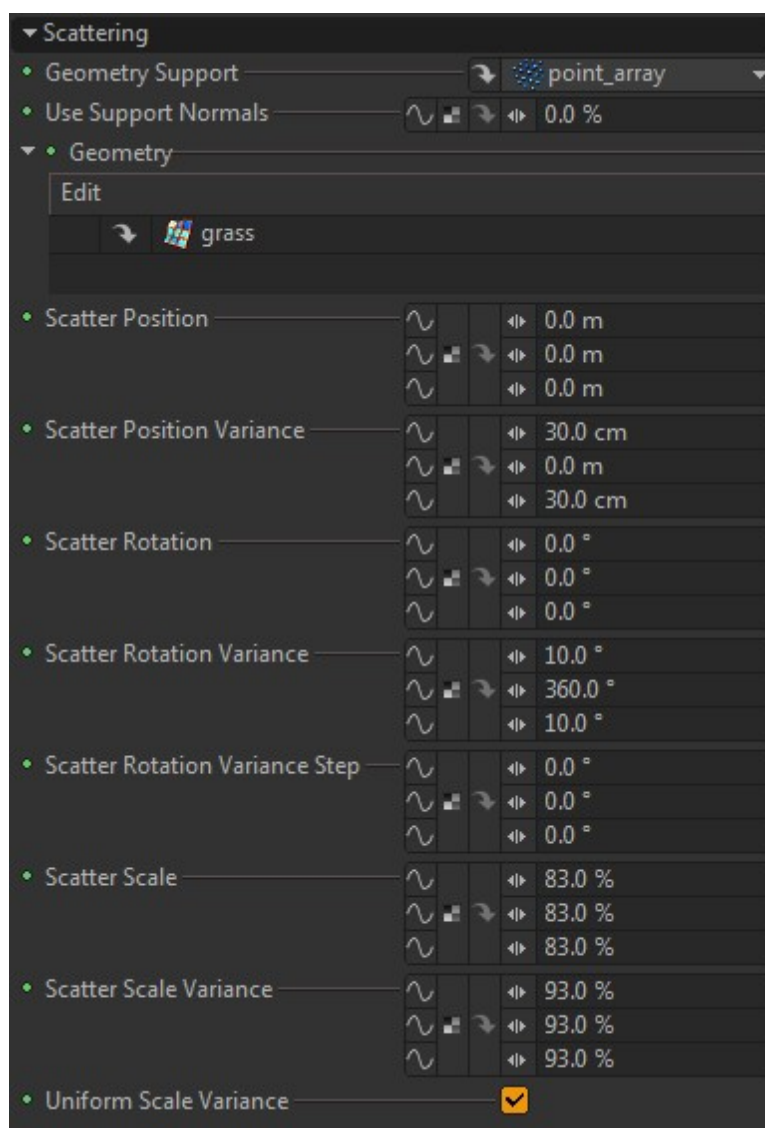
Now, create a *Point Array* object. Set its attributes to:

Size : 10cm 10cm 10cm

Count : 20 1 20

Add a *Scatterer*, rename it `grass_clump1`. Drag and drop the grass object in the scatterer *Geometry list*, and the *geometry support* to `point_array`.

Set the other attributes as follows:



The scatterer attributes

Now, copy and paste grass_clump1 twice to get a total of three clumps.

On grass_clump2, set the following attributes:



On `grass_clump3`, set the following attributes:



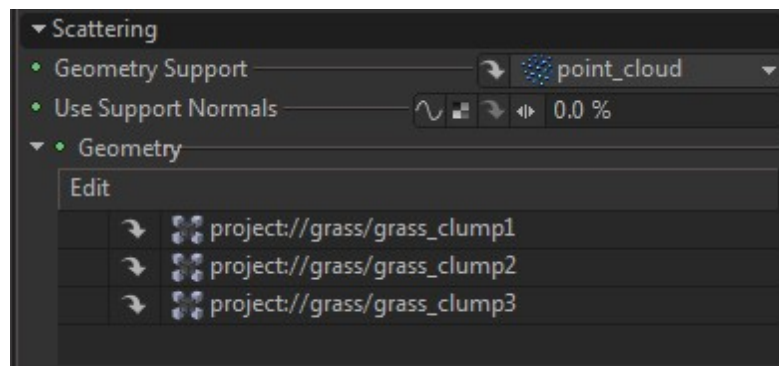
We now have three different grass clumps ready to be scattered on our ground.

Scattering the grass

Go back to the project's root and create a context named `scatter_grass`. Go in that context and drag and drop the context on the 3DView title bar.

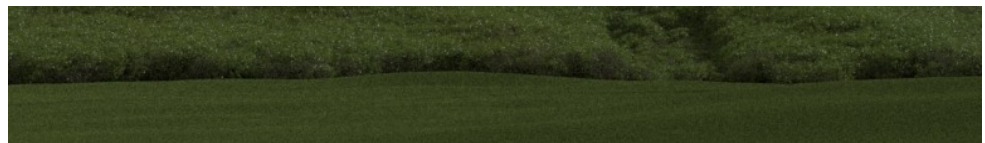
Create a point cloud. Set its *density* to 100, and the *geometry* to `project://terrain/grass_support`. Uncheck the *display visible attribute* (currently point clouds are one of the few things displayed using hardware OpenGL, which slows down Clarisse).

Create a *Scatterer*. Rename it `grass_scatter`. Connect its *Geometry Support* to the point cloud and drag and drop the three grass clumps in the *Geometry list*.



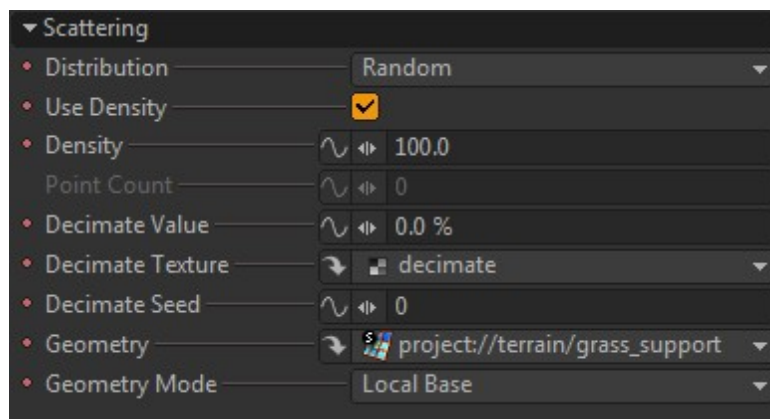
grass_scatter attributes

You can now instantiate `grass_scatter` in the render context to see your grass in the shot.



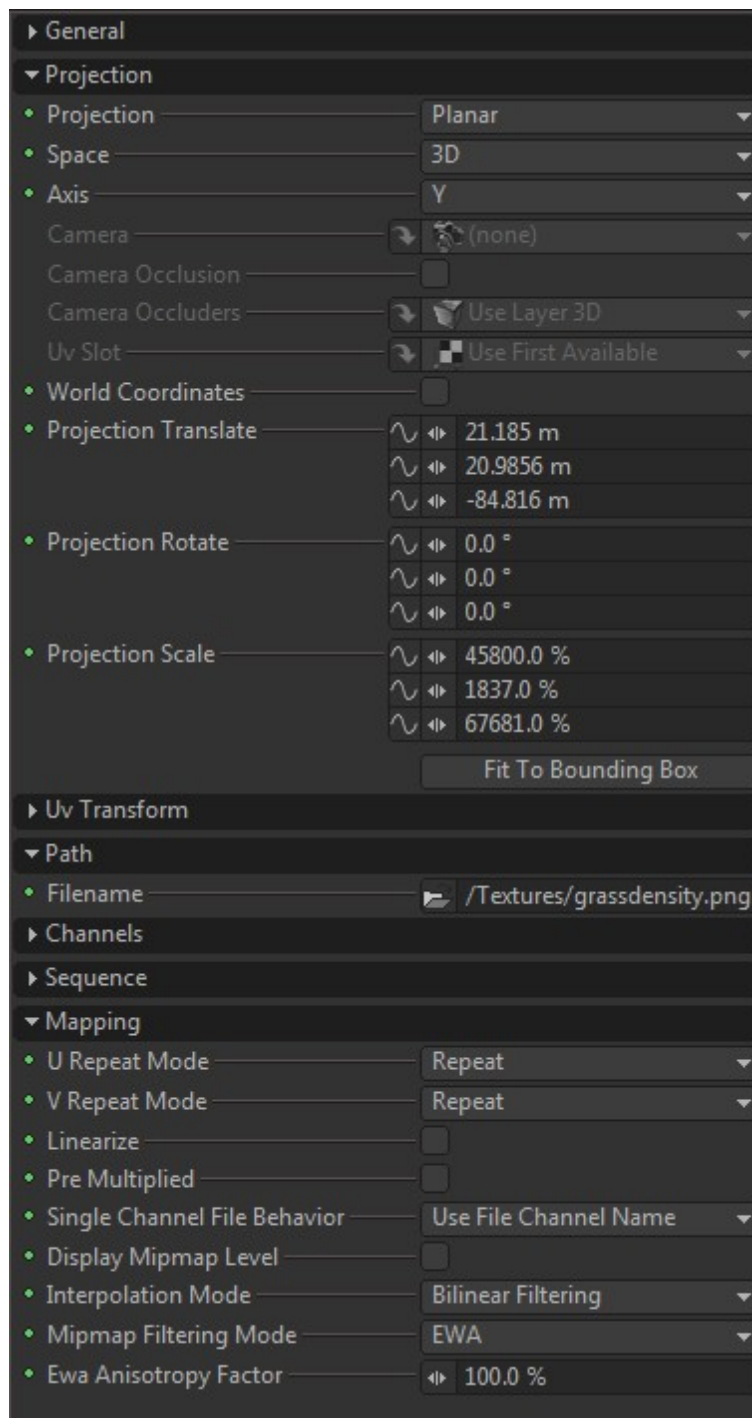
A limited region render of the grass

Return to the `scatter_grass` context and select `point_cloud`. Connect to the *Decimate Texture* attribute a new *Constant Color* Texture. Rename it `decimate`. This entry point will make it easier to manage our texture graph.



point_cloud attributes with the connected decimate texture

Import `grass_density.png` as a *Map File* and set its attributes like this:



grassdensity_map attributes

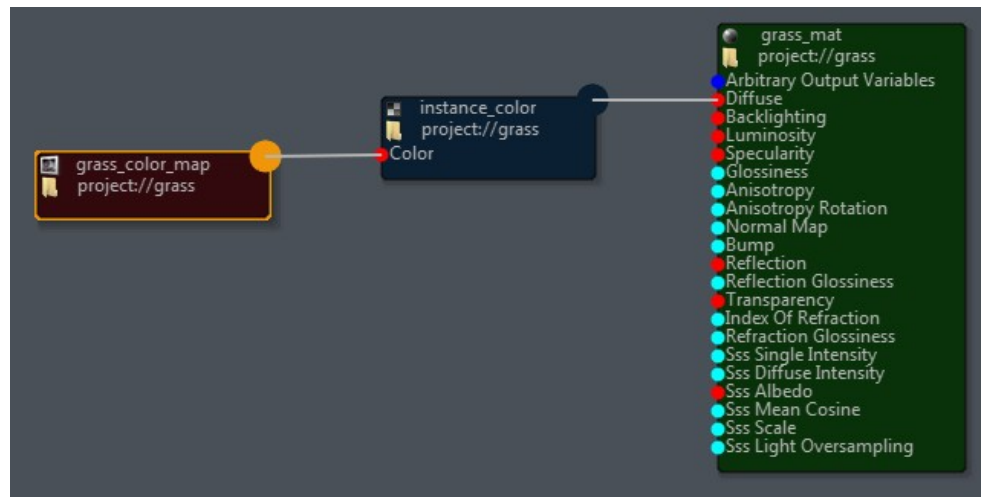
Connect grassdensity_map to the color input of decimate. You should now see grass density variations on the render.



The grass rendered with density map applied

The color is too uniform. We'll vary it using the same technique we used for the trees: the *Instance Color Texture*. Go to the grass context and select `grass_mat`. Open the Material Editor.

Import `grass_color.png` and drag and drop it in the Material Editor workspace. Create and connect to the *diffuse* an *Instance Color Texture*. Set the *Parent Level* to 2 and connect the *Color* attribute to `grass_color_map`.



Set the attributes of `grass_color_map` as follows:

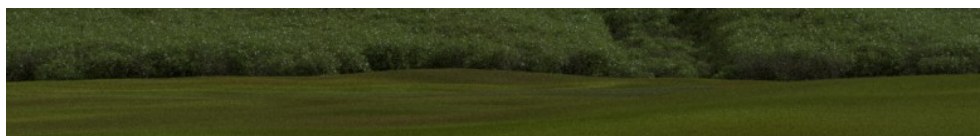
Projection: planar

Axis: Y

Projection Translate: 21.185m 20.9856m -84.816m

Projection Scale: 45800% 1837% 67681%

Connect a *Multiply Texture* to the *Backlighting*, then the multiply *Input1* to the output of `instance_color`. Set the multiply *Input2* attribute to 0.127 0.18 0.115.



The instance color texture result

The grass is ready.

Adding props and palmtrees

The purpose of this tutorial is not to describe how to shade all the basics materials, so we'll directly import the next object with already made materials.


Go to the project's root and import `cabin_ref.project`. In the `cabin_ref` context, drag and drop `cabin` directly where you want to place it in the image view. It automatically creates an instance of the cabin in the render context. If you want it at the same place as shown in the tutorial scene, select the instantiated cabin and set the *Transform* this way:

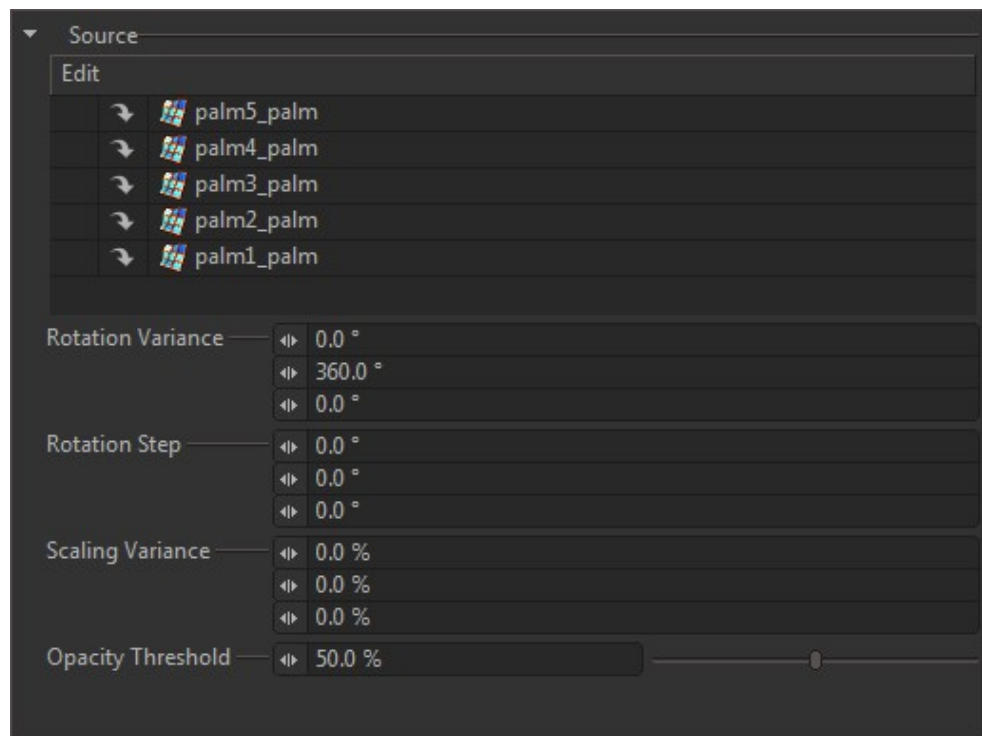
Translate: -4.7218m 18.7065m 596.2479m

Rotate: 0 -10.5 0

Scale: 150% 150% 150%

Let's import palm trees the same way. Go to the project's root and import `palmtree_ref.project`. This time, we'll use a different technique to place our objects on the terrain. Of course we could instantiate them one by one like we did for the cabin, but the *Clone Tool* will be more appropriate.

Enable the *Clone Tool*  from the toolbar and open its options panel (**windows>tool options**). You can see a list, very similar to the *Scatterer Geometry list*. Drag and drop all palm trees into the list. You can also set the *Rotation Variance* to 360 on the Y axis .

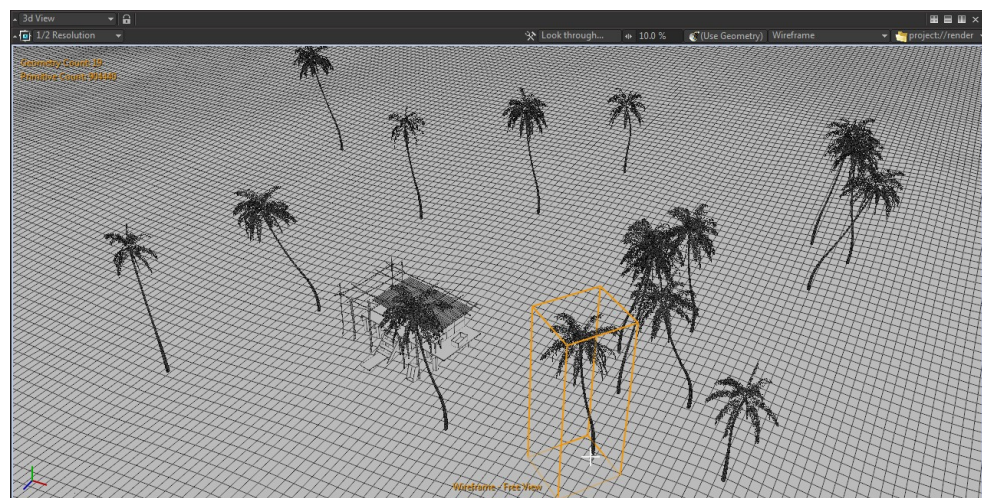


The tools option windows

Go to the render context and create a palmtree context. Go in there.

Note

For maximum interactivity, it is better to temporary move tree_sc and grass_scatter (grass and trees scatterers) away from the render context. That way, you will be able to manually place all your foreground objects more easily, without billions of polygons of the grass and trees.



Clone 3D in action

Now, in the Image View or 3DView, according to you, click where you want to place your palm trees. Each time you click, Clarisse randomly picks up an object from the list, and instantiate it.



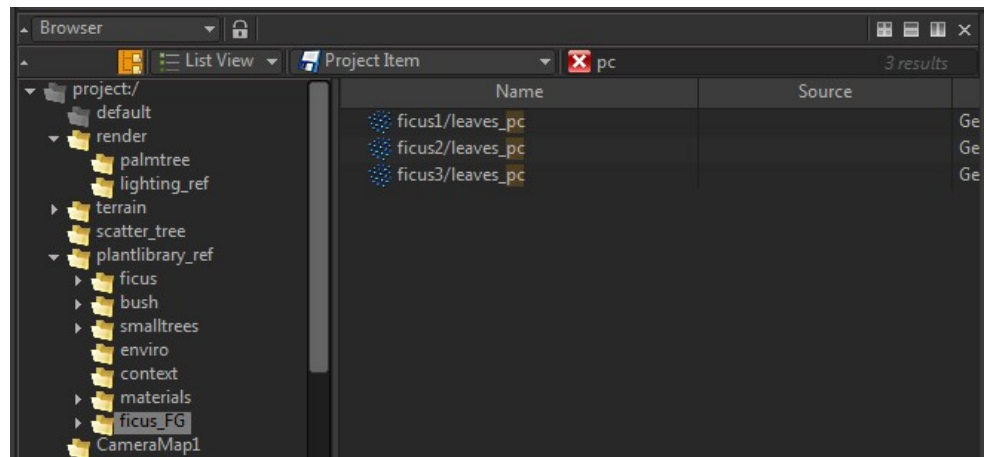
The rendered palm trees

More plants and trees

Let's do the same with more plants we used for the background. Nevertheless, they were tuned to look good from a far distance. In theory, if the object is perfectly setup, you should get a perfect result from both a close up and a fair distance. Often you want more control on the foreground objects, so let's make separate versions of the tree for the foreground.

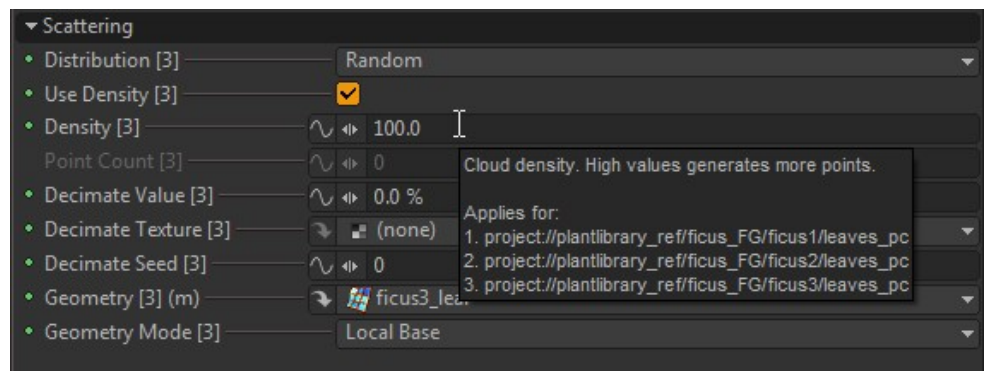
Go to `plantlibrary_ref` context, and copy/paste the `figus` context. Rename the copy `figus_FG`.

Go to `figus_FG` context and search for `pc` in the browser.



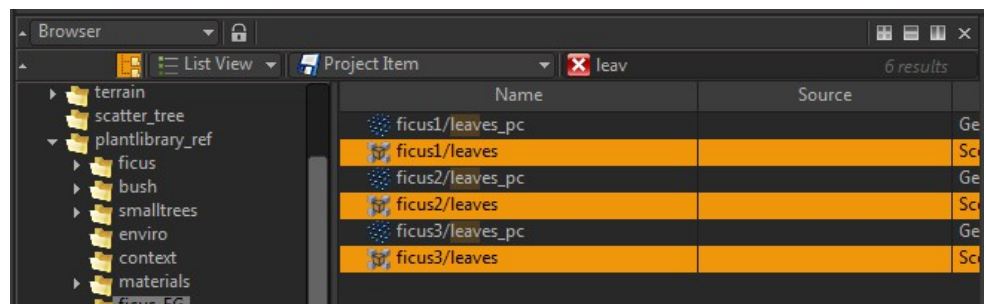
The browser search feature

Select the three point clouds (ficus1/leaves_pc, ficus2/leaves_pc, ficus3/leaves_pc) and change the *Density* to 100, to get more leaves.

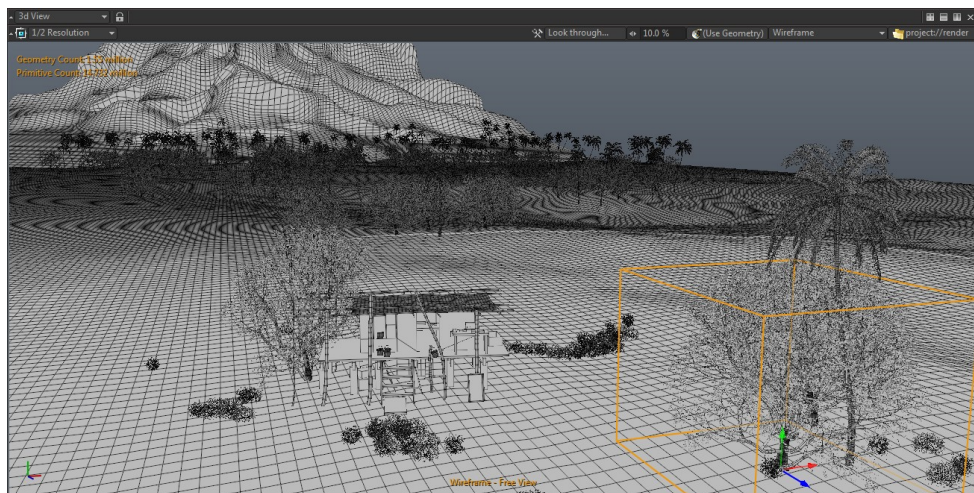


Multiple selection density modification

Do the same for the three leaves scatterer, except this time you'll change the *Scatter Scale* attribute to 70%.



We now have more detailed trees for the foreground. Use them with the clone tool to build a nice layout. If you are not satisfied with the placement, rotation or scaling of some of your trees, you can edit any of them manually using the *Transform Tools*.



Finish the work with smaller plants located in the bush plant library context.



The final layout rendered

Lighting and compositing

We are getting close to the end and this is time to complete the lighting. Go to the render context and delete the lighting context. Load

lighting_ref.project to replace it by a premade one.



The final lighting applied

This lighting setup is very simple: the sun is simulated by a spot, far away from the scene. Its soft cone angle gives more light to the center of the scene as if the sun was passing through the clouds. A *Fractal Noise* in the color will complete this effect. The light rays coming from the sky are created with an *Ambient Occlusion* light, using a cold gray color. A *Global Illumination* light adds some bounces to get more realism (especially on the house. The vegetation alone can be very convincing without GI).

To complete the project, we added some foreground plants very close to the camera so that we get nice depth of field effect (the DOF is already enabled in the tutorial camera, but if you created your own, it is located in the camera attributes. Use a *F Stop* of 1.8 and *Focus Distance* of 100 m). In the completed project, these objects are isolated in a different 3d layer so you can cheat and make it darker with 2d filters.



The final image

A 2D sky background is added too, with a fog effect done with a *Z_depth* material override.

Technical Specifications

Isotropix assumes no responsibility or liability for any errors or inaccuracies that may appear in this document and this document is subject to change without notice. The content of this document is furnished for informational use only.

Minimum System Requirements

- Intel or AMD based x86-64 CPU supporting SSE2
- 2 GB RAM)
- 500 MB disk space available
- **64-bit operating system**
- Windows Vista Business SP2 or Mac OS X 10.6 (Snow Leopard) or Linux Red Hat/Centos 6
- OpenGL 2.0 compliant graphics card with 32 MB of video memory
- Display supporting 1280 x 1024 pixel resolution in 24 bit color
- Network card
- Three-button mouse

General Features

- | | |
|--|--|
| • Full construction history | • Drag and Drop |
| • Fully procedural object graph | • Ascii or binary object representation |
| • Fully multithreaded evaluation | • Contextual attribute specialization |
| • Virtually texture and animate any attributes | • Event based attribute update notification |
| • On demand dependency graph evaluation | • FCurve driven attribute value |
| • On demand automatic data loading | • Texture driven attribute value |
| • On demand automatic data evaluation | • Gradient driven attribute value |
| • 64-bit floating point precision | • Curve driven attribute value |
| • Cyclic dependencies safeguards | • Attribute driven attribute value |
| • Unified resource management | • Object instancing and attribute overriding |
| • Realtime redundant data tracking | • Object can nest other objects |
| • Realtime file tracking | • Objects can be static |
| • Asynchronous object creation | • Objects can be read-only |
| • Asynchronous resource creation | • Objects can be hidden |
| • Asynchronous attribute value update | • Objects can be private |
| • Slot based selection | • Objects supports user tags |

• Command based undo/redo	• Dynamic class based object description
• Cut/copy/paste	• Project file inclusion

File Formats

3D File Formats

• Alembic	• LightWave 3D Motion (MOT)
• Wavefront OBJ	• LightWave 3D Scenes (LWS)
• GoZ**	• LightWave 3D Deformation (MDD)
• LightWave 3D Objects (LWO)	

** Available to maintenance subscribers some time in 2013

2D File Formats

• TIFF	• JPG
• OpenEXR	• TGA
• PNG	• HDR
• DPX	• Cineon
• FITS	• BMP
• ICO	• RMan Zfile
• PIC	• DDS
• SGI	• Maya IFF
• Field3d	• WebP
• PSD	• RLA
• Ptex**	

** Available to maintenance subscribers some time in 2013

User Interface

• Fully customizable layout
• Multiple user interface schemes
• User configurable metric
• Custom layout presets**
• Application Color space modes (linear, sRGB)

** Available to maintenance subscribers some time in 2013

Editors and Viewers

• Explorer	• Layer Editor
• Browser	• Tools
• 3D View	• Tool Options
• UV View	• Log
• Material Editor	• Timeline
• Material Linker	• History Editor

- | | |
|--------------------|-----------------|
| • Image View | • Color picker |
| • Image Gallery | • Resource View |
| • Attribute Editor | • Variable View |
| • Graph Editor | • Progress View |
| • Script Editor | |

Geometry and Scene Object

- | |
|---|
| • Implicit, Polygonal, Subdivision Surface, Curve geometry support |
| • Extensive subdivision surface support (linear, smooth/corners/borders) |
| • Extensive UV interpolation support (linear, smooth/corners/borders/discontinuities) |
| • Box |
| • Cylinder |
| • Fur (linear, curved, ribbon, smoothed) |
| • Point Array, Point Cloud, Point Volume |
| • Polyfile |
| • Polygrid |
| • Polysphere |
| • Sphere |
| • Combiner |
| • Scatterer |
| • Deformer modifier stack |
| • Texturable displacements |
| • Per shading group visibility |
| • Density based point cloud generation on any object (combiners, scatterers included) |
| • Recursive object scattering |

Animation

- | | |
|---|----------------------------|
| • FCurves support | • FCurve modifier stack** |
| • Linear interpolation | • Animation modifier stack |
| • Stepped interpolation | • Orient Constraint |
| • Hermite (TCB) interpolation | • Parent Constraint |
| • Bezier interpolation | • Point Constraint |
| • Virtual Keyframe preview | • Scale Constraint |
| • Animatable Pivot (translate, rotate, scale) | • Target Constraint |
| • User configurable rotation order | • LightWave 3D MOT Player |

** Available to maintenance subscribers some time in 2013

Compositing

Image

- | |
|---|
| • 32-bit/16-bit/8-bit precision per channel for the full image pipeline |
|---|

- Unlimited number of channels
- Image Canvas and Image Offset
- Dynamic Mipmapping
- Filtering (nearest, bilinear, bicubic, trilinear, EWA)
- Image space object picking
- Layered image resulting from a layer stack
- Layer 3D
- Layer file
- Layer image, layer referencing an image from the project
- Layer color
- Image Filter modifier stack

Blendings

- | | |
|------------------|----------------|
| • Normal | • Linear dodge |
| • Alpha replace | • Overlay |
| • Alpha add | • Difference |
| • Alpha subtract | • Hue |
| • Alpha divide | • Saturation |
| • Add | • Color |
| • Multiply | • Brighthness |
| • Screen | |

Filters

- | | |
|----------------------|-----------------|
| • Add | • Invert |
| • Brightness | • Multiply |
| • Color Space | • Noise |
| • Contrast | • Subtract |
| • Cut Off | • Defocus Blur |
| • Gamma | • Gaussian Blur |
| • Hue and Saturation | |

Rendering

- Per object rendering flags
- Material Override at object, combiner/scatterer, 3D layer level
- Interactive raytracer
- Interactive panoramic renderer (360 degrees)
- Interactive previz renderer
- Image space object picking
- Subdivision surfaces render modes: fixed and on the fly tessellation, surface evaluation
- Motion blur
- Depth of field**
- Layer file

- Renderers decouple geometry antialiasing from shading antialiasing
- Command line renderer

*** Available to maintenance subscribers some time in 2013*

Lighting

- | | |
|-------------------------------|--|
| • Per object visibility flags | • Dome |
| • Per object light linking | • Geometry** |
| • Per light object visibility | • Global Illumination Irradiance Cache** |
| • Ambient | • Global Illumination Monte Carlo |
| • Ambient Occlusion | • Linear |
| • Area | • Point |
| • Distant | • Spot |

*** Available to maintenance subscribers some time in 2013*

Shading and Texturing

- Multiple built-in animatable, texturable, materials (uber, lambert, phong, blinn, matte, gradient...)
- Per material Reflection Group and Refraction Group settings to explicitly specify what the material "sees" during evaluation
- Multiple built-in texture (perlin noise, worley noise, map, checker, blend, fresnel, color...)
- Textures supporting projection (Planar, Cylindrical, Spherical, Cubic, Camera, Parametric...)
- Subsurface scattering
- Mip Mapping and on demand mip map generation
- Any Clarisse image (even with layer 3d) can be used as texture map

Licensing

- Network-based licensing

Third Party Licenses

This chapter lists third party libraries used in Clarisse, along with their licenses

FLTK

Clarisse GUI Library is based in part on the work of the FLTK project (<http://www.fltk.org>).

OpenImageIO

OpenImageIO and all code, documentation, and other materials contained therein are:

Copyright 2008 Larry Gritz and the other authors and contributors. All Rights Reserved.

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 in the documentation and/or other materials provided with the distribution.
- Neither the name of the software's owners nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY

THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

(This is the Modified BSD License)

Alembic

TM & © 2010-2012 Lucasfilm Entertainment Company Ltd. or Lucasfilm Ltd. All rights reserved.

Industrial Light & Magic, ILM and the Bulb and Gear design logo are all registered trademarks or service marks of Lucasfilm Ltd.

© 2010-2012 Sony Pictures Imageworks Inc. All rights reserved.

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 in the documentation and/or other materials provided with the distribution.
- Neither the name of Industrial Light & Magic nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE

USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

ALEMBIC ATTACHMENT A —
REQUIRED NOTICES FOR DISTRIBUTION

The Alembic Software is distributed along with certain third party components licensed under various open source software licenses ("Open Source Components"). In addition to the warranty disclaimers contained in the open source licenses found below, Industrial Light & Magic, a division of Lucasfilm Entertainment Company Ltd. ("ILM") makes the following disclaimers regarding the Open Source Components on behalf of itself, the copyright holders, contributors, and licensors of such Open Source Components:

TO THE FULLEST EXTENT PERMITTED UNDER APPLICABLE LAW, THE OPEN SOURCE COMPONENTS ARE PROVIDED BY THE COPYRIGHT HOLDERS, CONTRIBUTORS, LICENSORS, AND ILM "AS IS" AND ANY REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER ORAL OR WRITTEN, WHETHER EXPRESS, IMPLIED, OR ARISING BY STATUTE, CUSTOM, COURSE OF DEALING, OR TRADE USAGE, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, ARE DISCLAIMED. IN NO EVENT WILL THE COPYRIGHT OWNER, CONTRIBUTORS, LICENSORS, OR ILM AND/OR ITS AFFILIATES BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE OPEN SOURCE COMPONENTS, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Boost C++ Libraries

Boost Software License – Version 1.0 August 17th, 2003 Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the

Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS 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, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MurmurHash3

The MIT License (MIT)

Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS 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 NON-INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,

WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

GLEW

The OpenGL Extension Wrangler Library

Copyright (C) 2002-2007, Milan Ikits

Copyright (C) 2002-2007, Marcelo E. Magallon

Copyright (C) 2002, Lev Povalahev

All rights reserved.

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 in the documentation and/or other materials provided with the distribution.
- * The name of the author may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Mesa 3-D graphics library

Version: 7.0

Copyright (C) 1999-2007 Brian Paul All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS 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. IN NO EVENT SHALL BRIAN PAUL BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Copyright (c) 2007 The Khronos Group Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and/or associated documentation files (the "Materials"), to deal in the Materials without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Materials, and to permit persons to whom the Materials are furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Materials.

THE MATERIALS 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. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE MATERIALS OR THE USE OR OTHER DEALINGS IN THE MATERIALS.

Appendix A

Disclaimer

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

End User License Agreement

End User Licensing Agreement (EULA)

IMPORTANT: BY SELECTING THE "I ACCEPT" BUTTON AT THE END OF THIS AGREEMENT OR BY INSTALLING THIS SOFTWARE YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT DO NOT INSTALL, COPY OR USE THE SOFTWARE.

This END USER LICENSE AGREEMENT (this "Agreement") is made by and between Isotropix SAS, a company registered in France, ("Isotropix"), and you, as either an individual or a single entity ("Licensee").

Grant Of License

Subject to the terms and conditions of this Agreement, Isotropix hereby grants to Licensee a non-exclusive, non-transferable limited license to install and use a machine readable, object code version of this software program (the "Software") and accompanying user guide and other documentation (collectively, the "Documentation") solely for Licensee's own internal business purposes (collectively, the "License"); provided, however, Licensee's right to install and use the Software and the Documentation is limited to those rights expressly set out in this Agreement.

Restriction On Use

Except as otherwise specifically permitted in this Agreement, Licensee may not: (a) modify or create any derivative works of any Software or Documentation, including translation or localization; (code written with published Software API/SDK shall not be deemed derivative works); (b) copy the Software except as provided in this Agreement or elsewhere by Isotropix; (c) separate Software, which is licensed as a single product, into its component parts. (d) sublicense; (e) reverse engineer, decompile, or disassemble or otherwise attempt to derive the source code for any product the Software (except to the extent applicable laws specifically prohibit such restriction); (f) redistribute, encumber, sell, rent, lease, sublicense, use the Software in a timesharing or service bureau arrangement, or otherwise transfer rights to any Software. Licensee may NOT transfer the Software under any circumstances; (g) remove or alter any trademark, logo, copyright or other proprietary notices, legends, symbols or labels in the Product(s); (h) use the Software by more users than have been licensed, on more computers or computing devices than the number licensed. Licensee may install the Software on more than one computer, however, Licensee shall not at any one time have its number of concurrent users using the Software exceeds the number of total valid Software licenses purchased by Licensee.

If the software is a "Personal Learning Edition", the licensee purpose of use of the software should be strictly personal or dedicated to internal training or instruction, excluding any other purpose. These versions of the software exclude any commercial, professional or for-profit purposes. For any avoidance of doubt, providing training or instruction to third parties is also excluded.

If Licensee has purchased the Software under the terms of Isotropix's Educational Policy: (a) If licensee is an individual, Licensee may use the Software for strictly non commercial, non professional or non profit purposes; (b) If Licensee is not an individual, Licensee may use the Software as an organization only for training and instruction purposes, any other use being strictly prohibited; (c) Licensee will at all times comply with the Educational Policy which is available on request.

License Transfer Policy

A license key bound to a computer is required to access and enable the Software. The issuing of replacement or substituted license keys if the Software is moved from one computer to another is subject to and strictly in accordance with Isotropix's License Transfer Policy, which is available on request and which requires a fee to be paid in certain circumstances. Isotropix conserves the right to vary the terms and conditions from time to time of the License Transfer Policy.

Evaluation Version

If Isotropix identifies the Software as a demonstration, evaluation, trial, or not for resale version ("Evaluation Version"), Licensee may install and access a copy of the Software, only for the purpose of commercial evaluation and demonstration. Without limiting the foregoing, You may not use it for competitive analysis, or commercial, professional, or other for-profit purposes. The Evaluation Version may only be installed for a fifteen (15) days evaluation period, unless otherwise specified by Isotropix in writing.

Backup Copy

Licensee may install one (1) backup copy of the Software and Documentation on another file server computer solely for backup purposes in the event that Licensee's primary file server computer on which the Software is Installed becomes inoperable. Licensee may access such backup copy of the Software only in the event and for so long as the primary file server Computer on which the Software is installed becomes inoperable and Licensee would be otherwise unable to access the Software.

License Fee

Licensee understands that the benefits granted to Licensee hereunder are contingent upon Licensee's payment in full of the license fee payable in connection herewith (the "License Fee").

Upgrades And Enhancements

The Licensee's access to support, upgrades and updates is subject to the terms and conditions of the "Maintenance Subscription Program" available on request. Isotropix conserves the right to vary the terms and conditions from time to time of the Maintenance Subscription Program.

Limited Warranty

Isotropix warrants for a period of thirty (30) days after delivery of the Software: (a) the Software will perform substantially in accordance with the functions described in the Documentation, and that (b) the media, on which the Software is furnished, if any, is free of defects in materials and workmanship which could prevent its operation, subject to proper use and maintenance (c) to the best of Isotropix's knowledge, Licensee's use of the Software in accordance with the Documentation will not, in and of itself, infringe any third party's copyright, patent or other intellectual property rights. EXCEPT AS WARRANTED, THE SOFTWARE AND DOCUMENTATION IS BEING PROVIDED "AS IS", EXCEPT FOR THE WARRANTIES SET FORTH IN THIS LICENSE AGREEMENT, THE SOFTWARE IS PROVIDED WITH NO OTHER WARRANTIES WHATSOEVER, AND ISOTROPIX DISCLAIMS ANY OTHER WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, INCLUDING, BUT WITHOUT LIMITATION, THE USE, THE PERFORMANCE AND RESULTS LICENSEE MAY OBTAIN BY USING THE SOFTWARE, THE MERCHANTABILITY, THE SATISFACTORY QUALITY, THE FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. WITHOUT LIMITING THE FOREGOING, ISOTROPIX DOES NOT WARRANT THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR FREE.

Limited Remedy

Licensee's sole and exclusive remedy and the entire liability of Isotropix under this limited warranty will be for Isotropix to make commercially reasonable efforts within a reasonable period of time to (a) correct or replace any Software that does not comply with the above warranty, or (b) if Isotropix cannot remedy or replace such defective Software within such time period, then Isotropix will refund the fees actually paid to it for the Software license. Any replacement Software will be warranted for the remainder of the original warranty period.

Limitation of Liability

IN NO EVENT SHALL ISOTROPIX OR ITS LICENSORS HAVE ANY LIABILITY FOR ANY INCIDENTAL, SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS, REVENUE, DATA, OR COST OF COVER. IN ADDITION, IN NO EVENT SHALL THE LIABILITY OF ISOTROPIX OR ITS LICENSORS FOR ANY DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE SOFTWARE, DOCUMENTATION OR THIS AGREEMENT EXCEED THE AMOUNT PAID OR PAYABLE BY LICENSEE FOR THE SOFTWARE DIRECTLY RESPONSIBLE FOR SUCH DAMAGES. THE LIMITATIONS OF LIABILITY IN THIS SECTION SHALL APPLY TO ANY DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, WHETHER DERIVED FROM CONTRACT, TORT (INCLUDING, BUT NOT LIMITED TO, NEGLIGENCE), OR OTHERWISE, EVEN IF ISOTROPIX HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND REGARDLESS OF WHETHER THE LIMITED REMEDIES AVAILABLE HEREUNDER FAIL OF THEIR ESSENTIAL PURPOSE. ISOTROPIX SHALL HAVE NO RESPONSIBILITY OR LIABILITY WHATSOEVER ARISING FROM LOSS OR THEFT OF THE SOFTWARE OR THE MEDIA ON WHICH THE SOFTWARE IS FURNISHED TO LICENSEE. ISOTROPIX SHALL NOT BE OBLIGATED TO REPLACE ANY LOST OR STOLEN SOFTWARE OR SOFTWARE MEDIA. LICENSEE IS SOLELY RESPONSIBLE FOR SAFEGUARDING THE SOFTWARE AND THE MEDIA ON WHICH THE SOFTWARE IS FURNISHED.

Term and Termination

Subject to the terms and conditions of this Agreement, the license to use the Software is perpetual, unless the Software qualifies as an Evaluation Version, or is designated as a fixed-term license, a limited duration license or a rental license. In such cases, the term of the license shall be the term identified by Isotropix in the applicable Documentation (the “Designated Term”) or the term for which Licensee have paid, whichever is less. If Isotropix identifies the Software as licensed for a fixed term, limited duration or rental and does not specify a term, then the Designated Term shall expire ninety (90) days after the date Licensee's first install the Software. Use of this Software beyond the applicable license term, or any attempt to defeat the time-control disabling function in the Software is an unauthorized use and constitutes a material violation of this Agreement and intellectual property law. Without prejudice to any right to compensation for any damage suffered, Isotropix may immediately terminate this Agreement, without further obligation or liability, if Licensee or any of the users breaches any term or condition hereof and fails to correct such breach within thirty (30) days after Isotropix notifies Licensee in writing. Upon termination of this License Agreement for any reason, all rights granted to Licensee hereunder will cease, and Licensee will forthwith (i) purge the Software from all of Licensee's computer systems, storage media and other files, and (ii) destroy or return to Isotropix all copies of the Software and Documentation. Any obligation that by its nature shall persist will survive the termination of this License Agreement but shall in no event imply or create any continued right to use the Software after termination of this Agreement.

Entire Agreement

This Agreement and the applicable Documentation constitute the entire agreement between us and supersede any other previous or contemporaneous communications, agreements, representations, or advertising with respect to the Software and Documentation. Any modifications to this Agreement shall be invalid, unless made in a duly signed writing.

Applicable Law And Jurisdiction

THIS AGREEMENT SHALL BE CONSTRUED AND ENFORCED IN ACCORDANCE WITH FRENCH LAW. THE UNITED NATIONS CONVENTION ON THE INTERNATIONAL SALE OF GOODS SHALL NOT APPLY. ANY DISPUTE ARISING OUT THIS AGREEMENT SHALL BE SUBJECT TO THE EXCLUSIVE JURISDICTION OF THE COURTS OF FRANCE.

All Rights Reserved

EXCEPT AS EXPRESSLY PROVIDED OTHERWISE IN THIS AGREEMENT, TITLE, OWNERSHIP AND ALL RIGHTS AND INTEREST INCLUDING, WITHOUT LIMITATION, PATENTS, COPYRIGHTS, TRADEMARKS, TRADE SECRETS AND OTHER INTELLECTUAL PROPERTY RIGHTS, IN AND TO ISOTROPIX AND ANY AUTHORIZED COPIES MADE BY LICENSEE REMAIN WITH ISOTROPIX AND ITS LICENSORS. THE STRUCTURE, ORGANIZATION, AND CODE OF THE ISOTROPIX ARE VALUABLE TRADE SECRETS OF ISOTROPIX AND ITS LICENSORS AND LICENSEE SHALL KEEP SUCH TRADE SECRETS CONFIDENTIAL. THE SOFTWARE AND DOCUMENTATION ARE LICENSED, NOT SOLD.

Canada License

If Licensee purchased the license for this Software in Canada, Licensee agrees to the following: The parties hereto confirm that it is their wish that this Agreement, as well as other documents relating hereto, including Notices, have been and shall be written in the English language only. Les parties ci-dessus confirment leur désir que cet accord ainsi que tous les documents, y compris tous avis qui s'y rattachent, soient rédigés en langue anglaise.

Copyright (c) 2013 Isotropix SAS. All Rights Reserved. Do not duplicate.

Index

- 3 -

3D Layer 96, 105, 106, 107, 108, 109, 418
3D Navigation 96
3D View 55, 94, 117, 329, 459, 480

- A -

Abstract class 115
Adding Items 85, 86
Adding Scene Objects 96
Alpha Checkerboard 98
Animating 311, 312, 313, 314, 315, 316, 317, 322, 323, 329
Animation 89
Animation modifier 88
Antialiasing 99, 173, 174, 175
AOV 200
Applying Filters 110
Assets 67, 68
Assigning Materials 69, 83, 96
Attribute 51, 58, 59, 60, 62, 77, 78, 79, 83, 84, 89, 91, 92, 108, 169, 184, 294, 295, 416
Attribute Editor 33, 37, 40, 53, 58, 62, 76, 77, 90, 106, 110, 113, 317, 331, 393, 395, 397, 422, 423, 453, 454, 459, 467, 482
Attribute Modifiers 88
Attribute Types 77

- B -

Batch replace windows 78
Blending Modes 110
Blue Noise 123
Boolean 78
Bounding box 145
Bounds 409
Brightness 111
Browser 55, 395
Browsing 329
Buckets 171
Built-in Contexts 66
Bump 465

- C -

Camera attribute 105
Camera mapping 439
Canvas 93
Category 58, 60
Changing color 82
Changing User Interface 47, 48
Checkerboard icon 98
Class 51, 57
Class Explorer 53, 55
Class hierarchy 52, 53
Class inheritance 51, 52
Classes 51
Clipping 108
Color 79, 82
Color Channel 79
Color Correction 94, 98
Color Display Mode 48
Color Field 82
Color Layer 104
Color Picker 79, 81, 82
Color Scheme 47
Color Slider 82
Color Swatches 82
Combiners 159, 393, 482
Command Line Option 17
Compositing 111
Context 35, 51, 63, 64, 65, 454
Contrast 111
Copy 61
Copying Contexts 67
CPU 9, 10
CPU raytracer 331
Creating Contexts 66
Creating Layer 112
Creating Objects 54
Creating Widgets 46
Creating Windows 46
Curve 83, 326
Curve meshes 136
Customizing 32, 46

- D -

Deactivating Items 87

Default Context 66
Deformation 120
Deformer 316, 317, 459, 482
Delete Key 80
Deleting Contexts 67
Deleting Objects 57
Dependency Graph 63
Diffuse attribute 30
Displacement 145, 146, 147, 150, 152, 317, 407, 409, 410, 459, 482
Displacement shader 144, 146
Display 330
Distribution 122
Drag and drop 96, 97, 395, 397, 414
Drivers 9
Duplicating Objects 61

- E -

Editing 324
Editing Keys 326
Editing Objects 58
Embedded instance 91
Embedded layers 110
Embedded object 59, 86, 104
Environment 453
Evaluation 59, 93, 97, 98
Evaluation Engine 93
Evaluation Progress Bar 45
Evaluation Status 45
Explorer 55
Exporting renders 108

- F -

Far planes 108
Favorite 50
Favorites 50
FCurve Editor 324, 325
Field value 77
File 124
File dependency 70
File format 72
File Layer 104
File path 78
File referencing 141
Filename 78

Filters 93
Fitting the view 96
Flag 58, 59
Folder 50, 64
Fractal noise 30
Fur 125, 127
Fur file 136
Fur Support Color 125

- G -

Geometry 115, 117, 120, 122, 123, 127, 147, 148, 448
Geometry shading group 144
Geometry Support 480, 481, 482
Global mode 40, 120
Gradient 80, 81, 82
Graph Editor 89, 324
Grid 142
Groups 83, 395, 397, 459, 470, 472

- I -

ILISE 13, 15, 16, 17
Image 35, 93, 94, 95, 97, 469
Image Layer 109
Image Toolbar 96
Image View 33, 37, 55, 93, 94, 96, 97, 414, 466, 481, 482
Importing data 445, 448
Importing geometries 68, 69, 141
Importing Images 71
Importing Scenes 72
Incidence texture 465
Input Dependencies 55
Inserting 326
Inserting a key 80
Installation 9, 10, 11
Installing licenses 16
Instance 30, 51, 90, 92
Instancing Contexts 67
Instancing Objects 62
Interpolation 80
Items 454

- K -

Kernel Filters 111
Kernel image filter 110
Key Editor 326
Keyframe 89
Keys 326
knots 136

- L -

Label 77
Layer 93, 96, 104, 106, 447, 459, 467, 469, 474
Layer Editor 37, 111, 467, 474
Layering 421, 422, 423
Layout 466
License key 13
License server 13
Licensing 9, 13, 15, 16
LICINFO 16
LICMAN 17
Light linking 417, 418, 419, 420
Lighting 107, 171, 173, 174, 175, 182, 183, 184, 192, 193, 200, 201, 202, 203, 212, 214, 217, 218, 220, 222, 224, 227, 482
Limited Region 98
Linux 11
Local mode 40
Localize 148, 152
Localize In 91, 148
Localizing 62, 91, 423
Localizing Attributes 62

- M -

Mac OS X 11
Main window 33
Managing Visibility 106
Map Files 465
Master Image 97
Material 148, 231, 413, 414, 415, 416, 451, 458, 465
Material Editor 152, 157, 410, 456
Material Graph 157
Material Linker 144, 146, 148, 149, 150, 407, 415, 451, 480
Material Override 108, 416

Mattes 474, 478
Memory Manager 63
Menu Bar 33
Microsoft Visual C++ 2010 10
Modifier 58, 60, 62
Modifying attributes 77
Modifying Layout 46
Mono-threaded 123
Motion Blur 184, 185, 187
Moving Camera 96
Moving Contexts 67
Moving Objects 58, 67
Multiselection 35, 36

- N -

Name 58
Naming Objects 57
Navigating 330
Navigation 34, 152, 153
Navigation Buttons 36
Near planes 108
Next Selection Button 36
Nodal View 153
Nodes 30, 51, 153, 154, 155, 156, 157, 459
Numeric 78
Numeric mode 82

- O -

Object 30, 51, 55, 56, 58, 63, 76, 446
Object attribute 30
Object classes 51
Object List 86
Object path 65
Object visibility 65
Object-instanting 51
Object-localization 51
Object-oriented-programming paradigm (OOP) 51
Optimization 200
Orbiting Camera 96
Output Dependencies 56
Overlay 98
Overlay Icon 98

- P -

Panning 325
 Parent attribute 30
 Parenting 52, 58, 83, 127, 313
 Particles 117, 124
 Picker color 82
 Pixel Filters 111
 Pixel image filter 110
 Point Cloud 480, 481, 482
 Point samples 119
 Point Volume 401
 Points 115, 117, 118
 Polygons 136
 Preferences panel 36, 81, 326
 Preset 83
 Previz 192, 193
 Primitives 115, 116, 117, 144, 147
 Procedural environment 459
 Progress Bar 97
 Project 35, 67, 454
 Project items 90
 Project path 91
 Protection 59
 Python 2.7 10

- Q -

Quads 140

- R -

RAM 9, 10
 Raytracer 171, 178
 Reference 52, 68, 83
 Reference Layer 109
 Reference List 84
 Reflection 182, 470
 Refraction 183
 Removing Items 85, 87
 Removing Layer 112
 Renaming Contexts 67
 Renaming Objects 57
 Renaming Windows 57
 Render Channels 108
 Render Farm 371

Render Nodes 371
 Render Workshop 46
 Renderer 105, 178
 Rendering 93, 94, 98, 125, 139, 171, 173, 174, 175, 182, 183, 184, 185, 187, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 329, 331, 332, 371, 469, 470
 Reordering Items 88
 Reordering Layers 112
 Request key 13
 Requirements 10
 Resolution Multiplier 111
 resource 345
 Ressource View 63
 Rotate attribute 30, 311

- S -

Samples 122, 123, 124
 Sampling 119, 120, 185
 Sampling Quality 99
 Saving the Image 99
 Scale Attribute 312
 Scatterer 481
 Scatterers 400, 401, 479, 480, 482
 Scattering 159, 161
 Scene 35
 Scene item 30, 58, 311
 Scene object 105, 107, 108, 115, 158, 161, 419
 Select Key 80
 Selecting Contexts 67
 Selecting Dependencies 55
 Selecting Items 85, 86
 Selecting objects 55
 Selection 34, 36, 40, 97
 Selection field 35
 Selection History 36
 Selection Tool Bar 33, 34, 55
 Selection Tools 56
 Set Key Color 80
 Setting Layer Blending Mode 113
 Setting Layer Opacity 113
 Setting the Renderer 106
 Setting the Viewpoint 105
 Shading 194, 231, 232, 233, 234, 235, 239, 245
 Shading group 69, 70, 115, 146, 147, 148, 149, 150, 151, 152
 Shadows 470
 Shadows Layer 477

Shortcut 33, 111, 113
Shutter Curve 187
Single primitive 115
Slot mode 40
Source 90, 91
Source object 61, 62, 91, 92
Spans 142, 143
sRGB icon 98
Status Bar 33, 45
Stopping an Evaluation 94
Subdivision Surface attribute 138
Subdivision surfaces 136, 138, 139, 140
Sub-windows 46
System requirements 9

- T -

Tessellation 125, 140, 144, 145
Texture 89, 152, 410
Texture displacement 320
Texture maps 93
Texture modifier 88
Texture object 30
Texturing 231, 261, 262, 263, 264, 265, 266, 267
Tiles 171
Tools 337, 338, 339, 340, 341, 343
Transformation 120
Translate attribute 30, 58, 311
Translate tool 58
Transparency 178, 179, 180
Tutorials 387
Type 58, 59, 77

- U -

Unlocalize 148, 152
Unlocalizing 62, 92
Unlocalizing Attributes 63
Updating Geometry 70
User interface 32, 37, 46, 445
Using Clarisse 29
UV map 142, 143
UVs 125

- V -

Value 62

Vertex Maps 115, 274
Vertices 136
Viewpoint 105
Viewport 33, 37, 39
Viewport Toolbar 37

- W -

Widget 33, 37, 40, 148
Windows 10, 46
Workflow 30, 51
Workshop 46, 47

- Z -

Zooming 325