Scene file format dev

From LuxRender Wiki

Introduction

In order to pass scene data for rendering, either the luxrender API or the .lxs file format can be used. While the API and the .lxs file format are closely related, this document only covers the description of the .lxs file format. The API format uses the same basic commands, but has a different input syntax. See API Usage Example. This should aid both exporter writers and those wishing to gain more control over the rendering process.

Basics

An .lxs file is an ASCII text file with a full description of the scene to be rendered. As scene descriptions can get quite large, there is a mechanism to break the scene description down into more manageable pieces: the description can be spread over several text files. The suggested organization is the following:

1. A main scene file with extension '.lxs'. For example: 'scene.lxs'
2. An included Material file with suffif '-mat' and extension '.lxm'. For example: 'scene-mat.lxm'
3. An included Geometry file with suffix '-geom' and extension '.lxo'. For example: 'scene-geom.lxo'
4. An optional Volume file with suffix '-vol' and extension '.lxv'. For example: 'scene-vol.lxv'

You can include each additional file into the scene file via the "Include" statement:

```
Include "scene-geom.lxo"
```

Coordinate System

LuxRender uses a 3D coordinate system where the Z axis points upward and where positive increments of the Y coordinate move away from the camera. This is consistent with Blender's coordinate system. If you need to write an exporter from a program that uses a Y-up system, you need to swap the Y value with the Z value when writing the LuxRender file. In addition, if positive values of the original Z axis move the object toward the camera, you will need to multiply that value by -1 when exporting to Lux.

Basic Structure

While the contents of .lxs files may vary considerably, even the simplest scenes will have the following: a camera specification, at least one light source, a film specification, transformations, primitives, materials, textures and so on...

The following example .lxs file will be the example used in all following explanation:

```
#This is an example of a comment!
#Global Information
LookAt 0 10 100 0 -1 0 0 1 0
Camera "perspective" "float fov" [30]

Film "fleximage"
"integer xresolution" [300] "integer yresolution" [300]
```

```
PixelFilter "mitchell" "float xwidth" [2] "float ywidth" [2] "bool supersample" ["true"]

Sampler "metropolis"

#Scene Specific Information
WorldBegin

AttributeBegin
        CoordSysTransform "camera"
        LightSource "distant"
                "point from" [0 0 0] "point to" [0 0 1]
                "color L" [3 3 3]
AttributeEnd

AttributeBegin
        Rotate 135 1 0 0

        Texture "clouds_noise_generator" "float" "blender_clouds"
                "string coordinates" ["local"] "float noisesize" [2.15] "string noisebasis" "voronoi_crackle"

        Texture "clouds_diffuse" "color" "mix"
                "color tex1" [0.8 0.1 0.1] "color tex2" [0.1 0.1 0.8] "texture amount" "clouds_noise_generator"

        Material "matte"
                "texture Kd" "clouds_diffuse"
        Shape "disk" "float radius" [20] "float height" [-1]
AttributeEnd

WorldEnd
```

Global vs Scene Specific Information

Notice that in the .lxs file structure, global information is given before scene specific information.

Global information

This is information that describes how the scene is going to be rendered; it includes types of and settings for the camera, sampler and film and so on. While the position and rotation of a camera may be considered part of the specific scene description, the camera defines the final render area and is therefore more important to the rendering process than say, geometry. This data is given at the top of the .lxs file.

LookAt

The Lookat transformation takes 3 vectors as options:

```
LookAt 0 0 0
       0 0 0
       0 0 0
```

The first vector (0 0 0 in my example) is the position of the camera.
The second vector (0 1 0) is the position of the target of the camera.
The last vector (0 0 1) is the up vector.

This is quite a simple transformation, which allows you to specify 2 points, one where the camera is located and another point of 'interest'. This can also be given as a unit vector. The up vector is a unit vector which defines how the camera is help up, it should point upwards of your camera, eg if it is held straight, it will be 0 0 1 (+Z up)

Scene specific information

Geometry, materials, textures - these obviously affect the final image but have no bearing on how the scene is to be rendered. This data is given between the WorldBegin and WorldEnd statements after the global information. More on these statements and the 'Attribute' statements a little later.

The distinction between the global information and the scene specific information is not arbitrary. LuxRender processes and "locks" global information first before looking at scene specific information. This implementation would allow you to tessellate geometry based on what is viewed from the camera, for instance.

Components of a .lxs file

Notice these features occur frequently throughout the .lxs file:

- Comments

Any text on a line beginning with a # is ignored by the parser. Often these are useful to give readable descriptions of what you are doing!

- Statements

'WorldBegin' and 'AttributeEnd' are examples of statements. These are important in describing the properties of whole blocks of data.

- Identifiers

'Camera', 'Sampler', 'Texture' and 'Material' are some examples of identifiers. These indicate what aspect of the scene is being described.

- Types

These are given in quotes, often after an identifier to specify an implementation to use. For example, by giving the type "perspective" after the 'Camera' identifier, the renderer can use the intended camera model.

- Parameters

Parameters typically consist of two parts: the type of parameter followed by the parameter name in given in quotes, followed by the actual value of the parameter given. The type of a parameter can be an integer, a float, a point, a normal, a color, a bool or a string. Continuing our example, the field of view ('fov') needs to be passed to our camera and as this parameter is of type float, we type "float fov" [30]. Note that as 'Rotate' and 'LookAt' take a predetermined amount of numbers and types of arguments they do not need parameter types or quotes.

Importance of parameter types

It is important to explain why you need to specify whether you are giving an integer or float every single time you give a parameter. It may seem like that tedium should be taken care of automatically, but LuxRender would be far less flexible if that were the case. LuxRender is based on plugins, functional sections of code that the core renderer does not need to know anything about. By specifying the parameter type, LuxRender can handle new plugins in a uniform way, allowing developers to realise new features quicker. To reiterate, the parameter types are:

Type	Description	Example
integer	a whole number	1, 2, 3 ...
float	a number with a decimal point	[1.67], [2.54] ... Note the square brackets given around the values.
point	declares a point, given by three floating point values	[45.000 12.000 0.900]
vector	a vector, specified the same way as a point.	[12.000 3.120 1.100]
normal	a normal (unit vector)	same as a point.
color	RGB color	[r g b] expressed in decimal notation ([0.123 1.0 0.245])
bool	Boolean	["true"] , ["false"]
string	string, non-escaped, surrounded by quotes	"simple.exr", "lux.png".

Arrays are created by specifying more than one value inside brackets, separated by space. So a point or a vector is actually an array of floats. In this tutorial we will indicate this by using [n] after the type name, where n is the number of elements required. For example, a valid float[4] would be [0 1 2 3].

Transformations

Any number of transformation can be applied to a section of the scene. Unless specified each transformation will be concatenated (right multiplied) with the current transform. To make the transformation "local" it's advisable to enclose them between "TransformBegin" and "TransformEnd". These two commands define a block that is then removed when the TransformEnd command is reached. From that point on, the previously defined, possibly global, transformations will be restored. For example:

A simple rotation:

```
TransformBegin
  Rotate 90 0 1 0

  Shape "cone"
TransformEnd
```

A more complex (and realistic) transform:

```
Identity
Translate 5 0 0
TransformBegin
        # Rotate in addition to translation.
        Rotate 90 0 1 0
        Shape "cone"
TransformEnd
# Only the translate transformation is current at this point
```

One can store the current transformation as a named coordinate system using "CoordinateSystem". Using this the above example can be written as:

```
TransformBegin
        Identity
        Translate 5 0 0
        Rotate 90 0 1 0
        CoordinateSystem "mytransform" # store the current transform as "mytransform"
TransformEnd

...
CoordSysTransform "mytransform"   # set the current transformation to "mytransform"
Shape "cone"
```

The following transformation commands are available:

Command	Description	Example
Identity	Replaces the current transformation with the identity transformation.	
Rotate	Rotates an entity by a given number of degrees around a given axis. The first parameter is the rotation in degrees, the following parameters indicate the X, Y and Z axis. A 0 means to leave the axis unaffected, a 1 means to rotate on that axis.	`Rotate 180 1 0 0` Rotates the object around the X-axis 180°
Scale	Adjusts the size of an entity by a given factor.	`Scale .1 .2 .3` Scales in the X-direction to 10%, the Y-direction to 20%, and the Z-direction to 30%
Translate	Moves an entity in a given direction by a vector.	`Translate 10.1 20.2 30.3` Moves the center 10.1 units in the X-direction, 20.2 in the Y-direction, and 30.3 in the Z-direction
Transform	Replaces the current transformation with the given transform. It takes a single 16-element array as parameter, each element corresponds to an item in a 4x4 matrix, column by column. The last column, and thus last 4 values, typically holds the translation part.	`Transform [2.0 0.0 0.0 0.0 0.0 2.0 0.0 0.0 0.0 0.0 2.0 0.0 -3.0 2.0 -1.0 1.0]` Replaces the current transform with one that scales the object by 2 and translates it by -3.0 along the X-direction, 2.0 along the Y-direction and -1.0 along the Z-direction.
ConcatTransform	Similar to Transform, except concatenates (right multiplies) the given transform with the current transform.	See Transform
Lookat	The LookAt transformation takes 3 vectors as parameters. The first vector (0 0 0 in the example) is the position of the camera. The second vector (0 1 0) is the position of the target of the camera. The last vector (0 0 1) is the up vector. This is quite a simple transformation, which allows you to specify 2 points, one where the camera is located and another point of 'interest'. The up vector is a unit vector which defines how the camera is help up, it should point upwards of your camera, eg if it is held straight, it will be 0 0 1 (+Z up)	`LookAt 0 0 0 0 1 0 0 0 1` Orients the entity such that it is located at the origin and looking in the +Y direction, with +Z as up.
CoordSysTransform	Replaces the current transformation by a named transformation defined earlier using CoordinateSystem.	`CoordSysTransform "mytransform"`

Alternatively you can use a transformation matrix within an Attribute or Object block:

Transform [scale_x 0.0 0.0 0.0 0.0 scale_y 0.0 0.0 0.0 0.0 scale_z 0.0 move_x move_y move_z 1.0]

Motion Blur

LuxRender includes a system for motion blur of camera and object transforms. Multiple steps are supported for both, but deformation motion blur is not currently supported. It is important to remember that since LuxRender is a progressive renderer, motion blur is effectively "free" as the trail noise will normally converge before other elements in the scene. (Thus, there is also no concept of a "time sample count" or similar)

Motion of an object is handled by having a special transform block that uses the keywords MotionBegin and MotionEnd instead of TransformEnd and TransformBegin. Unlike TransformBeginEnd statements, a MotionBegin/End block allows multiple transforms at different times, and interpolates between them. A block following the MotionBegin statement (same line) includes the times of each transform statement to follow. These times are technically unitless, but to maintain physicality it's best to consider them as being in seconds. Camera motion blur is handled in the same fashion, using either LookAt or Transform statements as usual. You can use MotionBegin and MotionEnd statements to define the camera's position over time, and camera motion blur will be interpreted from this.

The strength of the blur is determined from the shutteropen and shutterclose parameters for the film. These describe

the amount of time the shutter is open for. Like the timings for transform steps described in the previous paragraph, these are technically unitless but should generally be considered as being in seconds.

It is important to note that shutterclose timings are independent of the motion step time. This allows LuxRender to simulate the real-world scenario of the shutter being open for only a portion of the frame. In general, the first step in your motion definition should be set to 0.0, the beginning of the frame. The last step in the motion definition should always be 1 divided by the current frame rate. For example, if the current animation frame rate is 24fps, you should have the first step at 0.0, the final step at 0.04167 (1 divided by 24), and any addition steps at even intervals between them. Since most 3D packages allow arbitrary frame rates, it is important that you divide by the actual scene frame rate value rather than using a fixed time for a set of common frame rates. See your 3D application's documentation on animation export and frame rates.

Taking the example from the Transform section, here we use two motion steps to rotate our cone over a 24fps frame. For objects being translated in a curved path, you will need at 3 transform statements for a curved trail. Think of this as subdividing the motion trail.

```
MotionBegin [0.0, 0.04167]
  Rotate 90 0 1 0
  Rotate 180 0 1 0

  Shape "cone"
MotionEnd
```

As a final note, the values for shutteropen/shutterclose act as an arbitrary scaling factor on motion blur. Normally you want to use 0.0 for shutteropen and set shutterclose to a reasonable exposure time, such as 1/30th of a second (0.033), or 1/100th of a second (0.01) to mimic the normal timings of a camera. The scale is calibrated to match real world motion blur when used in this way. If you would like to add a non-physical "blur scale" you can apply it simply by multiplying the shutterclose value by the scaling value. (Note: there is also no requirement that the linear tonemapper's exposure value match shutterclose, although to fully simulate a real digital camera, you should use matching values).

Attributes

- **WorldBegin**

The *WorldBegin* attribute indicates the end of the global information and the beginning of the scene specific information.

- **WorldEnd**

The *WorldEnd* attribute indicates the end of the scene specific information and marks the end of the file (if no comments follow). Upon reaching the *WorldEnd* statement, LuxRender will assume the scene description is complete and will begin rendering.

- **AttributeBegin**

The *AttributeBegin* attribute indicates the beginning of the description of an unnamed object. In case the object needs to be referenced (for example by instances), use *ObjectBegin* instead.

- **AttributeEnd**

AttributeEnd marks the end of the description of an unnamed object.

- **TransformBegin**

Indicates the beginning of a named coordinate system. Usually within this block you will write *Identity*, *Transform*, and *CoordinateSystem*. *Identity* is used to reset the coordinate system to zero, the *Transform* defines the transform of the coordinate-system you wish to name, and *CoordinateSystem* followed by a string names the transform defined in this block.

- **TransformEnd**

TransformEnd marks the end of a named coordinate system.

- **ObjectBegin**

The *ObjectBegin* attribute indicates the beginning of the description of a named object, which is useful for creating instances. It must be followed by a string, indicating the name of the object.

- **ObjectEnd**

ObjectEnd marks the end of the description of a named object.

- **ObjectInstance**

If a named object is created, it can be referenced in multiple objects. These objects can be ordinary unnamed objects that use an *ObjectInstance* line instead of a shape definition. It must be followed by a string with the name of the object to instance.

- **MotionBegin**

MotionBegin defines the start of a block of animated transforms, and includes the times of each transform to follow. See the Motion Blur section above.

- **MotionEnd**

MotionEnd marks the end of a block of animated transforms.

- '*PortalInstance*

Works in a similar manner to ObjectInstance but is a substitute for PortalShape. This allows instancing of portal geometry.

Using the attributes

To give an idea on some more complex uses of these statements, here's an example a motion blurred light with nested instances.

```
# define a named light object, it is not yet included in the scene
ObjectBegin "test"
LightSource "point" "point from" [0 0 0]
ObjectEnd

# define a named compound object comprising a sphere and a reference to the previous light
# they are not yet included in the scene
ObjectBegin "test2"
Shape "sphere"
ObjectInstance "test"
ObjectEnd
# add an instance of the compound in the scene
ObjectInstance "test2"

# add a moving instance of the light in the scene
MotionBegin [0 0.04167]
Identity
Translate 1 0 0
MotionEnd
ObjectInstance "test"
```

Identifiers

The following identifiers can be included in the scene:

- Camera
- Sampler
- Film
- PixelFilter
- Shape
- Accelerator
- Material
- Texture
- Volume
- Light
- SurfaceIntegrator
- VolumeIntegrator

Each of these shall be covered in turn.

Camera

The following camera types are available. Each of them is a plugin. The default camera type is 'perspective'.

- **perspective** (default)

The most common camera, resemble most closely what is seen by the human eye or a pinhole camera.

- **environment**

Denotes a type of camera which maps the whole of the environment around the camera to an image.

- **orthographic**

The standard orthographic camera: parallel lines stay parallel and a sense of depth is lost.

- **realistic**

A physically-based camera model that loads lens descriptions files to realistically generate camera rays and weights.

Common Camera Parameters

There are parameters specific to a particular camera implementation and general parameters for all cameras. Here are the common parameters:

name	type	description	default value	theoretical range	suggested range
hither	float	Geometry closer than this distance will not be rendered	10^{-3}	0 - ∞	0 - 10 000
yon	float	Geometry further than this distance will not be rendered	10^{30}	0 - ∞	0 - 10 000 (always bigger than hither)
shutteropen	float	The time in seconds at which the virtual shutter opens	0.0		0
shutterclose	float	The time in seconds at which the virtual shutter closes	1.0		0.001 - 1, larger (1-60 or more) for "long exposure" blur
lensradius	float	Default (0) for pinhole camera with whole scene in focus, increase for depth of field and focus effects. Ignored by environment and realistic cameras	0.0	0 - ∞	0 - 1
focaldistance	float	Focal distance of the virtual lens. Use the control focus in depth of field effects. Ignored by environment and realistic cameras	10^{30}	0 - ∞	0 - 10 000
frameaspectratio	float	This is normally computed from resolution but can be overridden here. Ignored by realistic camera	computed from given resolution	0 - ∞	1 - 10
screenwindow	float[4]	Specifies area in camera that the virtual film occupies - the bounds in screen space. Crops the camera viewport. (*Ignored by realistic camera*.) Values are: xmin xmax ymin ymax.	entire screen space	dependent on frame aspect ratio (xresolution/yresolution): If aspect_ratio > 1: -aspect_ratio aspect_ratio -1 1 Otherwise: -1 1 -1/aspect_ratio 1/aspect_ratio	

Specific Camera Parameters

'environment' cameras do *not* support depth of field parameters lensradius and focaldistance as indicated above. 'realistic' cameras don't support lensradius, focaldistance, frameaspectratio and screenwindow either. The orthographic camera supports the above parameters, but has no others. However, the 'perspective' and 'realistic' cameras do have some specific parameters:

perspective

name	type	description	default value	theoretical range	suggested range
fov	float	This is the field of view for the camera - this specifies the solid angle viewed. The fov value is the angle spanned by the **smallest** dimension of the image (width/height). An fov of 49.13 is equal to a focal length of 35mm.	90	0 < x < 180	1 - 179
lensradius	float	This value defines the lens radius. Values higher than 0 enable DOF and control its amount. To calculate the lensradius based on the f/stop number use the following equation. lensradius = (focallength_in_mm / 1000.0) / (2.0 * fstop) Example: 0.00625 = (35 / 1 000.0) / (2.0 * 2.8)	0.00625	x ≥ 0	0 - 0.4
autofocus	bool	Enable/disable the autofocus feature. It is useful to automatically calculate the focal distance when lens radius > 0.0 (i.e. depth of field enabled).	false		
focaldistance	float	This value controls the focal point of the scene, the distance from the camera at which objects will be in focus. It has no effect if the Lens Radius is set to 0. It is specified in meters. This value is not needed when using autofocus.	0.0	0 < x	0 - 200
blades	integer	This value controls the number of blade edges of the aperture, values 0 to 2 defaults to a circle.	6		
distribution	string	This value controls the lens sampling distribution. Non-uniform distributions allow for ring effects. Valid values include: "uniform", "exponential", "inverse exponential", "gaussian", and "inverse gaussian".	uniform		
power	integer	This value controls the exponent for the expression in exponential distribution. Higher values gives a more pronounced ring effect.	1		

realistic

This camera type is not currently functional and should thus not be present in the user interface.

name	type	description	default value
specfile	string	The .dat lens specification file which can be found in the cameras/realistic directory	""
filmdistance	float	Distance from the film to the backmost lens component (milimeters)	70.0
aperture_diameter	float	The aperture diameter of the aperture stop	1.0
filmdiag	float	Film diagonal size	35.0

Sampler

Here are the sampler types. Each one is a separate plugin.

- **metropolis**

Generates random values that are then slightly mutated, according to the Metropolis algorithm. In most cases, this sampler should be used by default.

- **sobol**

Progressive quasi-random sampler, uses a sobol sequence. Based on the sobol sampler in Blender Cycles. This sampler gives good results on simple scenes.

- **random** (default)

Completely random sampler. Set to default as a fallback, not production-useful in most cases.

- **lowdiscrepancy**

Generates a low discrepancy sequence that evenly samples values, given a target number of samples per pixel (must be a power of 2). In most cases, the sobol sampler will yeild better results due to its progressive sample stratification.

- **erpt**

Similar to the Metropolis sampler but the *Energy Redistribution Path Tracing* algorithm is used. This sampler is largely a failed science experiment and shouldn't be used.

Noise-aware sampling

All samplers (except for ERPT) support an option for noise-aware adaptive sampling. If this mode is enabled, LuxRender will create a "heat map" of the areas of the image it believes are not fully converged. These areas will become more likely to be sampled.

In addition to this, there is an option for loading a user-supplied sampling map as well. If this option and noise-aware are both used, the sampling weights will be combined.

Parameters

name	type	description	default value	theoretical range	suggested range
noiseaware	bool	Enables or disables the noise-aware mode	False	False	True, unless using bidirectional integrator, where it is not always helpful

Sampler Parameters

These samplers accept the parameters below. Valid values for *pixelsampler* are *hilbert* (since v0.6), *linear*, *vegas*, *lowdiscrepancy*, *tile* (or alias *grid*), and *random*. "tile", "linear" and "hilbert" modes offer slightly better performance due to cache coherency, but obviously lack the quick feedback of seeing the whole image at once.

metropolis

name	type	description	default value	theoretical range	recommended range
maxconsecrejects	integer	Number of consecutive rejects before a next mutation is forced. Low values can cause bias	512	x ≥ 0	0 - 32768
largemutationprob	float	Probability of generating a large sample mutation	0.4	0 ≤ x ≤ 1	0 - 1
mutationrange	float	Maximum distance in pixel for a small mutation	$\frac{(render_{width} + render_{height})}{32.}$		
usevariance	bool	Use the variance hint provided by some integrators to alter sample acceptance	false		
usecooldown	bool	Ramp the largemutationprob from 0.5 to the provided value during the first few seconds of rendering	false		

sobol

The sobol sampler has no parameters of its own, just the noise-aware options

random

name	type	description	default value	theoretical range	suggested range
pixelsamples	integer	Number of samples per pixel / pass	4	x ≥ 1	1 - 512
pixelsampler	string	Pixel sampler algorithm to use: hilbert, linear, vegas, lowdiscrepancy, tile (or alias grid) or random	vegas		

lowdiscrepancy

name	type	description	default value	theoretical range	suggested range
pixelsamples	integer	Number of samples per pixel / pass	4	x ≥ 1	1 - 512
pixelsampler	string	Pixel sampler algorithm to use: hilbert, linear, vegas, lowdiscrepancy, tile (or alias grid) or random	vegas		

erpt

Note: The ERPT sampler is largely a failed science experiment and probably shouldn't be used.

name	type	description	default value	theoretical range	recommended range
initsamples	integer	Number of warm-up samples	100000		
chainlength	integer	Number of mutations from a given seed	2000	x ≥ 1	1 - 32768
mutationrange	float	Maximum distance in pixel for a small mutation	$\frac{(render_{width} + render_{height})}{50.}$		

Film

There is currently only one film type:

- **fleximage** (default)

A flexible film, being able to output several buffers, with some image normalization options.

Specific Film Parameters

fleximage

name	type	description	default value	theoretical range	recommended range
xresolution	integer	The number of pixels in the xdirection	800	x ≥ 1	1 - 10000
yresolution	integer	The number of pixels in the ydirection	600	x ≥ 1	1 - 10000
cropwindow	float[4]	Values describing render region that range from min (0) to max (1) in order xmin, xmax, ymin,ymax. (0,0) is top left	(0,1,0,1)	all values: 0 ≤ x ≤ 1	
gamma	float	Gamma correction value	2.2	x ≥ 0	2.2, or your monitor gamma if it's not 2.2. Other values will lead to a non-physical response of light gain (non-linear workflow)
premultiplyalpha	bool	Multiplies the pixel colours by the pixel's alpha value before writing the image. Used to encode alpha in images without an alpha channel	false		
colorspace_red	float[2]	The xy chromaticity coordinates of the red primary in the target color space. Used for low dynamic range output (and OpenEXR if write_exr_applyimaging is enabled). Default color space is SMPTE.	[0.63 0.34]		
colorspace_green	float[2]	Same as colorspace_red but for the green primary.	[0.31 0.595]	both values: 0 ≤ x ≤ 1	both values: 0 ≤ x ≤ 1
colorspace_blue	float[2]	Same as colorspace_red but for the blue primary.	[0.155 0.07] both values: 0 ≤ x ≤ 1	both values: 0 ≤ x ≤ 1	
colorspace_white	float[2]	Same as colorspace_red but for the white point.	[0.314275 0.329411] both values: 0 ≤ x ≤ 1	both values: 0 ≤ x ≤ 1	
write_exr	bool	Toggles the .exr (OpenEXR) output	false		
write_exr_channels	string	Which channels to write for OpenEXR output. Valid values are "RGB", "RGBA", "Y" and "YA". "RGBA" is RGB with alpha, "Y" is luminosity, and "YA" is luminosity with alpha.	"RGB"		
write_exr_halftype	bool	Indicates if the half-float (16bit) type should be used for OpenEXR output.	true		
write_exr_compressiontype	string	Which compression scheme to use for OpenEXR output. Valid values are "RLE (lossless)", "PIZ (lossless)", "ZIP (lossless)", "Pxr24 (lossy)" and "None".	"PIZ (lossless)"		

write_exr_applyimaging	bool	Specifies if the imaging pipeline (for tone mapping, bloom etc) should be used for OpenEXR output. Use "false" to get the raw HDR output. Image will be in linear gamma regardless of this setting or the gamma setting.	"true"
write_exr_gamutclamp	bool	Specifies if out-of-gamut colors should be clamped for OpenEXR output.	"true"
write_exr_ZBuf	bool	Specifies if the Z-Buffer should be used for OpenEXR output. Will be added as a separate channel in the OpenEXR file.	"false"
write_exr_zbuf_normalizationtype	string	Specifies which normalization, if any, is applied to the Z-Buffer before being added to the OpenEXR file. Valid values are "Camera Start/End clip", "Min/Max" and "None".	"None"
write_png	bool	Toggles the .png low dynamic range tonemapped output.	true
write_png_channels	string	Same as write_exr_channels, but for PNG output.	"RGB"
write_png_16bit	bool	Toggles 16 bit per channel PNG output.	false
write_png_gamutclamp	bool	Toggles clamping of out-of-gamut colors for PNG output.	true
write_tga	bool	Toggles the .tga low dynamic range tonemapped output	false
write_tga_channels	string	Same as write_exr_channels, but for TGA output.	"RGB"
write_tga_gamutclamp	bool	Toggles clamping of out-of-gamut colors for TGA output.	true
write_tga_zbuf_normalization	string	TBD	N/A
ldr_clamp_method	string	Method used to clamp out-of-gammut colors after tonemapping. Valid values are "lum", "hue" and "cut". First two tries to preserve luminosity and hue, respectively, of the color. "cut" will simply clip large values.	"cut"
write_resume_flm	bool	Toggles the rendering resume output. Luxrender will auto-resume a rendering if the resume file exists. (**NOTE1:** the resume feature doesn't include any sanity check, do not forget to erase the file if you are starting a new rendering or you have modified the scene description. **NOTE2:** the resume file is usually read between 10-30 secs after the start of the rendering. **NOTE3:** this feature works well only with progrsive pixel	false

		samplers.)			
restart_resume_flm	bool	Disables starting from an existing flm file, whilst still writing a new one if above param is true.	false		
filename	string	The suffix used for all file output	"luxout"		
writeinterval	integer	The interval in seconds between all image writes	60	x ≥ 1	60 - 36000
flmwriteinterval	integer	The interval in seconds between flm file writes	60	x ≥ 1	60 - 36000
displayinterval	integer	The interval in seconds between display updates	12	x ≥ 1	10 - 36000
outlierrejection_k	integer	Strength value for the firefly rejection feature. 0 = disabled	0	x ≥ 0	3-10
tilecount	integer	Number of film buffer tiles to use. 0 = automatic	0	x ≥ 0	0
haltspp	integer	Stop the rendering when the average number of samples per pixel is higher or equal to the chosen value. The rendering will stop at the next valid Sampler/PixelSampler state in order to not introduce bias (i.e. few more samples can be required to reach that state). A value less or equal to 0 disables this option.	0	x ≥ 0	0 - 50000
halttime	integer	Stop the rendering when the time elapsed from the start is higher or equal to the chosen value (defined in seconds). The rendering will stop at the next valid Sampler/PixelSampler state in order to not introduce bias (i.e. few more samples can be required to reach that state). A value less or equal to 0 disables this option.	0	x ≥ 0	0 - 86400
haltthreshold	float	Adaptive-quality rendering option. Stops the rendering when the number of pixels failing the perceptual convergence test falls below this value. A value of 0 disables this option. Enabled noise-aware in the sampler block to automatically focus-fire the failing pixels.	0	x < 1	0.01 - 0.0000001
convergencestep	float	Interval (in samples per pixel) that the convergence test updates. This test is used to supply quality data for the haltthreshold option and noise-aware sampling.	32	x ≥ 4	8 - 128
tonemapkernel	string	The tone mapping kernel used for low dynamic range output (and OpenEXR if write_exr_applyimaging is enabled). Valid values are	"autolinear"		

		"reinhard", "linear", "autolinear", "contrast" and "maxwhite".		

Tonemapper Parameters

Fleximage accepts the following parameters for the various tone mapping kernels:

reinhard

name	type	description	default value	theoretical range	recommended range
reinhard_prescale	float	Defines the amount of pre-scale	1.0	x ≥ 0	0 - 8
reinhard_postscale	float	Defines the amount of post-scale	1.0	x ≥ 0	0 - 8
reinhard_burn	float	Defines the amount of burn	6.0	x ≥ 0	0 - 8

linear

name	type	description	default value	theoretical range	recommended range
linear_sensitivity	float	Sensitivity of the film (in ISO).	50.0	x ≥ 0	6 - 6400
linear_exposure	float	Exposure time of the film in seconds (currently not tied to shutter opening).	1.0	x > 0	0.0005 - 60
linear_fstop	float	F-stop value.	2.8	x ≥ 0	0.7 - 96
linear_gamma	float	Gamma value (separate from the FlexImageFilm's gamma parameter).	1.0	x ≥ 0	0 - 5

contrast

name	type	description	default value	theoretical value	recommended value
contrast_ywa	float	Ywa parameter for the contrast tone mapping kernel.	1.0	x > 0	0 - 20000

maxwhite

The maxwhite kernel has no parameters.

autolinear

The autolinear kernel has no parameters

Filter

Here are the pixel filter types. Each one is a separate plugin:

- **blackmanharris**
- **box**
- **catmullrom**
- **triangle**
- **gaussian**
- **mitchell** (default)
- **sinc**

Filter Parameters

blackmanharris

name	type	description	default value	theoretical range	recommended value
xwidth	float	Radius of the filter in the x direction	1.65?	x > 0	3.3 / 1.65?
ywidth	float	Radius of the filter in the y direction	1.65?	x > 0	3.3 / 1.65?

box

name	type	description	default value	theoretical range	recommended range
xwidth	float	Half-width (or radius) of the filter in the x direction	0.5	x > 0	0.1 - 2
ywidth	float	Half-width (or radius) of the filter in the y direction	0.5	x > 0	0.1 - 2

catmullrom

name	type	description	default value	theoretical range	recommended value
xwidth	float	Radius of the filter in the x direction	?	x > 0	?
ywidth	float	Radius of the filter in the y direction	?	x > 0	?

gaussian

name	type	description	default value	theoretical range	recommended range
xwidth	float	Half-width (or radius) of the filter in the x direction	2	x > 0	0.1 - 2
ywidth	float	Half-width (or radius) of the filter in the y direction	2	x > 0	0.1 - 2
alpha	float	Gaussian rate of falloff. Lower values give blurrier images.	2	x > 0	0 - 10

mitchell

name	type	description	default value	theoretical range	recommended range
xwidth	float	Half-width (or radius) of the filter in the x direction	2	x > 0	0.1 - 3
ywidth	float	Half-width (or radius) of the filter in the y direction	2	x > 0	0.1 - 3
B	float	B parameter for the mitchell filter in the range	1/3	0 ≤ x ≤ 1	0 - 1
C	float	C parameter for the mitchell filter in the range 0..1.	1/3	0 ≤ x ≤ 1	0 - 1
supersample	bool	Use filter sumpersampling, image will be filtered first with a grid of 4 pixels for each pixel	True		Don't turn this off. Seriously, don't.

sinc

name	type	description	default value	theoretical range	recommended range
xwidth	float	Half-width (or radius) of the filter in the x direction	4	x > 0	0.1 - 5
ywidth	float	Half-width (or radius) of the filter in the y direction	4	x > 0	0.1 - 5
tau	float	Width of the filter window. Should not be larger than the radius of the filter.	3	x > 0	0.1 - 5

triangle

name	type	description	default value	theoretical range	recommended range
xwidth	float	Half-width (or radius) of the filter in the x direction	2	x > 0	0.1 - 2
ywidth	float	Half-width (or radius) of the filter in the y direction	2	x > 0	0.1 - 2

Renderer

The Renderer is a sort of wrapper for the surface integrator. It is used to switch between possible operating modes of LuxRender. The following renderers are available

- **sampler**
- **hybrid**
- **sppm**

The "classic" LuxRender rendering modes are all contained within the the "sampler" renderer. There are two other available renderers, "hybrid" or "hybridsampler" (both call the same renderer), and "sppm". Hybrid is a GPU-accelerated renderer, that offloads ray-triangle intersections to OpenCL compute devices. SPPM is an experimental implementation of stochastic progressive photon mapping.

"Sampler" and "sppm" have no parameters of their own, all parameters are taken from the surface integrator. "Hybrid" has several parameters dealing with GPU options:

name	type	description	default value	theoretical range	recommended range
configfile	string	Path to a configuration file containing Renderer settings. This file should have the same format as used by SLG, and reads only the parameters defined in this table.			
opencl.platform.index	integer	The OpenCL platform index to use for rendering.	0	x ≥ 0	The "correct" platform is most likely platform 0 or 1, but it may not be
opencl.gpu.use	bool	Tell the renderer to use the OpenCL GPU devices. Otherwise, native threads are used.	True		True
opencl.gpu.workgroup.size	integer	Set the work-group size for the OpenCL GPU. The default (0) will attempt to automatically detect the correct size.	0		0 is automatic, but some cards will require this be set manually to 64
opencl.devices.select	string	List of OpenCL devices to use. Default is to target all available GPU devices unless this parameter specifices otherwise.			Generally, it's best to leave this blank unless the user has a specific rendering card.
statebuffercount	integer	Number of buffers the renderer will use to prepare rays for the GPU. Higher values can improve efficiency but may significantly increase memory use	1	x > 0	1 or 2
raybuffersize	integer	Number of rays per buffer fed to the GPU	8192	x ≥ 2	4096-16384, must be a power of 2
accelerator.qbvh.stacksize.max	integer	Max stack depth for the GPU QBVH (different from scene accelerator)	32	x > 0	24 or 32, 32 is faster, but not all cards can handle it

Surface Integrator

The surface integrator is responsible for actually propagating the ray through the scene. The following types are available:

- **bidirectional (default)**
- **path**
- **exphotonmap**
- **directlighting**
- **igi**
- **distributedpath**
- **sppm**

Common Parameters

The following parameter is common to all integrators except for SPPM

name	type	description	default value	theoretical range	recommended range
lightstrategy	string	Light sampling method. See LuxRender_Render_settings#Light_Strategy	auto	Values strings are "auto", "one", "all", "importance", "powerimp", "allpowerimp", "autopowerimp", and "logpowerimp"	"auto" or "autopowerimp". Hybrid bidir can only use "one"
shadowraycount	integer	Specifies the number of shadow rays traced at each intersection for each sampled light. Supported by the following surface integrators: directlighting, exphotonmap, path.	1		

Specific Surface Integrator Parameters

Bidirectional

Bidirectional path tracer. Available with hybridsampler or sampler. This should be the default integrator. NOTE: Light strategy parameter (see common parameters) must be declared as "one" when used with hybridsampler

name	type	description	default value	theoretical range	recommended range
lightdepth	integer	Maximum recursion depth of a light path	8	x > 0	Values below 2 can give very strange results, avoid using higher values (16+) with hybrid renderer
eyedepth	integer	Maximum recursion depth of an eye path	8	x > 0	Values below 2 can give very strange results, avoid using higher values (16+) with hybrid renderer
lightrrthreshold	float	Light-path russian roulette control. 0.0 uses standard efficiency-based RR, 1.0 disables RR entirely. Values in-between are a combination of the two	0.0	x ≥ 0	Values other than 0.0 are only useful in certain situations
eyerrthreshold	float	Eye-path russian roulette control. 0.0 uses standard efficiency-based RR, 1.0 disables RR entirely. Values in-between are a combination of the two	0.0	x ≥ 0	Values other than 0.0 are only useful in certain situations
lightpathstrategy	string	The strategy used for sampling the lights when tracing light paths (same than what lightstrategy does for direct lighting)	"auto"	Values strings are "auto", "one", "all", "importance", "powerimp", "allpowerimp", and "logpowerimp"	

Path

Classic path tracing integrator. Available with sampler or hybridsampler.

name	type	description	default value	theoretical range	recommended range
maxdepth	integer	Maximum recursion depth of a path	16	x > 0	Maxdepth=1 is identical to the direct lighting integrator, values of 2 or greater should be used.
rrstrategy	string	Defines the russian roulette strategy used. Can be "none, "probability", or "efficiency".	efficiency		Efficiency normally gives the best results
includeenvironment	bool	Show environment light sources.	True		
directlightsampling	float	Use direct light sampling to help find lights. Disable for "brute force" path tracing (can be faster with only environment map lighting)	True		

Direct

Classic Whitted raytracer, available only in the sampler renderer. Calculates direct lighting only, and is mainly useful for previews (such as seeing how your textures look in the actual scene). This integrator has no ability to calculate or fake indirect lighting, and thus is not terribly useful for most final renders.

name	type	description	default value	theoretical range	recommended range
maxdepth	integer	Maximum recursion depth of a path	16	x > 0	3-5 is often enough, but there is little speed penalty for going higher, as this integrator does not continue diffuse paths. Extra bounces are for specular/transmissive surfaces.
rrstrategy	string	Defines the russian roulette strategy used. Can be "none, "probability", or "efficiency".	efficiency		Efficiency normally gives the best results

SPPM

Experimental stochastic progressive photon mapping integrator. Only available with the sppm renderer. It renders the image using a series of passes. In the first half of each pass, it runs a direct-light render and stores the diffuse-surface hit points in a map. In the second half of the pass, photons are traced from the lights, and any that fall into a hitpoint are considered to have illuminated that hit point. The process is then repeated using a smaller hit point search radius.

name	type	description	default value	theoretical range	recommended range
photonsampler	string	The sampler used to trace photons.	"halton"	"halton" (standard quasi random sampler) and "amc" (adaptive sampler)	"halton"
lookupaccel	string	The data structure used to store hit points.	"hybridhashgrid"	"hashgrid", "kdtree", "hybridhashgrid" and "parallelhashgrid"	"hybridhashgrid"
parallelhashgridspare	float	Only used for parallelhashgrid	1		
pixelsampler	string	The scheme used to sample the image plane. Possible strings are "hilbert", "liner", "tile" and "vegas"	"hilbert"		
maxeyedepth	integer	Maximum recursion depth of an eye path	16	x > 0	3-5 is often enough, but there is little speed penalty for going higher, as this integrator does not continue diffuse paths. Extra bounces are for specular/transmissive surfaces.
maxphotondepth	integer	Maximum recursion depth of a photon path	16	x > 0	16-64
photonperpass	integer	Number of photons to gather before moving on to the next pass	1000000	x > 0	250000-5000000
startradius	float	Search radius uses during the first pass	2.0	x > 0	~.5-5
alpha	float	Scaling factor to tighten search radius by on passes after the first. 1.0 will defeat the radius reduction effect	0.7	x > 0	0.5-1.0
includeenvironment	bool	Include the light from the environment when it is directly visible	true		
directlightsampling	bool	Do a direct lighting pass instead of relying only on photons	true		
useproba	bool	Use the radius reduction behavior from "PPM: a probabilistic approach". Due to the reduced hitpoint size this may improve performance.	true		

wavelengthstratification	integer	The number of initial passes with special wavelength sampling in order to reduce color artifacts	8	> 0	8
debug	bool	Debug mode	false		
storeglossy	bool	Allow storing of hit points on glossy surfaces, it might help rendering or ruin the render depending on the scene	false		

Volume Integrator

The volume integrator is responsible for adding volumetric effects to the result computed by the surface integrator. The following types are available:

- **None**

A sham integrator which disables volumetric scattering entirely. Can improve performance by avoiding any calculations for unspecified volumes (which are thus a vacuum with no effect on the scene)

- **Emission**

Calculates absorption, as well as light emission from the (largely deprecated) PBRT volumes. Scattering will be ignored

- **Single**

Calculates a single bounce of scattering, as well as all effects for the emission integrator

- **Multi**

Calculates all volumetric effects. Rays passing through a homogeneous medium will be allowed to scatter repeatedly until they exit the volume or are terminated by the surface integrator.

Common Parameters

The following parameter is common to all volume integrators. There are no integrator specific parameters.

name	type	description	default value	theoretical range	recommended range
stepsize	float	Ray marching step length, in meters. This parameter is only used for the old volumes, as the medium system does not currently support heterogeneous volumes	1.0	x > 0.0	0.1-2

Scene specific information

Geometry, materials, textures - these obviously affect the final image but have no bearing on how the scene is to be rendered. This data is given between the WorldBegin and WorldEnd statements after the global information. More on these statements and the 'Attribute' statements a little later.

The distinction between the global information and the scene specific information is not arbitrary. LuxRender processes and "locks" global information first before looking at scene specific information. This implementation would allow for view-optimized tesselation of geometry, for instance.

In addition to the identifiers described above in the Transformations section and the Attributes section the following identifiers can be included in the scene:

- AreaLightSource
- LightSource
- LightGroup

- Shape
- PortalShape

- Material
- MakeNamedMaterial
- NamedMaterial

- Texture

- MakeNamedVolume
- Interior
- Exterior

Each of these shall be covered in turn.

A Word on Colors and Gamma

In LuxRender, colors are always described as linear RGB floats. For example, [0.80000001 0.80000001 0.80000001]

Some 3D applications will give a linear RGB float color when queried for the value of a color, but others may not. If they do give a linear RGB float triplet, you can simply write it to the scene file as-is, your app and LuxRender "speak the same language" for colors. However, this may not be the case. You might instead receive a 8bit integer triplet, such as [255, 255, 255]. This can be converted to the equivalent float by dividing by 255.

However, this might not give the color the user thought they were inputting. Some applications will return color queries with a value that already has gamma correction applied. Returning integer colors and gamma corrected colors often goes hand-in-hand. If an app returns a float, there is a good chance it is linear gamma. If it returns an integer, it is probably NOT linear gamma. Neither is guaranteed, however, and you should check any relevant api documentation for the application you are working with. If your application returns values with gamma correction applied to them, you should apply reverse gamma correction prior to writing them to scene file, in order to ensure that linear values are written out.

As a physically based renderer, linear workflow is fairly important for LuxRender, and is strongly enforced. All color inputs are assumed to be linear. Image textures are assumed to not be linear and have reverse gamma correction applied to them (the gamma parameter on the image map texture sets this, it should be set to the gamma the texture was created in for color textures, but left at 1.0 for float textures). The film is saved in linear XYZ color space, and gamma corrected for viewport display or when saving PNG or TGA output (this is done according the film gamma setting). The OpenEXR output is NOT gamma corrected, as the OpenEXR spec states that these files should always be linear gamma. The write_exr_applyimaging flag does NOT affect this behavior, it causes everything except gamma correction and clamping to be applied.

Light

General Syntax:

```
LightSource TYPE <parameters>
```

Where TYPE in one of the following:

- **distant**
- **goniometric**
- **infinite**
- **point**
- **projection**
- **sky**
- **sky2**
- **spot**
- **sun**

For area lights the syntax is slightly different:

```
AreaLightSource "area" <parameters>
```

There is no default type.

Note that the **area** type should be within a normal shape definition, which defines the shape of the light.

For convenience, it's possible to specify the **sunsky2** light which automatically creates a **sun** and **sky2** light with the same parameters. Generally, it's best to have your 3D package's distant/sun/environment lamp translate to the **sunsky2** light by default.

The **sky** and **sunsky** light types are legacy lights that use a outdated spectrum model (the Preetham model instead of the Hosek and Wilkie model used by sky2/sunsky2). This light should probably not be exposed in the UI, or at least not the default.

The point and spot lamps are non-physical, and generally do not give good results in most cases.

Examples

Here are some examples of light declarations:

Type	Code
Point	`Texture "pL" "color" "blackbody" "float temperature" [6500.0]` `LightSource "point" "texture L" ["pL"] "float gain" [1.0]`

Spot	`Texture "Spots::L" "color" "blackbody" "float temperature" [3200.00]`
	`LightSource "spot" "point from" [0 0 0] "point to" [0 0 -1] "float coneangle" [70] "float conedeltaangle" [5]`
	`"texture L" ["Spots::L"] "float gain" [1.50]`

Common Light Parameters

These parameters are applicable to all light plugins:

name	type	description	default value
importance	float	It is part of the Rendering Hints framework. It is a user defined value. It can be used by a light strategy as weight for choosing which light source to sample more. A value of 0.0 will completely disable sampling of that light, assuming a light strategy which uses render hints was selected	1.0
gain	float	Overall intensity multiplier for the light	1.0

Specific Light Parameters

These parameters are specific to individual light plugins:

Area

Defines a light emitting shape. This lamp definition must be enclosed within a shape definition, that shape will define the geometry. Note that as of v1.2, instances CANNOT be area lights!

name	type	description	default value
area::L	color/texture	The color of the light.	1 1 1
area::nsamples	integer	The suggested number of shadow samples when computing illumination from the given light.	1
area::power	float	Lamp output power in watts. Setting 0 for both power and efficacy bypasses this feature and uses only the lamp gain	100.0
area::efficacy	float	Luminous efficacy in lumens/watt. Setting 0 for both power and efficacy bypasses this feature and uses only the lamp gain	17.0
area::iesname	string	Path to an IES file to use. Note that it will be deformed by the shape of the light, using a small sphere primitive for the geometry is highly recommended when using IES profiles with area lights	

Distant

A generic directional light source located at infinity

name	type	description	default value
distant::L	color	The color of the light.	1 1 1
distant::from/to	point	The two points defining the light direction. Default is down the z axis.	0 0 0 and 0 0 1
distant::theta	float	Half angle of the light source in radians. Must be > 0 for soft shadows.	0

Goniometric

name	type	description	default value
goniometric::I	color	The color of the light	1 1 1
goniometric::mapname	string	The filename of the goniometric file (goniometric diagram of light distribution)	no default

Infinite

An omnidirectional environment light. Also used for attaching HDRI maps.

name	type	description	default value
infinite::L	color	The color of the light.	1 1 1
infinite::nsamples	integer	The suggested number of shadow samples when computing illumination from the given light.	1
infinite::mapname	string	The filename of the environment map for an infinite area light. If not provided, a solid color is used.	no default
infinite::mapping	string	The map type of environment maps. Options are latlong, angular and vcross	latlong

Point

A classic point lamp. This lamp is non-physical and should be avoided in most cases. A sphere primitive with an area light statement can be used for a more realistic omnidirectional light.

name	type	description	default value
point::I	color	The color of the light.	1 1 1
point::from	point	The location of the point light.	0 0 0
point::power	float	Lamp output power in watts. Setting 0 for both power and efficacy bypasses this feature and uses only the lamp gain	100.0
point::efficacy	float	Luminous efficacy in lumens/watt. Setting 0 for both power and efficacy bypasses this feature and uses only the lamp gain	17.0
area::iesname	string	Path to an IES file to use.	

Projection

Projects an image, acting as something of a square spot light. Useful for simulating an actual image projector.

name	type	description	default value
projection::I	color	The color of the light.	1 1 1
projection::fov	float	The field of view in terms of angular spread along the shorter image axis.	45
projection::mapname	string	The filename of the image to project.	required - no default

Sky

An older physical sky lamp. Deprecated, use Sky2 instead.

name	type	description	default value
sky::gain	float	Gain (aka scale) factor to apply to sun/skylight.	0.005
sky::nsamples	integer	The suggested number of shadow samples when computing illumination from the given light.	1
sky::sundir	vector	Direction vector of the sun.	0 0 1
sky::turbidity	float	Turbidity can go from 1.0 to 30+. 2-6 are most useful for clear days.	2.0
sky::aconst/bconst/cconst /dconst/econst	float	Perez function multiplicative constants.	1.0

Sky2

A lamp for simulating the light output of a terrestrial sky. Uses the model from Hosek and Wilkie's paper "An Analytic Model for Full Spectral Sky-Dome Radiance". Can also be used with the sunsky2 light type, which will create matching sun and sky2 lamps. sunsky2 accepts all properties of both sky2 and sun.

name	type	description	default value
sky2::gain	float	Gain (aka scale) factor to apply to sun/skylight.	0.005
sky2::nsamples	integer	The suggested number of shadow samples when computing illumination from the given light.	1
sky2::sundir	vector	Direction vector of the sun.	0 0 1
sky2::turbidity	float	Turbidity can go from 1.0 to 30+. 2-6 are most useful for clear days.	2.0

Spot

A traditional CG spot light. Like the point light, it is non-physical and should be avoided if possible.

name	type	description	default value
spot::I	color, texture	The color of the light.	1 1 1
spot::from/to	point	Points defining the axis of the spot light. Default is down the z-axis.	0 0 0 and 0 0 1 respectively.
spot::coneangle	float	The angle in degrees of the spotlight cone.	30
spot::conedeltaangle	float	The angle at which the spotlight intensity starts to fade from the edge.	5
spot::gain	float	The intensity of the spot light	
spot::power	float	Lamp output power in watts. Setting 0 for both power and efficacy bypasses this feature and uses only the lamp gain	100.0
spot::efficacy	float	Luminous efficacy in lumens/watt. Setting 0 for both power and efficacy bypasses this feature and uses only the lamp gain	17.0

Sun

A special distant lamp that simulates the spectrum of the sun. Color is set automatically by angle.

name	type	description	default value
sun::gain	float	Gain (aka scale) factor to apply to sun/skylight.	0.005
sun::nsamples	integer	The suggested number of shadow samples when computing illumination from the given light.	1
sun::sundir	vector	Direction vector of the sun.	0 0 -1
sun::turbidity	float	Turbidity can go from 1.0 to 30+. 2-6 are most useful for clear days.	2.0
sun::relsize	float	Relative size to the sun.	1.0

Attaching Materials, Volumes, and Lights

Materials are attached to objects within the object/attribute blocks. They are each their own lines, separate from the transform and shape definition. To add light emission, add an area light statement. This is a normal light statement like those described in the previous section, for the lamp type "area".

The syntax for materials:

```
NamedMaterial "examplematerial"
```

Where "examplematerial" is a material defined by a MakeNamedMaterial statement. See Materials for more info.

There are 3 possible commands, any or all can be used on a particular object:

- NamedMaterial (surface shader, this should always be used)
- Interior (interior volume material)
- Exterior (exterior volume material)

Shape

Here are the shape types. Each one is a separate plugin.

- **cone**
- **cylinder**
- **disk**
- **heightfield**
- **hyperboloid**
- **lenscomponent**
- **loopsubdiv**
- **nurbs**
- **paraboloid**
- **plymesh**
- **stlmesh**
- **sphere**
- **trianglemesh**
- **mesh**
- **hairfileshape**

Common Shape Parameters

name	type	description	default value
name	string	A string for identifying the mesh to the user. The contents of this string are not used by the renderer, names can be repeated or used arbitrarily. It is solely for human-readable output	""

Common Mesh Shape Parameters

The following parameters are shared among the mesh shapes, mesh, trianglemesh, plymesh, and stlmesh

name	type	description	default value
generatetangents	bool	Generate tangent space using miktspace, useful if mesh has a normal map that was also baked using miktspace (such as blender or xnormal)	false
subdivscheme	string	Subdivision algorithm, options are "loop" and "microdisplacement"	"loop"
displacementmap	*string*	DEPRECATED *Name of the texture used for the displacement. Subdivscheme parameter must always be provided, as load-time displacement is handled by the loop-subdivision code.*	*none - optional. (loop subdiv can be used without displacement, microdisplacement will not affect the mesh without a displacement map specified)*
displacementmap	float	Texture used for the displacement. Subdivscheme parameter must always be provided, as load-time displacement is handled by the loop-subdivision code.	none - optional. (loop subdiv can be used without displacement, microdisplacement will not affect the mesh without a displacement map specified)
dmscale	float	Scale of the displacement (for an LDR map, this is the maximum height of the displacement in meter)	0.1
dmoffset	float	Offset of the displacement.	0
dmnormalsmooth	bool	Smoothing of the normals of the subdivided faces. Only valid for loop subdivision.	true
dmnormalsplit	bool	Force the mesh to split along breaks in the normal. If a mesh has no normals (flat-shaded) it will rip open on all edges. Only valid for loop subdivision.	false
dmsharpboundary	bool	Try to preserve mesh boundaries during subdivision. Only valid for loop subdivision.	false
nsubdivlevels	integer	Number of subdivision levels. This is only recursive for loop subdivision, microdisplacement will need much larger values (such as 50).	0

Specific Shape Parameters

These parameters are specific to individual shape plugins.

Cone

name	type	description	default value
cone::radius	float	The radius of the cone.	1.0
cone::radius2	float	The radius of the cone upper face in case of a cone frustum *(available since v0.7)*.	0.0
cone::height	float	The height of the cone - along the z axis	1.0
cone::phimax	float	The angle (degrees) swept out by the cone round its circular base	360.0

Cylinder

name	type	description	default value
cylinder::radius	float	The radius of the cylinder	1.0
cylinder::zmin	float	The position of the bottom of the cylinder along the z axis	-1.0
cylinder::zmax	float	The position of the top of the cylinder along the z axis	1.0
cylinder::phimax	float	The angle (in degrees) swept out by the cylinder	360.0

Disk

name	type	description	default value
disk::height	float	The location of the disk (which is flat) along the z-axis	0.0
disk::radius	float	The radius to the outer rim of the disk	1.0
disk::innerradius	float	The radius to the inner rim of the disk - defines a "hole" in the disk	0.0
disk::phimax	float	The angle (degrees) swept out by the disk	360.0

Height Field

name	type	description	default value
heightfield::nu	integer	Number of points in the u axis	none - must be specified
heightfield::nv	integer	Number of points in the v axis	none - must be specified
heightfield::Pz	float[nu*nv]	The height of each point in the grid	none - must be specified

Hyperboloid

name	type	description	default value
hyperboloid::p1	point	The hyperboloid lies between p1 and p2. Here p1 defines the start of the hyperboloid	0 0 0
hyperboloid::p2	point	The hyperboloid lies between p1 and p2. Here p2 defines the end of the hyperboloid	0 0 0
hyperboloid::phi	float	The angle (in degrees) swept out by the hyperboloid	360.0

Subdivision Surfaces

Deprecated, instead use subdivscheme parameter on trianglemesh/mesh/plymesh/stlmesh shapes as applicable. This will allow for microdisplacement support.

name	type	description	default value
loopsubdiv::nlevels	integer	The level of subdivision applied to the base mesh	3
loopsubdiv::indices	integer[n]	Indices of the cage mesh - using the same format as used in the triangle primitive	none - must be specified
loopsubdiv::P	point[n]	Vertex positions of the cage mesh - using the same format as used in the triangle primitive	none - must be specified
loopsubdiv::uv	float[2*n]	Vertex texture coordinates of the cage mesh - using the same format as used in the triangle primitive	none - optional
loopsubdiv::displacementmap	*string*	DEPRECATED *Name of the texture used for the displacement. Subdivscheme parameter must always be provided, as load-time displacement is handled by the loop-subdivision code.*	*none - optional. (loop subdiv can be used without displacement, microdisplacement will not affect the mesh without a displacement map specified)*
loopsubdiv::displacementmap	float	Texture used for the displacement. Subdivscheme parameter must always be provided, as load-time displacement is handled by the loop-subdivision code.	none - optional. (loop subdiv can be used without displacement, microdisplacement will not affect the mesh without a displacement map specified)
loopsubdiv::dmscale	float	Scale the output of displacement map by the specified value (use negative value in order to invert the map)	0.1
loopsubdiv::dmoffset	float	Translate the output of displacement map by the specified value	0.0
loopsubdiv::dmnormalsmooth	bool	Interpolate normals in order to get a smooth output	true
loopsubdiv::dmsharpboundary	bool	The output mesh will have the same boundary of the cage mesh if this flag is true, otherwise the boundary will be interpolated as well	false

Lens Component

This shape is primarily used by the realistic camera. Lens are approximated by a sphere section.

name	type	description	default value
lenscomponent::radius	float	The radius of the sphere	1.0
lenscomponent::zmin	float	The position of the bottom of the sphere along the z axis	-radius
lenscomponent::zmax	float	The position of the top of the sphere along the z axis	radius
lenscomponent::phimax	float	The angle (in degrees) swept out by the sphere	360.0
lenscomponent::aperture	float	The aperture of the lens	1.0

NURBS

name	type	description	default value
nurbs::nu	integer	Sets the number of control points in the u direction	none - must be specified
nurbs::nv	integer	Sets the number of control points in the v direction	none - must be specified
nurbs::uorder	integer	The order (surface degree + 1) in the u direction Defines how many nearby control points affect each control point	none - must be specified
nurbs::vorder	integer	The order (surface degree + 1) in the v direction. Defines how many nearby control points affect each control point	none - must be specified
nurbs::uknots	float[nu+uorder] (eg. if nu = 2 and uorder = 2 we have a float[4] such as (1,1,1,1))	The knot vector in the u direction	none - must be specified
nurbs::vknots	float[nv+vorder]	The knot vector in the v direction	none - must be specified
nurbs::u0/nurbs::v0	float	The u and v coordinates from which NURB surface evaluation starts	uknots[uorder-1]/vknots[vorder-1]
nurbs::u1/nurbs::v1	float	The u and v coordinates from which NURB surface evaluation ends	uknots[nu]/vknots[nv]
nurbs::P	point[nu*nv]	P specifies regular control points on the NURBS surface	none - one of either P or Pw must be specified
nurbs::Pw	float[4*nu*nv]	Pw specifies rational control points on the surface which include a weight value for each control point	none - one of either P or Pw must be specified

Paraboloid

name	type	description	default value
paraboloid::radius	float	The radius of the paraboloid	1.0
paraboloid::zmin	float	The position of the bottom the paraboloid along the z axis	0.0
paraboloid::zmax	float	The position of the top the paraboloid along the z axis	1.0
paraboloid::phimax	float	The angle (in degrees) swept out by the paraboloid	360.0

PLY Mesh Loader

name	type	description	default value
plymesh::filename	string	The .ply filename to load	none
plymesh::smooth	bool	Toggles smoothing of the mesh faces	false

Plymesh example:

```
AttributeBegin # Plymodel
    NamedMaterial "plymodel"
    Shape "plymesh"
    "string filename" ["c:\\blender\\models\\plymesh.ply"]
AttributeEnd
```

STL Mesh Loader

Same as plymesh, but loads STL files instead.

name	type	description	default value
stlmesh::filename	string	The .stl filename to load	none
stlmesh::smooth	bool	Toggles smoothing of the mesh faces	false

Sphere

name	type	description	default value
sphere::radius	float	The radius of the sphere	1.0
sphere::zmin	float	The position of the bottom of the sphere along the z axis	Set to - sphere::radius
sphere::zmax	float	The position of the top of the sphere along the z axis	Set to - sphere::radius
sphere::phimax	float	The angle (in degrees) swept out by the sphere	360.0

Triangle Mesh

There are two types of triangle mesh: *barytrianglemesh* and *waldtrianglemesh*. Both have the same below parameters but waldtrianglemesh uses a better intersection algorithm. Using only *trianglemesh* defaults to the *barytrianglemesh* which is better tested.

name	type	description	default value
trianglemesh::indices	integer[n]	An array indexing the triangles found in the P array. The three vertices of the ith triangle are given by 3i, 3i+1, 3i+2. This provides the ordering of the triangles	none - must be specified
trianglemesh::P	point[n]	An unordered list of vertex positions in the triangle mesh. Ordered with trianglemesh::indices	none - must be specified
trianglemesh::N	normal[n]	Per vertex normals over the triangle mesh	none - optional
trianglemesh::S	vector[n]	Tangents given at each vertex. Tangents are perpendicular to normals	none - optional (deprecated)
trianglemesh::uv	float[2*n]	Texture coordinates (per vertex)	none - optional
trianglemesh::st	float[2*n]	Texture coordinates (per vertex)	none - optional

Mesh

Available since v0.6.

name	type	description	default value
mesh::N	normal[n]	Per vertex normals over the mesh	none - optional
mesh::P	point[n]	An unordered list of vertex positions in the mesh.	none - must be specified
mesh::acceltype	string	Accelerator structure to use (auto, bruteforce, grid, kdtree, none, qbvh)	"auto"
mesh::quadindices	integer[4*q]	An array indexing the quadrilaterals found in the P array. The three vertices of the ith quadrilateral are given by 4i, 4i+1, 4i+2, 4i+3. This provides the ordering of the quadrilaterals	none - must be specified
mesh::quadtype	string	Select the intersection routine used for quadrilaterals (quadrilateral)	"quadrilateral"
mesh::triindices	integer[3*t]	An array indexing the triangles found in the P array. The three vertices of the ith triangle are given by 3i, 3i+1, 3i+2. This provides the ordering of the triangles	none - must be specified
mesh::tritype	string	Select the intersection routine used for triangles (auto, bary, wald)	"auto"
mesh::uv	float[2*n]	Texture coordinates (per vertex)	none - optional

Hairfileshape

Available since v1.3. A special shape for loading hair and fur particles. It accepts baked hair systems in the cemyuksel hair file format: http://www.cemyuksel.com/research/hairmodels/

Note that hairfileshape is merely a strand primitive, NOT a hair simulator. The hair system should be simulated elsewhere and "baked" to the cemyuskel format at export time

Strands are loaded and tessellated as either camera-facing ribbons or cylinders. An adaptive tessellation version of both modes is available, and in most cases should be used

name	type	description	default value	recommended value
hairfileshape::filename	string	Path to the hair file		
hairfileshape::camerapos	point	Location of the camera, used to orient ribbons towards the camera, and for adaptive tessellation distance		
hairfileshape::tesseltype	string	Tessellation method for the strands. One of "ribbonadaptive", "ribbon", "solidadaptive", and "solid". "Ribbon" modes use camera-facing planes, "solid" modes use cylinders.	"ribbonadaptive"	ribbonadaptive is best in most cases. solidadaptive can be preferable with a small number of thick strands
hairfileshape::acceltype	string	Accelerator used for strands. One of "kdtree" or "qbvh"	qbvh	
hairfileshape::adaptive_maxdepth	integer	Max tessellation depth for adaptive tessellation modes	8	
hairfileshape::adaptive_error	float	Max error tolerance for adaptive tessellation modes	0.1	
hairfileshape::solid_sidecount	integer	Solid modes only, number of faces on each cylinder	3	
hairfileshape::solid_capbottom	bool	Solid modes only, adds a triangle fan cap to the strand root	false	
hairfileshape::solid_captop	bool	Solid modes only, adds a triangle fan cap to the strand tip	false	

Material

Here are the material types.

- **carpaint**
- **cloth**
- **glass**
- **glass2**
- **glossy**
- **glossycoating**
- **glossytranslucent**
- **glossy_lossy** (deprecated, use glossy)
- **layered** (not well tested, may be unstable)
- **matte** (default)
- **mattetranslucent**
- **metal** (deprecated, use metal2)
- **metal2**
- **mirror**
- **mix**
- **null**
- **roughglass**
- **shinymetal** (deprecated, use mirror+glossycoating or metal2 as needed)
- **scatter**
- **velvet**

See http://www.luxrender.net/wiki/LuxRender_Materials for detailed descriptions of each material

Common Material Parameters

Here are the common parameters for materials. Note that float and color textures are described in the 'Texture' section.

name	type	description	default value
bumpmap	float texture	The floating-point texture for use as a bump map	None

Integrator Dependant Parameters

The following material parameters are available on all materials for the stated Integrators:

name	type	integrator	description	default value
compo_visible_material	bool	distributedpath	Material will not be directly visible (it will have alpha-0), but will reflect light and be visible in reflection and transmission	true
compo_visible_emission	bool	distributedpath	If object is emitting, it will not appear to shine when viewed directly	true
compo_visible_indirect_material	bool	distributedpath	XXX	true
compo_visible_indirect_emission	bool	distributedpath	XXX	true
compo_override_alpha	bool	distributedpath	Override the alpha value in the output image for hits on this material	false
compo_override_alpha_value	float	distributedpath	Value to set alpha to when overridden	0.0

Specific Material Parameters

These parameters are specific to individual material plugins.

Car Paint

The available names for the carpaint material are:

- "ford f8"
- "polaris silber"
- "opel titan"
- "bmw339"
- "2k acrylack"
- "white"
- "blue"
- "blue matte"

Other parameters are only necessary if no name given.

name	type	description	default value
carpaint::name	string	The name of the car paint model to use	"ford f8"
carpaint::Kd	color texture	Diffuse component	"ford f8" data
carpaint::Ks1 / carpaint::Ks2 / carpaint::Ks3	color texture	Specular component of layers	"ford f8" data
carpaint::R1 / carpaint::R2 / carpaint::R3	float texture	Fresnel constants of layers	"ford f8" data
carpaint::M1 / carpaint::M2 / carpaint::M3	float texture	Microfacet roughness of layers	"ford f8" data

Cloth

name	type	description	default value
cloth::warp_Kd	color texture	The coefficient of diffuse reflection in one weave direction	0.5
cloth::warp_Ks	color texture	The coefficient of specular reflection in one weave direction	0.5
cloth::weft_Kd	color texture	The coefficient of diffuse reflection in the opposite weave direction	0.5
cloth::weft_Ks	color texture	The coefficient of specular reflection in the opposite weave direction	0.5
cloth::repeat_u	float	Thread count in the u direction	100.0
cloth::repeat_u	float	Thread count in the v direction	100.0
cloth::presetname	string	Name of the fabric type. One of "silk_charmeuse", "denim", "cotton_twill", "wool_gabardine", "polyester_lining_cloth", or "silk_shantung"	"denim"

Glass

name	type	description	default value
glass::Kr	color, texture	The reflectivity of the surface	[1.0 1.0 1.0]
glass::Kt	color, texture	Fraction of light transmitted through the surface	[1.0 1.0 1.0]
glass::index	float, texture	The index of refraction for the glass surface	1.5
glass::cauchyb	float, texture	Cauchy B coefficient	0.0
glass::film	float, texture	The thickness in nanometers of the thin film coating (0.0 disables) on the surface	0.0
glass::filmindex	float, texture	The index of refraction of the thin film coating on the surface	1.5
glass::architectural	bool	Disables refraction during transmission, improves rendering speed with thin sheets	False

Glass2

Glass2 is an "empty shell" for volumes, it will be invisible if no interior medium is specified

name	type	description	default value
glass2::architectural	bool	Disables refraction during transmission, improves rendering speed with thin sheets	False
glasss2::dispersion	bool	Enables chromatic dispersion. Should be used with a volume that has a fresnel texture other than "constant"	False

Glossy

name	type	description	default value
glossy::Kd	color texture	The coefficient of diffuse reflection	0.5
glossy::Ks	color texture	The coefficient of specular reflection	0.5
glossy::Ka	color texture	The coefficient of absorption of the coating layer	0.0
glossy::uroughness	float texture	The roughness of the surface in the u direction	0.1
glossy::vroughness	float texture	The roughness of the surface in the v direction	0.1
glossy::d	float texture	The depth (thickness) of the coating layer for absorption effects. (0 = disables)	0.0
glossy::index	float texture	IOR of the coating. IOR overrides color Ks if both are specified	0.0
glossy::multibounce	bool	Simulation of asperity (velvet-like reflection) on the glossy surface	false
glossy::separable	bool	If true, material acts internally as matte+glossycoating. This is somewhat more efficient to render, and allows use of the sigma parameter	true
glossy::sigma	float texture	The sigma parameter of the Oren-Nayer base shader in degrees. Zero for pure Lambertian reflection. Ignored if "separable" option is false	0.0

Glossycoating

Glossycoating in an "empty glaze" shader. It takes another material as a base, and puts a glossy coating over the top of it. Any other material can be used a base material, including another glossycoating. (however, the parameter cannot be left blank, so the last glossycoating in the stack must have some other material as its base)

name	type	description	default value
glossycoating::basematerial	string	The name of another MakeNamedMaterial block that describes the base material	
glossycoating::Ks	color texture	The coefficient of specular reflection	0.5
glossycoating::Ka	color texture	The coefficient of absorption of the coating layer	0.0
glossycoating::uroughness	float texture	The roughness of the surface in the u direction	0.1
glossycoating::vroughness	float texture	The roughness of the surface in the v direction	0.1
glossycoating::d	float texture	The depth (thickness) of the coating layer for absorption effects. (0 = disables)	0.0
glossycoating::index	float texture	IOR of the coating. IOR overrides color Ks if both are specified	0.0
glossycoating::multibounce	bool	Simulation of asperity (velvet-like) on the glossy surface	false

GlossyTranslucent

name	type	description	default value
glossytranslucent::Kd	color texture	The coefficient of diffuse reflection	0.5
glossytranslucent::Ks	color texture	The coefficient of specular reflection	0.5
glossytranslucent::Kt	color texture	The coefficient of transmission	1.0
glossytranslucent::Ka	color texture	The coefficient of absorption of the coating layer	0.0
glossytranslucent::uroughness	float texture	The roughness of the surface in the u direction	0.1
glossytranslucent::vroughness	float texture	The roughness of the surface in the v direction	0.1
glossytranslucent::d	float texture	The depth (thickness) of the coating layer for absorption effects. (0 = disables)	0.0
glossytranslucent::index	float texture	IOR of the coating. IOR overrides color Ks if both are specified	0.0
glossytranslucent::onesided	bool	Disable seperate coating for backface	True
glossytranslucent::backface_Ks	color texture	The coefficient of backface specular reflection	0.5
glossytranslucent::backface_Ka	color texture	The coefficient of absorption of the backface coating layer	0.0
glossytranslucent::backface_uroughness	float texture	The roughness of the backface surface in the u direction	0.1
glossytranslucent::backface_vroughness	float texture	The roughness of the backface surface in the v direction	0.1
glossytranslucent::backface_d	float texture	The depth (thickness) of the coating layer for backface absorption effects. (0 = disables)	0.0
glossytranslucent::backface_index	float texture	IOR of the backface coating.	0.0

Layered

A special material that realistically layers other materials. The model is will be computed as though each layer actually forms a separate surface. There are a maximum of 4 slots. Mix, glossycoating, and other layered materials can all be assigned, however, effectively allowing an unlimited number of layers. Only the first slot is required, though there is no practical use for only filling one slot. This is due to a rather severe speed penalty when using the layered material, it is

far more practical to use that material by itself. Additionally, this material will perform quite poorly compared to the mix or glossycoating materials, so you should use those material instead if possible.

name	type	description	default value
layered::namedmaterial1	string	The name of the first (base) material in the stack	None
layered::namedmaterial2	string	The name of the second material in the stack	None
layered::namedmaterial3	string	The name of the s material in the stack	None
layered::namedmaterial4	string	The name of the second material in the stack	None
layered::opacity1	float texture	The value or texture to determine transmission through the base layer. 1.0 = no added transmission.	0.5
layered::opacity2	float texture	The value or texture to determine transmission through the layer 2. 1.0 = no added transmission.	0.5
layered::opacity3	float texture	The value or texture to determine transmission through the layer 3. 1.0 = no added transmission.	0.5
layered::opacity4	float texture	The value or texture to determine transmission through the layer 4. 1.0 = no added transmission.	0.5

Matte

name	type	description	default value
matte::Kd	color texture	The diffuse reflectivity of the surface	1.0
matte::sigma	float texture	The sigma parameter in the Oren-Nayer shader in degrees. Zero for pure Lambertian reflection	0.0

MatteTranslucent

name	type	description	default value
mattetranslucent::Kr	color texture	The diffuse reflectivity of the surface	1.0
mattetranslucent::Kt	color texture	The diffuse transmitivity of the surface	1.0
mattetranslucent::sigma	float texture	The sigma parameter in the Oren-Nayer shader in degrees. Zero for pure Lambertian reflection	0.0
mattetranslucent::energyconserving	bool	Enabling forces energy conservation by making Kt = Kt*(1-Kr) (the glossytranslucent material does not have a similar flag, it acts as though it was always enabled)	False

Metal

These are the possible inbuilt metal names:

- "amorphous carbon"
- "silver"
- "gold"
- "copper"
- "aluminium"

name	type	description	default value
metal::name	string	The name of the desired inbuilt metal or nkdata file.	1.0
metal::uroughness	float texture	Roughness of the surface in the u direction from 0 to 1. Rough surfaces have blurry highlights	0.001
metal::vroughness	float texture	Roughness of the surface in the v direction from 0 to 1. Rough surfaces have blurry highlights	0.001

Metal2

Note that "fresnel" is a fresnel texture input, not a float, even though the command for a constant value is "float fresnel".

name	type	description	default value
metal2::fresnel	fresnel texture	Fresnel input for the metal's nk profile. Can accept constant values, but is meant primarily to be used with the fresnelcolor (constant color) and fresnelname (presets/measured data) textures.	5.0
metal2::uroughness	float texture	Roughness of the surface in the u direction from 0 to 1. Rough surfaces have blurry highlights	0.001
metal2::vroughness	float texture	Roughness of the surface in the v direction from 0 to 1. Rough surfaces have blurry highlights	0.001

Mirror

name	type	description	default value
mirror::Kr	color texture	The reflectivity of the mirror	1.0
mirror::film	float texture	The thickness in nanometers of the thin film coating (0.0 disables) on the surface	0.0
mirror::filmindex	float texture	The index of refraction of the thin film coating on the surface	1.5

Mix

name	type	description	default value
mix::namedmaterial1	string	The name of the first material to mix	None
mix::namedmaterial2	string	The name of the second material to mix	None
mix::amount	float texture	The value or texture to mix by	0.5

Null

The Null material takes no options.

Rough Glass

name	type	description	default value
roughglass::Kr	color texture	The reflectivity of the surface	1.0
roughglass::Kt	color texture	Fraction of light transmitted through the surface	1.0
roughglass::uroughness	float texture	Roughness of the surface in the u direction from 0 to 1. Rough surfaces have blurry highlights	0.001
roughglass::vroughness	float texture	Roughness of the surface in the v direction from 0 to 1. Rough surfaces have blurry highlights	0.001
roughglass::index	float texture	The index of refraction for the glass surface	1.5
roughglass::cauchyb	float texture	Cauchy B coefficient	0.0
roughglass::dispersion	bool	Enables accurate dispersion computation	false

Shiny Metal

name	type	description	default value
shinymetal::uroughness	float texture	Roughness of the surface in the u direction from 0 to 1. Rough surfaces have blurry highlights	0.001
shinymetal::vroughness	float texture	Roughness of the surface in the v direction from 0 to 1. Rough surfaces have blurry highlights	0.001
shinymetal::Ks	color texture	The coefficient of glossy reflection	1.0
shinymetal::Kr	color texture	The coefficient of specular reflection	1.0
shinymetal::film	float texture	The thickness in nanometers of the thin film coating (0.0 disables) on the surface	0.0
shinymetal::filmindex	float texture	The index of refraction of the thin film coating on the surface	1.5

Scatter

name	type	description	default value
scatter::Kd	color texture	Color of the material	1.0
scatter:g	float texture	Value from -1 to 1.0 that sets the asymmetry of the scattering	0.0

Velvet

name	type	description	default value
velvet::Kd	color texture	Color of the material's fuzz	1.0
velvet:thickness	float texture	Height of the fuzz	0.10
velvet:p1	float texture	Polynomial that influences the fuzz	-2.0
velvet:p2	float texture	Polynomial that influences the fuzz	20.0
velvet:p3	float texture	Polynomial that influences the fuzz	2.0

Texture

General Syntax

```
Texture NAME color|float|spectrum|fresnel TYPE <values>
```

Where NAME is a quoted string, and TYPE is one of the types listed below. The second parameter for a texture denotes if this texture defines values of type color, float or fresnel data.

Textures are declared differently from what we have seen so far. An example of a texture declaration is:

```
Texture "checks" "color" "checkerboard"
```

'checks' is the name given by the user to the texture. This name is arbitrarily chosen by the user to refer to in the material declaration. In order to tell a "matte" material to use a checkerboard texture for diffuse reflectivity, in this example you would type:

```
Material "matte" "texture Kd" "checks"
```

Again, note that you refer to a texture, you type **"texture *param*" "*name*"** where *param* is the parameter you want the texture to modulate and *name* is the name you have assigned the texture.

color declares the texture type which can be one of the following:

- color - RGB color or some sort of spectral power distribution (the latter is given by special textures, see next section). Note the US English spelling of the word "color".
- float - Single-channel data, used to texture numeric inputs.
- fresnel - A specialized form of "float" used to set optical datasets.

Some textures have more than one possible type available, a handful can even be all 3 (see next section). Note that the type declaration must be included regardless.

Texture Types

There is no default texture type. The texture types are:

Type	Variants	Description
add	float/color	Addition of color or texture to another texture.
band	float/color /fresnel	"Gradient" texture. This is a more sophisticated version of the mix texture that allows more than 2 values. Native gradient/ramp nodes should translate to this texture.
bilerp	float/color /fresnel	Bilinear interpolation texture, takes 4 values and interpolates between them
blackbody	color	Definition of the color temperature expressed in Kelvin. For example: `Texture "LampTemp" "color" "blackbody" "float temperature" [6500]`
brick	float/color	Procedural pattern for generating bricks and tile
cauchy	fresnel	Allows defining of fresnel inputs via Cauchy's equation
checkerboard	float	A checkerboard. Useful to visualize the geometry flow.
cloud	float	?
colordepth	color	A texture for simplifying absorption. Takes a target color and the depth that light should penetrate before becoming that color. This is a curve, light that penetrates shorter distances will be lighter and less saturated, and vice versa
constant	float/color /fresnel	Outputs a constant value. For example, the color version simply generates a color: `Texture "SolidColor" "color" "constant" "color value" [1.000 0.910 0.518]`
dots	float	Generates a pattern of dots
equalenergy	color	Generates a spectral power distribution at a single intensity level
fbm	float	A fractal brownian motion pattern
fresnelcolor	fresnel	A color > nk converter. Takes an input color and outputs an IOR distribution so that a reflective object reflects the input color. (primarily used for metal2 material)
fresnelname	fresnel	IOR data loader. Can load an NK file in the Sopra or Luxpop formats, or one of several presets. (this should be used in place of the luxpop or sopra textures)
gaussian	color	Defines a spectrum along a gaussian curve
harlequin	color	Geometry test pattern. Colors each face differently
imagemap	float/color	A texture defined with an external image.
lampspectrum	color	A loader of spectral presets for various light sources.
luxpop	fresnel	A Luxpop NK file loader. Deprecated in 1.0, use fresnelname instead.
marble	color	Procedural marble. Note that this is the only 3D noise texture that uses a color output
mix	float/color /fresnel	A mix of two textures based on a third value that can be either a constant or bitmap-based. Useful to combine textures or define transparency maps.
multimix	float/color /fresnel	A textures for adding other textures. Each input texture is scaled by its "weight" value, then the results are summed.
normalmap	float	A special version of imagemap for loading normal maps.
scale	float/color /fresnel	Multiplies the two input values/textures. This is the multiply/divide node for the texture pipeline.
sellmeier	fresnel	Allows defining of fresnel inputs via the Sellmeier equation
sopra	fresnel	A Sopra NK file loader. Deprecated in 1.0, use fresnelname instead.
subtract	float/color	Subtraction of color or texture from another texture. Please note that the order of the textures is important. Subtract tex2 from tex1.
tabulateddata	color	Loads external datafile that defines color across a spectrum
uv	color	A UV test pattern
uvmask	float	Generates a mask based on UV coverage
windy	float	A fractal pattern for waves on water surfaces. It has no parameters.
wrinkled	float	A procedural noise texture, good for producing wrinkling or clouding on surfaces.
blender_blend	float	Blender's "blend" texture, generates a 3-dimensional gradient.

blender_clouds	float	Blender's clouds procedural noise. Has various Perlin/Voronoi clouds patterns.
blender_distortednoise	float	Blender's distorted noise texture. Distorts one noise pattern by another.
blender_musgrave	float	Blender's Musgrave procedural noise.
blender_magic	float	Blender's "magic" texture. Produces a sort of layered cross-hatch pattern
blender_marble	float	Blender's "marble" texture. Produces bands of noise
blender_noise	float	Blender's noise texture. Produces a fine, high-frequency noise.
blender_stucci	float	Blender's Stucci procedural noise
blender_voronoi	float	Blender's Voronoi procedural noise
blender_wood	float	Blender's "wood" texture.

Common Texture Parameters (2D)

Only 2D textures share these parameters: these are 'bilerp', 'imagemap', 'normalmap', 'uv', 'checkerboard' and 'dots'. Note that checkerboard can also be set as a 3D texture, and in that case the UV coordinates are not used.

name	type	description	default value
mapping	string	Texture mapping to use. Can be "uv", "spherical","cylindrical" or "planar".	"uv"
uscale	float	The scaling of the u texture coordinates. Only applies to "uv" mapping.	1
vscale	float	The scaling of the v texture coordinates. Only applies to "uv" mapping.	1
udelta/vdelta	float	u/v offset for "planar" or "uv" texture mappings.	0
v1/v2	vector	The two vectors that define the plane for "planar" mapping	(1,0,0) and (0,1,0) respectively

Common Texture Paramaters (3D)

name	type	description	default value
coordinates	string	Coordinate system for 3D texture space. Options are "global", "local" "global normal", "localnormal" and "uv"	"global"
translate	vector	Translate transformation for 3D texture space	0 0 0
rotate	vector	Rotate transformation for 3D texture space	0 0 0
scale	vector	Scale transformation for 3D texture space	1 1 1

Specific Texture Parameters

These parameters are specific to individual texture plugins:

add

Example of use:

```
Texture "Added" "color" "add" "color tex1" [0.0 0.5 0.0] "texture tex2" ["myTexture"]
```

name	type	description	default value
add::tex1	color/float texture	The first of the textures to add together	1.0,1.0,1.0
add::tex2	color/float texture	The second of the textures to add together	1.0,1.0,1.0

band

Defines a gradient of colors or textures that can be selected using, for example, a B&W noise texture. This texture is the way you can convert a float texture to a color one. By setting the colors, or "bands", in the inputs for this texture, you assign what color is used for black, white and all possible intermediate values.

Example:

```
# Create a noise pattern using the Blender clouds procedural shader
Texture "noise" "float" "blender_clouds"
        "float bright" [1.0]
        "float contrast" [1.0]
        "string noisetype" ["soft_noise"]
        "string noisebasis" ["blender_original"]
        "float noisesize" [0.25]
        "integer noisedepth" [2]
        "string coordinates" ["global"]
        "vector scale" [1.0 1.0 1.0]

# The band texture defines four colors, equally "spaced". Black in the noise texture
# selects tex1, white selects tex4, grey points in between are selected based on their proximity
# the textures in between
Texture "Texture" "color" "band"
        "texture amount" ["noise"]
        "color tex1" [0.71054006 0.00000000 0.04896442]
        "color tex2" [0.00264841 0.00000000 0.30318916]
        "color tex3" [0.19658694 0.67383397 0.16353604]
        "color tex4" [0.70282871 0.88290709 0.84896249]
        "float offsets" [0.0 0.33 0.66 1.0]
```

name	type	description	default value
band::tex1..texN	color/float /fresnel	Any number of textures starting from tex1	none
band::offsets	float array	A series of floats, one for each texture, which selects the texture form the previous parameter. For example, value 0 for tex1 will indicate that that texture is used when the amount, see below, is set at 0.	none
band::amount	texture/float	The point picked in the range. Normally this is set to a float texture, like a noise pattern. Each pixel in the pattern becomes a selector in the band texture	none

bilerp

name	type	description	default value
bilerp::v00	color/float/fresnel	First value to bilinearly interpolate.	0
bilerp::v01	color/float/fresnel	Second value to bilinearly interpolate.	1
bilerp::v10	color/float/fresnel	Third value to bilinearly interpolate.	0
bilerp::v11	color/float/fresnel	Fourth value to bilinearly interpolate.	1

brick

name	type	description	default value
brick::brickbond	string	Determines the brick pattern. One of: "stacked", "flemish", "english", "herringbone", "basket", "chain link".	"stacked"
brick::brickwidth	float	width of the brick.	0.3
brick::brickheight	float	Height of the brick	0.1
brick::brickdepth	float	Depth of the brick	0.15
brick::mortarsize	float	Size of the mortar strips	0.01
brick::brickbevel	float	Bevel for the brick	0.0
brick::brickrun	float	Amount of linear offset down the row between adjacent rows. It works only for stacked, english, and flemish. For correct result with flemish brickrun must be 0.75	0.75
brick::bricktex	color/texture	Color or texture of the brick face	1.0
brick::brickmodtex	float texture	Modulation texture for the brick face	1.0
brick::mortartex	color/texture	Texture for the mortar	0.2

checkerboard

name	type	description	default value
checkerboard::dimension	integer	Specifies whether a 2D or 3D checkerboard texture is being used.	Values can be either 2 or 3. Defaults to 2.
checkerboard::tex1/tex2	float texture	The texture to use for even/odd squares of the checkerboard.	1 and 0 respectively
checkerboard::aamode	string	For 2D checkerboard only - the antialiasing mode to use: "closedform","supersample" or "none".	"closedform"

cloud

This is a 3D texture meant to define cloud volumes. The cloud will be defined in a (0, 0, 0)-(1, 1, 1) cube and can then be scaled and placed using standard 3D mapping transform options.

name	type	description	default value	theoretical range	suggested range
cloud::radius	float	The overall cloud radius inside the base cube.	0.5	0 < x	0 < x < 0.5
cloud::noisescale	float	The strength of the noise.	0.5	0 <= x	0 <= x <= 1
cloud::turbulence	float	The size of the noise displacement.	0.01	0 <= x	0 <= x <= 0.1
cloud::sharpness	float	Noise sharpness (the higher the more spikes you'll get).	6.0	0 < x	0 < x
cloud::noiseoffset	float	Parameter to change the random sequence used so that you can get reproducible yet unique clouds.	0.0	0 <= x <= 1	0 <= x <= 1
cloud::spheres	integer	If it is greater than 0, the cloud will consist of a bunch of random spheres to mimic a cumulus, this is the number of random spheres. If set to 0, the cloud will consist of single displaced sphere.	0	0 <= x	0 <= x
cloud::octaves	integer	The number of octaves for the noise function.	1	0 <= x	0 < x
cloud::omega	float	The amount of noise per octave.	0.75	0 <= x <= 1	0 < x < 1
cloud::variability	float	Amount of extra noise.	0.9	0 <= x <= 1	0 <= x <= 1
cloud::baseflatness	float	How much the base of the cloud is flattened.	0.8	0 <= x <= 1	0.67 <= x <= 0.9
cloud::spheresize	float	The maximum size of the cumulus spheres.	0.15	0 <= x	0 < x < 1-radius

colordepth

Many users find it's useful for LuxRender exporters to have an option to set volume absorption by result color rather than by the actual absorption color/spectrum. This texture simplifies this, so exporter writers don't have to code the math themselves. It also makes it possible to specify a resulting spectrum (by texturing color Kt) and letting LuxRender calculate the absorption spectrum from that.

name	type	description	default value
colordepth::Kt	color texture	Transmitted color at the target depth.	0.0
colordepth::depth	float	The depth in meters a ray must penetrate to turn to the color set by Kt. This is a scaling factor, shorter distances will give lighter and less saturated colors, and vice versa	1.0

densitygrid

This texture is meant to define volumes using density data obtained for example from a smoke simulation. It will be defined in a (0, 0, 0)-(1, 1, 1) cube and can then be scaled and placed using standard 3D mapping transform options. It can also be repeated like an image map.

name	type	description	default value	theoretical range	suggested range
densitygrid::nx	integer	The number of values alongside the x axis.	1		
densitygrid::ny	integer	The number of values alongside the y axis.	1		
densitygrid::nz	integer	The number of values alongside the z axis.	1		
densitygrid::density	floats	The array of values in the order x, y, z. There must be exactly nx*ny*nz values.			
densitygrid::wrap	string	The wrapping mode when outside of the base cube. "repeat" will repeat the volume informations in subsequent cubes, "black" will set the texture to 0 outside the base cube, "white" will set it to 1, "clamp" will extend the values on the cube boundaries outside the cube.	"repeat"		

dots

name	type	description	default value
dots::inside	color/float texture	The colour of the dots.	1
dots::outside	color/float texture	The colour of the background.	0

exponential

name	type	description	default value	theoretical range	suggested range
exponential::origin	point	The reference point to compute the exponential decay	(0, 0, 0)		
exponential::updir	vector	The direction of the exponential decay.	(0, 0, 1)		
exponential::decay	float	The reference distance for the exponential decay expressed as exp(-decay*Dot(p-origin, updir))	1.0		

fbm

name	type	description	default value
fbm::octaves	integer	The number of octaves to use in the spectrum.	8
fbm::roughness	float	The "bumpiness" of the texture.	0.5

fresnelname

This texture will first attempt to load the value for filename as a sopra nk file, then as a luxpop nk file, and failing this, will try to find a preset matching the value for name. If all of this fails, it will output the aluminum preset. The possible strings for name are:

- **amorphous carbon**
- **silver**
- **gold**
- **copper**
- **aluminium**

name	type	description	default value
fresnelname::filename	string	The path of a Sopra or Luxpop nk file to use.	
fresnelname::name	float	The name of the metal preset to use	aluminium

fresnelcolor

Examples of use:

```
Texture "ABC.fresnel" "fresnel" "fresnelcolor" "texture Kr" ["ABC.Kr.scale"]
```

or

```
Texture "ABC.fresnel" "fresnel" "fresnelcolor" "color Kr" [0.23 0.5 0.1]
```

name	type	description	default value
fresnelcolor::Kr	color/texture	The color or texture that should be reflected by the textured object.	0.5

gaussian

Defines a spectrum along a gaussian curve

name	type	description	default value
gaussian:energy	float	Energy of the spectrum	1.0
gaussian:wavelength	float	Frequency at the peak of the curve (in nm)	550.0
gaussian:width	float	Width of the curve (in nm)	50.0

imagemap

name	type	description	default value
imagemap::filename	string	Path to the image to use.	Required. No default.
imagemap::wrap	string	How to wrap the texture using "repeat", "black" (black beyond texture) or "clamp" (clamps to border color)	Required. No default.
imagemap::filtertype	string	one of: "bilinear", "mipmap_trilinear", "mipmap_ewa", "nearest".	"bilinear"
imagemap::maxanisotropy	float	The eccentricity of the ellipse used in the EWA algorithm	8
imagemap::trilinear	bool	Toggle trilinear filtering on from the default (better but slower) EWA algorithm.	false
imagemap::channel	string	one of: "mean", "red", "green", "blue","alpha", "colored_mean".	"mean"
imagemap::gamma	float	Input gamma value to use for reverse gamma correction	2.2
imagemap::gain	float	Scaling factor for the image	1.0

marble

name	type	description	default value
marble::octaves	integer	The number of octaves to use in the spectrum.	8
marble::roughness	float	The "bumpiness" of the texture.	0.5
marble::scale	float	Scaling factor for the noise input.	1
marble::variation	float	moderates perturbation magnitude	0.2

mix

name	type	description	default value
mix::tex1/tex2	color/float/fresnel texture	The two textures (of the same type) to mix with the "mix" texture.	0 and 1 respectively
mix::amount	float texture	The degree of mix between the two textures while linearly interpolating between them.	0.5

normalmap

A special version on imagemap for loading of normal maps. The image specified should be a tangent-space normal map (other spaces are not supported). It can be used with math/utility textures such as add or multimix to combine with with a simple bump map. It should be attached (after mixing, if needed) to the "bumpmap" slot of the surface material. Attaching it to other material slots causes it to act as a bump map.

name	type	description	default value
normalmap::filename	string	Path to the image to use.	Required. No default.
normalmap::wrap	string	How to wrap the texture using "repeat", "black" (black beyond texture) or "clamp" (clamps to border color)	Required. No default.
normalmap::filtertype	string	one of: "bilinear", "mipmap_trilinear", "mipmap_ewa", "nearest".	"bilinear"
normalmap::maxanisotropy	float	The eccentricity of the ellipse used in the EWA algorithm	8
normalmap::trilinear	bool	Toggle trilinear filtering on from the default (better but slower) EWA algorithm.	false
normalmap::channel	string	one of: "mean", "red", "green", "blue","alpha", "colored_mean".	"mean"
normalmap::gamma	float	Input gamma value to use for reverse gamma correction	1.0
normalmap::gain	float	Scaling factor for the image	1.0

scale

Multiplies 2 inputs

name	type	description	default value
constant::value	color/float/fresnel texture	The constant value of this texture.	1
scale::tex1/tex2	color/float/fresnel texture	The textures to multiply by the "scale" texture	1

subtract

Example of use:

```
Texture "Subtracted" "color" "subtract" "texture tex1" ["myTexture"] "color tex2" [0.0 0.5 0.0]
```

name	type	description	default value
subtract::tex1/tex2	color/float texture	The textures to subtract. tex2 is subtracted from tex1	1.0,1.0,1.0

tabulateddata

name	type	description	default value
tabulateddata::filename	string	Path to tabulated data file	

windy

Windy has no parameters aside from the 3D transform parameters

wrinkled

name	type	description	default value
wrinkled::octaves	integer	The number of octaves of to use in the spectrum.	8
wrinkled::roughness	float	The "bumpiness" of the texture.	0.5

Blender procedural textures emulation

LuxRender includes some code imported (with author's permission) from the Blender (https://web.archive.org/web/20170127203318/http://www.blender.org/) sources in order to emulate Blender procedural textures. The list is of supported textures is:

- **blender_clouds**
- **blender_distortednoise**
- **blender_noise**
- **blender_magic**
- **blender_marble**
- **blender_musgrave**
- **blender_stucci**
- **blender_wood**

- **blender_voronoi**

A usage example:

```
Texture "mat_proc::col1" "color" "constant" "color value" [1.0 1.0 1.0]
Texture "mat_proc::col2" "color" "constant" "color value" [1.0 0.0 0.0]
Texture "mat_proc::blender_musgrave" "float" "blender_musgrave"
Texture "mat_proc::mix" "color" "mix" "texture tex1" ["mat_proc::col1"] "texture tex2" ["mat_proc::col2"] "texture amount ["mat_proc::blender_musgrave"]
Texture "mat_proc::Kd.scale" "color" "scale" "texture tex1" ["mat_proc::mix"] "color tex2" [0.900000 0.900000 0.900000]
MakeNamedMaterial "mat_proc" "string type" ["matte"] "texture Kd" ["mat_proc::Kd.scale"] "float sigma" [0.000000] "float bumpmap" [0.000000]
```

blender_clouds

name	type	description	default value
blender_clouds:noisetype	string	Noise type. One of: 'soft_noise', 'hard_noise'	soft_noise
blender_clouds:noisebasis	string	Noise basis type. One of: 'blender_original', 'original_perlin', 'improved_perlin', 'voronoi_f1', 'voronoi_f2', 'voronoi_f3', 'voronoi_f4', 'voronoi_f2_f1', 'voronoi_crackle', 'cell_noise'	blender_original
blender_clouds:noisesize	float		0.25
blender_clouds:noisedepth	integer		2
blender_clouds:bright	float	Brightness	1.0
blender_clouds:contrast	float		1.0

blender_distortednoise

name	type	description	default value
blender_distortednoise:type	string	Distort noise by this noise pattern. One of: 'blender_original', 'original_perlin', 'improved_perlin', 'voronoi_f1', 'voronoi_f2', 'voronoi_f3', 'voronoi_f4', 'voronoi_f2_f1', 'voronoi_crackle', 'cell_noise'	blender_original
blender_clouds:noisebasis	string	Noise basis type. One of: 'blender_original', 'original_perlin', 'improved_perlin', 'voronoi_f1', 'voronoi_f2', 'voronoi_f3', 'voronoi_f4', 'voronoi_f2_f1', 'voronoi_crackle', 'cell_noise'	blender_original
blender_distortednoise:noisesize	float		0.25
blender_distortednoise:distamount	float	Amount of distortion	1.0
blender_distortednoise:noisedepth	integer		2
blender_distortednoise:bright	float	Brightness	1.0
blender_distortednoise:contrast	float		1.0

blender_magic

name	type	description	default value

blender_marble

name	type	description	default value
blender_marble::noisesize	float	Blender Texture Button(F6) -> Marble -> NoiseSize	0.25
blender_marble::noisedepth	integer	Blender Texture Button(F6) -> Marble -> NoiseDepth	2
blender_marble::turbulence	float	Blender Texture Button(F6) -> Marble -> Turbulence	5.0
blender_marble::type	string	Blender Texture Button(F6) -> Marble -> Soft/Sharp/Sharper (valid values: soft, sharp, sharper)	soft
blender_marble::noisetype	string	Blender Texture Button(F6) -> Marble -> Soft Noise/Hard Noise (valid values: soft_noise, hard_noise)	hard_noise
blender_marble::noisebasis	string	Blender Texture Button(F6) -> Marble -> Sin/Saw/Tri (valid values: sin, saw, tri)	sin
blender_marble::noisebasis2	string	Blender Texture Button(F6) -> Marble -> Type (valid values: blender_original, original_perlin, improved_perlin, voronoi_f1, voronoi_f2, voronoi_f3, voronoi_f4, voronoi_f2_f1, voronoi_crackle, cell_noise)	blender_original
blender_marble::bright	float	Blender Texture Button(F6) -> Colors -> Bright	1.0
blender_marble::contrast	float	Blender Texture Button(F6) -> Colors -> Contrast	1.0

blender_musgrave

name	type	description	default value
blender_musgrave::h	float	Blender Texture Button(F6) -> Musgrave -> H	1.0
blender_musgrave::lacu	float	Blender Texture Button(F6) -> Musgrave -> Lacu	2.0
blender_musgrave::octs	float	Blender Texture Button(F6) -> Musgrave -> Octs	2.0
blender_musgrave::gain	float	Blender Texture Button(F6) -> Musgrave -> (Ridget Multifractal, Hybrid Multifractal) Gain	1.0
blender_musgrave::offset	float	Blender Texture Button(F6) -> Musgrave -> (Ridget Multifractal, Hybrid Multifractal, Hetero Terrain) Ofst	1.0
blender_musgrave::noisesize	float	Blender Texture Button(F6) -> Musgrave -> NoiseSize	0.25
blender_musgrave::outscale	float	Blender Texture Button(F6) -> Musgrave -> iScale	1.0
blender_musgrave::type	string	Blender Texture Button(F6) -> Musgrave -> Type (valid values: multifractal, ridged_multifractal, hybrid_multifractal, hetero_terrain, fbm)	multifractal
blender_musgrave::noisebasis	string	Blender Texture Button(F6) -> Musgrave -> Type (valid values: blender_original, original_perlin, improved_perlin, voronoi_f1, voronoi_f2, voronoi_f3, voronoi_f4, voronoi_f2_f1, voronoi_crackle, cell_noise)	blender_original
blender_musgrave::bright	float	Blender Texture Button(F6) -> Colors -> Bright	1.0
blender_musgrave::contrast	float	Blender Texture Button(F6) -> Colors -> Contrast	1.0

blender_noise

This texture has no parameters of its own, only the 3D transform properties

blender_stucci

name	type	description	default value
blender_stucci:type	string	Pattern type. One of: 'plastic', 'wall_in', 'wall_out'	plastic
blender_stucci:noisetype	string	Noise type. One of: 'soft_noise', 'hard_noise'	soft_noise
blender_stucci:noisebasis	string	Noise basis type. One of: 'blender_original', 'original_perlin', 'improved_perlin', 'voronoi_f1', 'voronoi_f2', 'voronoi_f3', 'voronoi_f4', 'voronoi_f2_f1', 'voronoi_crackle', 'cell_noise'	blender_original
blender_stucci:turbulence	float		5.0
blender_stucci:bright	float	Brightness	1.0
blender_stucci:contrast	float		1.0

blender_wood

name	type	description	default value
blender_wood:type	string	Pattern type. One of: 'bands', 'rings', 'bandnoise', 'ringnoise'	bands
blender_wood:noisetype	string	Noise type. One of: 'soft_noise', 'hard_noise'	soft_noise
blender_wood:noisebasis	string	Noise basis type. One of: 'blender_original', 'original_perlin', 'improved_perlin', 'voronoi_f1', 'voronoi_f2', 'voronoi_f3', 'voronoi_f4', 'voronoi_f2_f1', 'voronoi_crackle', 'cell_noise'	blender_original
blender_wood:noisebasis2	string	One of: 'sin', 'saw', 'tri'	sin
blender_wood:noisesize	float		0.25
blender_wood:turbulence	float		5.0
blender_wood:bright	float	Brightness	1.0
blender_wood:contrast	float		1.0

blender_voronoi

name	type	description	default value
blender_voronoi:distmetric	string	Distance Metric, one of "actual distance", "distance_squared", "manhattan", "chebychev", "minkovsky_half", "minkovsky_four", "minkovsky"	actual_distance
blender_voronoi:minkowsky_exp	float	Minkowski exponent	2.5
blender_voronoi:noisesize	float	Noise size	.25
blender_voronoi:nabla	float		.025
blender_voronoi:w1	float	Weight 1	1.0
blender_voronoi:w2	float	Weight 2	0.0
blender_voronoi:w3	float	Weight 3	0.0
blender_voronoi:w4	float	Weight 4	0.0
blender_voronoi:bright	float	Brightness	1.0
blender_voronoi:contrast	float		1.0

PBRT Volumes (deprecated)

These volume types are deprecated! Instead you should use normal shapes with Interior/Exterior defined. See Mediums for more information.

Here are the volume types. There is no default type.

- **exponential**
- **homogenous**
- **volumegrid**
- **cloud**

Common Volume Parameters

Here are the common parameters for the old volumes.

name	type	description	default value
sigma_a	colour	The absorption cross section.	0
sigma_s	colour	The scattering cross section.	0
g	float	The phase function asymmetry parameter.	0
Le	colour	The volume's emission spectrum.	0
p0	point	One corner of the bounding box defining the volume calculations.	0 0 0
p1	point	The second corner of the bounding box defining the volume calculations.	1 1 1

Specific Volume Parameters

These parameters are specific to individual volume plugins:

name	type	description	default value
exponential::a/b	float	exponential::a and exponential::b are the parameters in the ae^{-bh} formula	1
exponential::updir	vector	The "up" direction along which to compute height.	(0,1,0)
volumegrid::nx/ny/nz	integer	Each of these parameters denote the number of voxels in the x,y and z directions respectively.	1
volumegrid::density	float[nx*ny*nz]	The array of density values.	0

Mediums (Object Volumes)

The mediums are essentially "volume materials", they are applied to a shape independent of the surface materials described above. That shape then becomes the bounds for the described volume. All shapes accept both an interior and exterior volume. Which side is the "interior" for meshes is determined by the surface normal. A ray striking the front a face and transmitting is considered to be entering the interior volume set for that face, striking the backface is considered to exiting into the exterior volume set for that face.

All mediums allow a fresnel input to specify the index of refraction within that volume. (Currently, this property is only accounted for when intersecting an object with the glass2 material). Relative index of refraction between the interior and exterior mediums is calculated automatically, so the index of refraction specified should always be the absolute IOR (that is, the value representing the speed of light in that volume relative to a vacuum)

Here are the medium types, there is no default type.

- **clear**
- **homogeneous**
- **heterogeneous**

Example

Here is a sample clear volume named "water":

```
MakeNamedVolume "water" "clear"
    "float fresnel" [1.3332999944468689]
    "color absorption" [0.68557313 0.43993088 0.35927745]
```

Note that absorption (clear) and sigma_a (homogeneous) represents the absorption spectrum of the volume (that is, the color lost passing through the volume). As a result, the actual color the volume appears will be the opposite of the color given for absorption. If you are finding this difficult to work with, you can attach the colordepth texture to absorption/sigma_a. Colordepth allows you to set any color input as the resulting color after transmitting a specific depth, the proper absorption spectrum will be computed automatically.

Common Medium Parameters

name	type	description	default value	theoretical range	recommended range
fresnel	fresnel	The refractive index of the volume	1.0	x > 1.0 (the core will accept values below 1.0, but the result will not be correct	1.0 - ~2.5

Specific Medium Parameters

clear

name	type	description	default value	theoretical range	recommended range
absorption	color	The absorption cross section.	0	x ≥ 0	~0.1-20

heterogeneous

name	type	description	default value	theoretical range	recommended range
sigma_a	color	The absorption cross section.	0	x ≥ 0	~0.1-20
sigma_s	color	The scattering cross section.	0	x ≥ 0	.001-.1 (as exterior, for atmospheric effects), 50-10000 (as interior, for SSS)
g	float	The phase function asymmetry parameter. This is a float triplet, with asymmetry for red, green, and blue ends of the spectrum.	0	-1.0 - 1.0	~-0.2 - ~0.2
stepsize	float	Size in meters of the ray marching steps used to evaluate integrals through the medium	1	x ≥ 0	size of sigma_s features

homogeneous

name	type	description	default value	theoretical range	recommended range
sigma_a	color	The absorption cross section.	0	x ≥ 0	~0.1-20
sigma_s	color	The scattering cross section.	0	x ≥ 0	.001-.1 (as exterior, for atmospheric effects), 50-10000 (as interior, for SSS)
g	float	The phase function asymmetry parameter. This is a float triplet, with asymmetry for red, green, and blue ends of the spectrum. Despite using the "float" command for constant values, it accepts color textures, not float textures.	0	-1.0 - 1.0	~-0.2 - ~0.2