Chapter7 Lighting and Rendering 2

Chapter7 Lighting and Rendering

Everything we see, we see because of light. Although light is invisible, it is the interaction of light with the surfaces of objects in the real world that makes things visible. Everything we see is either light reflected from surfaces or the absence of that light.

The same is true of our virtual world: in order to see objects, we need to have light. To make objects appear to have form and substance, there need to be specific light sources within our virtual space.

So lighting is very important, and not just as a means of giving objects form or making them visible. Light also places objects within a context. Daylight and outdoor lighting have a dramatically different effect than interior lighting. The way an object is lit tells us as much about the object's story as the object itself. 3D modeling and rendering can be seen as sculpting with light.

7.1 Lighting in Workspace (DX9)

The real-time engine in trueSpace is capable of creating fabulous lighting. A range of possible light types can be added to your scene, to achieve the lighting effects you are looking for. Lights do not differ from other 3D objects and can be manipulated in the same ways (moved, rotated, etc).

Real-time scene with projector light.

7.1.1 Real-time Light Types

Ambient light

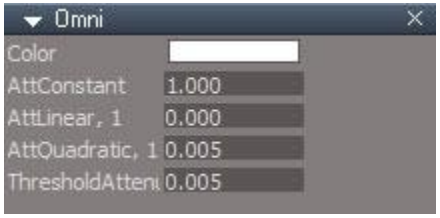
Ambient light provides equal lighting conditions for every pixel of the scene.

- **Color** – Controls the color (and intensity) of the light.

Scene lit by only an Ambient light (by Marcel Barthel).

Omnidirectional light

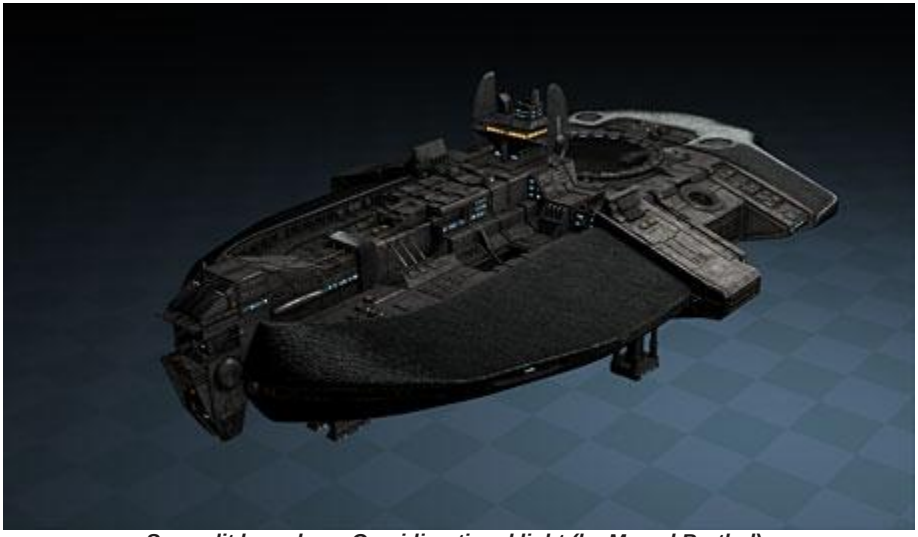
Omnidirectional lights cast the same amount of light in rays starting at the center of the light and emanating in all directions equally.



- **Color** – Controls the color (and intensity) of the light.
- **AttConstant** – Constant attenuation does not depend on distance. Values greater than 1.0 will make the light darker (2.0 means half intensity), and values in the range 0..1 will make the light brighter (0.5 means double intensity).
- **AttLinear –** Specifies a linear attenuation coefficient. The light grows less intense with distance, with a linear relationship between distance and intensity.
- **AttQuadratic –** Specifies a quadratic attenuation coefficient. The light grows less intense with distance, with a relationship to the square of the distance (light will grow less intense more rapidly than with a linear attenuation).

- **ThresholdAttenuation –** This specifies a final light intensity that is considered as zero – any pixel in an area with intensity of less than this value is considered "not lit". This parameter acts to speed up the processing of the light (a higher value will class more points as unlit, and so reduce processing).

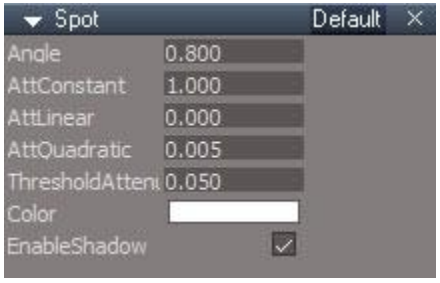
The various attenuations can be freely mixed to produce the desired attenuation level and behavior.



Scene lit by only an Omnidirectional light (by Marcel Barthel).

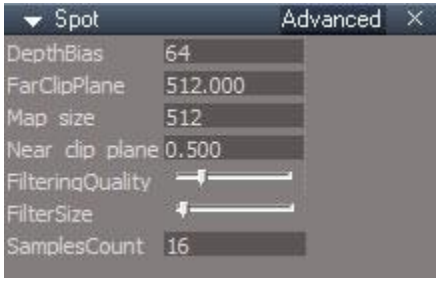
Spotlight

A Spotlight works the same way as an Omnidirectional light, but its area of incidence is limited to a **cone** with a radius set by the *Angle* parameter. A Spotlight can also have shadows in real-time. The number of active shadow casting lights may be limited by your video card's memory.



- **Angle** – Controls the radius of the cone of light.
- **AttConstant** – Constant attenuation does not depend on distance. Values greater than 1.0 will make the light darker (2.0 means half intensity), and values in the range 0..1 will make the light brighter (0.5 means double intensity).
- **AttLinear –** Specifies a linear attenuation coefficient. The light grows less intense with distance, with a linear relationship between distance and intensity.
- **AttQuadratic –** Specifies a quadratic attenuation coefficient. The light grows less intense with distance, with a relationship to the square of the distance (light will grow less intense more rapidly than with a linear attenuation).

- **ThresholdAttenuation –** This specifies a final light intensity that is considered as zero – any pixel in an area with intensity of less than this value is considered "not lit". This parameter acts to speed up the processing of the light (a higher value will class more points as unlit, and so reduce processing).
- **Color** – Controls the color (and intensity) of the light.
- **Enable Shadow** – If checked, the light will cast shadows in real-time. Shadow parameters are found in the Advanced aspect.



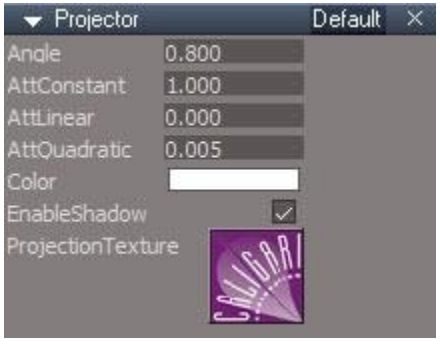
- **Depth Bias** –The main effect is to reduce the moiré pattern artifacts that appear based on the Near Clip Plane parameter. See section 7.1.3 for full details.
- **Far Clip Plane**– Specifies the maximum distance from the light that is still lit. See section 7.1.3 for full details.
- **Map Size –** Larger values give better shadows. See section 7.1.3 for full details.

- **Near Clip Plane –** Specifies the minimum distance of a point from the light source that has lighting calculations performed for it. See section 7.1.3 for full details.
- **Filtering Quality**– Controls the quality of shadows. Setting this to maximum activates Poisson disk shadows if supported by your hardware. See section 7.1.3 for full details.
- **Filter Size** – Only has an effect when Filtering Quality is set to maximum. Larger values give a softer edge to the shadow, but may need a higher samples count. See section 7.1.3 for full details.
- **Samples Count** – Only has an effect when Filtering Quality is set to maximum. Larger values increase the accuracy of soft edges shadows, but at the expense of more processing being required from the GPU. See section 7.1.3 for full details.

Scene lit by only a Spotlight (by Marcel Barthel).

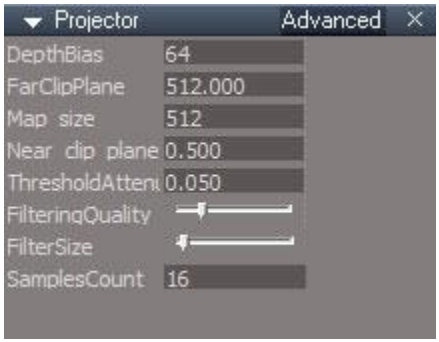
Projector light

Projector lights are a special implementation of spotlights that project a user-defined texture into the scene. Projector lights have all of the same parameters as Spotlights and also include a new one called *Projection Texture* that specifies the image the light is to project. Like Spotlights, they can cast real-time shadows. The number of active shadow casting lights may be limited by your video card's memory.



- **Angle** – Controls the radius of the square of light.
- **AttConstant** – Constant attenuation does not depend on distance. Values greater than 1.0 will make the light darker (2.0 means half intensity), and values in the range 0..1 will make the light brighter (0.5 means double intensity).
- **AttLinear –** Specifies a linear attenuation coefficient. The light grows less intense with distance, with a linear relationship between distance and intensity.
- **AttQuadratic –** Specifies a quadratic attenuation coefficient. The light grows less intense with distance, with a relationship to the square of the distance (light will grow less intense more rapidly than with a linear attenuation).
- **Color** – Controls the color (and intensity) of the light. Note that this color will combine with the colors contained within the Projection Texture (think of it as shining a light of this color through a transparent sheet with the image on it).
- **Enable Shadow** – If checked, the light will cast shadows in real-time. Shadow parameters are found in the Advanced aspect.

- **Projection Texture** – Sets the image that the light will project.



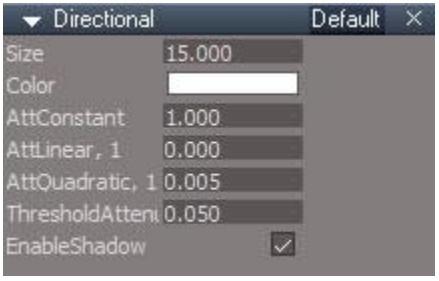
- **Depth Bias** –The main effect is to reduce the moiré pattern artifacts that appear based on the Near Clip Plane parameter. See section 7.1.3 for full details.
- **Far Clip Plane**– Specifies the maximum distance from the light that is still lit. See section 7.1.3 for full details.
- **Map Size –** Larger values give better shadows. See section 7.1.3 for full details.
- **Near Clip Plane –** Specifies the minimum distance of a point from the light source that has lighting calculations performed for it. See section 7.1.3 for full details.

- **Threshold Attenuation –** This specifies a final light intensity that is considered as zero – any pixel in an area with intensity of less than this value is considered "not lit". This parameter acts to speed up the processing of the light (a higher value will class more points as unlit, and so reduce processing).
- **Filtering Quality**– Controls the quality of shadows. Setting this to maximum activates Poisson disk shadows if supported by your hardware. See section 7.1.3 for full details.
- **Filter Size** – Only has an effect when Filtering Quality is set to maximum. Larger values give a softer edge to the shadow, but may need a higher samples count. See section 7.1.3 for full details.
- **Samples Count** – Only has an effect when Filtering Quality is set to maximum. Larger values increase the accuracy of soft edges shadows, but at the expense of more processing being required from the GPU. See section 7.1.3 for full details.

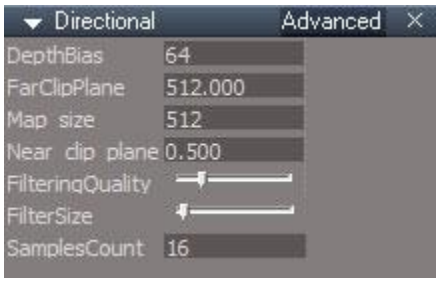
Scene lit by only a Projector Light (by Marcel Barthel).

Directional light

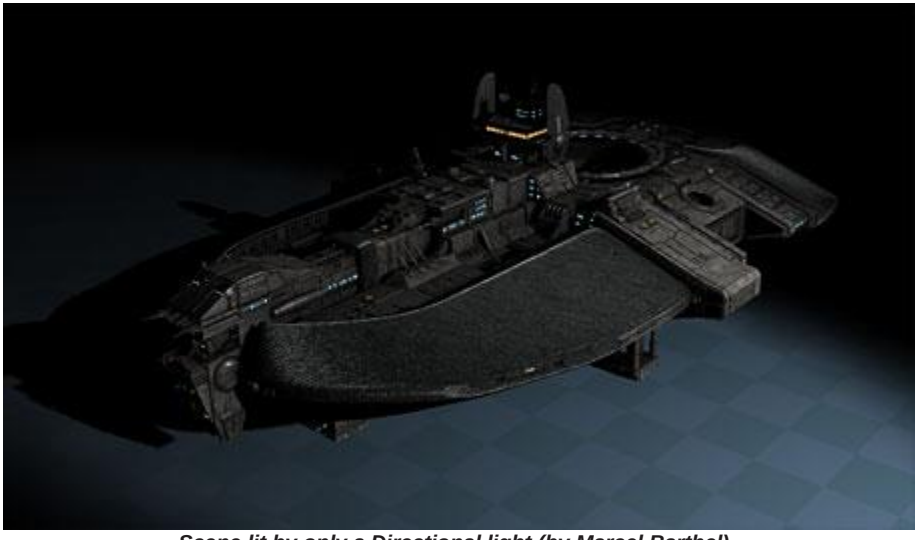
Directional lights cast light rays only in one direction and only within a **cylinder** shape specified by the *Size* parameter. Attenuation parameters work the same as for Omnidirectional lights. Like Spotlights and Projector Lights, Directional lights can cast shadows.



- **Size**– Controls the radius of the cylinder of light.
- **Color** – Controls the color (and intensity) of the light.
- **AttConstant** – Constant attenuation does not depend on distance. Values greater than 1.0 will make the light darker (2.0 means half intensity), and values in the range 0..1 will make the light brighter (0.5 means double intensity).
- **AttLinear –** Specifies a linear attenuation coefficient. The light grows less intense with distance, with a linear relationship between distance and intensity.
- **AttQuadratic –** Specifies a quadratic attenuation coefficient. The light grows less intense with distance, with a relationship to the square of the distance (light will grow less intense more rapidly than with a linear attenuation).
- **Threshold Attenuation –** This specifies a final light intensity that is considered as zero – any pixel in an area with intensity of less than this value is considered "not lit". This parameter acts to speed up the processing of the light (a higher value will class more points as unlit, and so reduce processing).
- **Enable Shadow** – If checked, the light will cast shadows in real-time. Shadow parameters are found in the Advanced aspect.



- **Depth Bias** –The main effect is to reduce the moiré pattern artifacts that appear based on the Near Clip Plane parameter. See section 7.1.3 for full details.
- **Far Clip Plane**– Specifies the maximum distance from the light that is still lit. See section 7.1.3 for full details.
- **Map Size –** Larger values give better shadows. See section 7.1.3 for full details.
- **Near Clip Plane –** Specifies the minimum distance of a point from the light source that has lighting calculations performed for it. See section 7.1.3 for full details.
- **Filtering Quality**– Controls the quality of shadows. Setting this to maximum activates Poisson disk shadows if supported by your hardware. See section 7.1.3 for full details.
- **Filter Size** – Only has an effect when Filtering Quality is set to maximum. Larger values give a softer edge to the shadow, but may need a higher samples count. See section 7.1.3 for full details.
- **Samples Count** – Only has an effect when Filtering Quality is set to maximum. Larger values increase the accuracy of soft edges shadows, but at the expense of more processing being required from the GPU. See section 7.1.3 for full details.

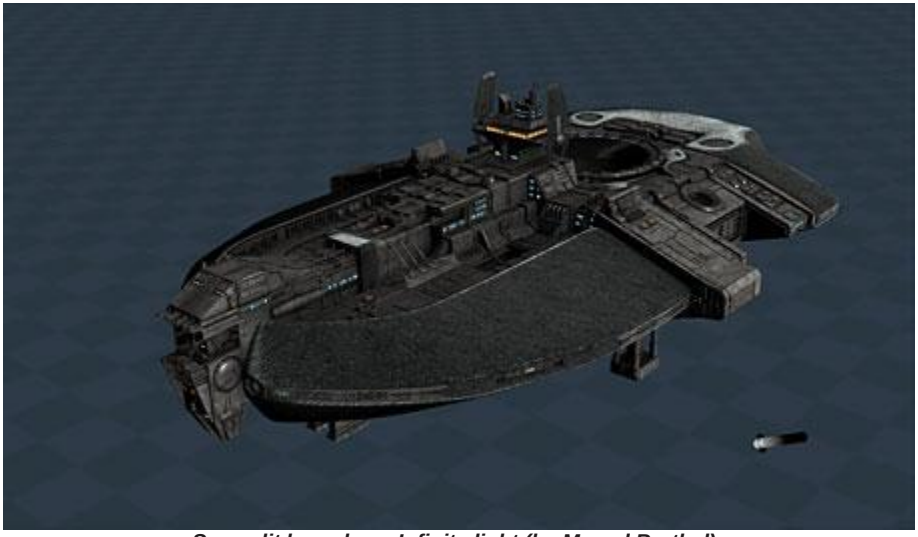


Scene lit by only a Directional light (by Marcel Barthel).

Infinite light

Infinite lights are directional and cast light rays in one direction without any size limitations. The light is spread evenly and directionally within the scene. The infinite light has only the *color* parameter.

- **Color** – Controls the color (and intensity) of the light.

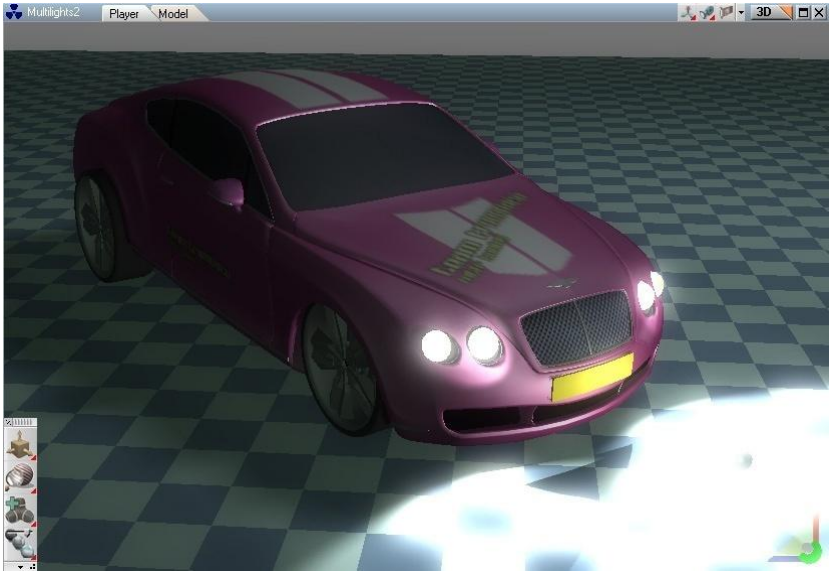


Scene lit by only an Infinite light (by Marcel Barthel).

7.1.2 Examples

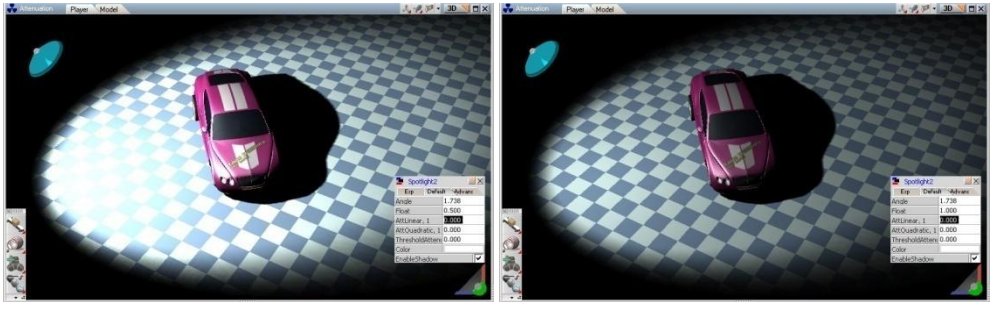
Mixing Light Types

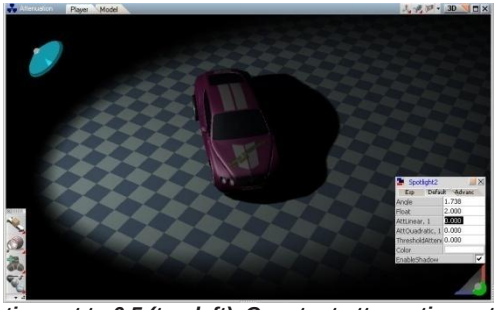
The light types are not meant to be used in isolation. Mixing more than one light type in the same scene will give you the most impressive results, as can be seen in the image below where several spotlights create the headlights, a directional light simulates moonlight, and an ambient light adds just the slightest amount touch of light to the shadows. The image also uses real-time bloom and glow.



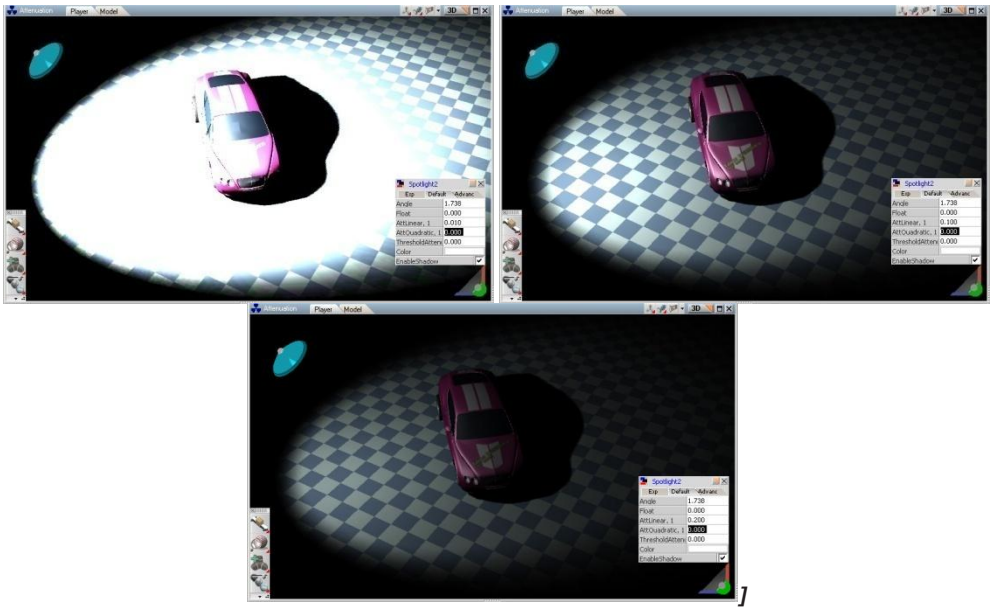
Attenuation examples

This section illustrates some of the different effects achieved using attenuation. Attenuation adjusts the brightness of the light, and is similar to the fall off parameters in other light types. Constant Attenuation simply reduces the intensity of the light; Linear Attenuation reduces the intensity of the light based on distance from the light source; Quadratic Attenuation reduces the intensity of the light based on the square of the distance from the light source. The images below illustrate different values for these parameters using a spotlight.



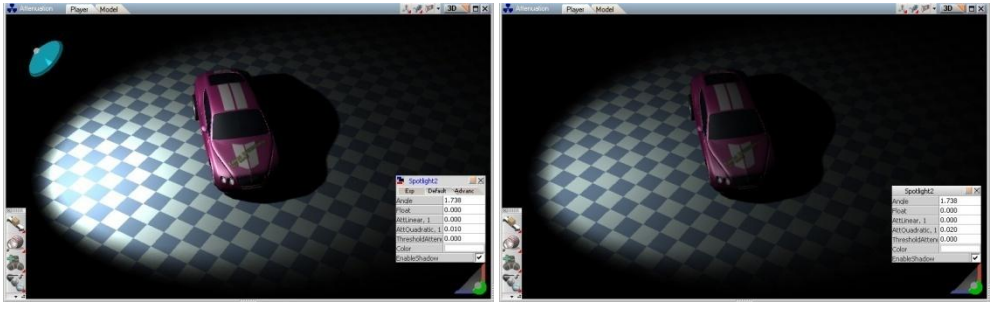


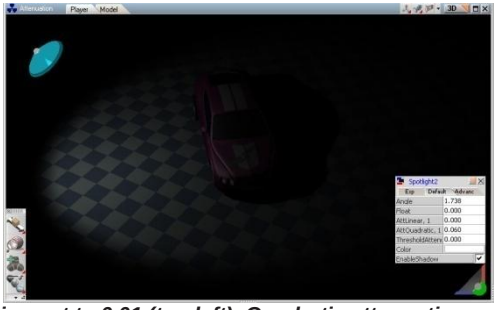
Constant attenuation set to 0.5 (top left), Constant attenuation set to 1.0 (top right) and Constant attenuation set to 2.0 (bottom).



]

Linear attenuation set to 0.01 (top left), Linear attenuation set to 0.1 (top right) and Linear attenuation set to 0.2 (bottom).





Quadratic attenuation set to 0.01 (top left), Quadratic attenuation set to 0.02 (top right) and Quadratic attenuation set to 0.06 (bottom).

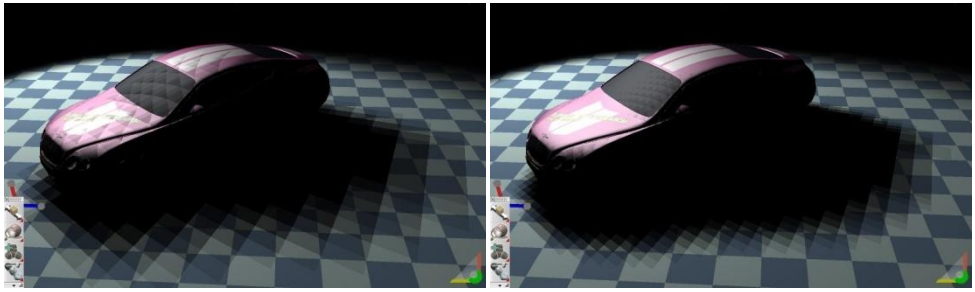
7.1.3 Working With The Real-time Shadow Parameters

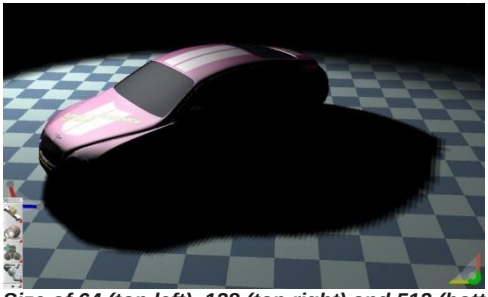
When using DX9 lights in the real-time view, there are many ways to control the shadows cast by the light that let you balance quality against speed and performance. The parameters interact with each other, so it is important to understand each of them.

Map Size:

Map Size specifies the size of the shadow map texture. A larger size will mean more accurate shadows with less boundary errors, and less of a "stair step" look to the shadow. However, it will also mean greater memory usage and slower rendering speed. You can use a lower Map Size value to optimize for speed and reduce the "stair step" effect by using a carefully chosen value in the Filtering Quality parameter.

The examples shown here all used the third level of Filtering Quality. Map Size is somewhat similar to the Map Size parameter for Mapped Shadows in the Lightworks renderer and works in a similar way, although note that for real-time light sources the Map Size does not need to be a power of two (although that has been used here in the examples for convenience).



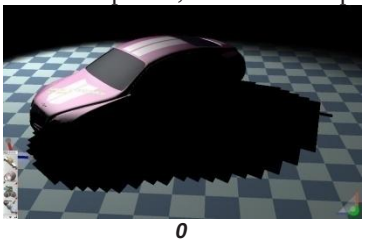


Map Size of 64 (top left), 128 (top right) and 512 (bottom).

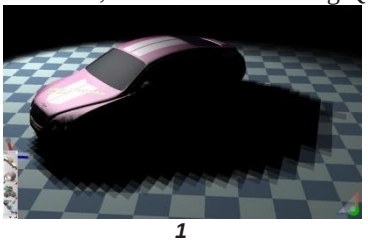
Filtering Quality:

This parameter plays a double role. At all settings except the maximum, it adjusts the quality of the filtering done on the shadows. Careful adjustment of this parameter can let you achieve better results from smaller Map Size settings (which can be preferable as lower Map Size settings allow for better real-time performance).

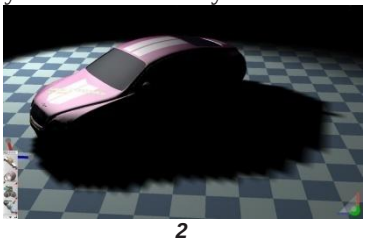
These non-maximum settings for Filtering Quality will have the most noticeable effect on shadows that use a smaller Map Size, and as the Map Size increases, the effect of Filtering Quality can become barely noticeable.



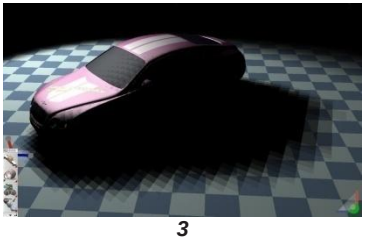
0



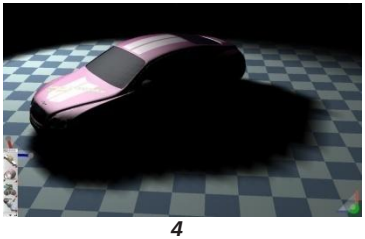
1



2



3



4

How Filtering Quality affects the shadows.

When set to maximum, the parameter activates Poisson disk shadow filtering. This filtering uses random distribution of shadow map samples (similar to internal distribution of receptors in human eye) to provide higher quality soft shadows. This can provide particularly good results, particularly where Hardware Shadow Filtering is not supported.

When set to maximum, the two parameters Filter Size and Samples come into play.

Filter Size:

When the Filtering Quality is set to maximum, then this parameter will have an effect on the shadows. Larger values will give a softer and broader edge to the shadow. Note that as the value increase, you are likely to see some "separation" in the shadow, as if multiple light sources were casting different shadows. You can either reduce the Filter Size to lower this effect, or increase the Samples Count (see below).

Samples Count:

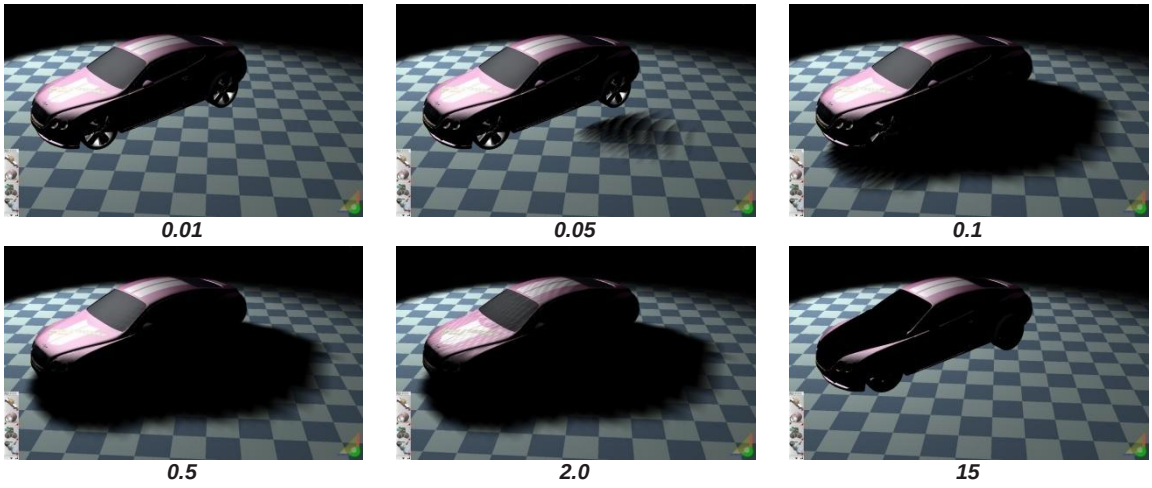
When Filtering Quality is set to maximum, then this parameter will have an effect. It controls the number of samples used to generate the soft shadows. Higher values will give more realistic and detailed soft edges, but at the expense of requiring more processing power and slower real-time performance. Larger Samples Count values are likely to be needed with an increase in Filter Size, and lower Samples Count values can be used when Filter Size is set lower.

Near Clip Plane:

This parameter specifies the minimum distance of a point from the light source that has lighting calculations performed for it. Points with which are nearer to the light source than the NearClipPlane value do not have the lighting calculations performed for them. This value will affect the shadow quality considerably - too low a value will result in missing shadows, or shadows with rough edges; too high a value will start to cause missing shadows, and missing areas of illumination, and may also cause unwanted shadow artifacts in a moiré pattern on a surface.

The effect of this parameter is closely tied to the DepthBias parameter, as higher values of DepthBias will reduce the shadow artifacts on a surface and allow for higher values to be used in the NearClipPlane parameter.

The best balance for this parameter is for it to be as large as possible without introducing unwanted shadow artifacts on an object's surface. The value will depend on how far the light is traveling from the source in the particular scene, so there is no preset good and bad values that can be recommended as it depends on the scale of your scene, and the range of effect a particular light needs to have.



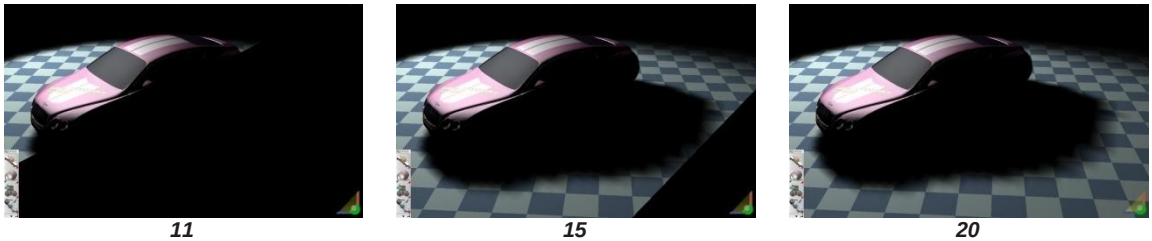
How Near Clip Plane values affect the shadows.

Far Clip Plane:

This parameter specifies the maximum distance of a point from the light source that has lighting calculations performed for it. Points with which are further from the light source than the FarClipPlane value do not have the lighting calculations performed for them.

This value will not affect the shadow quality, but it will affect which areas are illuminated by the light source. Too high a value may result in unnecessary calculations for areas that are too far from the light source to be affected by it, and too low a value will result in a sudden end to the illumination.

The best balance for this parameter is for it to be as low as possible while still lighting every object in the scene that is required to be affected by the light source.

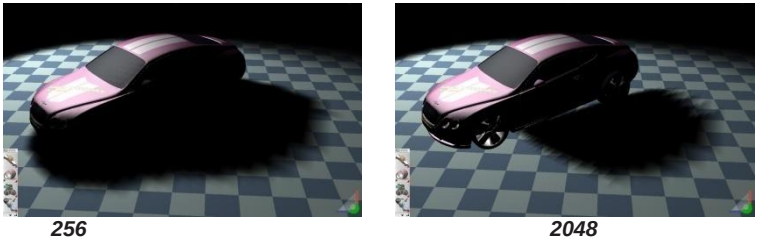


11 ***15*** ***20***

How Far Clip Plane values affect the shadows.

Depth Bias:

This parameter specifies the value that gets subtracted from the point distance when the shadow is computed. The main effect is to reduce the moiré pattern artifacts that appear based on the Near Clip Plane parameter - too small a value will leave those artifacts visible; while too high a value will add artifacts in the shadow itself or make shadows disappear altogether. Adjusting this parameter is best done in conjunction with changes to the NearClipPlane parameter.



256 ***2048***

How Depth Bias affects the shadows.

Suggested Workflow For Setting Real-time Shadow Parameters

A good guide to achieving the desired settings for a scene is as follows:

1. Add the light to the scene, and position and scale it to give the desired coverage.

2. The next thing to set up are the shadows themselves. This is always a tradeoff between quality and performance. Keep in mind who your intended audience will be – are they likely to have powerful hardware and graphics cards released within the last year or two, or are they likely to have older hardware?

3. Begin by adjusting the Map Size. You may want to start with a low value such as 64, especially if you are being performance conscious.

4. Now adjust the Filtering Quality parameter. Higher values will help remove "stair step" appearances to the shadows, but may make them too blurred or diffuse (sometimes though a soft shadow like this may be the required end result of course). Setting this value to maximum will enable Poisson disk shadows, and if you use these, be sure to experiment with the Filter Size (to make the edge of the shadow softer with larger values) and Samples Count (to make the soft edge more pleasing with higher Filter Size values).

5. If you can't find a good result by changing the Filtering Quality, then raise the Map Size. You may want to increase this by values like 32, 64, etc. Note that map sizes of 512 and above are quite large for some hardware, and likely to cause poor performance on older GPUs. Generally, map sizes of 256 or 512 give excellent quality with reasonable performance, with 64 or 128 being good for better performance (for older hardware, or if you plan on having more shadow casting lights in a scene).

6. After raising the Map Size, adjust the Filtering Quality again (and the Filter Size and Samples Count if you set Filtering Quality to maximum) and see if you find a good setting. Raising the Map Size will make the shadows more accurate. This means you can use less blurring or softening of the edges, so that you can have a sharper shadow without unpleasant "stair step" artifacts in it.

7. This may be as much as you want to adjust so you can stop here. However, you may want to go on and try to optimize some more for performance. If so, then set the Near Clip Plane parameter to be as large as possible while keeping all objects of interest illuminated by this particular light source. Good values to use start at 1.0 and the larger, the better.

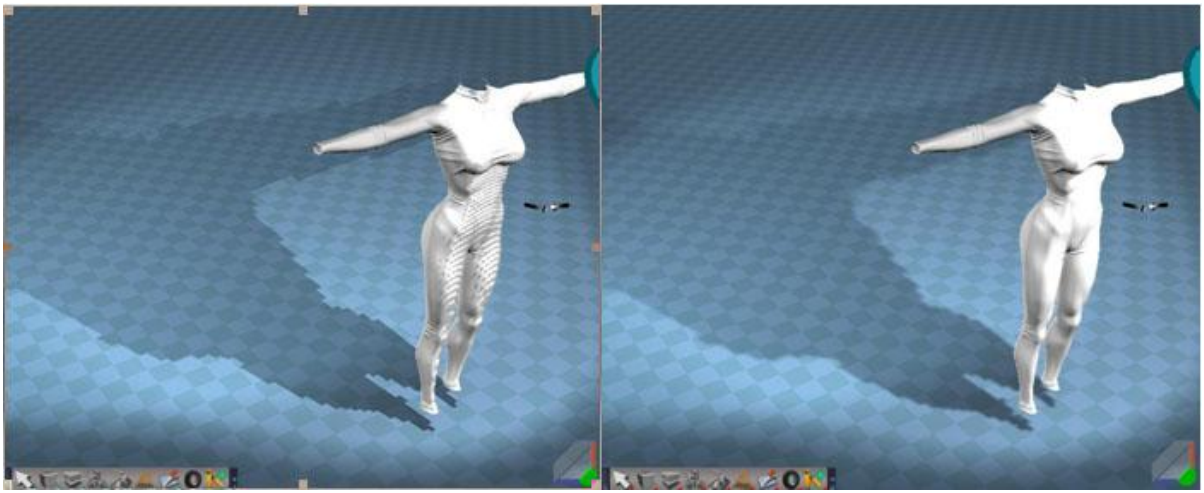
8. Next set the Far Clip Plane parameter to be as low as possible while keeping all objects of interest illuminated by the light source.

9. Now choose the smallest Depth Bias value that does not produce shadow moire errors on the object's surface. You may be able to revisit step 1 and adjust the Near Clip Plane parameter to use an even higher value (though you would not need to redo step 2, just steps 1 and 3)

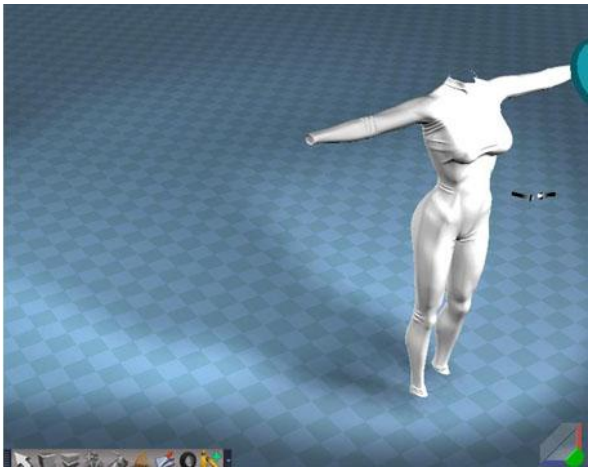
7.1.4 Shadowing improvements

You can further improve the quality of the shadows in the real-time scene by using Hardware Shadow Filtering. You can enable this in the File -> HW Settings menu, though this option can only be enabled on GPUs that support it (currently nVidia graphics cards only). Activating this option performs free bilinear filtering and hardware comparison for each shadow sample. This means that:

- Even lowest quality shadows are filtered, so you can reduce shadow settings to get better performance, yet still get good results
- You can use fewer shadow map samples to get results previously available only with bilinear filtered 2x2 and 3x3 modes.
- Depth comparison is more precise. This is especially visible on nVidia GeForceFX 5xxx cards that did not support high quality shadows.
- Shadows support true depth slope bias, which means less moiré with lower Depth Bias values. This is also visible on the image below.

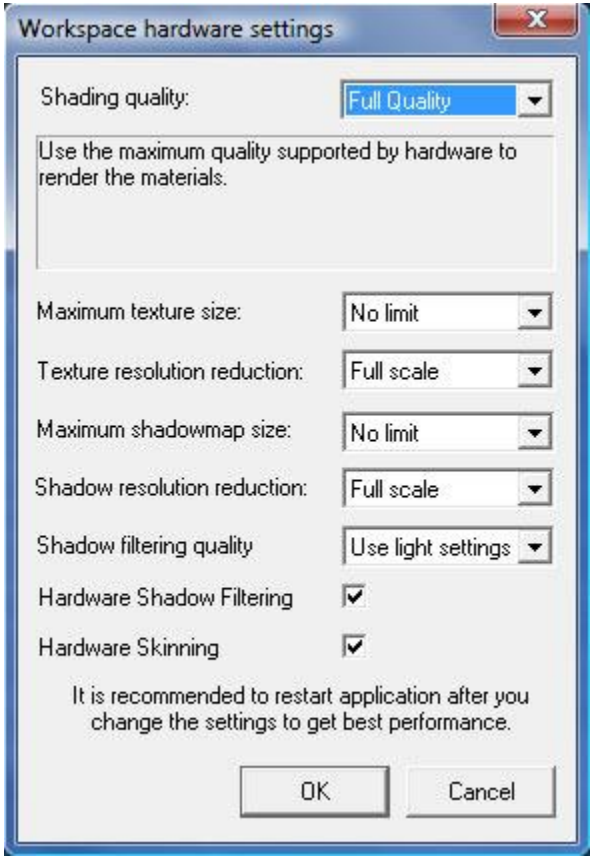


Left lowest quality shadows without HW filtering, right with filtering.
Shadow map size is intentionally small to show the difference.



Workspace Hardware Settings

Here you can control how your graphics hardware handles rendering the real-time view. This lets you achieve the balance of quality versus performance that best suits your hardware.

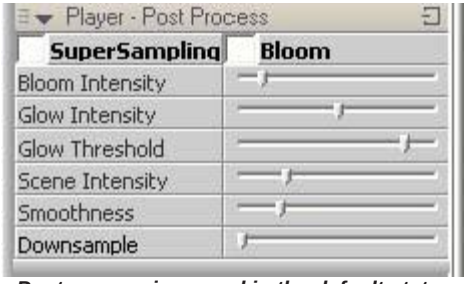


- **Shading Quality**– You can set trueSpace to display using particular shader versions. Lower versions give better performance but less visual quality. This may be necessary to maintain good performance on graphics cards that are more than 2 or 3 years old. "Full Quality" will use the best pixel shader version that your hardware can support – remember that while your hardware may support a particular version, switching to an earlier version will still give you better performance.
- **Maximum Texture Size** – Overrides any textures in the scene, and specifies the maximum size to be sent to your graphics card. This can help on graphics cards with lower amounts of memory (less than 256Mb), but will give a more blurred look to any textures in the scene.
- **Maximum Shadow Map Size** – This will override any Shadow Map sizes set for shadow casting lights in the scene. If a scene has been created for higher quality, you can quickly reduce the demands on your hardware using this setting, without having to find and edit the light sources yourself.
- **Shadow Resolution Reduction** – This reduces the overall shadow quality for all lights in the scene, giving better performance but lower quality and broader, more blurry shadows.

- **Shadow Filtering Quality** – Overrides the setting in the individual lights, and lets you specify what settings to use. Generally you use this to lower the filtering quality to give better performance, but you could use it to force Poisson disk shadows for all lights in the scene even if the scene was not set up to use them.
- **Hardware Shadow Filtering** – Enables Hardware Shadow Filtering (discussed in section 7.1.4, Shadowing Improvements). You may need to disable this manually if you experience poor performance, or if your hardware / driver does not support this feature and this has not been properly detected by trueSpace.
- **Hardware Skinning** – This uses the GPU to perform calculations relation to skinned objects (objects that are controlled and animated by skeletons). Enabling this will reduce the load on the CPU, and give better performance with newer graphics card hardware. The effect will be most noticeable on either complex models, or where there are many models in the scene being animated by skeletons. Note that it has no effect on an object that is currently selected – selected objects must be processed by the CPU, so be sure to deselect any models animated by skeletons for best real-time animation playback.

7.1.5 Real-time Post Processing

Workspace supports real-time post-processing of rendered images to achieve better and more atmospheric images. The post-processing settings panel can be displayed by switching to the Preferences aspect of the Stack View while the workspace window is active. The following image shows the post-processing settings panel in its default state.



Post-processing panel in the default state.

Supersampling

Supersampling can be used to produce anti-aliased images. The scene is rendered at double the resolution and is then down-sampled to the original resolution. The down-sampling process averages every four pixels (2x2 pixel regions) of the double-sized image and uses this average value as the pixel color in the final image. The resulting rendered scene is shown with anti-aliased edges.

The cost of such a process is that it requires more video card memory to store the double-sized image, and it also renders four times more pixels than the original mode, so can be slower on some video cards and systems.

The supersampling mode relies on the capabilities of the graphics hardware, and therefore it might not work with some video cards or at certain window sizes. If the video card does not support supersampling, or if there is not enough video card memory to enable it, then the effect is silently ignored and scene is rendered without it. For example, ATI Radeon cards like the X800 and all previous versions and model modifications (X300, X600, Radeon 9700, 9800, etc.) can support surfaces up to the size 2048x2048; therefore the maximum window size for the enabled supersampling effect is 1024x1024.

If supersampling does not work after being enabled, try decreasing the size of the window. If there is enough memory then the effect will be enabled as soon as the required maximum window size is reached.

The following figures illustrate the effect of rendering with and without supersampling enabled.

Scene with supersampling disabled (left) and a detail from the scene (right).

Scene with supersampling enabled (left) and a detail from the scene (right).

Bloom effect

The bloom effect actually consists of two parts: Bloom and Glow. Both effects can be freely combined and blended when Bloom is enabled in the post-processing panel. The Bloom effect adds the appearance of a slightly over-bright, unfocused view to the scene. This can significantly change the mood of the rendered image. The Glow effect, on the other hand, finds pixels with the greatest luminance and adds the effect of very high light reflection coming from these pixels. The following figures illustrate the difference in a scene with the Bloom/Glow effect disabled and enabled (with default settings).

Scene with Bloom/Glow disabled.

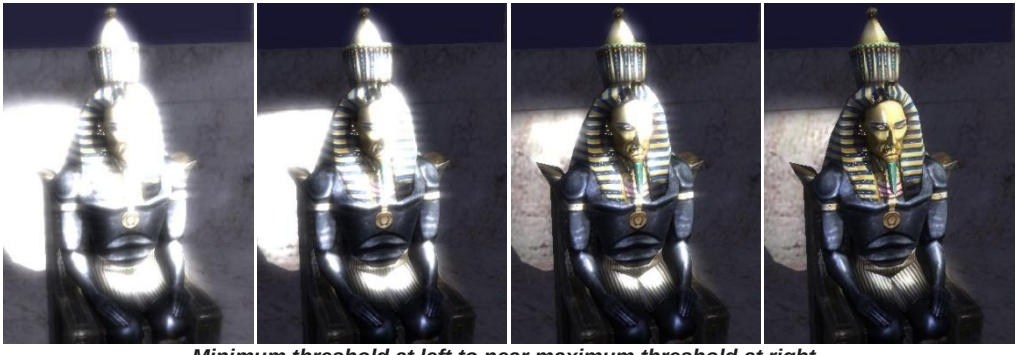
Scene with Bloom/Glow enabled.

Bloom/Glow Settings

The *Smoothness* slider controls the smoothness of both the bloom and glow effects. The higher the Smoothness value, the wider the area of the screen that bloom and glow can affect. The smoothness value also directly affects the performance of the algorithm. Higher smoothness values result in slower rendering times because more filtering passes have to be applied. The following figures illustrate how different values of the smoothness parameter can affect Bloom and Glow – note how too high a smoothness setting can cause the effect to become so smoothed out as to be hard to see. In images with more intense and larger highlighted areas, the effect could be more useful than here, where the aim is to achieve a glow just off the highlights on the statue.

Minimum smoothness at left to maximum smoothness at right.

The *Glow Threshold* slider specifies the minimum luminance that is considered as a highlight. Pixels considered as highlights are processed by the Glow part of the filter. The following figures show how different values of the threshold parameter affect the Glow effect, and control which parts of the image are affected and which not.



Minimum threshold at left to near maximum threshold at right.

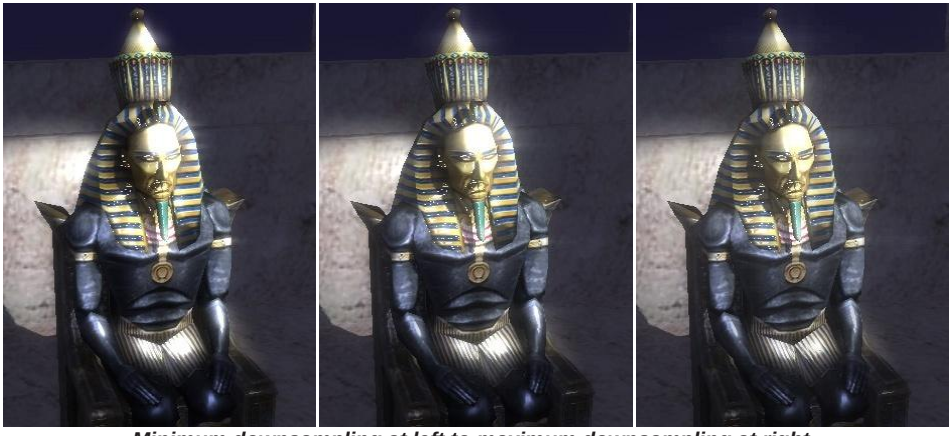
The final image is computed by blending three components: the bloom effect, the glow effect, and the rendered image of the original scene. The *Bloom Intensity* parameter specifies the amount of the computed Bloom that will be added to the final image. *Glow Intensity* specifies the amount of the computed Glow to be added to the image. The *Scene Intensity* parameter specifies the intensity of the original scene image.

The final parameter is the *Downsample* slider. The original image is always downsampled using the ratio specified by the Downsample slider, and the Bloom and Glow effects are performed on this smaller image. The downsampling step allows much faster computation of the effect. The slider allows you to specify three different

values. The minimum value means that the effect will be performed on an image that is half the size of the original image. The middle value downsamples to one quarter of the original size. The maximum value means the effect is processed on an image one eighth of the original image size.

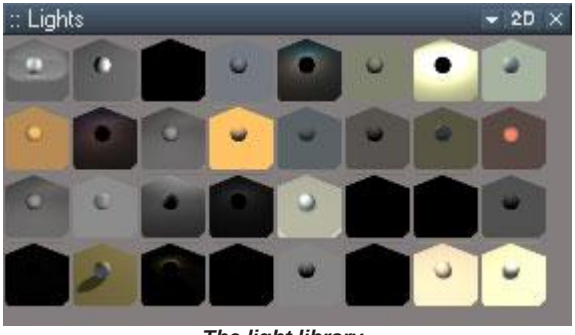
The downsampling step not only allows you to use a smaller image for the effect (and therefore significantly faster rendering speeds), but it can be also used to produce wider Bloom and Glow effects in place of the smoothness slider (while the smoothness slider decreases the speed of the rendering, downsampling actually increases it, but the trade off is that the result is less accurate).

The following figures illustrate how downsampling affects the image – note how the highlights are more clearly and sharply defined with less downsampling (you can see the difference between the individual gold and blue stripes on the mask), while more downsampling smooths out the highlights until they can barely be seen.



Minimum downsampling at left to maximum downsampling at right.

7.1.6 Real-time Light Libraries



The light library.

You can create light libraries to store your favorite lighting set ups, which you can then apply to any scene with the click of a button. The default light library that comes installed with trueSpace contains a range of lighting set ups you can choose from.

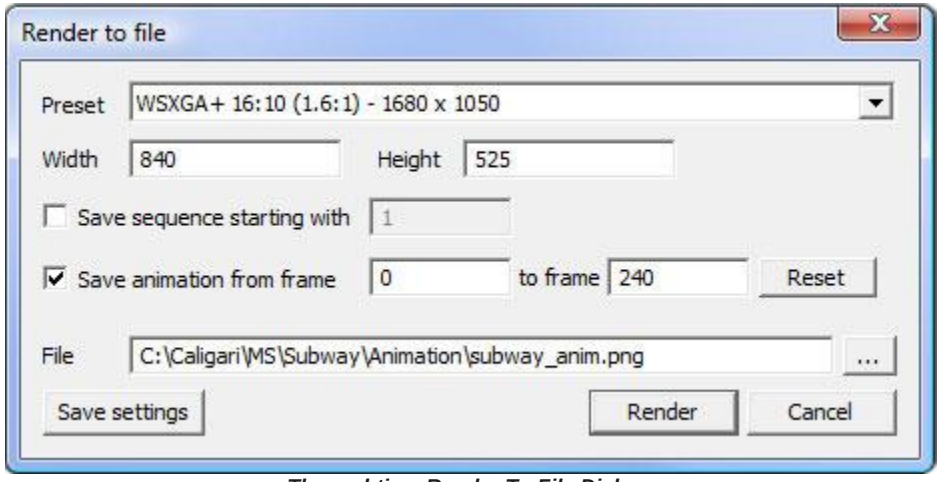
You can load a saved light set up by double clicking it in the library. To save a new set up, right click in an empty space in the library and choose Insert; to replace an existing saved set up, right click on the set up and choose Replace.

There are some important things to note about lighting set ups. First, loading a light set up will erase all lights stored at the scene level that are already present. If a light is encapsulated inside another object, then it will not be erased. Similarly, when saving a light set up, only those lights at the scene level are stored – any lights that are grouped or encapsulated inside other objects will not be saved.

7.1.7 Real-time Render To File

The capabilities of the real-time render engine are good enough to produce images and animations that you will want to keep. Not only that, but the real-time engine can let you produce 1 or 2 frames a second (on a good GPU), compared to 2 or 3 minutes per frame in an offline engine. You also get to use the anti-aliasing, supersampling, blooms and glows in the final image or animation.

To save an image or animation to file, click on the Render To File icon in the workspace (shown on the left), and this will open the save dialog (see below).



The real-time Render To File Dialog.

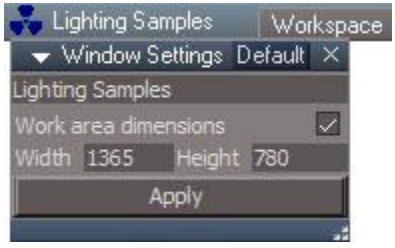
You can now, or you can enter a width and height manually. You can choose to start an image sequence at a particular number

- **Preset** – choose from a range of preset image sizes, selected to match various TV, film and video standards.
- **Width / Height** – Manually enter a particular width and height for the render.
- **Save Sequence Starting With** – This will let you render repeated images, saving them to a new file name each time without the need to manually type in a new name. This is useful if you are making a series of separate renders, perhaps from different views, or adjusting the lighting, etc. For instance, if you check this box and start at value 1, giving it a file name of MyRender, the first time you render will produce a file name MyRender1. The dialog will then update to show 2 in the Save Sequence Starting With box, and the next time you render it will be named MyRender2 automatically. If unchecked and you are rendering a single image, then no number will be

appended to the file name – in our example, you would save a file MyRender, and the next time you rendered, you would be asked if you wanted to overwrite the MyRender file (unless you type in a new name yourself in the File field). You cannot check this option and the Save Animation From Frame option at the same time.

- **Save Animation From Frame** – If checked, will render an animation from the start frame to the end frame. If unchecked, trueSpace will render just a single image. When rendering an animation, each image has the frame number appended to its file name automatically. For example, if you rendered to a filename of MyRender, then the image representing frame 0 in the animation will be named MyRender00000, and image for frame 1 would be MyRender00001, and so on. This means if you start rendering at a later frame, then your images will be numbered from that frame – e.g. if you render from frame 50, then the first image will be named MyRender00050.
- **Reset** – Will reset the From Frame and To Frame to match the range displayed in the Animation Editor. Note that this is not necessarily the same as the first and last keyframe of your animation.
- **File** – Specify the file location and name. Note that numbering will be applied to the file name if Save Animation is checked, or if Save Sequence Starting With is checked.
- **Save Settings** – Lets you capture your chosen settings so they automatically appear in the Render To File dialog next time you open it.

A Note On 3D Window Sizes



If you want to ensure your window matches the size settings for a Render To File preset, or if you want to use the size of your window in the Render To File dialog, then you can open the Window Settings dialog in the top left of your real-time window.

This dialog will let you either set your 3D window to the appropriate size by entering values and clicking on Apply, or will show you the Width and Height details to enter into the Render To File dialog.

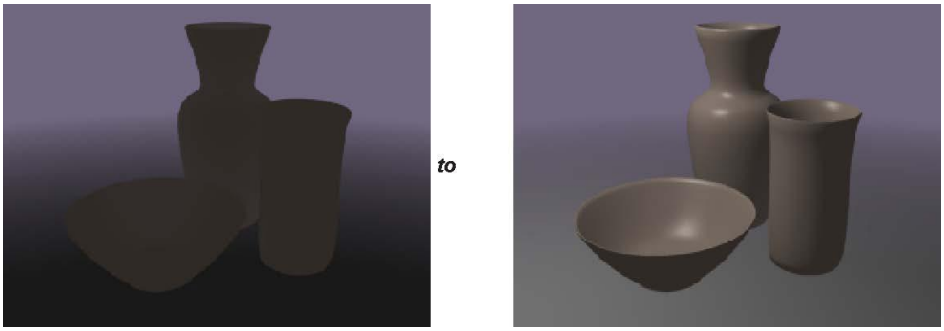
7.2 Lighting in the Model side (LightWorks and Virtualight)

7.2.1 Tutorial: A Basic Lighting Setup

Most of the conventions of lighting in CG come from photography and cinematography. One of the simplest and most commonly used lighting setups is the 3-light arrangement consisting of:

- A **Key Light** which provides the bulk of the highlight and shadow, and gives objects form.
- A **Fill Light** which brings out detail in objects that would otherwise be lost in shadow.
- A **Back Light** which brings objects into the foreground, and gives the overall image some depth.

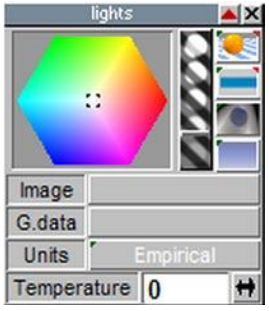
Adding a Key Light



1. Open the file pots1.scn, and **render** the file - the objects are barely visible and have no real form.

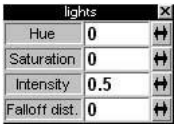
2. Add a local light source by clicking on the **Local Light** icon.

3. Select the newly created light by clicking it, then right-click on the new light to open the lighting panel.



4. Click and hold the **Falloff** icon (set at "No falloff with distance" by default) to make the flyout appear. Choose the "Inverse Squared Falloff" icon.

5. Right-click inside the lighting panel to open the lighting properties dialog.



6. Set the *Intensity* to 1.5 by entering the value into the Intensity field or by moving the **Intensity** slider until the value shown is 1.5.

7. Set the *Falloff Distance* to 10 meters by entering 10 in the *Falloff Distance* field.

8. Move the light until its position is roughly X 2, Y 1, Z 4.

9. Render the scene - the objects are now clearly visible with their forms clearly defined.

Adding a Fill Light

1. Although the objects are visible and appear to have form they lack any real detail.

2. Add an **Infinite Light** by clicking on the **Infinite Light** icon.

3. Select the newly created light by clicking it, then right-click to open the lighting panel.

4. Right-click inside the Lighting panel to open the lighting properties dialog.

5. Set the *Intensity* to 0.5 by entering the value into the *Intensity* field or by moving the slider until the value shown is 0.5.

6. Adjust the angle of the light to about X -20, Y -20, Z -45.

7. Render the scene - the objects now appear far more detailed and solid.

Adding a Back Light


to


1. The image still seems a little flat and lacking in depth

2. Add an **Infinite Light** as before, but this time set the *Intensity* to around 0.25 and the angle of the light to X 110, Y 0 , Z 0.

3. Render the scene - although the difference is subtle, the back lighting separates the objects from the background and from one another.

Adding Shadows

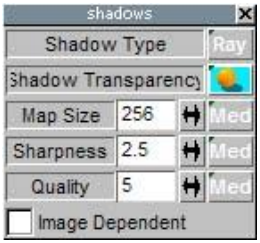

to


1. All that is missing to complete the sense of depth and form in the scene are shadows.

2. Select the Local Light source created earlier, and right-click it to open the lighting panel.

3. Enable shadowing for the light source by selecting the **Enable Shadows** icon.

4. Right-click the Enable Shadows icon to open the Shadow Properties panel.



5. Set the *Shadow Type* to “Map”.
6. Set the *Map Size* to “High”.
7. Set the *Sharpness* to “Med”.
8. Set the *Quality* to “High”.
9. **Render** the scene .

Lightening the Shadows

1. The shadows are a little dark
2. Select the **Local Light** source and reduce its *Intensity* to 0.75
3. Copy the light source then disable shadowing on the copy by selecting the **Disable Shadows** icon .
4. Render the scene .

7.2.2 Light Types and Parameters

All light types can have the following parameters:

- **Intensity:** The overall intensity of the light.

- **Shadowing:** Enabled or Disabled.
- **Physical Light:** Enabled or Disabled - this determines whether a light source will use real world values (see below) or color temperature (Kelvins).

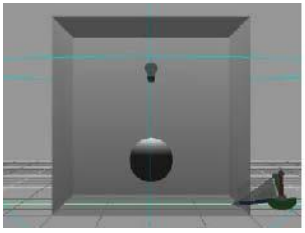
When **Physically Based Lighting** is enabled, the following units are used for the intensity setting of light sources:

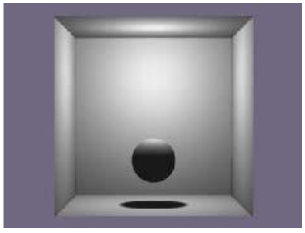
- **Empirical:** The user has no interest in intensity units and they wish it to have no effect on brightness of the emitted light.
- **Lumen:** The unit of luminous power: the lumen (lm).
- **KiloLumen:** Kilolumens(klm); each of these is 1000 lumen.
- **Lux:** The unit of luminous power per square meter: the lux (lm/m*m).
- **KiloLux:** Kilolux; each of these is 1000 lux.
- **Footcandle:** The unit of luminous power per square foot: the footcandle (lm/ft*ft).
- **Candela:** The units of luminous intensity (i.e.,power per unit solid angle): candela (lm/sr).
- **KiloCandela:** Kilocandela; each of these is 1000 candela.

Color temperature is used to describe the spectral distribution of the light. The color temperature is a positive number which defines a temperature in Kelvin derived from an ideal black body emitter (the color of the glow emitted at different temperatures). Values smaller than 150K will be ignored by light sources. If a color other than pure white (255,255,255) is set for a light source, then the final emission color is determined by combining the color and the color temperature values.

The following is a list of the different light types with descriptions of their function, operation, and additional parameters.

Local Light





Light emitted from a single arbitrary point in space. The light reaching an object's surface is determined by both surface-to-light angle and surface to light distance. Local lights are suitable for any interior light source that does not have complex spatial characteristics, for example, simple light bulbs, mantles or candle-light.

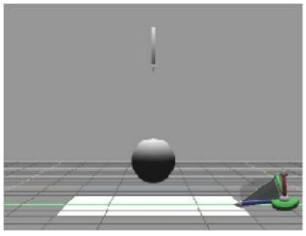
Additional parameters:

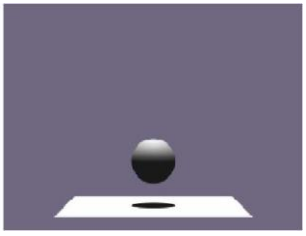
- **Falloff Type:** None, Linear or Inverse Square.
- **Color**
- **Falloff Distance:** The distance at which the light Intensity will be reduced to half its value when Inverse Square falloff is set.
- **Volumetric Enable:** Includes light in calculation of volumetric fog and shadow effects when the

Volumetric Foreground Shader is enabled.

- **Lens Flare Enable:** Renders the light as a lens flare when the **Lens Flare Post Process Shader** is enabled.

Infinite Light



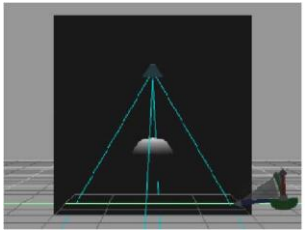


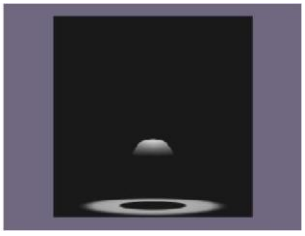
Light with only a directional reference, as if it were emitted from an infinitely distant source. The effect of the light on an object's surface is determined by surface-to-light angle. Infinite lights are ideally suited to outdoor scenes.

Additional parameters:

- **Color**
- **Volumetric Enable:** Includes light in calculation of volumetric fog and shadow effects when the **Volumetric Foreground** Shader is enabled.

Spot Light





Local light source masked by a directional "cone" which shadows the light as though it were a spot light. The light reaching an object's surface is determined by surface-to-light angle, surface-to-light distance, and by whether the object falls either inside or outside the light cone. The circular masking at the end of the cone can be adjusted to create a feathering or softening of the spot.

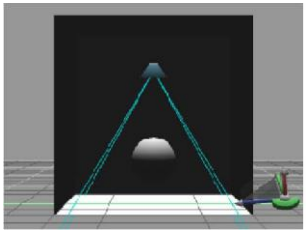
Spot lights work well for any kind of light which casts a narrow or controlled beam, such as flash lights or head lights.

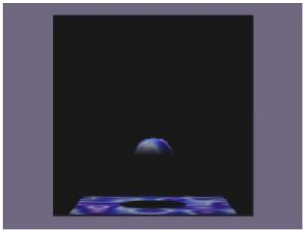
Additional parameters:

- **Falloff Type:** None, Linear or Inverse Square
- **Color**

- **Falloff Distance:** The distance at which the light Intensity will be reduced to half its value when Inverse Square falloff is set
- **Volumetric Enable:** Includes light in calculation of volumetric fog and shadow effects when the **Volumetric Foreground Shader** is enabled
- **Lens Flare Enable:** Renders the light as a lens flare when the **Lens Flare Post Process Shader** is enabled.

Projector Light





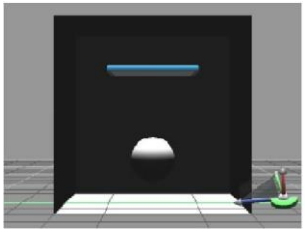
Local light source masked by an image that is then projected into the scene. The light reaching an object's surface is determined by surface-to-light angle, surface-to-light distance, by whether the object falls either inside or outside the projection mask, and by the coloring of the image.

Projector lights can also be used to create shadowing "Gobos" – images are used to cast a false shadow into a scene, hinting at detail outside of camera.

Additional parameters:

- **Falloff Type:** None, Linear or Inverse Square
- **Color**
- **Falloff Distance:** The distance at which the light Intensity will be reduced to half its value when Inverse Square falloff is set
- **Volumetric Enable: I**ncludes light in calculation of volumetric fog and shadow effects when the **Volumetric Foreground Shader** is enabled
- **Lens Flare Enable:** Renders the light as a lens flare when the **Lens Flare Post Process Shader** is enabled
- **Image:** The filename of the image to be projected.

Area Light



Light emitted from multiple points in a defined area to simulate lighting with a spatial distribution. Light is emitted from only one side of the area light object and therefore has a directional property similar to that of spot lights. The light reaching an object's surface is determined by both surface-to-light angle and surface-to-light distance.

Area lights recreate complex specular and shadowing effects by taking multiple samples from within the area lights bounds as though multiple light sources were being used. The number of samples used in raytraced rendering is determined by the radiosity quality setting. **Area lights** are especially good for simulating strip lighting and lighting panels.

Area lights are a special case as they are visible within a scene. They also work differently when **radiosity** is enabled. In radiosity solutions an area light is calculated as a mesh of a known quality setting which acts as an emitter, giving out light in much the same way as a diffuse surface but with much higher values.

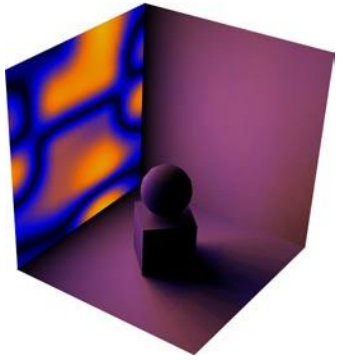
The main benefit of area lights is the reproduction of a phenomenon known as **shadow attenuation** – shadows spread and soften as the distance between the shadow surface and the casting object increases.

Note: Area lights can take a very long time to render, especially when shadowing is enabled. Shadow maps should not be used with area lights as rendering times and memory requirements will become very high.

Additional parameters:

- **Falloff Type:** None, Linear or Inverse Square.
- **Color**
- **Falloff Distance:** The distance at which the light Intensity will be reduced to half its value when Inverse Square falloff is set.
- **Volumetric Enable:** Includes light in calculation of volumetric fog and shadow effects when the **Volumetric Foreground Shader** is enabled.

Multi-colored Area Lights

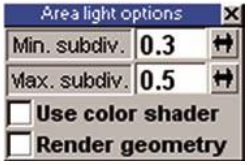


It is possible to assign a color shader to the area light. The color information from the material is used to filter the light and determine its intensity and color. This can be used to simulate radiosity effects (such as color bleeding) in scanline or raycast rendering mode using area lights

To create a multi-colored area light:

1. Paint the light a material that uses a color shader using the **Paint Object** tool. Alternatively, in solid draw mode you can drag and drop a material from a library onto the area light. Note that you can use the **Inspect** tool to query the material painted onto the light as well. Animated materials are also supported.
2. Right-click the area light (or press the 'L' key while the light is active) to open its options panel. Along with the normal lighting options panel, a panel specific to area lights will appear. Enable "Use color shader" in the Area Light options panel.

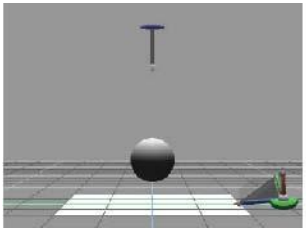
Area Light Options Panel

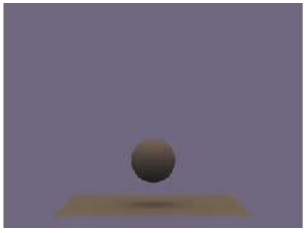


- **Min. subdiv.:** Minimum level of detail for area source decomposition. This parameter determines the initial sampling for lighting calculation and visibility analysis. No matter what the value of the other argument is, the light will always do as much work as is specified for every point being illuminated. If the parameter is too low, then shadow boundaries may not be reproduced correctly. If it is too high, then rendering times will be excessive.
- **Max. subdiv.:** Maximal level of detail for area source decomposition. This delimits the maximum amount of work the shader will carry out for any point being illuminated. If set to 0, no adaptive decomposition is performed, and the shader may fail to sample densely enough in regions of rapidly-varying irradiance (such as inside penumbrae).
- **Use color shader:** When enabled, the area light will use the color information from the material painted on it to filter the light that it casts. When this option is disabled, the area light will take its color from the normal color picker.
- **Render geometry:** When enabled, the area light geometry and its material will be visible when rendered.

Note: The process of sampling the area light sources (in other words, determining where the point lights approximating it are located, and how many of them are being used) is dynamic and rather complex. Thus instead of specifying a single value (area light sampled with 7 point lights, for example), the number and location of the "samples" is determined based on the overall conditions: point/object being illuminated, color shader attached to the area light (especially whether it is uniform or not). The sampling process itself is governed by the parameters mentioned above. As a general rule, you may want to go for low settings for both parameters to achieve the best speed (at the expense of inaccurate sampling), and increase the values if the artifacts due to area lights are visible.

Sky Light





Sky lights simulate the light reflected by and refracted through the atmosphere, as if light is being scattered within an infinitely large hemispherical volume, with the angle of the light source representing the position of the sun, and the amount of light decreasing as the angle falls away from this position.

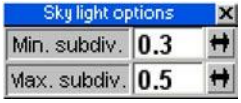
When raytracing, a sky light can be used to produce more accurate rendering of specular highlights. In **radiosity** simulations, a sky light acts as an ambient light source.

Note: Sky lights simulate sunlight and are best used in conjunction with an **Infinite Light**. Sky lights can take a very long time to render, especially if shadowing is enabled.

Additional parameters:

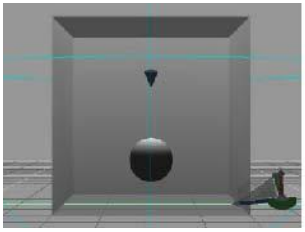
- **Falloff Type:** None, Linear or Inverse Square.
- **Color**
- **Falloff Distance:** The distance at which the light Intensity will be reduced to half its value when Inverse Square falloff is set.
- **Cover:** Clear, Intermediate and Overcast - these settings determine the effects of cloud coverage.

Sky Light Options Panel



- **Min. subdiv.:** Minimum level of detail for area source decomposition. This parameter determines the initial sampling for lighting calculation and visibility analysis. No matter what the value of the other argument is, the light will always do as much work as is specified for every point being illuminated. If the parameter is too low, then shadow boundaries may not be reproduced correctly. If it is too high, then rendering times will be excessive.
- **Max. subdiv.:** Maximal level of detail for area source decomposition. This delimits the maximum amount of work the shader will carry out for any point being illuminated. If set to 0, no adaptive decomposition is performed, and the shader may fail to sample densely enough in regions of rapidly-varying irradiance (such as inside penumbrae).

Goniometric Light



Simulates light sources with complex spatial characteristics using a 2D function to interpolate data that describes

how light is emitted from the source in every direction. The data is available in one of four text formats from many lighting manufacturers. These are:

- **CIE:** Commission Internationale de l'Eclairage. An international standard.
- **IESNA:** Illuminating Engineering Society of North America. A North American standard.
- **CIBSE:** Chartered Institution of Building Services Engineers. A British standard.
- **EULUMDAT:** A German standard.

The main purpose of goniometric lights is to provide accurate simulation of light sources such as luminaries, lamps and mantles for architectural visualization.

Additional parameters:

- **Falloff Type:** None, Linear or Inverse Square
- **Color**
- **Falloff Distance: T**he distance at which the light Intensity will be reduced to half its value when Inverse Square falloff is set
- **Volumetric Enable:** Includes light in calculation of volumetric fog and shadow effects when the Volumetric Foreground Shader is enabled.
- **G.Data:** Filename of the Goniometric Data file to be used. If no filename is supplied, the Goniometric Light behaves like a Local Light with some limited directional properties. **Note:** IES files can be found from many sources on the internet. Try entering "IES" or "photometric data" in a search engine to start locating libraries of IES files.

Image Based Light

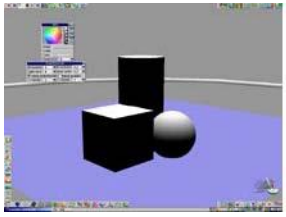


Image based lights (IBL) use an image mapped on the inside of a sphere as a source for multiple lights to recreate the complex ambient and diffuse characteristics of natural light. Points are sampled from the interior of the sphere and their color and brightness used as the basis for virtual light sources. Image based lights are particularly useful for rendering objects which are to be incorporated into photographic images, where both lighting and shadows have to be accurately reproduced. Outdoor scenes are very well suited to using IBL lights.

Use of an IBL light is quite simple; add one to your scene and scale it so that it completely encloses the objects you wish to be lit by it.

Note: Image Based Lights are computationally expensive and can take a long time to render, especially when shadowing is enabled. There are controls to reduce the complexity and accuracy of the lighting, allowing a tradeoff between precision/quality and speed.

Additional parameters:

- **Falloff Type:** None, Linear or Inverse Square.
- **Color:** If no Image is specified, the color of the light source will be determined by the color picker.
- **Falloff Distance:** The distance at which the light Intensity will be reduced to half its value when Inverse Square falloff is set.
- **Volumetric Enable:** Includes light in calculation of volumetric fog and shadow effects when the **Volumetric Foreground Shader** is enabled.
- **Image:** The image file to be used.
- **IBL Resolution:** Controls the number of samples to be taken from the image sphere.
- **Light Cutoff:** Determines how bright an image sample needs to be before it is included in lighting calculations
- **Shadow Cutoff:** Determines how bright an image sample needs to be before it is included in shadowing calculations.
- **IBL Fuzziness:** Controls the amount of random variation of the spacing of the samples - used to compensate for errors which may result from a low IBL Resolution.
- **Show Environment:** Determines whether the sphere will be rendered as a visible object. If the environment is set to be visible, you should place the camera inside the image based light sphere.
- **Below Ground:** Determines whether parts of the IBL sphere which sit beneath the ground plane will be sampled.
- **U Repeats and V Repeats:** Tiling values for the mapping of the image across the surface of the sphere.

Note: It is strongly recommended that you avoid the use of Shadow Maps with Image Based Lights as both memory usage and render times will be very high.

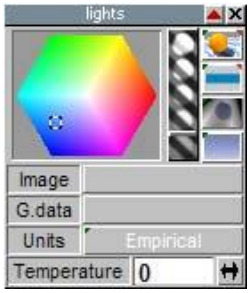
Light Arrays

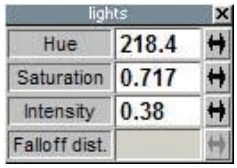
A light array is created by taking a single light source and then duplicating and arranging the light source until you have a large number of lights in a fairly small region. The intensity of those lights should be the overall desired intensity divided by the number of lights in the array. The lights in the array can be arranged in any shape desired. This is a useful tactic for creating soft shadows (using ray-traced shadows) and creating more realistic lighting if area lights and object lights (see **Artist Guide Chapter 4: Surfacing**) do not produce the desired result.

7.2.3 Light Options

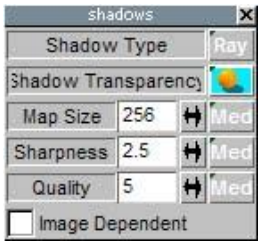
Lighting is controlled through the four panels.

The **Lights** panel is accessed by right-clicking on the currently selected light source. The main controls for light color, intensity and shadowing can be found here together with controls for falloff and lighting effects such as volumetric lighting and lens flares. Right-click the Light panel to open the numeric entry panel.





Shadows



On the main Lights panel, use the **Toggle shadow casting by current light** button to enable or disable shadow casting for the current light.

Shadows Enabled

Shadows Disabled

- **Shadow Type (Ray/Map):** Specifies whether hard-edged shadows (produced by ray casting) or soft-edged shadows (produced by shadow mapping) are created.
- **Shadow Transparency (Solid/Transparent):** Specifies whether the transparency of the object is taken into account when computing shadows. If this is set to solid, then all objects (including transparent objects) will cast solid shadows. Otherwise, shadow transparency depends on object transparency.
- **Map size:** Specifies the resolution of the shadow map. The larger the map size, the more detailed the shadows, and therefore the longer the processing time.
- **Sharpness:** Determines how soft the boundaries of mapped shadows appear. A larger value creates sharper shadows (which can cause jagged shadow edges if the map size is small), while lower values create soft shadows (which can hide the negative effect of a low resolution map to some extent).
- **Quality:** Determines the quality of the shadow by controlling how much effort trueSpace will put into computing precise shadows. This parameter works in conjunction with the other shadow parameters, especially map size and sharpness, to determine the final quality of the shadows.

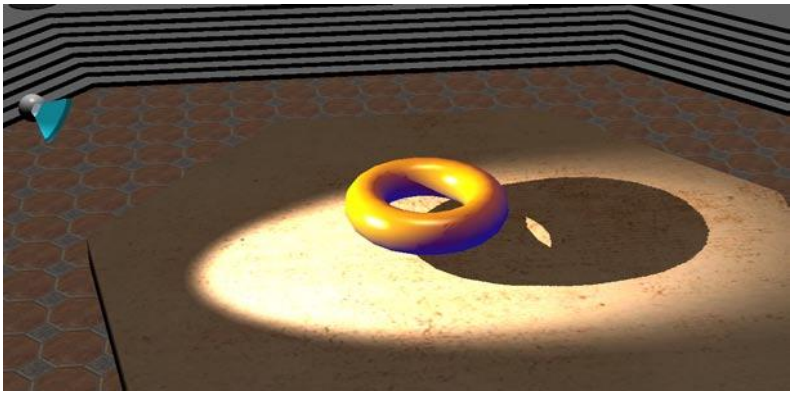
- **Image Dependent:** When enabled, the shadow map size will be image dependent, meaning that the larger the resolution of the rendered image, the greater the size (and therefore processing time) of the shadow map.

Falloff

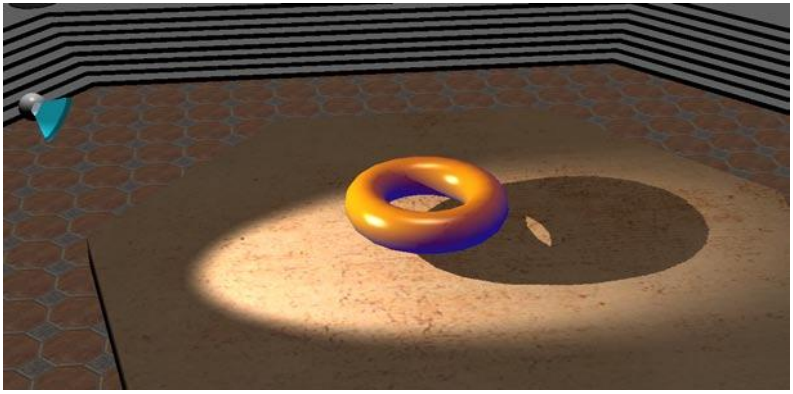
Local lights, area lights, spot lights, projector lights, and goniometric lights all have controllable falloff characteristics. Falloff is the term used to describe how the illumination from a light source is reduced over distance.

There are 3 falloff types available in trueSpace:

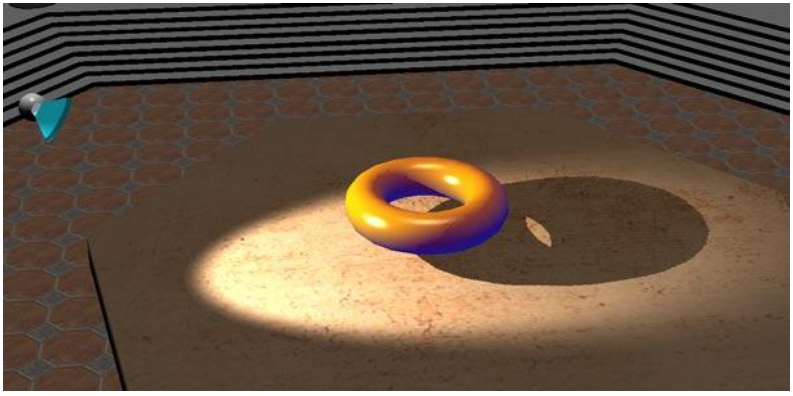
No Falloff: Illumination remains constant irrespective of the distance between the light source and the object(s).



Linear Falloff: Illumination is inversely proportional to the distance between the light source and the object(s).



Inverse Square Falloff: Illumination is inversely proportional to the square of the distance between the light source and the object(s). This most closely imitates nature.



In addition to setting a falloff type, **Falloff Distance** can also be set. This is the distance at which illumination will be half the value of the original light intensity. This value can be changed by entering a new value in the *Falloff* field (light parameters panel), using the spinner control next to the falloff field, or by clicking and dragging the falloff 'cage' that is displayed around light sources that support falloff. This cage will only be visible when "Linear" or "Inverse Square" falloff is selected.

Effects and Lighting Units

Enable/Disable Lens Flare

Select and click this icon to set a light source as the center point for a lens flare. The **Lens Flare** post process shader must be enabled for this effect to be visible.

Volumetric Light

Click this icon to either include or exclude a light source from volumetric fogging and shadowing effects. Either the **Simple Volumetric** foreground shader or the **Advanced Volumetric** foreground shader must be enabled for these effects to be visible.

Physically Based Light

Select and click this icon to enable or disable physically based lighting.

Atmosphere (for Sky Lights)

Overcast

Select and click this icon to set the **Sky Light** type to "Overcast." The sky light effect will be that of a cloudy atmosphere.

Clear

Select and click this icon to set the **Sky Light** type to "Clear." The effect will be that of a clear and cloudless sky.

Intermediate

Select and click this icon to set the **Sky Light** type to "Intermediate." The effect will be a balance between a cloudy and a cloudless sky.

7.2.4 Tutorial: Interior and Exterior Lighting

The coloring of the local light source used for the key illumination in the image above gives the image a slightly warmer cast. This subtle difference together with the soft shadows helps to enhance the illusion of an interior scene.

1. Open the file pots4.scn.
2. Select the two **Local Light** sources which are acting as **Key Lights** in the scene and delete them.

3. Create an **Infinite Light**  and alter its angle until the light is coming from roughly the same direction as before.

4. Select the newly created light, then right-click on it to open the **Lights** panel.

5. Right-click on the Light panel to open the **Light Properties** dialog.

6. In the Light Properties dialog, set the *Hue* of the newly created light to about 240 and the *Saturation* to about 0.05.

7. Set the *Intensity* to about 0.75.

8. Copy the light and enable shadowing for the copy by selecting the **Shadow Enable**  icon.

9. Right-click the **Shadow Enable** icon to open the **Shadow Properties** panel.

10. Set the *Shadow Type* to "Ray."

11. **Render** the scene  . It should now look as though the scene is set outside and illuminated by sunlight.

The left image uses an area light, and the right image uses a light array.

7.2.5 Radiosity

In real life, light also bounces around in a scene off walls and objects. Usually when you render an image, you see only the effect of light coming directly from a light source onto an object with no bouncing of the light.

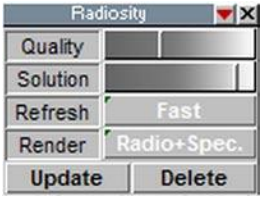
Radiosity lets you add in the effects of light bouncing around in a scene. This results in a more detailed and natural image by removing harsh shadows and stark lighting.

To use radiosity, you will find the list of parameters below, along with a suggested workflow, followed by a more detailed discussion about radiosity.

Radiosity Panel, Basic Parameters

The radiosity panel in trueSpace can be accessed by going to the real-time **Draw Objects** toolbar and right-clicking on the **Draw objects as radiosity** icon. By default, the radiosity panel is collapsed, meaning it will show only the most commonly used parameters. We'll look at these first, and then later we'll look at the advanced options that can be found under the expanded radiosity panel.

Draw as Radiosity



Quality

This parameter controls the **resolution** of the radiosity meshing grid. The higher the number, the better the quality of the radiosity solution The end result is more accurate lighting and color bleeding. Higher values will require more memory and time to process

Solution

This parameter controls how far through the calculations trueSpace will go, and is shown as a percentage in the **Progress Report**. (Open the Progress Report by clicking its icon on the toolbar located to the left of the Link Editor by default.) A value of 100 means trueSpace will keep calculating until all the light bouncing around in the scene has been accounted for (or until the user stops the calculations.)

Not all of the light bounces in a scene will have a noticeable effect on the final image, so you can set this to less than 100 if you like. However, values lower than 100 have less chance of showing color bleeding. To ensure the best results, a radiosity solution should be processed at least up to 80%. In many scenes though, color bleeding may not be visible until 95% or higher.

Refresh

This parameter does not affect the radiosity calculation. Instead, it controls how often trueSpace will display an updated image showing the current result of the radiosity solution. You have the following options:

- **None:** This option will not refresh the radiosity solution while it is being calculated; trueSpace will work out the radiosity solution, but will not create any images to show the Radiosity solution until the solution is finished This means that the radiosity solution will calculate the faster since trueSpace does not pause to create an image along the way.

- **Slow:** This option will refresh the selected view only occasionally.

- **Fast:** This option refreshes the selected view frequently, so you get to see more steps of the radiosity calculation. This comes at the cost of having trueSpace take time out from the Radiosity calculations more

often.

Note: The **Refresh** option can be set after loading a radiosity solution file as well as after stopping an ongoing radiosity processing solution.

Render

This parameter controls how a rendered image will be produced. You can set whether you want just the results of radiosity, or whether you want to use raytracing in combination with radiosity. The specific options are discussed below:

- **Pure radio:** This option sets the radiosity solution to render the scene using radiosity alone. You will only see diffuse lighting, with no specular highlights or mirror-like reflections.

- **Hybrid radio:** In this mode, trueSpace will compute **Pure Radio** first, and then subtract the direct illumination from that result at render time. The **Raycast** or **Raytrace** rendering then puts back the direct illumination along with specular highlights and mirror-like reflections. This gives you all the benefits of both radiosity and raytracing in the final image.

- **Radio+Spec:** Radiosity takes care of the diffuse lighting, and raytracing then adds in specular highlights, but not mirror-like reflections. This is useful when you don't have any objects with mirror-like reflections.

Update

This parameter starts the radiosity calculations; either beginning them for a scene or resuming from where they were stopped (including resuming from a loaded solution).

Delete Radiosity Solution

This parameter clears any radiosity solution for the scene. The radiosity solution can only be recovered by calculating it again, or loading it from a saved file, so be careful to have saved any solution that you want to keep! You will want to use this if you want to start a new solution with a different **quality** setting, for instance.

Radiosity Panel, Advanced Parameters

By default, the radiosity panel only shows the essential parameters. You can expand it to show the advanced settings which modify the way radiosity works in the scene or as rendered.

The new parameters on the expanded radiosity panel are shown below.

View

This parameter sets whether the meshing for radiosity for the currently loaded scene will be computed as view-dependent or view-independent. You will find a more detailed discussion on the meaning of those two terms in the technical section at the end. This has two options:

- **Independent:** This is the default radiosity view mode. The radiosity meshing for all surfaces is the same, as controlled by the parameters. There is no selective subdivision of the radiosity mesh based on the visibility of an object. This is the option to use if you want to render images from more than one viewpoint within the same scene.

- **Dependent:** If this options is used, trueSpace will perform additional subdivisions on all the surfaces that are directly visible, giving it a much finer radiosity meshing than for an object which is not visible. This method is faster if you are going to render from only one viewpoint using the radiosity solution, and you will not be moving the camera and rendering again (otherwise rendering from a different viewpoint would require you to calculate the radiosity solution again to get a good result.)

Subdiv

This parameter controls the kind of subdivision that will be used for the Radiosity mesh, and has two settings:

- **Adaptive:** This method actively subdivides the radiosity mesh into smaller, finer areas when there is a big difference in illumination values at the corners of the patch. If such a large difference exists at the corners of the patch, then the lighting levels must change significantly within that patch so it is subdivided in order to more accurately capture that change in illumination.

  Please note that this parameter is dependent upon the initial *Quality* parameter. Even if you have adaptive subdivision, if the initial mesh Quality setting is coarse (a low value), no amount of adaptive subdivision will improve the final result without the Quality setting being increased.

- **Initial:** This method performs the calculations using the initial radiosity meshing only. The whole radiosity calculation will only rely on the resolution of the initial mesh (*Quality/Max Area* parameter) and will not create detailed finer mesh subdivisions. This is useful for inspecting and analyzing the results of direct

illumination in the scene, but probably is not good enough for a final rendering if accurate light bouncesand color bleeding are required.

Iterate

This parameter controls which areas of the radiosity solution are calculated. This parameter defaults to "Complete" and has three options:

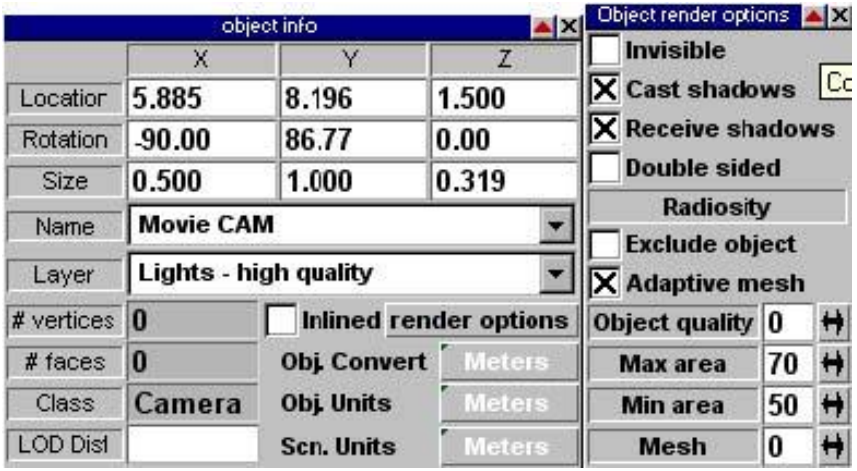
- **Meshing:** This option means that only the initial meshing (*Quality* setting) is performed. This is mainly used for evaluating the setting of the *Max Area* parameter to see if it creates a dense enough mesh to adequately capture the light changes across important surfaces. No actual calculation of illumination is carried out, but rather it simply performs the subdivision so that you can check the wireframes and see whether the density is acceptable.

- **All Lights:** This option only processes the effects of the lights in the scene and ignores any light bouncing between objects. This parameter is good for evaluating the light setup and distribution in a radiosity scene.

- **Complete:** This option is the default setting for this parameter. The "Complete" option tells trueSpace to process the radiosity meshing for all the surfaces (except for those which have been explicitly excluded) and then to calculate the lighting. If you do not use this setting, you will not see any radiosity effects in the render.

Individual Object Radiosity Meshing Parameters

The last three parameters are the individual object radiosity mesh settings, which are *Max Area*, *Min Area* and *Mesh* parameters. These settings ensure that important objects get enough accuracy in the calculations without minimizing the overall calculations to keep render times as low as possible. All of the per object radiosity meshing settings are only used when the object's radiosity mesh is set to anything other than zero – this option can be found under **Object Tool –> Render Options –> Object Render Options –> Radiosity –> Object Quality**.



Max Area

This parameter is for the initial resolution of the object's radiosity mesh and sets the largest mesh patch for the initial radiosity mesh. The lower the value, the denser the mesh, and the more accurate (but slower) the radiosity calculation will be. Use lower values for more important objects.

Min Area

This parameter sets the cut off after which no more adaptive subdivision will be done (see the *Subdiv* parameter information for more details on adaptive subdivision.) Once the mesh is subdivided to this level, no more subdivision will occur. This parameter mainly sets the smallest radiosity mesh while the solution is being calculated.

Mesh (accuracy)

This parameter controls the level of subsequent adaptive subdivision. In essence, this controls the "trigger" that lets trueSpace decided when to subdivide the radiosity mesh to achieve more detail, for example in areas where there is a large change in light energy. A higher setting will trigger more adaptive subdivisions, and allows finer adaptive subdivisions on areas that are important like shadow boundaries. To capture soft and diffuse **penumbras** (shadow edges), this parameter must be set to a high number.

A suggested workflow for using the per object radiosity meshing setting:

1. Set the global meshing (*Quality*) first:

   The Quality setting controls the overall level of the radiosity mesh resolution in the scene. Set this to a low level (around 5) to preview the way the lights affect the objects in the scene.

2. Process the radiosity solution:

   Start the radiosity calculations and observe the way radiosity affects the scene, especially in the areas where there is a drastic light energy change such as a shadow boundary. Let it run through a few steps to ensure that an adequate solution has been performed, and then stop the radiosity solution.

3. Inspect and analyze the scene:

   You should move around the scene in real-time (via **OpenGL/Direct 3D**), or move and render a few images from different locations to see which areas of the scene are the most important. Take note of the objects which have jagged shadows or lighting areas. These objects will need to have their individual meshing parameters changed (*Object quality*, *Max Area*, etc.)

4. Delete the existing radiosity solution in memory by clicking on the **Delete Radiosity Solution** button.

5. Change the *Min Area*, *Max Area*, and *Meshing* accuracy for the important objects:

   Do this for all of the affected objects and make sure that the most important objects have higher *Object quality* and the lowest *Max Area*, *Min Area* and *Mesh* settings.

6. Start the radiosity processing again and repeat step 3 to 5 if needed.

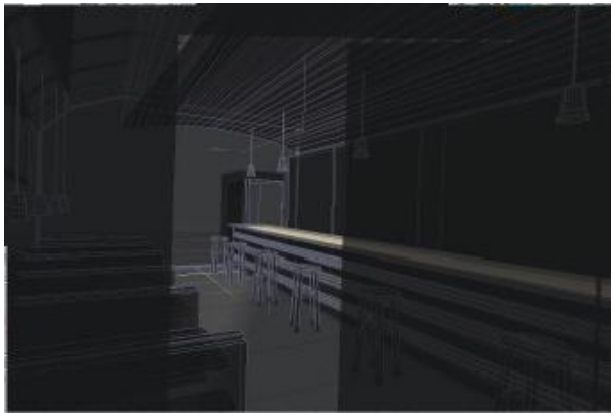
With the new settings, the radiosity solution might take longer than before.

7) Do a final Render and see the results. If you are only interested in the way the direct illumination affects the scene and your scene will not be animated, you can set the radiosity parameters to **View–Dependent**, **Subdiv–Initial Only** and **Iterate–All Lights**.

Real-time Radiosity preview with OpenGL or Direct 3D

While calculating the Radiosity solution, trueSpace can display how the scene looks with the results of the Radiosity calculations. This does not require an expensive video card, but rather a video card that supports OpenGL or Direct 3D with hardware acceleration.

1. Open the "Interior.scn." This is a relatively simple scene that has been optimized to show the real-time radiosity display in trueSpace.



The image above shows the scene with just direct illumination and shadows. Notice how dark it looks.

2. Now go to the **Radiosity** icon and right-click it to open the **Radiosity panel**, then right-click anywhere inside the panel to open the numerical entry panel. Change your settings to match those in the image below:

3. Click **Update** to start the Radiosity calculations. trueSpace will show in the **Progress Report** "Initial radiosity meshing X polygons" Where X is the number of polygons that trueSpace has found to be relevant and visible in the scene. trueSpace then shows the following:

   "Radiosity in progress: X steps, XX% completed...XXXXXX polygons"

This shows the number of steps and the percentage of the solution completed, while XXXXXX shows the number of polygons involved in the solution (including the adaptive subdivision meshing.)

The early radiosity solution just shows the 'direct illumination' energy from the downward pointed spotlights on the right side of the scene and the scene remains dark until the next steps happen (as shown in image above) where most of the direct illumination from the spotlights is accounted for and the first bounces are beginning to take place.

Now, the effect of light bouncing in the scene can be observed on wall to the right side. The direct illumination of the spotlights on the left side has also been started.

Light continues to bounce not just once, but two or more times for brightly lit surfaces. Note that the dark areas are now lighted and the scene is more open due to the secondary light bounce distribution.

4) If you want to see the full effect of the light bouncing in the scene, process the radiosity solution until at least 70% is reached. You can stop the process by a right-clicking twice or by pressing the ESC key. trueSpace will complete the current round of calculations before stopping so it may not appear to respond immediately.

> **Tip:** *Small black artifacts may appear in places like corners and sharp edges. For example, in the Interior scene used here, artifacts occur because the back wall and ceiling are two separate objects in a hierarchy rather than one object. This may cause shadow leaks.*

The solution is to Boolean union the wall and ceiling together, creating a correct Radiosity mesh for the scene and removing the shadow leaks completely. Be aware of overlapping objects that can cause such artifacts, and consider using the Boolean tool to make them into one object where possible.

Radiosity with Tone Mapping

It is not entirely possible to predict how a scene will look once a Radiosity solution has been completed, and sometimes it can appear very bright with washed out highlights. The **Tone Mapping** tool provides a good solution for this.

You can load (or re-use) the Radiosity solution from the previous tutorial. Alternatively, you can perform a quick Radiosity calculation by loading the interior.scn into trueSpace, right-clicking on the **Radiosity** icon, clicking Update to start the processing, and letting it run until at least 60% is reached.

Right-click on the Tone Mapping icon on the **Rendering** toolbar opens the **Post-Process Editor** panel. This reveals a preview window where the current view is rendered normally and then tone mapping is applied. You can see in the image above that we have a very dark image with just 60% Radiosity calculated. Experiment with Tone Mapping.

Click on the little strip in the upper right hand corner of the Post-Process Editor to open the options parameter. Expand the Tone mapping panel to see the other parameters:

The **Tone mapping panel** will show the following parameters

- **Brightness:** This parameter controls how intense and how dark the tones in the image will be.
- **Balance:** This parameter controls the tonal distribution in the scene between the light and dark areas, and determines where the 'tonal emphasis' will be in the image.
- **Auto-setup:** trueSpace determines which 'tonal range' will be used from the computed luminance level as determined by the radiosity solution processing. The default process is that trueSpace will render an image and then determine the maximum and minimum level of intensity in the scene. Sometimes, if you render a viewport (**Camera**, **Right**, **Left**, **Top** or **Perspective** view) the current state of tonemapping will not be applied and there will be a discrepancy between what is being shown by the Post-Process Editor and what is being rendered in the viewport. Click on the **Detect** button to let trueSpace know that you want the current Tone mapping settings to be applied to the rendering (viewport or otherwise).

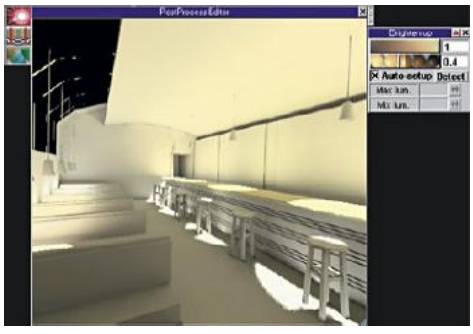
Change the Tone mapping settings to *Brightness* .90 and *Balance* to .49, and to see the result below:

Note that this has affected mostly the middle tones to the upper highlight tones. The *Brightness* parameter controls how intense the upper middle tone and the highlights will be.

Now change the Tone mapping to *Brightness* 1.0 and the *Balance* to .70, and you get this result:

This gives more weight to the dark areas of the scene, and shifts the emphasis onto the dark areas more than the light areas.

Finally change the *Balance* to .40 and keep the *Brightness* to 1.0:



This shifts the emphasis to the light areas of the scene, and makes the dark areas lighter since they have been pushed to the middle gray tones in the scene.

Tone mapping is great for changing the tonal distribution in the scene, but will never compensate for an inadequately processed radiosity solution as it can never add tones that were not there in the first place.

Radiosity in trueSpace

The manner in which radiosity works (in general) is as follows:

1. *Account for all visible surfaces.* All object hierarchies are collapsed, and the radiosity module inspects all of the visible surfaces in the scene based on the normal of each polygon. This makes the scene ready for the radiosity calculation.

2. *The whole scene is covered with a uniform radiosity mesh.* All surfaces are divided up into areas called patches (similar to a 2D image being made up of pixels – the surface has to be broken up into discrete areas of a set size.) The size of these patches is controlled by the *Quality* parameter (higher quality means smaller patches for a smoother and more accurate result). The sorting of all of the visible objects in the scene (step 1) plus the setting up of this mesh is the reason that large scenes may take a long time before they first refresh.

3. *Light energy is shot from the luminaires (lights).* The direct illumination from the scene is stored.

4. *The surfaces that received the initial light energy then become light emitters themselves.* All the surfaces that received direct illumination in step 3 then go on to emit light as if they were a light source. This is the essence of radiosity and lets light bounce around the scene.

5. *Light energy transfer is continually distributed into the scene until it finishes or is stopped.* Eventually all of the light energy has been bounced around in the scene and is "used up" or the user stops the calculations manually. The radiosity mesh may no longer be uniform because of the creation of additional patches and elements (called **adaptive subdivision**) for areas where the light is changing rapidly over the surface.

6. The *Radiosity solution is stored.* The radiosity solution is stored with the geometry either for further processing or for display. This radiosity solution storage is the reason for the intensive use of RAM and disk space when there is radiosity in the scene.

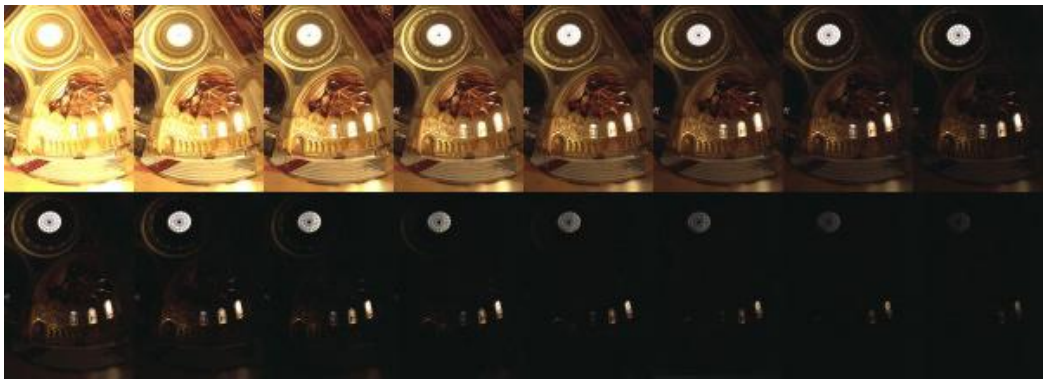
7. The *Radiosity solution is rendered.* The radiosity mesh is then converted to a visible form for viewing through real-time display or by rendering.

The radiosity module keeps all of this information in the geometry (vertices) of the scene. This is why you cannot have animated objects in the scene while there is radiosity in it. You will have to exclude (**Object Tool–Render Options–Object render options–Radiosity–Exclude object parameter**) the animated object from the radiosity solution.

This is also the reason you cannot point edit a mesh with a radiosity solution on it. If the loaded scene does not have a radiosity solution applied and it is loaded (.LWR), it will take time to re-map the saved, calculated radiosity mesh, so be patient when loading up a previously calculated solution!

7.2.6 HDRI

High Dynamic Range Images (HDRI) store brightness values as floating point numbers rather than as numbers between 0 and 1, and represent brightness levels visible in the real world



HDRI implementation

There are two possible ways to implement HDRI in a trueSpace scene:

- HDRI as an intensity scalable image
- HDRI as an environment (external light source)

In the first instance we can load an HDRI image as a standard picture material and use it for an object.

More commonly however, HDRI is used as a light source. In the second case, HDRI can be used as a global environment (foreground shader) with a scalable intensity value.

trueSpace supports the following high dynamic range image formats:

- **OpenEXR** is a high dynamic range image format that was developed and made freely available by Industrial Light and Magic (ILM). OpenEXR uses 16 bit floating point values rather than 32 bit floating point values since they easily provide sufficient range for image data.
- **HDR** is a format that represents each pixel by at most four bytes. This provides slightly less accuracy than OpenEXR, but is still sufficient for most applications. A larger number of images are currently available in this format.

HDRI Environment

A major application of HDRI has been the use of high dynamic range images as light sources. This provides a straightforward way to specify lighting environments that would otherwise be quite complex to define.

The advantages of using HDR images rather than regular images for this purpose are:

- Reflections will have much greater range of intensity, giving a more 'deep' and realistic image.
- It is possible to use HDR environments to light an object or a scene so that it matches the lighting conditions present when the environment image was captured.

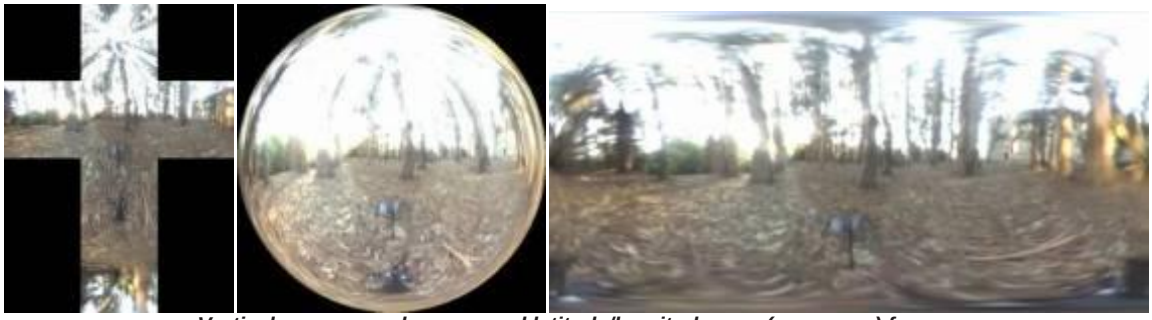
trueSpace supports environment maps created in the following ways:

- A '*cubic map'* created by reading six (possibly photographic) images.
- A '*cubic map'* created by reading a single image which consists of six sub-images laid out in the form of an unfolded cube (also known as a 'vertical cross' layout).
- A '*latitude/longitude map'* (sometimes known as a *panorama*) where the entire environment is flattened

into a single flat panoramic image, rather like a map of the globe.

- A '*spherical map'*, usually created by photographing a reflective sphere with a telephoto lens.
- An '*angular map'* (sometimes known as a 'light probe' image) is similar to a spherical map from a reflective sphere, except that the radial dimension is mapped linearly with the angle, instead of being squashed towards edge of the circle. This gives more accurate sampling around the edges of the image.

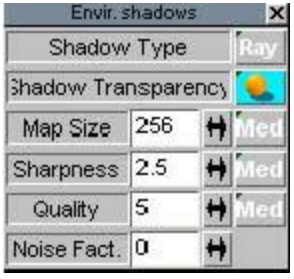
HDR images suitable for use as environments within trueSpace can often be found in several of these forms. For conversion from a spherical or cube form to vertical cross HDRI panoramas you can use the **HDR Shop** software available at http://www.debevec.org/HDRShop/.



Vertical cross, angular map, and latitude/longitude map (panorama) forms.

To activate HDRI as an environment you must first set a Global environment and then in the environment properties dialog select an HDRI bitmap. The global environment icon is found in the light group.

Global Environment



Global environment settings include:

- **Global Env:** Image for the global environment. This is designed to be used mainly with HDRI (High

Dynamic Range Image) images (.exr, .hdr), but it will also work with standard (low dynamic range) images (such as .jpg, .tif, .lwi, etc.).

- **Env. Type:** type of the image used for global environment. The following types are supported:
  - **Detect:** Let the rendering engine detect the type automatically. Note that the automatic detection works in many cases but not in all.
  - **Panorama:** panorama (latitude/longitude) image
  - **Probe:** light probe (angular) image
  - **Cross V:** cubical environment assembled in a cross shape - vertical
  - **Cross H:** cubical environment assembled in a cross shape - horizontal
  - **Cube:** cubical environment
  - **Sphere:** spherical environment
- **Light Intensity:** The Intensity value for the strength of the light from the HDRI. The range of this control is from 0 to 500. The default value is 2.0 – higher values will give brighter lighting, lower values will give darker lighting
- **Back. Intensity:** The Intensity value for the strength of the reflections from the HDRI, and the brightness of the background image if it is shown in the render. Higher values will give a brighter background image and stronger reflections, lower values will give a darker background image and weaker reflections.
- **Saturation:** Parameter for reducing (or increasing, if necessary) the degree of coloration of the scene caused by the environment lighting. The default value is 0.
- **Samples:** Provides an easy-to-understand and predictable trade-off between accuracy and speed. The default value is 10. The actual number of generated lights can be a bit smaller than requested by this parameter. A normal value for this parameter is about 100.
- **Sampling Angle:** Specifies the angle, in radians, over which the environment map is sampled for each pixel of background. This allows the environment map to be `blurred' slightly. By default the value is 0.0, meaning that the color of each pixel is established from a single point sample taken through the center of the pixel. Higher values will result in more blurring, which will be noticeable in reflections and refractions in the scene.
- **HDRI shadows:** Toggle for casting shadows from HDRI lights. Right click to open the shadows property dialog
- **Use only Envir. Lights:** If checked, then all lights in the scene are ignored except for the effect of the HDRI lighting. If unchecked, the effect of the HDRI lighting is combined with other lights in the scene.
- **Show Envir. Image:** When unchecked, an HDRI image is used only for the objects in the scene, and the background to the image is controlled through the regular background settings. Note that the HDRI background will still show in reflections and refractions. When checked, the HDRI image will be used as the background to the render, overriding the regular settings for background color or image.
- **Animate:** This is similar to the "Anim" checkbox in the texture dialog of the Material Editor:
  - If the name of the image used for HDRI contains a number, and there are similarly numbered images in the same texture folder, then these images are used during the animation - the HDRI image changes with the animation frame. If Animate is off, then only the loaded image is used.
  - If the image used for HDRI is an animation (AVI file), then individual frames of the AVI files will be used as HDRI images in individual frames of the animation. If the Animate checkbox is off, then only the first frame is used throughout the animation.

On the main Global environment panel, use Toggle shadow casting to enable or disable shadow casting for the current HDRI Image.

Global Environment Shadows:

- **Shadow Type (Ray/Map):** Determines whether hard-edged shadows (produced by ray casting) or soft-edged shadows (produced by shadow mapping) are created.
- **Shadow Transparency (Solid/Transparent/Advanced):** Determines whether the transparency of the object is taken into account when computing shadows. If this is set to solid, then all objects (including transparent objects) will cast solid shadows. Otherwise, shadow transparency depends on object transparency.
- **Transparent Shadows Advanced:** This is a switch for more advanced functionality, for those blockers whose transparency shaders require the initialization of more than the default set of shader globals. This is also for those blockers whose transparency shaders include effects due to displacement.
- **Map size:** Determines the resolution of the shadow map. The larger the map size, the more detailed the shadows, and therefore the longer the processing time.
- **Sharpness:** Determines how soft the boundaries of mapped shadows appear. A larger value creates sharper shadows (which can cause jagged shadow edges if the map size is small), while lower values create soft shadows (which can hide the negative effect of a low resolution map to some extent).
- **Quality:** Determines the quality of the shadow by controlling how much effort trueSpace will put into computing precise shadows. This parameter works in conjunction with the other shadow parameters, especially map size and sharpness, to determine the final quality of the shadows.
- **Noise Factor:** Allows you to exchange sampling artifacts for noise when using hard shadows. This parameter can have the effect of making shadow boundaries less visible. Its default value of 0.0 means that no jittering of visibility rays takes place. A value of 1.0 means that all visibility rays undergo jittering. Intermediate values result in some rays being jittered, and others not.

Examples of HDRI implementation in trueSpace

Without (left, using an IBL) and with an HDRI environment (right).

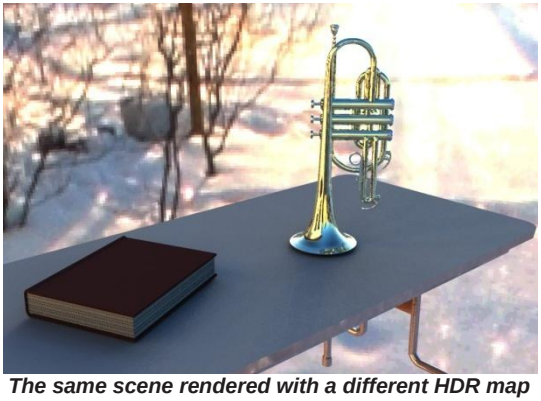
There are no lights in following scenes, just an HDRI environment. The images below show the effect of varying the Light Intensity parameter, which controls the strength of the diffuse lighting, and the Back. Intensity parameter, which controls the background image brightness and also the brightness and strength of the reflections.

Light Intensity 1.0

Light Intensity 2.0

Back. Intensity 0.5

Back Intensity 2.0



The same scene rendered with a different HDR map

HDRI data

HDRI data can be obtained from following sources:

- LightWorks HDRI Starter Collection (http://www.lightworks-user.com/hdri_starter_collection.htm): HDRI data from three of the best HDRI producers in the world, HDRImaps, Realtexture, and Sachform Technology. The image files contained within this collection are free for your use. You will find a sample of HDRI maps from Realtexture included as part of your trueSpace7 installation.
- HDRShop (http://www.debevec.org/HDRShop/): A Windows-only tool which is free for non-commercial use only.
- Photosphere (http://www.anyhere.com): A Macintosh image browsing and cataloging tool that can be used freely.
- hdrgen (http://www.anyhere.com): The HDR composition engine used within Photosphere is also available as a command-line tool for Linux as well as Macintosh.
- Photogenics HDR from Idruna Software (http://www.idruna.com): A commercial paint and photo editing package for Windows and Linux which supports high dynamic range images and their creation from multiple photographs.
- There are also some web sites offering an HDR creation service, such as WebHDR http://luminance.londonmet.ac.uk/webhdr/
- http://debevec.org/Probes/: Provides images in .hdr vertical cross form that can be used for testing purposes but which should not be published or distributed in any way without permission from the web site author.
- http://realtexture.com: Provides a large number of images in .hdr vertical cross form that are available for purchase.
- http://doschdesign.com: Provides a large number of images in .hdr vertical cross form (amongst others) that are available for purchase.

Usage hints and tips for HDRI lighting

1. **Render with default values of shader parameters.** By default, the "environment" light uses a small "number of samples" (10) and most parameters are turned off or do not affect the results (for example, "shadows" turned off, "saturation" set to 0). These values are a good starting point for adjustment and quick rendering. During the next few steps some parameters will be modified.

2. **Set the light intensity.** The "intensity" parameter determines the strength of the lights that are within the scene and how bright the rendered image will be. Typically you will want to match the brightness of lit objects with the brightness of the environment. Choosing a value that will match with lit objects in the background may require some experimentation.

3. **Set the saturation.** The environment light source shaders provide control over the degree of color saturation in the lighting. This allows you to reduce (or increase, if necessary) the degree of coloration of the scene caused by the environmental lighting. In the case of some HDRI environments, this effect may be more extreme than expected, so giving users the option to tone it down is sensible; however, it would in general be wrong to de-saturate too much, as this subtle color shift is precisely what makes the objects in the scene seem to match the color and lighting conditions in the HDRI environment. The value to be used for the "saturation" parameter therefore depends on the environment map being used and the effect that is desired.

4. **Set lighting quality.** The quality of lighting determines how good the lighting on objects will correspond to the environment. It depends mostly on the "number of samples", which is the number of lights used to simulate environment lighting. The more samples used, the more accurate the lighting. On the other hand,

increasing the number of samples increases rendering time. Some common artifacts that may suggest that you are using too few samples are:

- Objects appear to be lit from directions that do not correspond to bright areas in the environment map.
- Bright areas of the background do not seem to be influencing the lighting sufficiently.
- Lighting of objects changes rapidly.

The value for "number of samples" that will give good results can be different for each environment. If there are many small and bright areas, then more samples may be needed. If an environment is more or less uniform, then a smaller number of samples will suffice. Finding the correct value depends on the desired quality of lighting and on the environment. The best approach is usually to use a few tens of samples while experimenting to get the correct settings for the other parameters and then to increase the number of samples (possibly to as much as several hundred) for final rendering.

5. **Set shadows.** HDRI lighting is usually used to make objects look like they were really in the places represented by environment maps. Realistic shadows are therefore important. Increasing the number of samples introduces additional shadows (probably caused by weaker light sources) and makes stronger ones more accurate (by softening their boundaries, for example). The quality of shadows obtained with the same number of samples changes from one environment map to another. If there are many relatively small bright areas, then usually more samples are needed. On the other hand, large bright areas like windows are easier to simulate and in most cases require fewer samples. The number of samples needed to make the lighting of objects look acceptable may be not sufficient to produce nice shadows. If you see one of the following, the shadows are not accurate:
   - No shadows from bright areas in the environment
   - Shadows consist of visible layers

   If there are shadows missing, the only thing you can do is increase the number of samples. The second artifact is called "shadow banding," and there are two ways to reduce or eliminate it: either change the way shadows are calculated, or increase the number of samples.

   Environment light shaders support both hard and soft shadows. If hard shadows are used, noise may be applied to shadow boundaries by setting the "noise factor" parameter to a value greater than 0, which makes the layers of shadows less visible. This method does not increase rendering time and can reduce banding. The amount of noise needed depends on the scene and the environment. The disadvantage is that the noise can cause visible effects, especially when the shadows are being viewed from a short distance.

   If soft shadows are used, increasing the "shadow softness" parameter may reduce banding. Noise is not introduced, and rendering time is only slightly increased.

   In general, the best results can be obtained by using hard shadows with a high number of samples, but increasing the number of samples always makes shadows more accurate regardless of the shadow type. The obvious drawback is a longer rendering time.

   However, for some scenes it is difficult to eliminate banding even with many samples. This happens when an environment map contains many small bright areas. In such cases, a mixed approach may be used. Good results may be obtained by increasing the number of samples, and when banding is weak switching to soft shadows and smoothing them slightly. Similarly, some noise may be added when hard shadows are used.

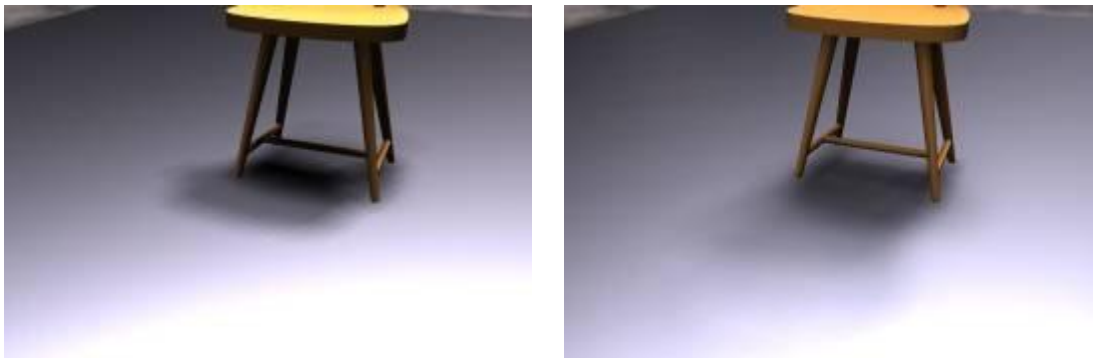
6. **Reduce or remove specular values in the reflectance shaders.** Specular highlights are calculated in rendered scenes in order to replace the reflections of bright lightsources that happen in the real world. With HDRI lighting, the environment is there to be reflected, including those lightsources, so it is advisable to reduce or even remove specular highlights from the reflectance shaders and rely on the reflections instead. For example, specular highlights are round when calculated in a reflectance shader, but the highlight caused by a window would in fact be square. Having bright specular highlights can obscure the more accurate reflection and reduce the realism of your scene. Using just a small amount of specular highlight processing in the reflectance shader can add a glow or bloom look to the result, though this normally does not need high values to achieve! Most of the examples shown in this manual used little or no specular values such as Shininess in their materials.

7. **HDRI does not mix with Volumetrics.** It is not possible to use HDRI lighting and volumetric foreground shaders at the same time. Trying to do so will result in no effect from the volumetrics, and a greatly reduced render speed, so be sure not to have both enabled at the same time.

Troubleshooting

Improper colors
If the environment lighting colors the image too much, use the "saturation" parameter to reduce the effect. The image on the left has a blue tint from blue in the environment image used. The results after setting "saturation" to −0.6 are visible on the right.

Shadow artifacts
If you get shadows that noticeably consist of layers ("shadow banding" artifact), use one of the solutions described earlier.



The image on the left shows shadows that are not smooth as a result of too few samples (10).
The image on the right shows the result of increasing the "number of samples" to 100.
In this scene it was possible to eliminate banding without needing to apply noise or to switch to soft shadows.

7.3 LightWorks Render Preferences

7.3.1 Render Toolbar

Two toolbars are available for setting options for and initiating the rendering process: the **Render Toolbar**, and the **Render Options Toolbar**.

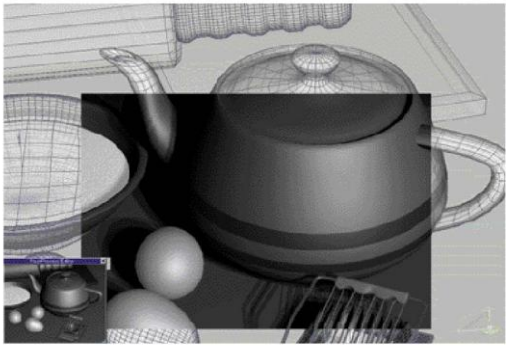
The **Render Options Toolbar** is the main toolbar for setting the various possible ways of rendering. With this toolbar you can change the way the image is rendered as well as set its quality and capabilities. The toolbar can be obtained by right-clicking any of the **Render** tools (**Render Current Object**, **Render Scene** etc.) on the **Render Toolbar**.

Area Render

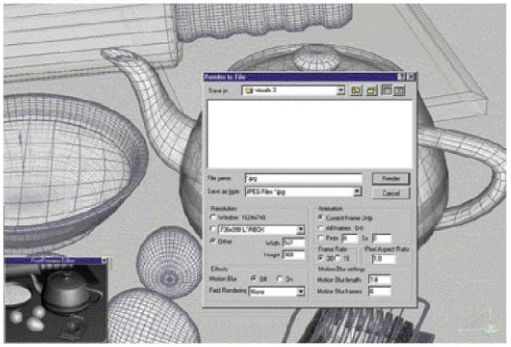
This button tells trueSpace to render the selected portion of the current viewport.

Here is a simple workflow for doing an **Area Render**.

1. Load a scene (.scn) in trueSpace.
2. Right-click the **Render Toolbar** to open the **Render Options toolbar**.
3. On the **Render Toolbar**, select the **Area Render** icon. Set the **Render Options toolbar** either to **Render Scene** or **Render to File**.
4. Click on the upper right corner of the section you want trueSpace to do an **Area Render** on, and then hold and drag the mouse to form the **Area Render** selection frame. This will be indicated by a gray rectangle.
5. trueSpace will then render that section of the scene. If you had the **Render to File** icon selected on the **Render Options toolbar**, trueSpace will then open the **Render to File** panel and will ask you for the image size, file format and filename etc.



Area Render (to screen)



Area Render (to file)

Area Render (to screen) ***Area Render (to file)***

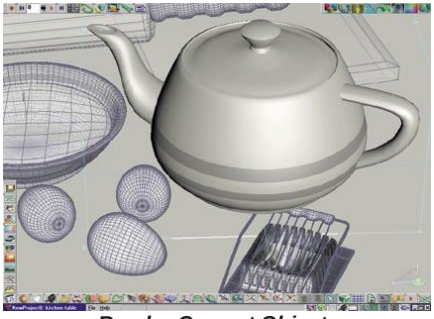
Tip: You can also render an area to file by CTRL+dragging the selection rectangle.

> **Note: Area Render** will only work with **Medium,** and **High** rendering modes. **Wireframe** or **Hidden Line** rendering modes will not work.

Render Current Object

This parameter tells trueSpace to render the selected object in the current viewport.

> **Note:** This icon will be grayed out if you have the **Render Options** toolbar set to **Render to File**. This is because trueSpace cannot do a rendering to file only of a single object. If you need to have a single object rendering, render the object with a plain background color to a file format that supports 32-bit depth, such as TARGA (.tga) or PNG (.png). These formats create an alpha channel that can be used in your image-editing program to create a selection around the object.



Render Current Object.

Render Scene

This button tells trueSpace to render the whole scene in the current viewport.

Note: If the **Render Options** toolbar is set so that **Render to File** is selected, clicking on **Render Scene** icon will open the **Render to File** panel. trueSpace will then ask you for the file name, file format and image size options to use as well as any needed animation related information if used.

Render Scene.

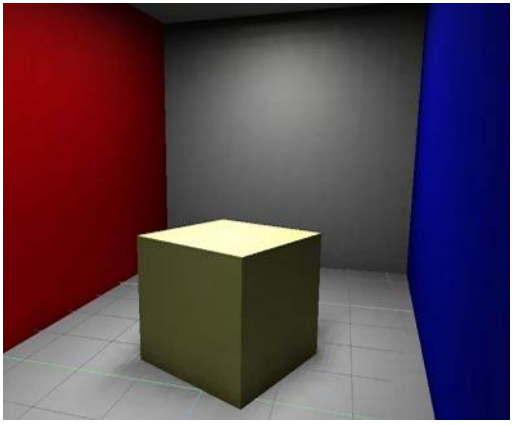
Render Scene to File

This button tells trueSpace to render the whole scene in the current viewport to file, just as if **Render to File** is enabled in the **Render Options** toolbar.

Texture Baking

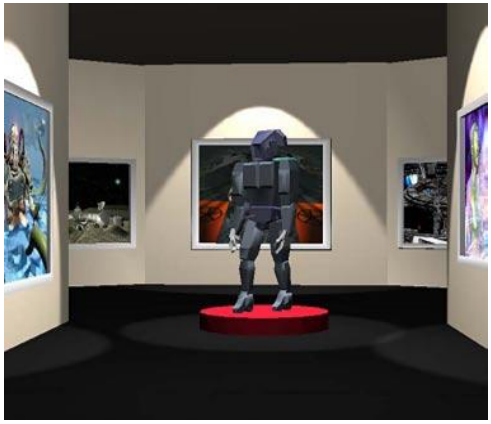
The texture from illumination (aka "texture baking") functionality allows you to convert extensive illumination effects (such as those found in radiosity solutions and/or renderings with the use of spot/projector light sources) into textures, which can be applied to the objects to reproduce the look in real-time rendering engines (including trueSpace itself). It is best used for objects that have relatively few polygons, like the floor and walls of a room; objects with high polygon counts require a large number of computations and may cause problems.

The following picture shows a radiosity solution for a simple scene converted into textures for few polygons. Note that in case of radiosity solutions, the subtle lighting effects (e.g. the illumination of the side walls) is represented by a fairly dense mesh - textures generated for the solution allow you to simplify the geometry to about 12 planar polygons while keeping a very similar appearance.



Texture computed from radiosity.

The next picture shows the real time screen grab for a non-radiosity case - note the effect of the spot lights on the walls of the gallery.



Texture computed from normal lighting.

Textures from illumination can be computed for selected objects using the **Compute textures from lighting** tool located in the rendering pop-up. This starts the process of texture generation for the currently selected objects The process can be started for radiosity data (the most frequent usage) or even for regular scenes. In the case of radiosity data the illumination stored in dense radiosity meshes is converted into textures, while in case of regular scenes with no radiosity the textures are generated based on the illumination from light sources (and materials applied to the objects).

Note: Once you use this tool to bake lighting textures onto an object, this essentially changes the scene so that recalculation of the radiosity solution is needed. To avoid the necessity of recalculating the radiosity solution each time, select all objects before executing the **Compute Textures from Lighting** tool.

Right-click the tool to bring up its properties panel:

- **Texel size:** The size of the texel relative to the size of the scene's bounding box. Lower values mean more detailed textures but longer processing time. A higher value means less detailed textures.
- **Run radiosity first:** If you want to include radiosity effects in the textures generated by the radiosity process, you can request the computation before the actual generation of the textures using this checkbox. Note that if a radiosity solution is available it will be used; otherwise, it is computed (taking into account the actual radiosity settings).
- **Squared textures:** Certain real-time rendering engines (e.g. DirectX) require textures to be squares as

opposed to rectangles (which may fit the geometry of the face better). Switching this setting on will force the generation of squared textures.

- **Power of 2 textures:** Similar to the previous setting, this switch causes all textures generated to have a width/height a power of 2, which is again a requirement for some of the real time rendering engines.
- **Prefix:** String which will be used as a prefix for the name of the textures generated. The textures are generated in the texture directory with prefix + the number, so this parameter should be used to distinguish between textures for different scenes.

You can easily switch between texture from illumination and standard (solid, wire, transparent, etc.) representation of the object using the "draw modes" of the object properties toolbar. Note that the resolution of the textures used by real time rendering modes of trueSpace is limited - you can set it manually in the display options panel. Thus if the textures do not look detailed enough in trueSpace, you may want to increase the setting there.

Generated textures can be deleted manually (if no longer needed) using the button in the object properties toolbar. Click and hold the **Draw object with textures from illumination** in the drawing modes toolbar to reveal the **Remove textures from illumination from the object** button.

It is possible to export the "texture from illumination" version of the scene into either regular trueSpace scene file (scn) or to Web formats (Viewpoint, Shockwave3D).

7.3.2 Output Options Group

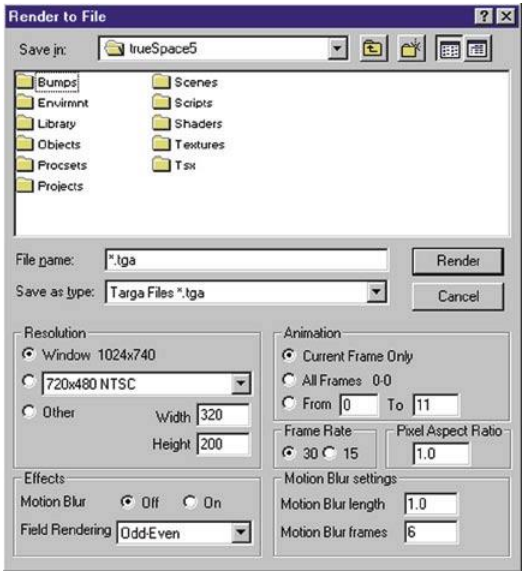
This is where the final output of the rendering is designated. It can be set as **Render to Screen**, or **Render to File**.

Render to Screen

This is the default rendering option. When a render command is issued, trueSpace will render to the screen.

Render to File

The **Render To File** button in combination with the **Render Scene** button will open the **Render to File** panel. This panel is used to render the loaded scene to a single image file or animation (.avi) in the image format of your choice. You can change the file type to be used by clicking on **Save as Type** and selecting a new image file format.



Render to file options panel.

The settings for the **Render to File** panel are as follows:

File Name: This parameter is for entering the name of the file you want to render with the appropriate file extension (.tga, .bmp, etc.) for both stills and animation files. The supported animation files are AVI and FLC.

Save as Type: The supported file formats for still images are Photoshop (PSD), Targa (.tga), Windows Bitmap (.bmp, .dib), JPEG (.jpg), TIFF (.tif), PNG (.png) and DDS (.dds). The supported output formats for animation are Video for Windows (.avi) and Flic (.flc).

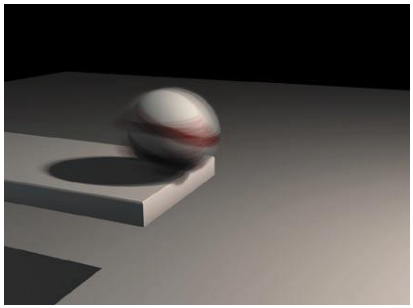
Resolution: This sets the resolution of the image file. This can either be the size of the view you are rendering, one of the preset resolutions or a custom size resolution.

- **Window:** This renders the image file to the same resolution as the window you are rendering.
- **Presets**
  - **720 x 480 NTSC, 720 x 488 NTSC**
  - **720 x 567 PAL, 768 x 567 PAL**
  - **20 x 200, 320 x 240, 640 x 480**
  - **352 x 240 NTSC MPEG1, 352 x 288 PAL MPEG1, 384 x 288 PAL MPEG1**
  - **704 x 480 NTSC MPEG2, 704 x 576 PAL MPEG2**
  - **736 x 398 LTRBOX**
- **Other (custom resolution):** This bullet choice has two parameters, **Width** and **Height**. This setting is normally used for very high resolution renderings. The maximum resolution that trueSpace can handle is 8,192 x 8,192 pixels. trueSpace will allow you to enter up to 6 digits but will revert to 8192 x 8192 resolution once it starts rendering and will not render to anything higher.

Effects

- **Motion Blur (On / Off):** This controls whether the frame to be rendered will have motion blur. If this parameter is turned ON, the whole visible scene will be blurred. This includes the shadows, transparent

objects, reflections etc.



Motion Blur.

- **Field Rendering:** When enabled, field rendering creates the alternate fields of a frame for television/video output. This means that trueSpace will generate two 'interlaced' fields for each frame of animation, one for the odd scan line and the other for the even scan lines. This makes the output animation smoother when played on broadcast television or video. The possible options for this parameter are **None**, **Even-Odd** and **Odd-Even**.

Motion Blur Settings

- **Motion Blur length:** This parameter controls the displayed animation time that will be used in a single frame. A value of 1.0 means that the interval between each frame will be blurred while a value of 2.0 means that two frames will be blurred in between etc. If you set the **Motion Blur length** to equal to the number of frames in the animation, the whole animation sequence will be blurred into one frame.
- **Motion Blur frames:** This parameter controls the quality of the motion blur, which is used together with the Motion Blur length parameter. This value controls the number of frames rendered and combined to produce a single motion blurred image. Having a short Motion Blur length will require a short Motion Blur frames setting as well and this goes the same for long Motion Blur length settings. However a frame value of one (1.0) is equivalent to having no motion blur.

Note: In animations where the whole frame is in motion together with the objects in the scene, it is advisable to turn off anti-aliasing and use the **Motion Blur** parameters to generate a pseudo-'anti-aliasing' by blurring the jagged edges using motion blur.

Animation

- **Current Frame Only:** When selected, trueSpace will only render a single frame/image. This will be the first frame by default.
- **All Frames 0-0:** Clicking and selecting this parameter will result in a rendered animation that includes all the frames in the scene as set in the **Scene Editor** or in the **Animation Control Panel**.
- **From X to X:** This parameter is for setting and limiting the animation to be rendered to just a select few frames as indicated by the beginning and end frame.

Frame Rate: trueSpace uses the 30 frames per second rate internally but this can be modified. A setting of 15-fps results in trueSpace not rendering (skipping) frames to get the correct frame rate, with 15 fps this means that only every second frame is recorded to file. This parameter only has two options, 15 or 30.

Pixel Aspect Ratio: This parameter is used for rendered images that will be used in desktop publishing applications. A change in the shape of the rendered pixels is needed by some desktop publishing applications to show the image without distortion or wrong proportions. **Note:** The width and height parameters will not preserve or automatically adjust the relative alignment and shape of the rendered image if you do not enter the proper ratio or multiples of the width/height relationship. Improperly entered resolution numbers without the correct width/height ratio will result in a distorted final image either vertically or horizontally and will look stretched.

Tip: The most common way of using this parameter is to generate print quality renderings, which are normally at 300 dpi. For example, your original image was rendered at 1024 x 768 resolution and you want to render a 300-dpi image. Just multiply your original width and height with the printed resolution requirement then divide it by 72.

Width: 1024 x 300 = 307,200
307,200 / 72 = 4266
Height: 768 x 300 = 230,400
230,400 / 72 = 3200

So for the Width and Height parameter on the Other button, you will enter:

Width: 4266
Height: 3200

to get a 300 dpi rendering from it. Of course the final rendered image file will need to be down-sampled to actually generate the 300 dpi file in an image editing program.

Note: Rendering to file, when interrupted, results in a correct (even though incomplete) image file / animation file being stored to the hard drive. Thus the preliminary results of the interrupted rendering can be seen in image/animation viewers without problems.

7.3.3 Quality Settings

The quality settings of the **Render Options** toolbar determine how good and how quickly the image will be rendered. The quality setting allows you to render your scene in various stages with different capabilities so that you can preview your scene much more easily than waiting for a long rendering. Normally the quality setting in the render panel is used progressively together with the other options in the **Render Options** toolbar to get an idea of the state of the scene, the lighting, the tones and the relative positions of each object in the scene.

Wireframe

This is the most primitive of all the types of rendering. This shows the edges and vertices that make up the object/polygon in the scene. The wireframe render will show everything in the scene including the inside and back sides of all the polygon geometry.

Note: Wireframe mode is good for previewing rendering problems to find out if an artifact is from a shading problem or a geometry problem. **Wireframe** is also great for inspecting the radiosity mesh formation. This aids in determining if a radiosity artifact is caused by the wrong meshing resolution or by another problem.

Hidden Line

This is similar to wireframe but it does not render the back and inner areas of polygon geometry. It only renders the visible parts of the scene from a certain viewpoint. **Hidden Line** will render the view from the selected viewport (Top, Left, Right, etc) and visible surfaces will be derived from that perspective. **Hidden Line** is useful for determining what is visible from a particular point of view without the distraction of the wireframe rendering.

Note: The **Hidden Line** render is not available and is not possible when a radiosity solution has been performed in the scene. It will be grayed out and will not be selectable.

Medium

The **Medium** render quality will render all textures, material shading and shadows. In combination with **Raytrace**, it will also show transparency.

Note: The **Medium Quality** render will not show transparency if the rendering visibility is set to **Raycast** without **Raytrace**. Also, if **Raytrace** is turned off and the **Rendering Visibility** is set to **Scanline**, the shadows and transparency will not be rendered. **Scanline** and **Raytrace** together with the **Medium Quality** render however will show both shadows and transparency.

High

The **High** render quality setting is for final production quality renders. It will render textures, shading, shadows and transparency.

Note: The **High Quality** render setting will only render shadows if **Raycast** is turned on. Also note that while **High** in combination with **Raycast** will generate shadows, it will not create reflections or transparency if **Raytrace** is not turned on.

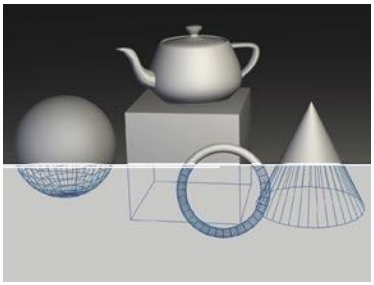
7.3.4 Rendering Visibility

The **Rendering Visibility** options are for setting the two different methods of scene generation. The two options here are **Scanline** and **Raycast**. The reason this section is called **Rendering Visibility** is because the two options differ significantly in the way they reconstruct and present the objects in the scene.

Scanline

Scanline is a fast and simple visibility-rendering algorithm. It analyzes the scene looking for the boundary (edges) of polygons and then figures out if the polygon is shaded or not. This is why it is called **Scanline**: it scans the scene line by line and sorts through the vertices and filled polygons it finds. **Scanline** rendering will not render shadows, reflections and transparency unless it is combined with the **Enable Raytracing** parameter. Also when combining it with the post-process editor's **Tone Mapping Brighten Up** option will result in a lighter tone-mapping distribution

when compared to **Raycast** rendering. Scanline rendering is advantageous in scenes that do not have transparency.



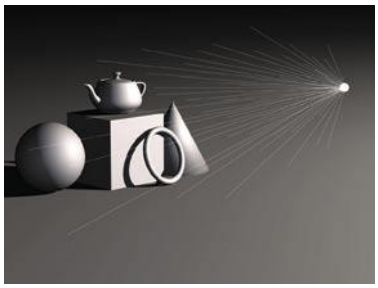
Scanline (representation).

Raycast

Raycast is the other visibility-rendering algorithm available in trueSpace. With raycasting, random rays are shot into the scene from a particular ideal point of view. These rays are then used to figure out the occlusion and depth of the objects in the scene from that selected view. This results in a more accurate scene representation than scanline, but also in a slower and longer rendering time. A good way of thinking about the raycasting process, is to imagine yourself as 'spraying light' randomly in the dark to investigate what is in the scene.

Raycast should not be confused with Raytracing, which is a more complete and global process of shooting random rays into the scene. Although raycasting uses rays to investigate the scene, those rays stop when they hit a polygon surface and thus are only used for determining the relative position of the objects in the scene and determining their shadows. Raytracing's random rays continue until they hit something straight on, graze something, or totally miss all the objects in the scene. These random rays are shot "backwards from the eye" from a virtual fine grid (mesh) and into the scene until they hit a surface. This is the so-called "visibility" part of the raytracing process, looking for occlusion and depth. Once it hits a polygon surface, it then asks, "What is visible around this point?" The type of material properties on the polygon surface determines if the ray should create a luminous (light emitting) object, reflective, or refractive surface. A new ray is then shot from this location if the material properties setting (shader) is reflective or refractive. New random rays are spawned whenever this new rays hit a surface with the right material properties. This process goes on until all the randomly shot rays no longer interact with anything in the scene. Those rays that do not interact with anything are shaded the same as the background color. This 'recursive' property of raytracing is what makes it a subset of Global Illumination.

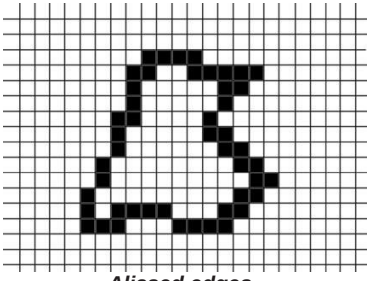
To get reflection, refraction and transparency with RayCast in trueSpace, Raytracing must be enabled. RayCast alone however will generate shadows and illumination information (light direction and intensity).



Raycast (representation).

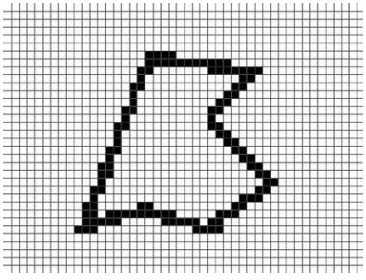
7.3.5 Anti-Aliasing

If you have ever looked at a printed picture closely on a magazine or zoomed in on a web photo you have seen "aliasing." The image looks as if it is seen through a coarse filter or a mosaic glass. It can be difficult to make out what is in the scene. You are then left to decide on what you really are seeing because you do not have any other information about what you are looking at. In computer graphics "aliasing" shows up in several ways, such as when a diagonal line is drawn in a stairstep pattern.



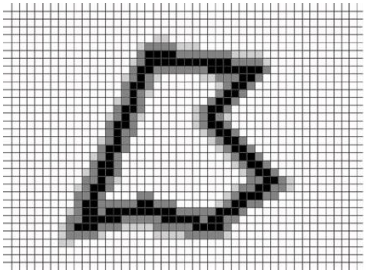
Aliased edges.

The solution is to either make those cells (pixels) smaller or to get rid of the high contrast edges created by the stairstep pattern. Changing the size of the cells (pixels) can be accomplished by changing the resolution, but it does not really solve the problem because it has complications of its own. By changing the resolution, the rendering time is increased and it does not totally solve the jagged edge problem unless the resolution is very high.



Making the cells smaller.

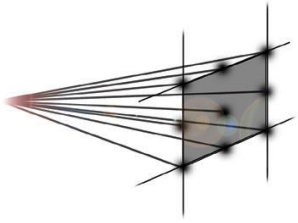
The solution is to create intermediate gray tones or shades between the black and white areas of the render.



Creating intermediate shades.

This is what anti-aliasing accomplishes in computer graphics. It gets rid of the high contrast jagged edges by creating an intermediate tone between two colors. This fools the eye into thinking that the line is not made up of broken cells in a stairstep pattern, but is actually composed of a line with smooth edges.

Analyzing each of the pixels to be rendered is anti-aliasing. This is done either by subdividing the pixel, inspecting each section and averaging it, or by inspecting the pixel. Inspecting and averaging the results is called supersampling.



Supersampling

Another way of doing anti-aliasing is to actually subdivide the pixels further only where there is a large difference in the contrast with the surrounding edges. This is called **Adaptive Anti-Aliasing**.

Adaptive Anti-Aliasing

Adaptive Anti-Aliasing is a task intensive procedure because it has to subdivide and analyze each of the pixel areas where there is a significant tonal change. However, adaptive Anti-Aliasing works well with boundary edges and is much faster than regular supersampling, especially when compared to 3X and 4X supersampling.

Anti-Alias Settings

The **Anti-Alias** setting in trueSpace determines the type and kind of anti-alaising performed on your scene.

Draft

The render output is done at half resolution. This means that it will do a very coarse render, which is good for checking the level of lighting and object position and the overall arrangement of the scene.

None

This renders the output at the default setting but without performing any anti-aliasing. This means that it will render at any resolution that you have set it, but will not remove the visible jagged edges in the rendered output.

2X AA

3X AA

4X AA

This parameter indicates the amount of supersampling performed on the rendered image. If you recall that supersampling is the inspection of the rendered pixels to judge what tone it will be, the 2X parameter uses 4 samples per pixel while 3X uses 9 samples per pixel, and 4X uses 16 samples per pixel.

Adaptive

This is adaptive anti-aliasing. This kind of supersampling analyzes the pixel and applies anti-aliasing only on the significant boundaries it finds.

Note: Adaptive anti-aliasing can only be used with the **High Quality** render settings. It is not functional with the **Medium Quality** render setting and will result in the image having artifacts.

7.3.6 Raytracing Options

This panel is where the various options associated with Raytracing are set.

Multi-threading

Multi-threading is the division of a computer job into two or more segments for processing by dual or multiple CPU's for faster and more efficient processing.

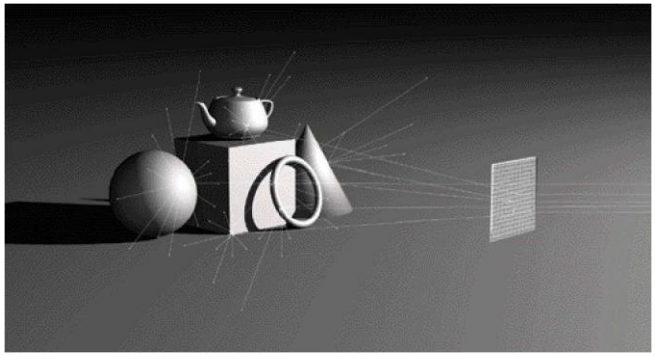
Single-Sided Rendering

Normally geometry in a scene is considered double-sided. In order to decrease rendering time, this parameter can be switched on and all polygons will be rendered as single-sided. If an object is a non-solid, however, some of its surfaces may have a missing polygon artifact. This is because some of these polygon faces might be oriented differently (face normals). Use this parameter with caution since it was designed for advanced rendering use only.

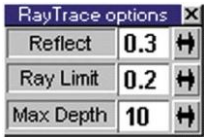
Triangulate

By default trueSpace triangulates all geometry prior to rendering to insure proper shading interpolation for concave surfaces. However, triangulating geometry also means that there is an increase in polygons, which not only results in increased data but also in increased rendering time as well. This parameter is in the OFF position by default but should be turned ON when there are concave surfaces in the scene and that are rendering with artifacts.

Enable Raytracing



This icon turns on **Raytracing** in the scene. As explained in the raycast section, raytracing uses multiple random rays that are traced backwards from the eye and spawn additional new rays when they hit a surface that has luminosity, reflection or refraction. **Raytracing** is what produces **reflection**, **refraction** and **transparency** in trueSpace renderings. Right-clicking the **Raytrace** icon will open the RayTrace options panel where you can modify the raytracing parameters. There are three options for this panel:



- **Reflect:** This parameter is for limiting the amount of reflections in the scene. It is related to the object's material properties *reflection* parameter. When an object's reflection is below the value of the Reflect setting, no reflections are rendered. This speeds up the rendering time as well as controls the amount of

reflections in the scene. This parameter can be animated with the key frame.

- **Ray Limit:** This parameter limits the spawning of new rays that will not contribute much to the appearance and quality of the scene. If this parameter is set to 1.0 there will be almost no raytracing. Setting this parameter to 0.0 will result in generating new random rays even if those rays will not contribute much to the rendering of the scene. Setting this parameter close to 1.0 results in trueSpace raytracing only the most significant and important reflections in the scene.
- **Max Depth:** This parameter sets the maximum number of rays that will be spawned and recursively traced from a particular pixel. This parameter's default of 10 will work for most scenes and should rarely be changed. Lowering the Max Depth setting results in simpler reflections and faster raytraced renderings. The maximum value for this parameter is 32.

7.3.7 Lightworks Post-Processing Settings

Note that almost all Post-processing shaders only work with the Lightworks render engine, and will give no results with Virtualight or with the optional V-Ray renderer. The exceptions to this are the Graduated Background shader which will work with Virtualight, and the Image Background shader and Color Background shader which will work with Virtualight and V-Ray.

Foreground Shaders

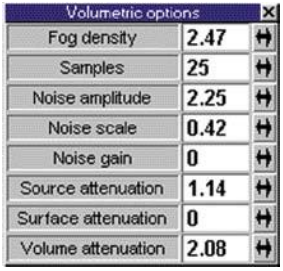
Foreground shaders enable you to add advanced special effects inside trueSpace itself without resorting to post-processing or compositing work.

None

This is the setting to use if you do not want any foreground shaders to be used.

Simple Volumetric

Volumetric lights simulate visible shafts of light due to scattering, which can add a new dimension to your scene. It generates realistic volumetric lights and shadows. Moving the lights, objects or camera in your scene will create animated volumetric lights. Selecting the **Foreground Effects Shader-Volumetric** on the **Render Option** toolbar enables volumetric lighting. Right-clicking on the **Volumetric** icon to opens the **Volumetric Options** Panel.



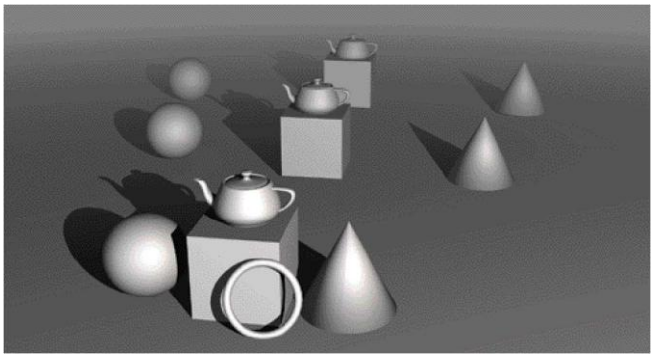
Note: In order for volumetric lights to show up, you also have to set the lights to have volumetric turned on In the **Light Options** panel. Only spotlights and projector lights work with **Simple Volumetric.**

Volumetric lights by nature require a lot of fine-tuning and tweaking to make them work since light intensity and scene situations may differ. The intensity, color, and falloff of your lights affect the volumetrics, and since volumetrics uses true sampling, it takes time to analyze the scene and render the results. The settings for the **Volumetric** options are as follows:

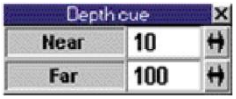
- **Fog Density:** This parameter controls the overall particle density of the volumetrics. It defaults to the value of 1.0. Increasing the Fog Density value results in a visible bright volumetric effect, while decreasing this value leads to more subtle volumetric effects. The Fog Density is affected by the dimensions/scale of your scene so this parameter will need to be adjusted for very large scenes.
- **Samples:** This parameter determines how many samples are performed per ray from each light source. A high sample value results in accurate shadow casting in the scene, but results in longer rendering time. Lower sample values render more quickly but may miss important fine detailed gaps between objects or not show proper occlusions in the scene.
- **Noise Amplitude:** This parameter adjusts for the presence of fractal noise particle effect or turbulence in the volumetric scattering. It simulates the swirling and clumping effect seen in smoke and dust particulate matter. Higher noise amplitude setting results in visible noise perturbations to the volumetric scattering. At the 0 setting, noise is not visible to the volumetric lighting in the scene, which results in quick render times.
- **Noise scale:** This parameter controls how large or small the perturbations appear in volumetric light. Small values result in fine granular clumps of noise, while larger values give subtle clusters of cloudy strands.
  Note: If **Noise Amplitude** is set to 0, this value will be ignored in the scene.
- **Noise gain:** This parameter controls the amount of visible noise in the scene, with values ranging from 0 to 1. This is the "noise visibility contrast" setting. High values result in clear and well-defined areas of noise, while low values result in a gradual fade. A value of 0.0 will turn the noise perturbation off. The default is a value is 0.5. Suggested useful values are between .2 and .8.
- **Source Attenuation:** This parameter controls the attenuation of the volumetrics from the light source. Attenuation is the gradual falloff of light from the source as it spreads out and covers more area. Larger values of Source Attenuation result in the volumetric intensity fading faster from the light source. If the volumetrics are intense for a particular scene, increasing this value will reduce the volumetric light intensity on the objects and limit it near the light sources. For better visibility of the subtle noise and scattering of the volumetric light, this parameter will need to be lowered.
- **Surface Attenuation:** This parameter controls the darkening of the surfaces of objects relative to the distance from the eye (viewport or camera view). This means that the light between the object's surface and the fog is attenuated (modified) by the surface of the fog. Value increases in Surface Attenuation result in far away objects appearing darker than the foreground objects within the volumetric influence. Beware that using a very high surface attenuation value will result in almost all the light reaching the eye from the object surfaces being absorbed, creating black areas. This is similar to having a very dense smoke where no light passes through. The default value is 0 (no attenuation).
- **Volume attenuation:** Volume attenuation controls the falloff of scattered light, as measured by the distance from the scattering point to the camera or eye view. In other words, adjusting this value controls how bright or how dim far-off scattering effects will appear. With higher volume attenuation values, beams of light will only be visible when located very close to the camera, and will get progressively dimmer the farther they get from the camera. The default value is 0 (no attenuation). Note: The exact amount of attenuation to use in different scenes depends mainly on scene size. You will want to experiment with the settings until you achieve the desired effect. Make subtle changes to these attenuation controls, starting

with low values such as 0.01 and slowly increasing to around 0.1 or 0.5.

Depth Cue

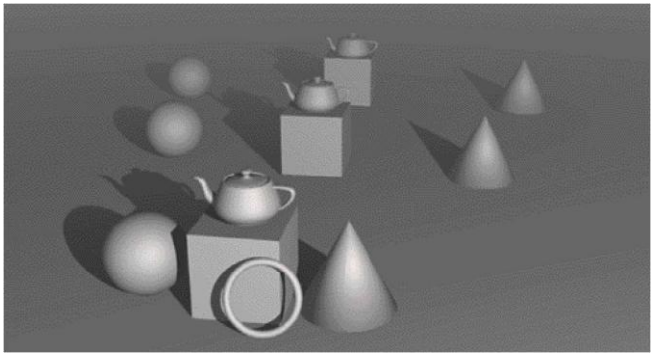


Depth cue shading is similar to the fog function. Surfaces that are far (deep) along the Z axis from the camera or eye view will fade to the background color, resulting in a gradual blending of foreground and background. Right-click on the **Depth Cue** button to bring up the **Depth Cue panel**:



- **Near:** This parameter is the starting point, in units, from the camera where depth cue shading begins. The lower the value, the sooner objects will be depth cue shaded. For example, setting a value of 10 means that all objects beyond 10 units from the camera start to receive this shading.
- **Far:** The parameter indicates the far distance that all objects are completely shaded by the background color. Any object outside this range will not be visible until the camera moves within the **Far** range.

Fog

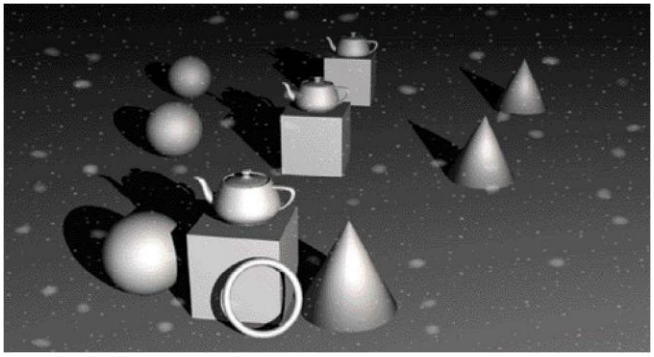


Fog allows you to add a "haze" to your scene that works much like depth cue: the farther away a surface is from the camera, the more it is washed out by fog. This heightens the realism of scenes that depict foggy or haze conditions. One important difference of the fog shader is that the fog color does not affect background shading, but only surfaces in the foreground. This makes it possible to use fog in conjunction with a background mask for compositing purposes.

Fog comes in two types: regular fog, which applies a consistent haze over the entire scene, and ground fog, which creates a haze that fades with height (as in the example below). You can open fog option parameters by right-clicking on the **Fog** button:

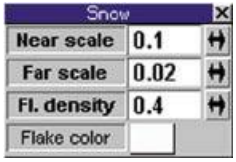
- **Fog Type:** The fog types are Regular and Ground. Ground fog is thickest at the ground level, gradually fading as it gains altitude. The level of fading vertically is controlled by the Height parameter.
- **Distance:** This parameter determines how far from the camera the fog reaches 100% opacity. High values will create a thin fog that spreads out for long distances, while a low value makes for very thick fog that obscures nearby objects.
- **Height:** The Height parameter is the vertical ceiling for the ground fog. As this value is increased, the vertical limit (altitude) of the fog also increases. Use lower values for a misty fog that is near the ground, or higher values for a fog that covers the scene more.
- **Fog Color:** The overall color of the fog is determined by this control. Clicking on the color block opens up the color picker, which is used to set the fog color. To change the base fog color, click on the desired portion of the color cube. To change the brightness, click and drag on the slider to the right of the cube. Note: Light sources will not become visible when using the Fog shader. If you wish to cast a beam of light from a light source, use one of the Volumetric foreground shaders instead.

Snow



The Snow shader creates the simulation of falling snow for still image use. The Snow parameter consists of two

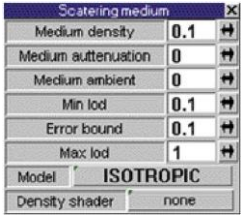
elements: nearby flakes, which appear larger and are sparsely scattered, and far flakes, which are smaller and more numerous. Animation of the Snow Shader is not possible. However the random way in which the actual snow changes in the Snow shader for every frame will produce a change in their relative positions.



- **Near Scale:** This parameter adjusts the size of the nearby flakes. Increase the value for larger nearby flakes, or lower the value to decrease and spread out the snowfall nearby.
- **Far Scale:** The Far Scale parameter changes the size of the far background flakes. As with the Near Scale, larger values will result in larger size flakes.
- **Fl(ake) Density:** The Flake Density parameter controls how many snowflakes are packed/shown in the scene when you render. Smaller values will simulate a light snowfall and larger values will result in heavy snowfall or storm-like conditions.
- **Flake Color:** Clicking on the Flake Color block brings up the color picker panel, which sets the color and intensity of the snowflakes.

Advanced Volumetrics

Advanced Volumetrics simulates the effect of participating media (smoke, mist, fog etc.) in the scene. This is a better alternative for most purposes than using simple Volumetric because it is not limited to spotlights or projector lights. You may recall that in the light transfer model, light interacts with intervening media (water vapor or particulate matter). The Advanced Volumetrics foreground shader simulates all the effects of light scattering. It processes visible light effects such as light falloff (attenuation) within the medium, light filtration though a medium that is colored, and directional 'first order' light scattering within the medium with volumetrics. First order scattering means the highly directional light scattering that is the primary light dispersion without secondary light bounces. It will not model light illumination caused by light scattering (indirect illumination by the media via scattering.) The **Advanced Volumetrics** foreground shader will work with all lights except the **Skylight**.



- **Medium density:** Determines how thick the intervening media is. This parameter for most purposes is independent of all the other Scattering parameters and is unaffected by them. However, it does affect the visibility of the other parameters.

- **Medium attenuation:** This parameter controls the amount of light absorption by the media. It affects all the light absorption in the scene except for the direct lighting on geometry surfaces. This parameter is also

affected by the color of the light shining on the media. Note: This parameter is very sensitive to your scene geometry's scale and size. Imagine modeling a dollhouse the scale of a stadium, and then even if you use the correct light intensities (kilolumens, etc.), a scene that size will most likely have minimally and realistic visible light scattering because of the wrong scene scale.

- **Medium ambient:** This parameter sets a uniformly dispersed scattering within the intervening media. As its name indicates, it is a kind of ambient illumination. It can be seen as introducing an even white light in the media. This parameter of course is affected by the color of lights that are in the scene. It is not affected nor does it depend on the Medium density parameter.

- **Medium color:** This parameter is the filtering and light scattering color parameter. This means that it controls the way the scattered light is shown. The greater the distance the light has traveled within the medium, the greater the light's color is attenuated and modified by the Medium color.

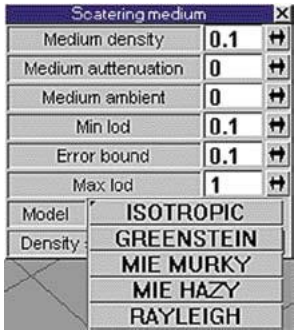
- **Medium shadows:** This parameter controls the shadowing of the medium, including 'self-shadowing'. This parameter is really a switch for turning on the medium's shadows. Note: Be warned that turning on Medium shadows results in long calculations, and thus a render time penalty.

- **Min lod:** This parameter determines the small amount of "details" in the visible scattering media. It controls the amount and quality of the small areas of visible scattered light. Setting this parameter to a low level results in a "noisy" scattering which is visible in the volumetric areas of the scattered media. This parameter is increased only in situations where the small details are invisible and where there is noise in areas of the medium that have rapid intensity changes. The default setting of 0.1 should be sufficient for most cases.

- **Error bound:** This parameter is for controlling the accuracy of the simulated scattered light. It determines the extent of variability and hence the visibility of the computed effect that is converted into visible pixels. The default value of 0.1 is adequate for most purposes. Setting this parameter to 0.0 will indicate to trueSpace to calculate the scattering up to the limit indicated in the Max lod parameter. Setting this parameter to 1.0 will disable it after the initial light distribution through the media. Useful values range from 0.0 to 0.5.

- **Max lod:** This parameter determines the maximum level of details in the scattering media. It really determines the extent of scene scattering calculation and investigation after the first order scattering has been performed and is controlled by the ***Error bound*** parameter. The default value of 1.0 is adequate for most purposes. Increasing this value will result in a visible change in the scattering simulation, but it is suggested that you decrease the Error bound level first before changing this parameter. This parameter also will control the upper limit of excessive and non-contributory scattering calculations.

- **Model:** This parameter controls what scattering model will be used in the scene. This is model is derived from real world light scattering physics models. The choice of model used affects the way the color (wavelength) is scattered in the media. The best way to remember how to use this parameter is this: light transfer has two properties, **isotropic** (which is a uniform scattering in all directions) and **anisotropic** (which is a highly directional and biased scattering). Anisotropic scattering depends on the position of the light source and the relative position of the viewer. This means that depending on where the viewer is looking, there will be a change in the visibility and quality of scattering. The **GREENSTEIN**, **MIE MURKY**, **MIE HAZY** and **RAYLEIGH** scattering models are all **anisotropic**.



- **Isotropic:** The **Isotropic** parameter is for using the uniformly distributed scattering model. This is the fastest scattering model of all, though it is not physically based and can produce unrealistic results. However, for most purposes this is the best scattering model to use.

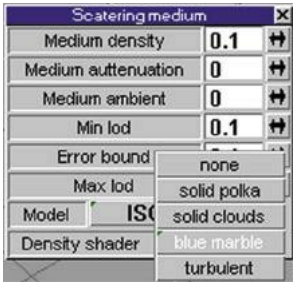
- **Greenstein:** This model is highly eccentric and is based on an analytical function model. The model's shape is a dual ellipsoid that is directional. It is a 'push-pull' scattering model where the light scattering can be directed (biased) at the opposite ends of the dual ellipsoid distribution. This is a so-called "forward and backward" scattering. It is based on Mie scattering and is a derivative of it, which is ideal for directional fog effects.

- **Mie Murky:** This model is mainly used for simulating a small particulate matter type of scattering, such as smoke. It is a highly directional scattering model where the scattered light is largest when the incoming light (incident light) is directed at the viewer. This model will have minimal visibility when the light direction is parallel to the viewer. The shape is like a thick elongated teardrop where the thin end is the origin of the light direction. Mie Murky predominantly models "forward scattering." This model is ideal for simulating thin fog effects with light sources.

- **Mie Hazy:** This model is similar to the **Mie Murky** model with one exception: it is less directional than the **Mie Murky** model. In short, the **Mie Hazy** model is thinner and more elongated than the **Mie Murky** model. This is ideal for simulating thick fog effects.

- **Rayleigh:** This model is used for simulating small molecules of water vapor and air. **Rayleigh** scattering affects the shorter wavelengths of light and scatters them more (greens and blues). This model is also highly directional, but is shaped like a peanut in its distribution pattern.

- **Density shader:** The Density shader introduces variability in the scattering media by modifying the Medium density according to a particular shader. Right-click any density shader option to set parameters for that shader. See **Artist Guide Appendix D: LightWorks Shaders** for full details on these shaders - they correspond to color shaders.



- **None:** This parameter turns off the density shader and indicates to trueSpace to not use any parameter that alters the way the ***Medium density*** of the scattering shader is rendered.
- **Solid polka:** This parameter uses the solid polka shader to modify the ***Medium density*** setting, which creates round empty spaces in the scattering model.
- **Solid clouds:** This parameter uses the solid clouds shader to modify the ***Medium density*** setting, which creates variable fractal wavy snakes in the scattering model.
- **Blue marble:** This parameter uses the blue marble shader to modify the ***Medium density*** setting, which creates stratified-layered curves in the scattering model.
- **Turbulent:** This parameter uses a chaotic function to modify the ***Medium density*** setting, which creates variable and random formations in the scattering model.

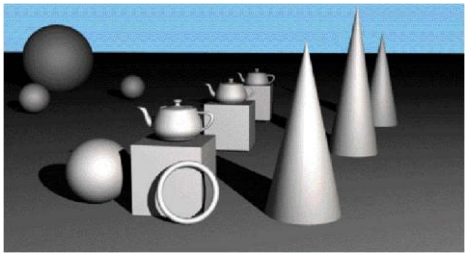
Note: With some shader settings, the scattering result will produce variable density shading, which results in longer rendering time due to calculations. If you do get highly random objectionable noise, the ***Min lod*** will need to be increased.

Background Shaders

The background replaces areas of the scene that are not covered by geometry with a background image or a procedural shader to simulate environments like sky, underwater, city backdrops etc.

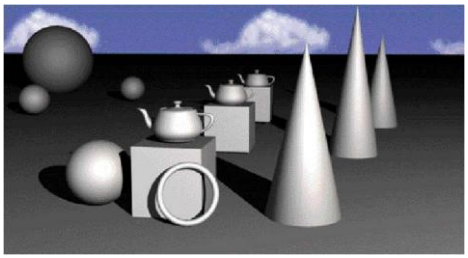
Color

The **Color** background shader allows you to choose a simple, solid color as your backdrop.

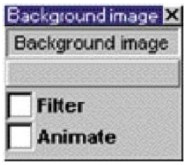


Right-click on the **Color** option to bring up the **Background Color** panel. To select your background color, click on the appropriate color area of the color picker (cube). To adjust the brightness of the color, use the slider to the right of the color cube.

Image

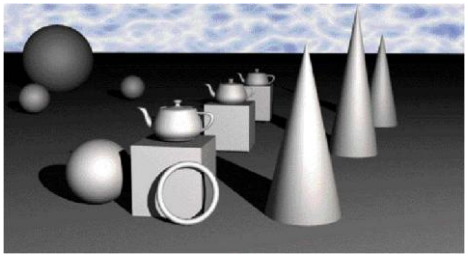


The **Image** setting is for using any supported image file (pictures, drawing etc.) as a background. This is especially useful for landscape renderings where you can use a picture of a sky as a background for your landscape scene or for creating star field effect in a space scene.



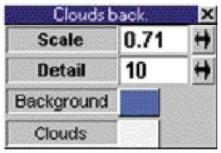
Right-clicking on the **Image** button brings up the **Background Image** file. To select a file as a background image, click on the button at the center of the panel to bring up the **Get Texture Map** dialog box. Use the browser to choose the appropriate image, then click Open to select. If the background image is an animation file, or a series of sequentially numbered still images (image 1.jpg, image2.jpg etc.), you can enable the ***Animation*** parameter to animate the background. When rendered, trueSpace will advance the animation one step per frame creating a changing animated background image in an animation file.

Clouds



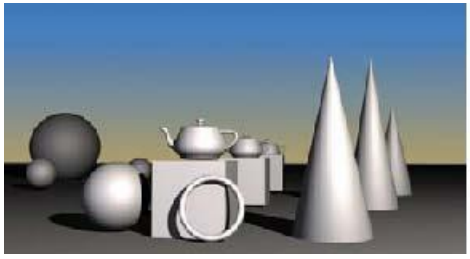
The **Clouds** shader offers the ability to select a two-color fractal cloud as the backdrop in your scene. With this shader, you can simulate different cloud conditions to create large, puffy clouds or veined, striated formations.

Use the following settings to adjust your cloudy sky:



- **Scale:** How large or how small the individual clouds in your background will appear. Larger scale values result in larger clouds that fill up more of your sky, while smaller settings result in more concentrated cloud formations. Values for ***Scale*** range from 0 to 10.
- **Detail:** The definition around the edges of cloud formations. Lower ***Detail*** settings result in smoother, rounder-looking clouds, while a higher ***Detail*** setting brings more wisps and curls around cloud borders. Values for ***Detail*** range from 0 to 10.
- **Background:** The background (sky) color of your scene.
- **Clouds:** The ***Clouds*** color control is used to determine the color of the clouds. For different variations on cloud formations, you might want to experiment by switching the background and cloud colors.

Graduated



The graduated, or gradient, shader allows you to choose two-tone (color) shading that blends from top to bottom to simulate sunsets, bright horizon effects with blue sky etc. Right-click on the Graduated button to open the Graduated options panel that shows and selects the two colors of the 'gradient'. As with other color options, click on either of the colored blocks to open the color picker for top and bottom colors.

Post Process Editor

The **Post-Process Editor** in trueSpace makes it easier to see effect of every parameter change that you make for

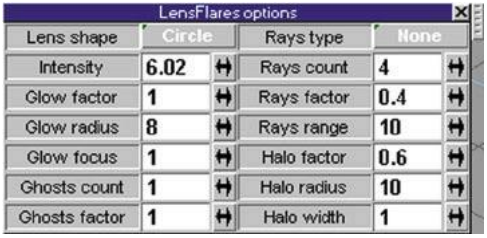
Glows and Lens Flare Effects, **Depth of Field** and **Tone Mapping** inside a real-time preview window.

Right-clicking any of the three icons of the shaders mentioned previously accesses the preview panel. The window is resizable.

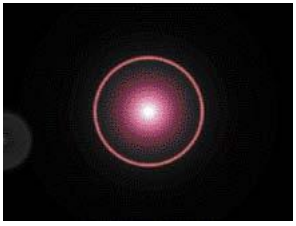
The preview window will refresh instantly for the **Depth of Field** and the **Glows and Lens Flare Effects**. It will not however refresh instantly for **Tone Mapping** because of the necessary internal calculations needed for every parameter change for this effect - it will be necessary to click inside the post process editor window to refresh it. Then you can adjust the tone mapping parameters and have the preview refresh.

Glows and Lens Flares

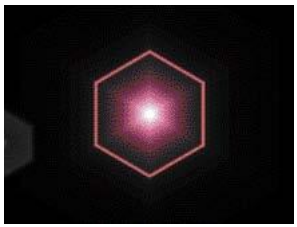
The **Glows and Lens Flare Effects** are for simulating the effects of internal light reflections in a camera when there is a bright light source in its field of view. This shader has several options. **Note**: This is a post-process rendering shader and will not create lens flares that are occluded by or are hidden by foreground geometry. This means that if an object is in front of the light in your rendering and that light source has a lens flare indicator turned on and there is an object in front of it, this shader will make the light visible and will not be covered by the geometry.



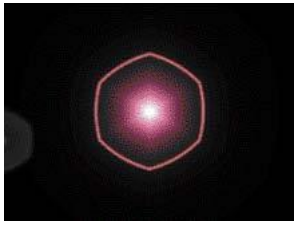
- **Lens Shape:** This parameter determines the form of the lens flare. This parameter has several options:
  - **Circle:** This creates round shaped halo.



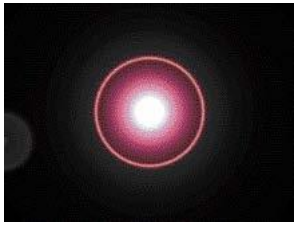
  - **Polygon:** This parameter creates angular edged shaped halo.



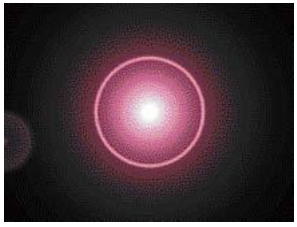
◦ **Arc:** This parameter creates round arc shaped halo.



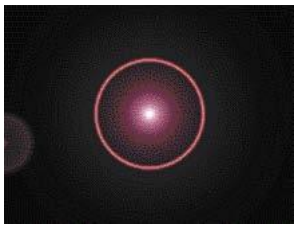
- **Intensity:** This parameter determines the level of brightness of all the lens flare parameters. The lower the value the lesser the visibility, and the larger the value the more brilliant and visible the lens flare will be.
- **Glow Factor:** This parameter determines the brightness of the glow that surrounds the central flare. Lower values result in a less intense glow while higher values yield bright and intense central hotspots.



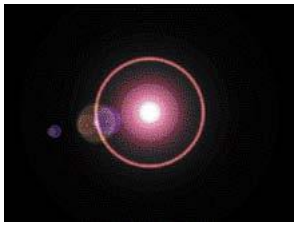
- **Glow Radius:** This parameter determines the size and extent of the glow of the lens flare. The lower values will create small glows while large values will increase the apparent diameter and extent of the glow.



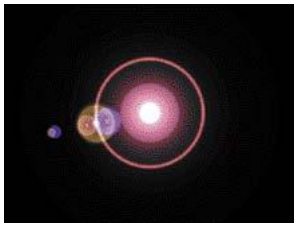
- **Glow Focus:** This parameter determines the extent and length of the lens flares' central hotspot. The low values will result in less intense central glows while larger values will brighten the glow effect.



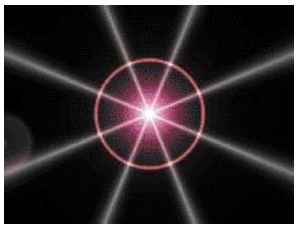
- **Ghost Count:** This parameter determines the number of lens element reflection (ghosts), which in the real camera result from the quality of the lens coating and the number of elements in the lens. In general, lenses that have long focal length have more lens elements than those with short focal lenses, but this is not always the case especially with zoom lenses. Keep this in mind when trying to simulate a lens flare. Decide if your camera's lens is far or near and if it is using a zoom lens or not, short or long focal length etc. The larger the value, the more ghosts that will be visible. Lower values will decrease the visible ghosts in the flare.



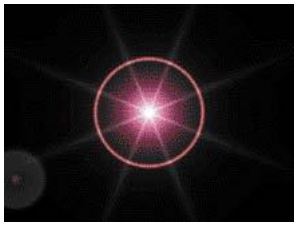
- **Ghosts Factor:** This parameter controls the intensity and visibility of the 'ghosts' effect in the lens flare. The lower value will make the ghosts less visible while the high value will make them stand out more intensely, which is unnatural for most instances. This parameter should only be increased if the camera is far from the light source and the 'ghosts' need to be made more visible.



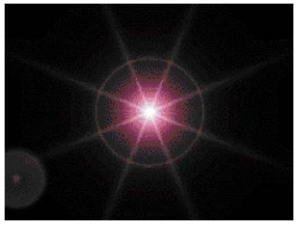
- **Rays Factor:** This parameter determines the intensity and brightness of the rays that emanate from the central hotspot area of the lens flare. A high value will make the rays more visible and intense while a low value will make them dim and subtle.



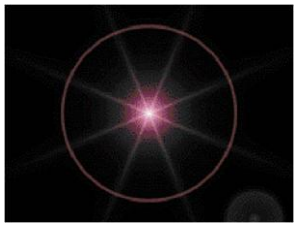
- **Rays Range:** This parameter determines how far the rays spread out from the central hotspot. This parameter in a way determines the 'attenuation' level of the rays. Lower values will make them have high attenuation resulting in the rays being shorter, while a high value will make them spread out farther from the light source.



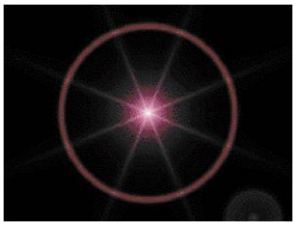
- **Halo Factor:** This parameter determines the brightness and intensity of the halo of the lens flare. Increasing the value of results in the halo being more noticeable and brighter, while lowering the value results in dimming of the halo. In general, halos should not be brighter than the central glow/hotspot.



- **Halo Radius:** This parameter determines the diameter and extent of the halo from the central glow/hotspot. High values will create a halo that is far from the central glow while lower values will produce a halo that is close to the central glow/hotspot.



- **Halo Width:** This parameter determines the 'thickness' of the halo. That means it controls the 'width' of the halo. High values result in the halo being thicker and more noticeable.



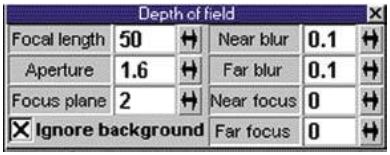
All these parameters can work collectively as well as individually depending on your setting in the **Glows and Lens Flare Effects** shader. Just remember that this shader should never attract attention to itself and should be used sparingly. You are simulating an artifact that real world photographers and cinematographer are trying hard to avoid. Use with discretion.

Depth of Field

This parameter simulates the effect of having varying degrees of sharpness within a frame or image due to the aperture opening of the lens used. Since there is no real lens used in trueSpace, the depth of field is simulated by a post process effect. The parameter is controlled in two ways, either through the description of the camera lens used (***Focal Length***, ***Aperture*** and ***Focal plane***) or directly (***Near blur***, ***Far blur***, ***Near focus***, ***Far focus***). Having two ways of setting the depth of field is to ensure that those who know photography have a familiar setting. This also makes it possible for somebody who does not know the photography approach to be able to set the **Depth of Field** through the near and.far parameters together with the near and far focus which is easier. Using the non-photography-based model of the **Depth of Field** makes it possible to create impossible but interesting effects.

> **Note:** You can only use either the photographic model (***Aperture*** and ***Focus plane***) or the simple model (***Near focus***, ***Far focus***). If you use the ***Aperture*** parameter, the ***Near focus/Far focus*** parameters are ignored and vice versa.

This panel has the following options:



Photographic Model (Camera):

- **Focal Length: (Values: 0.1 - 300):** This parameter determines both the photographic and simple camera lens' effect on the rendered scene. This means that it is taken into account by both processes either by the ***Aperture*** or the ***Near focus*** and ***Far focus*** parameters. It is measured in millimeters (mm). Smaller focal lengths (>50mm) in general will have more things in focus while longer focal lengths (<80mm-300mm) will register more objects in the scene as soft and fuzzy.
- **Aperture: (Values: 0 - 22):** This parameter directly corresponds to the extent of possible aperture (f/stops) for an ideal camera lens. In a typical 50mm real camera lens, the range is from f/1.4, f/2.8, f/4, f/5.6, f/8,

f/16 and f/22 and sometimes up to f/32 and f/45. The thing to remember is the larger the f/stop numbers (>f/5.6) the deeper the depth of field, which translates into images being sharper from the foreground to the background. Smaller f/stop numbers (<f/5.6) will translate into a narrower depth of field, which creates fuzzy and soft images in some or most areas of the image.

- **Focus plane: (Values: 0 - 500):** This parameter is the plane of "critical focus." This means it sets where the sharpest areas of the rendering will be. Smaller values mean that the focal plane is near the camera. That is, the critical focus is nearby while higher values mean that the focus is farther away.

Simple Model (Direct)

- **Near Focus: (Values: 0 - 500):** This parameter sets the foreground focus of the camera lens. The values range from 0-500.
- **Far focus: (Values: 0 - 500):** This parameter sets the background far focus of the camera lens. The values range from 0-500.

Additional independent parameters

The ***Near blur*** and ***Far blur*** parameters operate independently of the other parameters (**Photographic model** or **Simple model**) for additional creative camera purposes. These two parameters are set to zero (0) by default. When they are used they act as multipliers to the blurring effect as gauged from the camera location.

- **Near blur:** (-20.0, 20.0)
- **Far blur:** (-20.0, 20.0)

Note: When the camera is in "look at" mode and the **Depth of Field** post-process is used, then the distance from the selected camera to the scene object will be automatically set at the focal distance of the currently selected camera.

Depth of Field Workflow:

1. Set the **Tone mapping** first if your scene needs the proper tonal balance. This is to ensure that everything that should be visible is accounted for and is not too dark or too washed out.
2. Now, you have to decide if you want to use the **photographic model** or the **simple model**.

Photographic model:

a) Since the parameters for the photographic model are set at ***Focal Length***: 50, ***Aperture***: 1.6 and the ***Focal plane***: 2, for most scene this is wrong because it creates a blurred rendering.

b) The first thing you have to decide is the ***Focal Length*** used. ***Focal Length*** determines how 'narrow' or 'deep' the depth of field is in relation to the Aperture opening. This normally means the larger your ***Focal Length*** (>50mm) is, the narrower the depth of field is. That is, the far and near planes that determine the depth of field are closer to each other. Smaller focal lengths (<50mm) will yield 'deep' depth of field. The focal length of 50 mm is approximately what our eyes see.

c) Once you have decided on the ***Focal Length***, you have to set the ***Focal plane***. The ***Focal plane*** is where the 'critical focus' is, meaning the area where you want the camera to have sharp focus. Since this parameter is measured in meters, you must know the proper distance from the camera to the subject in meters to use this parameter well. You can use a box to find the distance by scaling the box to the distance between the Camera and the subject and looking at the **Object Info's** dimension.

d) The last step is to set and change the ***Aperture***. The default setting of 1.6 surely ensures that some if not all areas of the scene are blurred. The middle aperture opening in photography is between f/4.5-f/8.9. However, for most CG scenes these numbers will not show visible DOF effect. Start with f/2.8 and go lower if you want more DOF effect, but for focal lengths that are higher than 80mm, ***Aperture*** settings of f/2.8-4.5 will show DOF effects.

Simple model:

a) Set the ***Focal plane*** first to let trueSpace know where is the critical focus. Once you have established this parameter make a note of it because you will be setting the ***Near focus*** and ***Far focus*** based on it.
b) Now set the ***Near focus*** to a smaller number than the ***Focal plane***. This is to indicate to trueSpace that you want the ***Near focus*** to be in front of the ***Focal plane***.
c) Then set the ***Far focus*** to a number higher than the ***Focal plane***. This will ensure than the far focus is beyond the range of the ***Focal plane***.
d) Then adjust either the ***Near focus*** or the ***Far focus*** to establish your depth of field effect.

3. Now set the ***Near blur*** or ***Far blur*** if you want further blurring or softness either in the front blurred areas or in the back blurred areas. The higher the number the more blur will happen. These two parameters just establish the extent and degree of blurring that is happening in the background or foreground areas that already have been blurred by either the ***Aperture*** setting or the ***Near focus/Far focus*** settings. Please note that these two parameters, ***Near blur*** and ***Far blur*** are used independently of the other **Depth of Field** parameters and will not matter if you used the photographic model or the simple model.

Note: If you have a **Camera** in your scene and you have the **Look at Object** parameter set, the ***Focal plane*** will then update to the distance of that object. This means that the **Look at** parameter overrides the ***Focal plane*** distance parameter in the Depth of Field panel.

Tonemapping

Tonemapping is the technique of plotting or distributing the computed luminance values to fit the limited dynamic range of the display device. In essence, tonemapping distributes the tones in the scene so that it can better represent the available range in the scene without "washouts" or "darkness" dominating the scene. **Tonemapping** in trueSpace is accessed by right-clicking the **Tone Mapping** icon. You can expand the **Tone mapping** panel by clicking on the red triangle:

Brighten Up: This is the default tonemapping setting that plots and converts the calculated luminance value to RGB

color values using a logarithmic curve that is perceptually based. This means that it tries to simulate the way we see things in the natural world. This parameter has the following options:

- **Brightness: (Values 0.0- 1.0)** This parameter affects the overall intensity and darkness of the rendered image. Setting the value to zero (0) results in a completely dark image, while setting the value to 1 will result in a white washout image. The default is the mid value of 0.5
- **Balance: (Values 0.0- 1.0):** This parameter affects the balance between the whitest whites and the blackest blacks with the middle grays. Also it affects and determines the intermediate gray (middle gray) tones more than any other tonalities in the scene. You will notice that setting this to 1.0 will not always produce a completely dark image, but will leave out the intense and white areas of the scene. Also, setting this parameter to 0.0 will not produce a completely washed-out white image, but will leave some of the colored objects as well as the dark areas and objects intact. This is because it controls mainly the middle grays and the adjacent tones and affects the extremes (dark grays and tinted whites) less unless the value approaches the limits of the parameter value (0.0 or 1.0).
- **Auto-setup:** This on and off parameter (X) is for indicating to trueSpace that you want it to detect the maximum and minimum luminance levels that will be used for plotting the tone mapping distribution. If you are not satisfied with the tonal range that trueSpace selected, you can turn off ***Auto-setup*** and trueSpace will now let you select the tonal range. If you turn off ***Auto-setup*** you will be able to access two other parameters for the **Brighten Up** panel.
- **Max lum:** This parameter determines the highest level of luminance that will be used by trueSpace either from the radiosity solution or from the light intensity level as indicated in the lights/luminaires in the scene.
- **Min lum:** This parameter determines the lowest level of luminance that will be used by trueSpace either from the radiosity solution or from the light intensity level as indicated in the lights/luminaires in the scene.

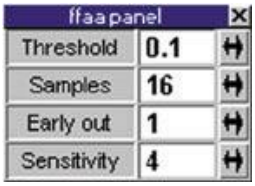
Note: On certain scenes the effect of changing the ***Min lum*** will not be as noticeable compared to the ***Max lum***. This is because the display device (video card/monitor) will not be able to accurately reflect the changes in the ***Min lum*** levels whereas it can easily show the changes with the ***Max lum*** levels because the changes there are more noticeable.

- **Detect:** This button is for telling trueSpace to use the currently computed changes to the **Tone mapping** preview as shown in the **Post-Process Editor** window in the currently selected viewports (Front, Left, Top, Perspective and Camera etc). Sometimes trueSpace will render the viewport differently than what is shown in the Post-Process Editor preview window. The **Detect** button is used to make trueSpace know that you want to use the currently computed information in the **Post-Process Editor** preview window for the viewport renderings as well as for final renderings.

Note: Tone Mapping, if used, will affect all parts of your scene and will function in all rendering modes including radiosity.

FFAA

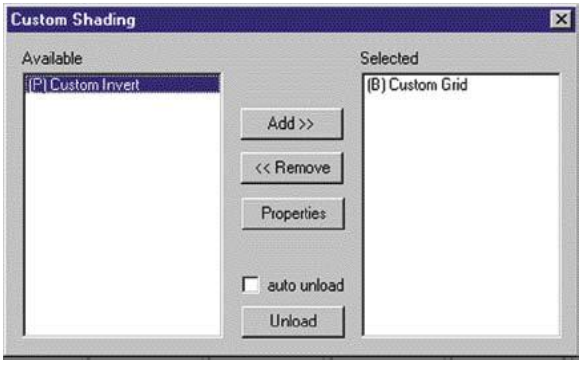
This parameter stands for **Feature-Following Anti-Aliasing**, which analyzes the features of several connected pixels in the rendered image by high-resolution sub-pixel sampling while ignoring insignificant pixel tone deviations, thereby accelerating the anti-aliasing process by performing it only on significant areas with visible features. This parameter has several options:



- **Threshold:** This parameter determines the range and limit of the FFAA process. Pixel tone differences that exceed this value are super-sampled, while other areas are ignored. This parameter should be set to a low enough value to ensure that all the pixels that contain significant features are indicated (tagged) as needing supersampling. It should not be set to so low of a value that it affects insignificant pixel tone deviations.
- **Samples:** This parameter controls the number of "ray samplings" that will be done at the corners of the pixel. The subsequent pixel tone deviations are calculated through the number of samples and compared to a perceptual table of the human eye. This means that it analyzes the pixel tone deviations then determines if that is something that can be seen or not. This is the way it decides if the connected pixel feature is significant or not. Setting this parameter to a very high value will result in additional CPU calculations without significant visible contributions to the scene.
- **Early out:** This parameter determines the number of samples used at the end of "feature analysis" and decides if the sampled area contains a significant feature or not.
- **Sensitivity:** This parameter works with the Threshold parameter to determine the level of detection of pixel tone deviation within the area being analyzed and whether it contains significant features that will need to be further super-sampled. This parameter should be set to a high enough value to make sure that all pixels that do contain significant features are tagged and detected. The higher the value, the more pixels are inspected.

Custom Global Shaders

This parameter includes the **(B)ackground**, **(F)oreground**, and **(P)ost-process** shaders. For the **Custom Global Shaders** to work, the desired shaders must be installed in the \Shaders\Global subdirectory of the main trueSpace directory or folder. This must be done prior to opening trueSpace. When these shaders are loaded, they are normally listed on the **Available** panel on the left side. You can **Add** or **Remove** the available shaders by clicking on the **Add** or **Remove** button. You can look at the available shaders by right-clicking on the **Custom Global Shaders** icon on the rendering toolbar. This will open the **Custom Global Shaders** panel.



To differentiate between the three different shader classes, **(B)ackground**, **(F)oreground** and **(P)ost-process**, the prefix of each of the classes is indicated before each shader name. The available shaders are listed in alphabetical order. To select the available shader to be used in the currently rendered scene, highlight it in the **Available** left column or double click the desired shader. This will move the selected shader to the Selected column.

To position a shader between two selected shaders, the position of the desired one in the list must be highlighted before the **Add** button is pressed. Shaders in the **Selected** list are listed in the order determined by the user with the exception that all **(B)ackground** shaders precede **(F)oreground** shaders, which in turn precede **(P)ostprocess** shaders. The CGS shaders are rendered last if the built-in foreground or background shaders are used.

They are processed in the following order.

1. Built-in Foreground and Background shaders (e.g. clouds, snow, fog, ...).
2. Custom foreground and background shaders. (CGS).
3. Built-in postprocess shaders (e.g. lens flares).
4. Custom post-process shaders. (CGS).

All Selected CGS shaders are loaded into memory. If the current trueSpace session is RAM and resource heavy, it is possible to remove in memory the currently loaded shader by selecting the desired **Custom Global Shader** and clicking on **Unload**. Checking the **Auto Unload** box will unload all unused shaders when a new shader (**Global** or **Material** shader) is used.

The currently used and loaded **Custom Global Shader** information in the current scene is saved with the scene file (.scn). When this scene is loaded back into trueSpace, trueSpace looks for and loads back the used **Custom Global Shaders** for that particular scene. When a used **Custom Global Shader** is not found, trueSpace will display a dialog box indicating that a particular **Custom Global Shader** is missing. If you want to make your own **Custom Global Shaders**, the software development kit (SDK) with documentation is available.

7.3.8 Multi-pass Rendering

You can render your scene into the popular **Adobe Photoshop** PSD format. At its simplest level, this can be used to produce a single-layer PSD file of the rendered scene. Yet the capabilities of this feature extend far beyond producing flat files. The output options presented at render time give you the ability to separate your render into **multiple layers** and/or **channels** based on features (such as reflections and shadows), lights, and layers. This provides you with great flexibility in compositing your images, or tweaking effects without re-rendering the scene.

For example, by outputting to a PSD using the “Per feature” option, you could strengthen the effect of reflections in the scene by manipulating or duplicating the “Reflection” layer, without needing to re-render the scene.

Once you select “*.psd” from the **Render to File** dialog and click the **Render** button, you are presented with the following options:

- **Layers:**
  - **Single:** The rendered output is stored as a single image layer in the PSD file. (It may contain multiple channels, though.)
  - **Per feature:** Each effect/rendering feature is stored as layer in the PSD file.
    - **Background:** Just a plain background
    - **Diffuse:** Diffuse lighting of the scene
    - **Specular:** Specular highlights in the scene
    - **Shadow:** Shadows in the scene
    - **Reflections:** Effect of reflected rays in the scene
    - **Refractions:** Effect of transmitted rays in the scene
    - **PostProcess:** Effect of postprocessing on the scene

      Individual layers can be turned on or off using the corresponding checkboxes. Please note that the effect of individual features is cumulative, meaning that a particular layer contains the effects of all the previous features. (For example, the shadow layer contains diffuse, specular and shadow effects.)
  - **Per light:** The effect of each light in the scene is stored as a layer in the PSD file. The number of layers will be the same as the number of lights in the scene (including implicit ambient light in the background layer).
  - **Per geometry layer:** Every geometry layer ("layer" in trueSpace) is rendered as a separate layer in the PSD file. The background layer consists of a plain background (no geometry), and all other layers contain a particular trueSpace layer with the appropriate layer blending mask.
- **Channels:**
  - **Alpha Mask:** Transparency mask
  - **Depth Mask:** Distance of the object from camera encoded in levels of grey
  - **Shadow Mask:** Levels of grey when shadow is present at particular point, black otherwise
  - **View angle mask:** Angle between view direction and objects's surface normal encoded in levels of grey
  - **Normal masks:** Length of the surface normal vector in world X,Y,Z direction encoded in levels of grey.

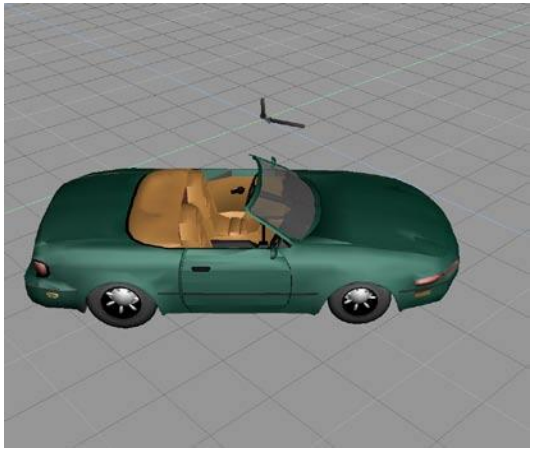
    Note: The effect of layer and channel rendering can be combined. When combined, the additional

channels are rendered into a composite image of the PSD file.

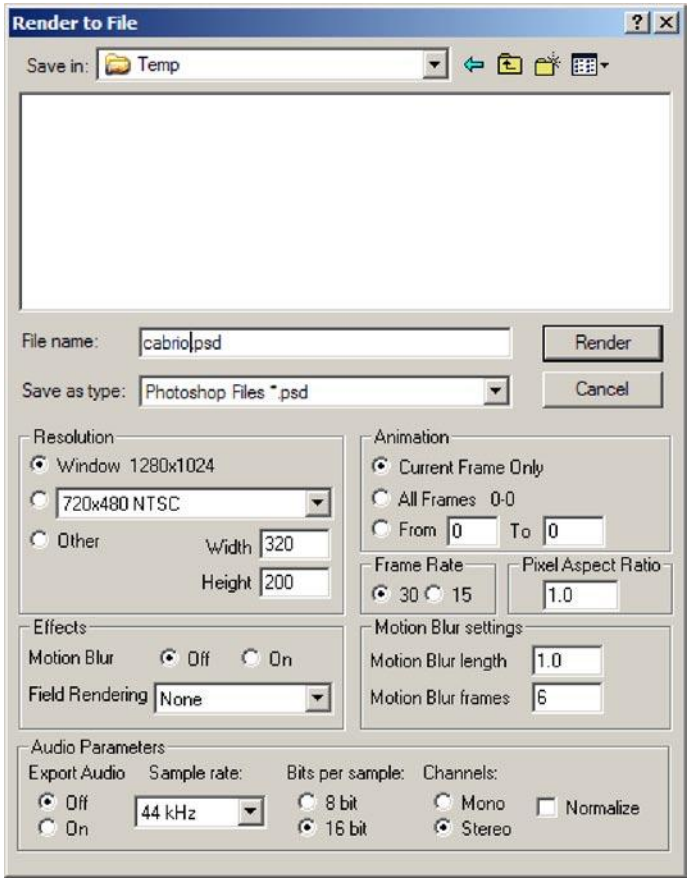
- **Channels as Layers:**
  - This option allows you to store additional rendering channel masks as individual layers in the PSD file. This is useful for applications which do not support PSD channel masks directly. The layers are set to be hidden so as not to interfere with the image itself.
- **Compatibility Mode:**
  - Per feature layers use the "linear light" blending mode, which is a new feature of Photoshop 7 that is not available in previous versions or some other applications. Compatibility mode uses the "hard light" blending mode instead.
  - Per light layers are integrated with Photoshop "linear dodge" blending. Because this is a new blending mode in Photoshop 7, previous Photoshop versions or other applications do not support it. In this case, enabling the "compatibility mode" checkbox causes the "screen" blending mode to be used instead. Individual layers are processed prior to being written to the file so that the "screen" blending of the layers in Photoshop gives the correct combined result. Please note that this approach is not perfect as "screen" blending means that the individual layer is not showing the effect of a single light. Instead, it contains the combined effect of individual light and lights stored in the layers below. Switching off a layer is not identical to switching off a particular light; the effects differ as the contribution of the layer is missing when layers above are being added to the composite image.
  - Per geometry layers are saved into the Photoshop file using clipping layers. This means that every geometry layer of trueSpace is stored as two layers in the Photoshop file: one layer with the actual geometry and the correct alpha mask for that geometry, and a second layer (below it) containing the same color information and the depth based mask. This mask is used for clipping to determine the visible portion of the layer.
  - It is possible to export geometry layers to a PSD file without clipping layers since certain applications (like PaintShopPro) do not support them. Check the "compatibility mode" box to exclude the clipping layers.

Tutorial: Exporting to a Multi-layered PSD

1. Load any scene containing multiple layers. The scene used in this example is "layer-cabrio.scn," which is included with your trueSpace install and is part of the "Layers" scene library.

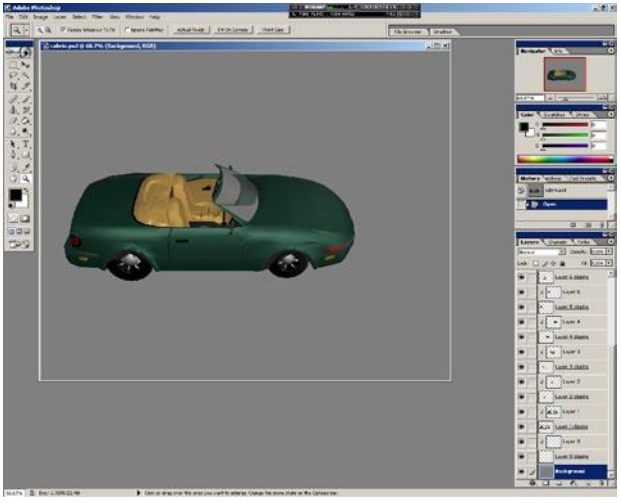


2. Render to File. Choose Photoshop PSD file format.

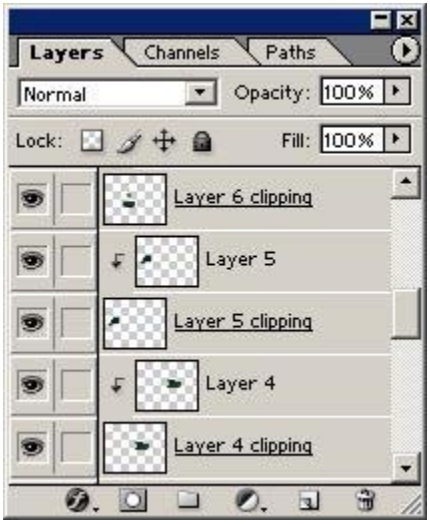


3. Choose “Per geometry layer” option for layer rendering.

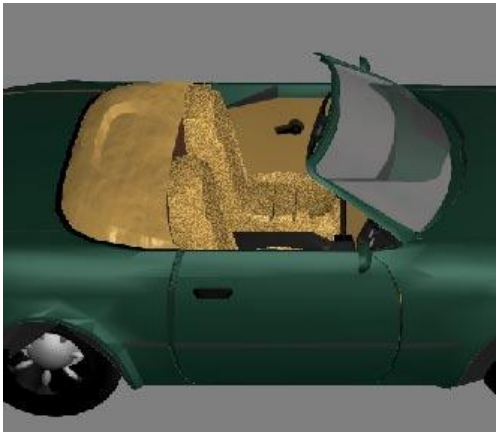
4. Load the file into Photoshop.



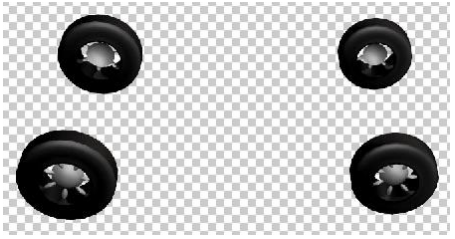
5. Try switching on/off individual layers to see the effect of the layers being combined into a resulting image.



6. Select “Layer 2” and apply any Photoshop filter (e.g. noise - monochromatic). Due to per layer masking, the effect is applied only to the specific objects on the layer.



7. Hide all layers except “Layer 8,” and select “Layer 8.” Choose Layer/Ungroup. The clipping layer is disconnected, resulting in the complete set of wheels (the complete trueSpace geometry layer) being made visible. To enable clipping again, select “Layer 8” and choose Layer/Group with Previous.

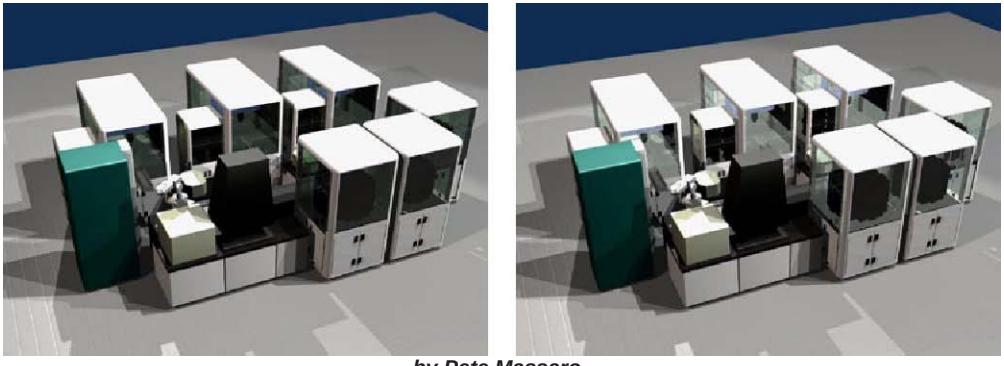


Here are some further examples of this feature in use:

"By rendering each feature layer I was able to adjust the image in Photoshop to taste. I particularly liked the way I could add a dreamy photo filter to the diffuse channel to 'soften up' what are perceived as cold looking CGI images and still retain the crispness of the shiny Goldfish bowl in the Family portrait shot."

by Paul Woodward.

"To bring up the interior light in the machines...I selected the refraction layer, increased the brightness and contrast.... and that was it! This would have taken an hour or more and not resulted in as good a result if I had to do the same thing from a single layer."



by Pete Massaro.

Limitations:

- Per geometry layers do not work very well with anti-aliased images; the blending cannot properly take into account the anti-aliased pixels at the objects' edges when overlapping objects exist on different layers. This is due to the fact that anti-aliasing in this case need to take into account objects on both layers simultaneously, which is not possible.
- Per geometry layer rendering does not take into account shadows and reflection/refractions of objects on

other layers.

7.4 Virtualight Render Preferences

In addition to the native **LightWorks** rendering engine, support for the **Virtualight** rendering engine has also been included in trueSpace as a fully integrated 3rd party rendering engine via the (slightly modified) ***virtuaOut plugin***. This gives the user access to high-end rendering features not supported by the **LightWorks** rendering engine, such as *Global Illumination* (*GI), photon* mapping, and *caustics*, while keeping the ease of use for rendering functionality.

Virtualight rendering engine

Virtualight (VL) is a rendering engine written by Stephane Marty. One of its main features is an extensive array of global illumination methods that can produce very high quality renderings. For full details, see the VL website at http://www.3dVirtualight.com/. The implementation of Virtualight as a trueSpace 3rd party rendering engine is based on the modified **VirtuaOut export** plugin by Simon Windmill. The major change though is that **virtuaOut** is now fully integrated into the trueSpace rendering workflow. Thus rather than existing as a trueSpace export plugin, it has become a trueSpace rendering extension.

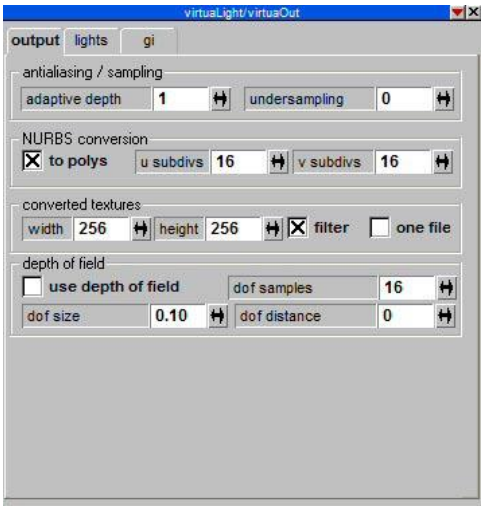
QuickStart Tutorial

1. Start trueSpace.
2. Open the render preferences panel by right-clicking one of the render buttons, like Render Scene.
3. Choose the ***Virtualight.tsr*** from the drop-down box.
4. Load your favorite scene.
5. Open the Material Editor (ME), change the objects' materials, and observe the effect of the material changes as they will appear in the final render.
6. Render the scene.

Options/Parameters

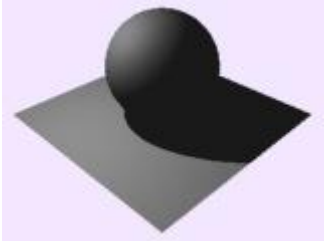
The **Virtualight** options panel can be opened by right-clicking on any of the trueSpace rendering buttons (*render scene, render scene to file, render object, render area*). This is consistent with the access to all of the other trueSpace rendering options (stored in the rendering tool), which can be also accessed by right-clicking on the render buttons.

The options panel consists of 3 tabs: *Output, Lights and GI*.

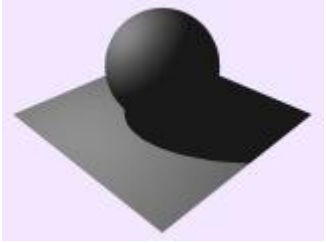


Output Parameters

- **Anti-aliasing**
  - *adaptive depth:* Allows you to specify how much adaptive anti-aliasing is done, as long as the anti-aliasing level is above 0. You can increase this value to get better anti-aliasing without having to go up to the next level of *anti-aliasing*.
  - *undersampling:* Does the opposite of anti-anti-aliasing. It allows you to generate a faster rendering by not rendering all points. A value of 0 switches this feature *off*, a value of 1 turns it *on* at the lowest quality/fastest rendering. Increasing the value increases the quality. This is intended to be used for previews only.

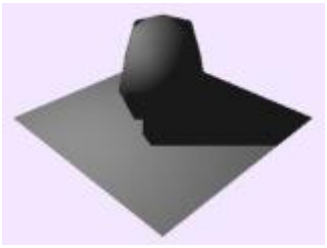


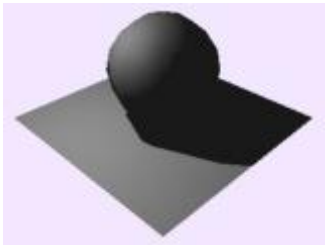
adaptive depth = 1

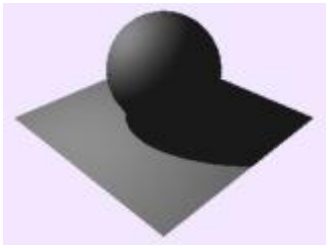


adaptive depth = 4

- **NURBs**
  - *to polys:* When enabled, converts all NURBS patches to polygons before exporting. This may be useful, for example, if you are seeing rendering issues with NURBS patches. It is usually best to keep it unchecked unless you have to *u subdivs* and *v subdivs* control the resolution of the VL NURBS patches. Increasing these values for higher quality output comes at the expense of file size and processing time.


u/v subdivs = 4

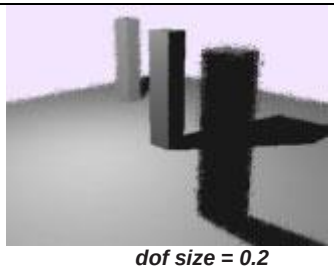

u/v subdivs = 8


u/v subdivs = 32

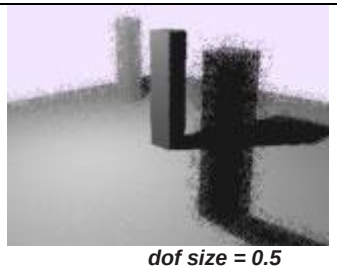
- **Converted textures (This section refers to the format of any images that are created from procedural color shaders used in the scene.)**
  - *Width* and *height:* Self-explanatory.
  - *filter:* Refers to whether or not there is smoothing of the texture when used in VL.
  - *one file:* Usually, a texture file will be created for every procedural color shader, for every frame in the animation, thus allowing you to convert animated textures. However, if you have no animated procedurals in the scene, it would be much faster and use less disk space if only one texture were created and used for every frame in the animation. Checking *one file* enables such behavior.
  - *export current texture:* Used to manually create a texture from the current material (whatever is currently in the trueSpace material panel). The *width* and *height* specified above will be used to generate the texture. You will be asked for a filename when you click the button.
- **Depth of field (VL renders DOF effects by using multiple sample rays when rendering.)**
  - *use depth of field:* Enables this feature.
  - *dof samples:* The number of samples to take for each pixel. Increasing this value will give you a smoother result but at an increased render time.
  - *dof size:* Controls the strength of the blur effect. If you increase this value, you will most likely need to increase the number of samples to avoid a dithered look.
  - *dof distance:* Specifies the focal point for the camera, i.e. the area that will be in focus. If you set this to 0, and the trueSpace camera has a look-at target, the object being looked at will be in focus. This behavior is similar to the way trueSpace's own depth of field works.


dof samples = 1

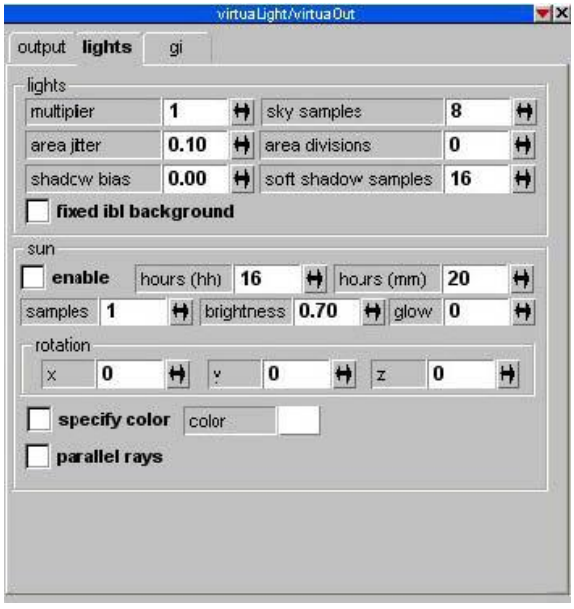

dof samples = 32



dof size = 0.2



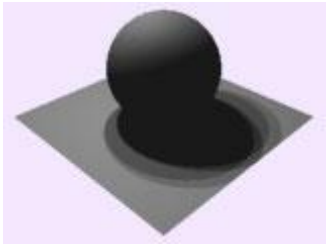
dof size = 0.5



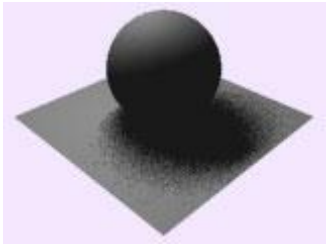
- **Lights**
  - *multiplier:* Allows you to increase or decrease the light levels in the VL render without having to change all of the lights in the scene. Setting it to 2 will make all lights twice as powerful; at 0.5 and they will be half as bright, and so on.
  - *area jitter:* Affects the shadows of area lights to help avoid banding. The side effect is a dithered look unless you increase the number of shadow samples.
  - *shadow bias:* Used to remove shadow artifacts at the border between light and dark on a curved surface. You might have seen an artifact in trueSpace where there is a faceted look at the edge of the shadow when viewing a shadow-casting light at an angle. With *shadow bias* set to 0, you may see the same thing in VL, but changing the value moves the shadow position slightly. You may wish to experiment to find the best result for a particular render, ideally it should be as close to 0 as possible without showing any defects. You should only need to change this setting infrequently.
  - *sky samples:* Controls the accuracy of the sky light calculations. A greater number should give a smoother, more accurate result but at an increase in render time.
  - *area divisions:* Controls the amount of internal subdivision when processing area lights. A greater number increases the accuracy of area light shadows at the expense of render time.
  - *soft shadow samples:* Affects several light types, including goniometric lights. A greater value will

give softer shadows with less banding effect, again at the expense of render time.

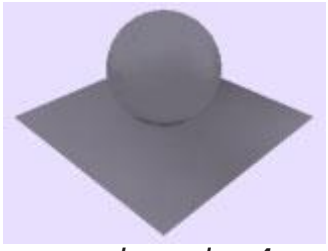
- *fixed ibl background:* Allows you to switch between two styles of rendering the image in an IBL light. When this box is *off*, the results are similar to the trueSpace method, i.e., the scene appears to be enclosed in a large sphere. This method, however, is not always desirable, as it proves hard to control what portion of the background image is displayed. When *fixed ibl background* is enabled, the background will be rendered as a flat 2D background, but the lighting, reflections and refractions will still be generated as if from a spherical environment.



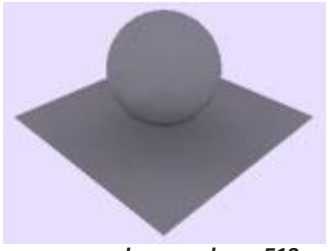
area jitter = 0



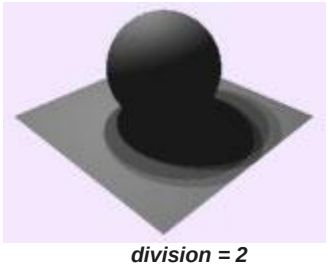
area jitter = 0.7



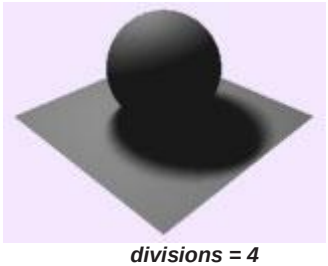
sky samples = 4



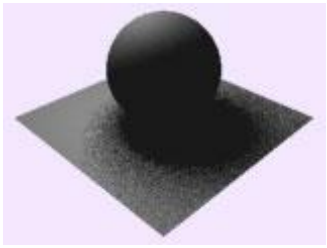
sky samples = 512



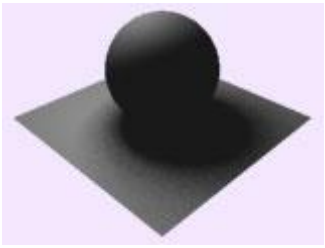
division = 2



divisions = 4



soft shadow samples = 0

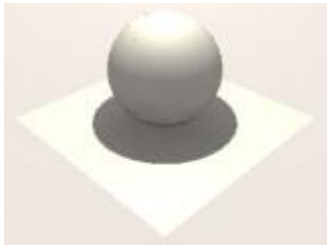


soft shadow samples = 128

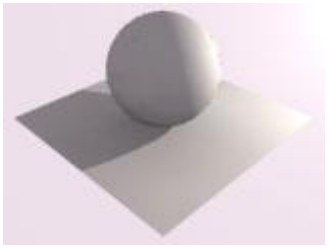
fixed IBL background = off

fixed IBL background = on

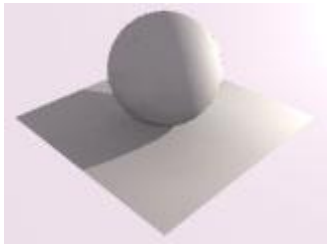
- **Sun** (**Note:** When you add a sky or IBL light to the scene, it adds illumination from the sky only. You won't see any sharp shadows cast; it is more of an ambient light producing subtle shading effects. This can be enough on its own, but sometimes you may want to add other lights. The real world has one large lightsource, of course: the sun.)
  - *enable:* Tells VL to calculate the effects of sunlight. The time of day can be specified using the 24 hour (or military) clock, and the position and color of the sun will be automatically set. For best results, set the sky light color and background color to a nice shade of blue, or whatever color the desired the sky should be. An image can be used if the scene has an IBL light instead of a sky light.
  - *brightness:* Increases or decreases the sunlight effect.
  - *glow:* Works together with the *brightness* setting to control the look of the sky. If *glow* is set to 0, the size of the sun's glow will be automatically calculated according to distance from the camera. This value can also be changed to affect the glow size relatively.
  - *samples:* Affects the softness of the sun's shadows. Ordinarily, the sun will cast sharp shadows. However, if softer shadows are desired, then increase the value.
  - *rotation:* Allows you to change the position of the sun in the sky to best fit the scene. Specify the number of degrees to rotate around each axis. For example, if you add sunlight in the *vl_ibl* example scene, the sun will come from the right. Change the Z rotation to 180 and it will come from the left.
  - *specify color:* Sets the sun's color. Ordinarily, the color of sunlight will be chosen automatically depending on time of day. However, if you wish to explicitly set the color, use the *color* setting. Note that you must have a sky or IBL light in the scene for sunlight to have any effect.



time: 12:00

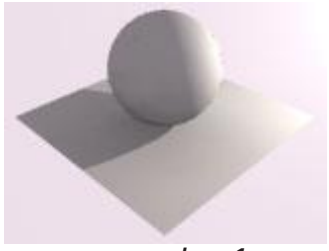


time: 16:20

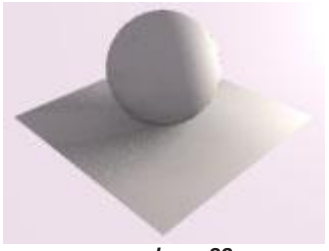


glow = 0.2

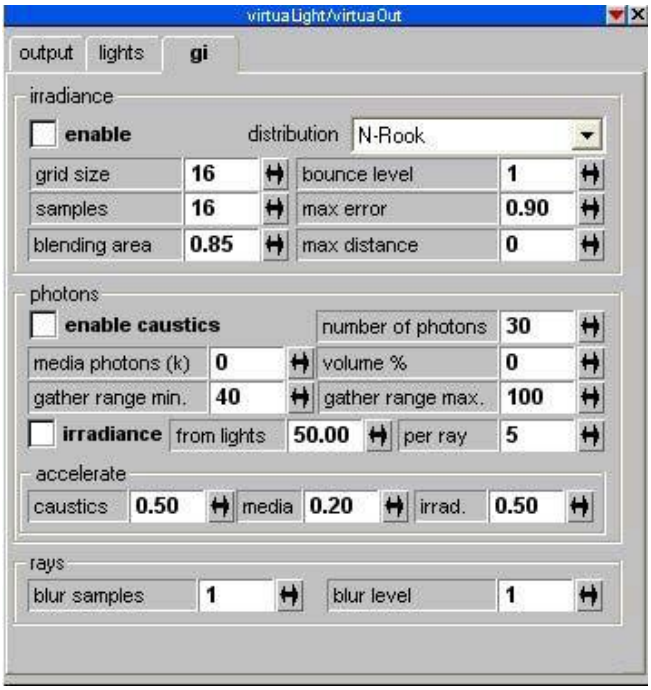
glow = 1.4



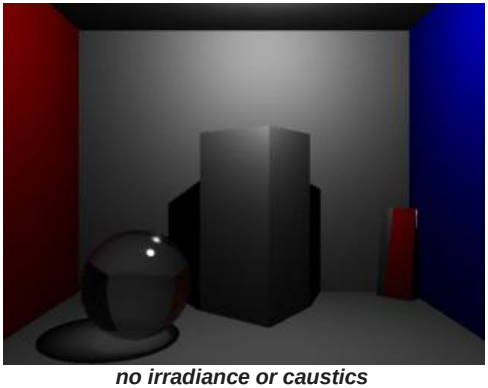
samples = 1



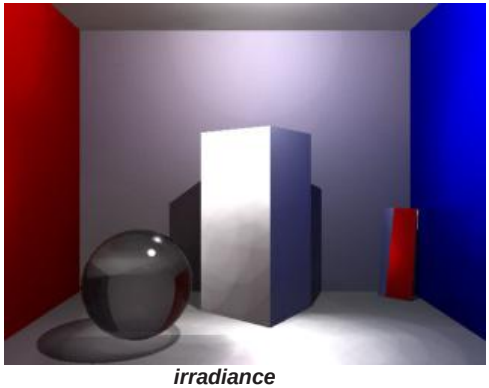
samples = 32



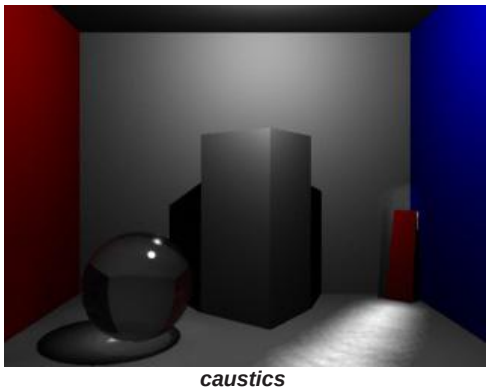
Virtualight has two main global illumination settings: *irradiance calculation*, and *caustic effects*. The irradiance calculations allow the light in the scene to creep into all corners, illuminating surfaces in a natural way. Caustics allow light to reflect off mirrors and refract through glass surfaces. It is strongly recommend that you read the Virtualight user guide for more information on what these settings actually mean. The *enable* checkboxes allow you to selectively choose irradiance and/or caustics. It is important to note that caustics can be generated without having to compute an irradiance solution.



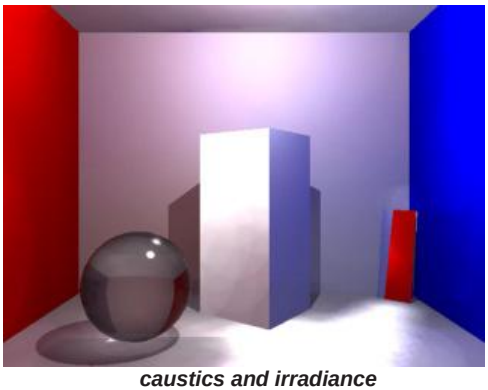
no irradiance or caustics



irradiance

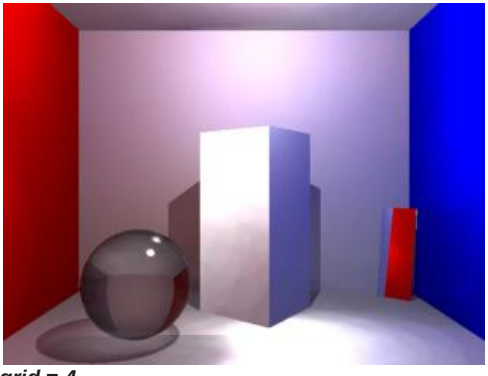


caustics

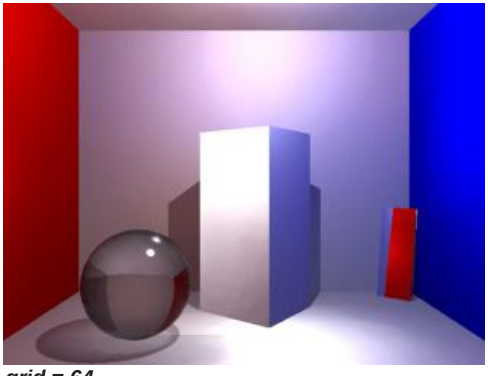


caustics and irradiance

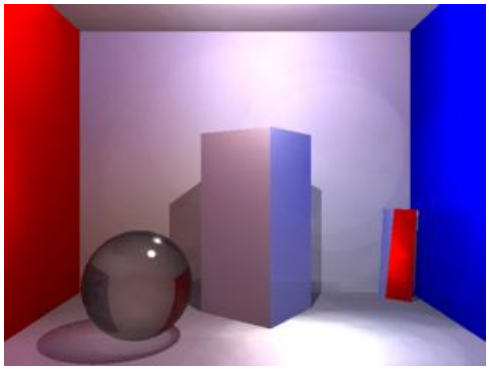
- **Irradiance**
  - *distribution:* Specifies how irradiance samples are taken. A full explanation with illustrations is given in the VL manual.
  - *grid size:* Specifies the initial accuracy of the irradiance calculation. A smaller value gives a more accurate result, but takes longer to calculate.
  - *samples:* Specifies the total number of irradiance samples to take for each point being evaluated. A larger value gives smoother results, with the usual speed tradeoff.
  - *bounce level:* Specifies the number of times a ray of light should be followed as it moves around striking different surfaces. Increasing this value will give a more accurate calculation, again at the expense of render time. If light is not reaching inside areas properly, try increasing this value.
  - *max error:* Controls how accurate the sampling should be. Smaller values are less accurate, but result in faster rendering.
  - *blending area:* Determines how well the irradiance samples are blended together to hide visible patches of irregularity. A smaller value will result in less blending; values over 0.5 are recommended.
  - *max distance:* This is another control for fine-tuning renderings and is related to the *max error* setting. Smaller values will result in more accuracy (and longer render times) if enough samples are used. A setting of 0 will allow VL to perform the calculations automatically.



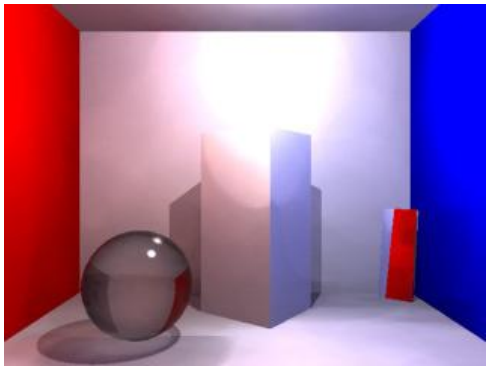
grid = 4



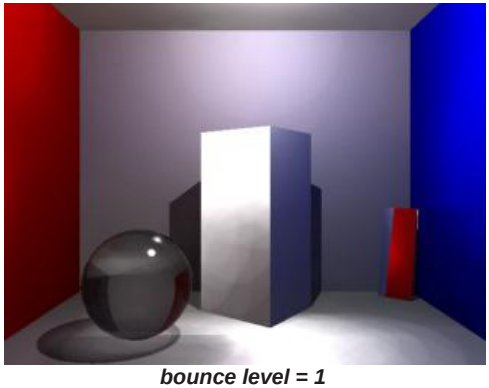
grid = 64



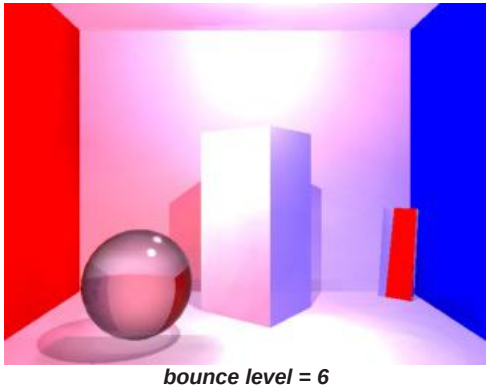
samples = 1



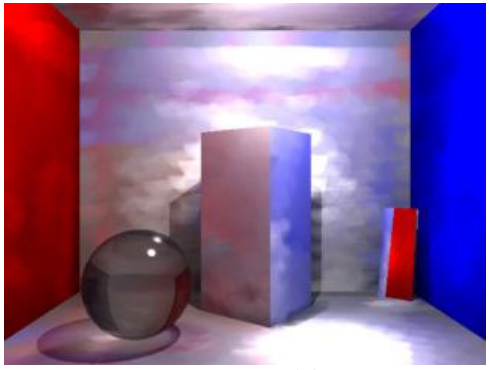
samples = 128



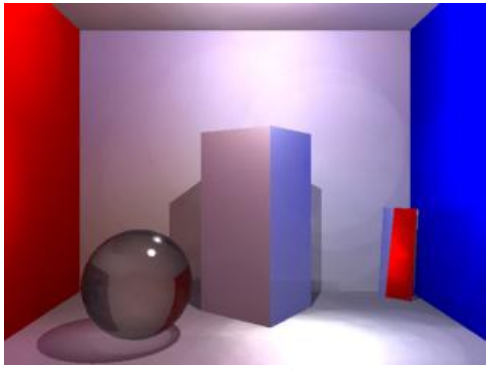
bounce level = 1



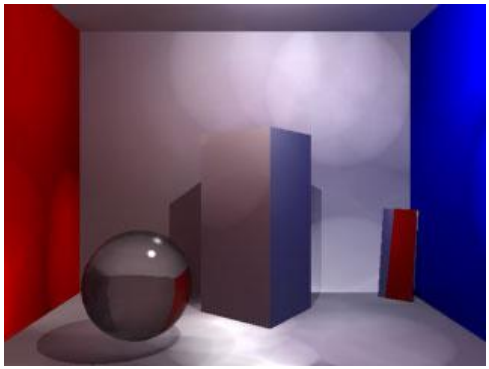
bounce level = 6



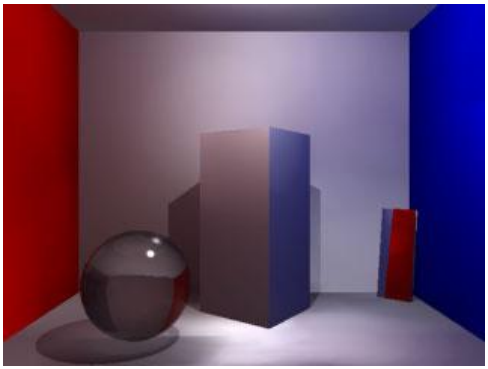
max error = 0.2



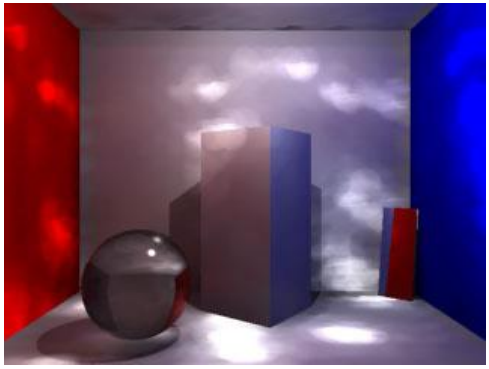
max error = 1.0



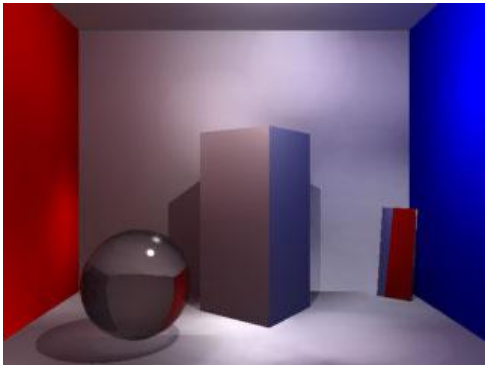
blending area = 0.1



blending area = 2.0



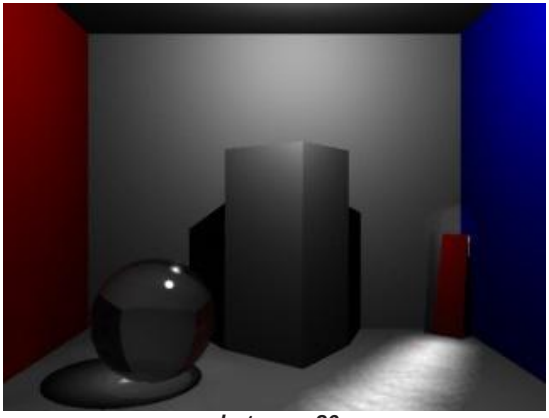
max distance = 0.4



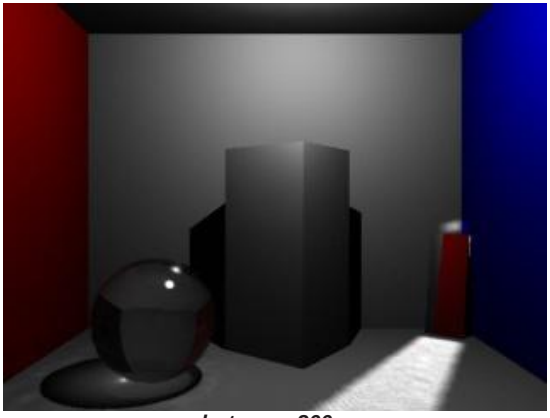
max distance = 1.0

- **Photon maps**
  - *number of photons:* Specifies the number of photons that will be shot when calculating caustics. This value is in thousands, and the number of photons that are actually shot will vary from scene to scene. In some scenes, you may need to increase this setting greatly (e.g. to over 5000) to generate enough photons to give accurate caustics.
  - *media photons:* This controls the number of photons that should be shot if volumetrics is enabled. When set to 0, this switch is effectively off.
  - *volume %:* Another setting that controls how many photons are fired, and refers to the number of photons covering a surface. If set to 0, the number is automatically generated, but you may need to adjust this value.
  - *gather range min* and *gather range max:* Control the number of photons to be gathered at each point during photon mapping. Increasing the values will increase the accuracy of the render.
    Note: The next photons settings do not refer to caustics generation, but are rather an alternative method of calculating irradiance for the scene instead of the *distribution* setting. You still need to have the enable box checked for irradiance, but you do NOT need to have *enable caustics* checked.

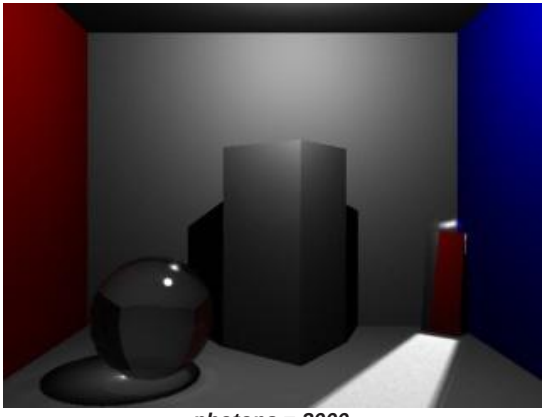
- *irradiance:* Switches on this alternative method.
- *from lights:* The number of photons that should be gathered for each light.
- *per ray:* The number of samples that should be taken during rendering.



photons = 30



photons = 300



photons = 3000

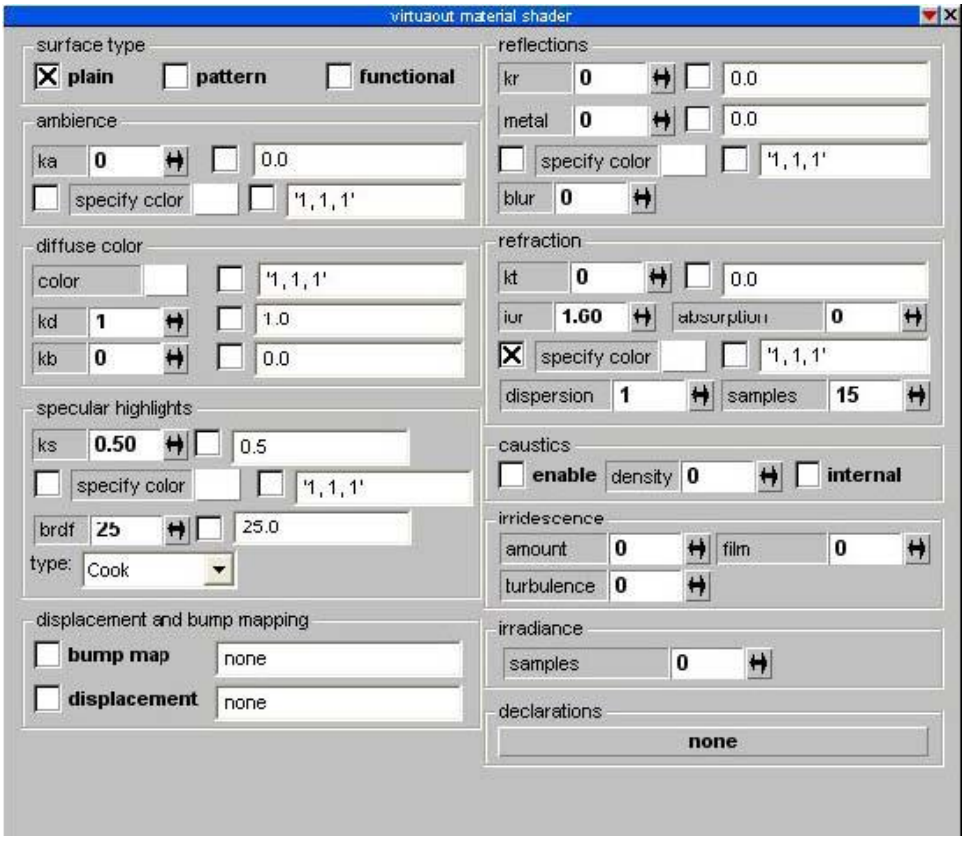
- **Rays:** These settings refer to the raytracing engine, and can be used even with irradiance and photons disabled.
  - blur samples and blur level: Used when rendering VL shaders with soft reflections and refraction effects.

Virtualight Reflectance Shader

In order to provide more control when working with the **Virtualight *rendering engine***, the **Virtualight *reflectance shader*** is provided. This documentation does not cover the format for the VL shading language. We encourage you

to read the VL reference manual first.

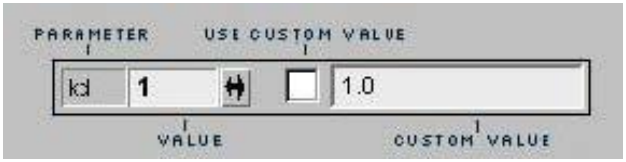
To use the *VL reflectance shader*, first load it into the ME from the **shader library** panel in the usual manner. Note that this shader can be loaded into the ME only if the LV rendering engine is loaded first. Otherwise it will be disabled.



- **Surface type:** The VL reflectance shader allows for 3 different types of surfaces:
  - *plain surface:* Will render the fastest, but can only have basic settings such as *simple color, ambience, diffuse*, etc.
  - *Functional surface:* Takes longer to evaluate but can contain many different calculations such as *Fresnel falloff* effects for reflections, procedural textures, and so on.
  - *Pattern surfaces:* Fall in between, but are not yet supported in this shader.

  Note: If you experience problems rendering a sample, try setting the surface type to functional. Complex parameters will not be used if the surface is set to plain.

- **Reflectance values:** These are the values for *ambience, diffusion, specular highlights, reflections, and refractions*. Most of these work in a similar manner as they do in trueSpace, but some explanation of the format of the settings is in order



As can be seen above, you can use the number spinner (or colorbutton) to set a regular value, or you can enter custom text to perform a calculation. The checkbox determines whether or not the shader will use the number spinner or the custom text for the shader. Please note: *either the spinner value or the text box must be chosen,* one or the other.

- **Ambience:** Controls how emissive the material is. This is the ***Ka*** value. You can choose to specify a color for this emissiveness by checking the *specify color box* or leaving it unchecked to use the base color. You can then decide whether to use the *colorbutton* or have a custom color text.
- **Diffuse color:** The base color of the material. The base color can be either a plain color chosen with the *color* button, or a custom color chosen by using the *custom color* text. For example, to use a texture map, you would enter something like:

  PlanarImageMapping(ImageFile("c:\images\grass.jpg", BILINEAR), (u, 0, v), 1)

  You set the diffuse amount with the ***Kd*** parameter. Related to this value is the ***Kb*** (brilliance) setting with which you can experiment with for metals and glasses.

- **Specular highlights:** Controls the highlights on a material for simulating rough, smooth, shiny, or dull surfaces.
  - *Ks:* The strength of the specularity.
  - *BRDF:* Similar to the roughness slider in trueSpace, lower values give you tighter highlights and thus a smoother, harder surface. VL allows you to choose from different BRDF models, which you can access from the *type* drop down list. *Phong* will give you a classic look. *Cook* will give nice glossy highlights for glass-type surfaces. Experiment to see what renders the best. The color of the highlights can also be specified if needed.
- **Reflections**
  - *Kr:* Akin to *mirror* in trueSpace and describes how reflective the surface is.
  - *metal:* Determines how much the base color affects the reflections for simulating metallic surfaces. You can specify a color for the reflections in order to attain certain results.
  - *blur:* Allows for blurred reflections. Set it closer to 1 for more blurred reflections. The number of samples used is controlled by the *blur samples* and *blur level* settings in the main **virtuaOut** interface in the ***gi*** panel.
- **Refractions**
  - *Kt:* Controls the amount of transmission in the surface.
  - *Ior:* Sets the value for the index of refraction. You can choose whether or not specify a color for refraction. This can be very useful if you are trying to create colored glass.
  - *dispersion:* Calculates chromatic dispersion for refraction (like the little rainbow effect you get on refractions in glass). A dispersion value of 1.0 will switch off this effect. Small increases are common; water is 1.007 and diamond is 1.035.
  - *samples:* Controls how smooth the dispersion will be. Higher is better but comes at a rendering time

cost.
- *absorption:* Used for simulation of translucent materials. Using a value greater than 0 switches this feature on.

- **Bump Mapping and Displacement**
  You can apply bump mapping to any object in VL. Bump maps can be set using an image file or procedurally. To use an image map, the following would be used:

  PlanarBumpMapping(ImageFile("c:\images\grassbump.jpg", BILINEAR), (u, 0, v), 2.0)

  The last "2.0" refers to the bump map amplitude which can be set to a positive value or a negative value to invert the bump mapping effect.

- **Displacement mapping:** Does not just give the effect of a bumpy surface, but actually modifies the geometry. Again, you can use an image file or a procedural effect. For example, entering fnoise(P) will give a simple displacement. If you want to use a file, you would use:

  *PixelHeight(ImageFile("c:\images\grassbump.jpg", BILINEAR), (u, 0, v), 0) * 0.005*

  Note: You can only use displacement mapping with NURBS objects. The virtuaOut output panel parameters *u subdivs* and *v subdivs* should be increased to at least 32 in order for the displacement to be effective.

- **Iridescence**
  This is a nice effect that can be used to simulate the look of irridescence, or the sheen of color found on the surface of a bubble, for example. The VL documentation has full information on the use of irridescence.
- **Caustics**
  - *caustics:* Can be controlled on a per-material basis. The *caustics* toggle must be checked and a value greater than 0 for *density* entered.
  - *Density:* Works as a percentage and refers to the surface area of the object that should be used for calculation. The internal flag can be used for more accurate (and slower) representation of glass surfaces.
- **Irradiance:** Specifies the number of irradiance samples the surface will use when doing *gi* renderings.
- **Declarations**

  Much of the power of the VL shaders comes from their ability to define (e.g.) color gradients and functions for procedural textures. However if we allowed for all possible instances where such shaders could be written, this shader interface box would be much larger than it already is. To account for this, trueSpace makes use of the fact that VL shaders can include parts of shaders from other files. You can define your color gradients and other material parameters in a separate file and save it as a .vs file (for example, *mygreatshaderstuff.vs*). Then, by clicking the *declarations* button and browsing to this file, you can reference anything from the file in other shader parameters. For example, in the declarations section, use the *alien.vs* file that is included with VL. It should be in the **Examples** folder where **Virtualight** is installed.

  The *alien.vs* file contains a procedural texture function called *swirl*, which can be used with a color gradient, also contained in the file. In the color section, enter *swirl_map[swirl]* in the *custom color* text box, and check the box to use a custom color. A procedural texture will now be used for the color.

Notes

- **Geometry:** All polyhedra are exported as triangulated meshes, so there should be no problems with holes in the polyhedra. NURBS patches can be exported as NURBS patches or they can be triangulated if the render is not as expected (NURBs as triangulated meshes is the default). The object render options *cast shadow* and *receive shadow* are fully supported.
- **Materials:** *Glass, dielectric*, and *mirror* shaders will result in VL shaders with caustics enabled. It should be noted that the *ambient* value of a reflectance shader controls how emissive the surface is when rendering with irradiance enabled.

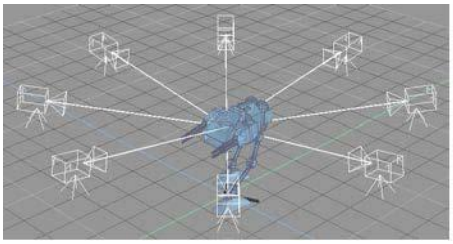
  Color procedural shaders can be converted to texture maps that VL can render. *Solid procedural* shaders will not look exactly the same, however. For best results 2D (wrapped/UV) shaders should be used in these instances. *Transparency* shaders are not fully supported in VL, however trueSpace materials with a transparency map, or plain transparency will be converted correctly. *Procedural rough* and *wrapped rough* shaders will be converted to the closest approximation in VL, but it is recommended that regular bump maps are used instead. The different shading types; *faceted, auto-facet*, and *smooth* shaded are supported fully for polyhedra, but NURBS are either completely smooth, or completely faceted. NURBS objects are also only exported with a single material.

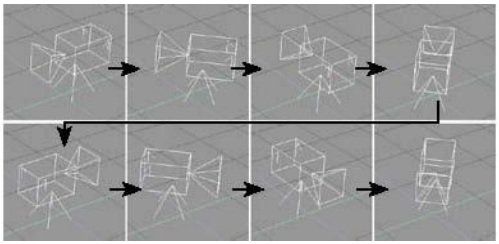
- **Lights**
  - Local, spot, and infinite lights are the same as in trueSpace.
  - Goniometric lights will be converted to spherical area lights.
  - Sky lights will be converted to a VL sky. Intensities less than 1.0 are recommended.
  - Area lights will be converted to VL flat area lights, but the conversion is not always straightforward due to differences in renderers and their representation in trueSpace.
  - IBL lights will be converted to VL sky lights, but using the image you used in the IBL light. There are a several differences, though. VL has no concept of the IBL orientation, so rotating will make no difference. Related to this, you may also find that the image is not aligned exactly as it is in trueSpace, and you may need to edit the image in an image editing program to make it fit correctly.
  - If you enable lens flares for a trueSpace local or goniometric light, the VL light will have a visible glow. The size of the glow is specified by the size of the tS light object.
  - Shadows are either on or off. Shadow maps are not supported in VL.
  - Non-specular lights are fully supported in VL. This type of light will still illuminate objects and cast shadows (if on), but will not cause a specular highlight to appear. These useful for fill lights. To turn off specular highlights for a particular light, uncheck the trueSpace option Physically based units for the light.
  - Negative tS lights will be written out as VL blackholes.
  - Projector lights are not supported in VL.
- **Global shaders**
  - Background color is correctly exported.
  - Gradient background is correctly exported.
  - Background image is correctly exported.
  - Simple Volumetric in trueSpace will convert to participating media settings in VL. Only the Samples and Density parameters apply. Note that you must still enable volumetrics for each light.
  - Fog is supported. Ground Fog is not supported.

7.6 Quicktime VR

Panoramic movies are constructed from a series of views taken from the same spot but at slightly different angles. These images are then combined into a single image to allow dynamic viewing.

Object movies are slightly different in that they consist of a number of views of a static object taken from different points around the object. This allows the user to "fly" around the object.



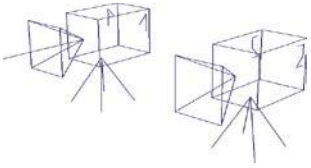


Object Camera Rendering Motion Panoramic Camera Rendering Motion

Notes:

- There is no output during the process of generating Quicktime VR movies, and the process cannot be canceled.
- The process of generation can take a long time.
- The software to view a Quicktime VR movie can be freely obtained from Apple via the internet. More information can be found at the Quicktime VR web site http://quicktimevr.apple.com

The Panoramic camera (Pcam) enables the creation of both panoramic and object Quicktime VR movies. The **Panoramic Camera tool** can be found on the main toolbar, in the same tool group as the regular camera. The Pcam (1 in the image below) has the same basic properties as the regular camera (2); only the visual representation is slightly different. The Pcam also has additional features.



Panoramic Camera

- **Movie:** Selects the mode in which Pcam works. It can be in either **Panoramic** or **Object** mode. The Pcam will generate a Quicktime VR movie according to the chosen mode.

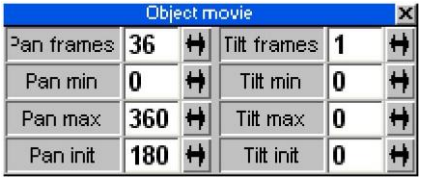
- **Panoramic mode:** A Quicktime VR Panoramic movie will be generated.
- **Object mode:** A Quicktime VR Object movie will be generated.

- **Quality level:** Indicates the quality level (compression) of the output image. Lower quality levels produce a smaller files, whereas higher quality levels produce larger files. By default, the value is set to the middle (85).
- **Render:** Starts the generation of the Quicktime VR movie and renders the active view.

Panoramic movie options

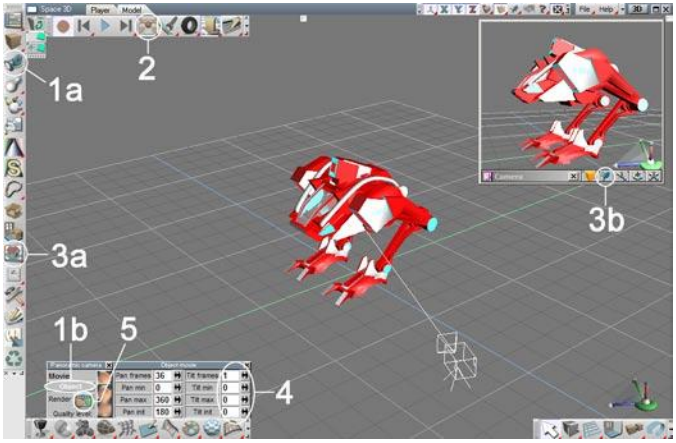
- **Slit:** Specifies the number of individual images that should be rendered to make up a full panorama. To avoid distortion in the resulting movie, it is not recommended to set this value too low. In addition, high values should be avoided because the gain in quality becomes minimal. Usually this control should be left at its default settings.
- **Width:** Specifies the width of the window that the movie will be displayed in.
- **Height:** Specifies the height of the window that the movie will be displayed in.

Object movie options



- **Pan frames (Tilt frames):** Number of frames to generate between the minimum and maximum horizontal pan (vertical tilt) positions. (number_of generated_frames = pan_frames x tilt _frames)
- **Pan Min (Tilt Min):** The minimum horizontal pan (vertical tilt) position the object will rotate to. (measured in degrees).
- **Pan Max (Tilt Max):** The maximum horizontal pan (vertical tilt) position the object will rotate to. (measured in degrees).
- **Pan Init (Tilt Init):** This parameter specifies the initial horizontal pan (vertical tilt) position in degrees. Indicates the starting horizontal (vertical) rotation for the object when the movie is first opened.

Example: How to create an Object movie.



1. The first step is to create the object and the Pcam.
   a) Use the **Panoramic Camera** button to create a camera.
   b) Choose the **Object** movie mode in the Panoramic camera option panel.

2. Bind the camera to the object using the **Look At** tool. You will see a line from Pcam to the center of the object.

3. a) Open a new auxiliary window.
   b) Set the new window to 'camera' view. Resize the window to the appropriate size. Move the camera to fit the whole object.

4. Set the Object movie control variables. (We want to rotate the object only in the horizontal direction.) :
   - **Tilt frames:** 1
   - **Tilt max:** 0
   - **Tilt min:** 0
   - **Tilt init:** 0

5. Select the active view and **Render movie**.