Contents

© 2011 Autodesk, Inc. All rights reserved.

Also note Appendix A: Copyrights.

Chapter 1

Rendering Reference

All global render settings in Turtle can be found in Maya's Render Settings window when Turtle is set as the current renderer. Common render settings, e.g for animation and image output, are located under the **Common** tab. Turtle specific settings can be found under the **Turtle** tab. The first control to set is the Render Type, which controls what type of rendering Turtle should do. Then there are five different categories of settings; **Sampling**, **Global Illumination**, **Environment**, **Options** and **Baking**.

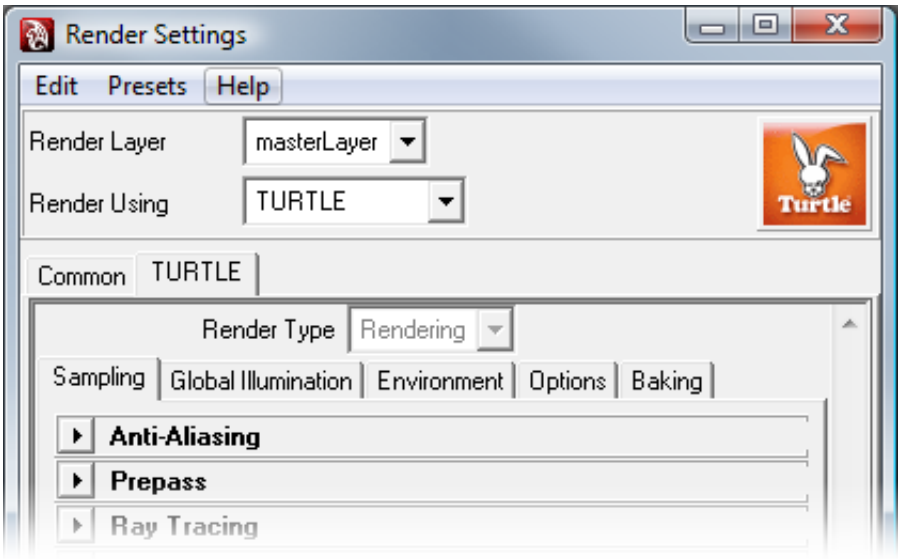

Figure 1.1: Render Settings Window

1.1 Render Type

With the **Render Type** control you select what type of rendering Turtle should do when a rendering is started in Maya. If **Rendering** is selected Turtle will do ordinary frame renderings from the current camera. A batch rendering in this mode will render all frames from

all renderable cameras according to the current animation settings. If **Baking** is selected Turtle will instead perform baking to texture maps or vertex colors. The current bake layer controls which objects and what type of lighting/shading to bake. A batch rendering in this mode will bake out all bake layers that is set to renderable. For more information on baking, see section 2.

1.2 Sampling

The Sampling tab holds controls for Anti-Aliasing as well as controls for Ray Tracing.

1.2.1 Anti-Aliasing

Min/Max Sample Rate

Turtle uses an adaptive sampling scheme that can perform both undersampling (less than one sample per pixel) and oversampling (multiple samples per pixel). A sampling rate from one sample per 256 pixels and up to 256 samples per pixel can be used. The minimum and maximum sample rate Turtle should use. The min to max number of samples taken are printed in text in the GUI.

Contrast threshold

If the contrast differs less than this threshold Turtle will consider the sampling good enough.

Diagnose Sample Rate

Enable this to diagnose the density of the sampling. The brighter a pixel is, the more samples were taken at that position.

Clamp Values

To work efficiently, the sampling algorithm can clamp the intensity of the image to the [0..1] range. When rendering to OpenEXR this is not desired. Clamp values should then be unchecked.

Filter Type

The available filter kernels are:

- Box
- Triangle
- Cubic
- Gauss
- Catmull-Rom
- Lanczos
- Mitchell

Filter Width

The width of the filter kernel in pixels. Range from 1.0 to 3.0.

Filter Height

The height of the filter kernel in pixels. Range from 1.0 to 3.0.

1.2.2 Prepass

Sets the sample rate used during the GI precalculation pass. The prepass is progressive, rendering from a low resolution up to a high resolution in multiple passes.

Min/Max Sample Rate

The **Min Sample Rate** sets the initial resolution, where a negative value means a lower resolution than the original image (in powers of two), e.g. -4 gives the original resolution divided by 16. The **Max Sample Rate** sets the resolution of the final prepass, e.g 0 giving the same resolution as the original resolution. These controls should normally be left at default values -4/0, however by decreasing them you can render very fast GI preview renderings.

1.2.3 Ray Tracing

Figure 1.2 shows the **Ray Tracing** interface in the Maya Render Settings window.

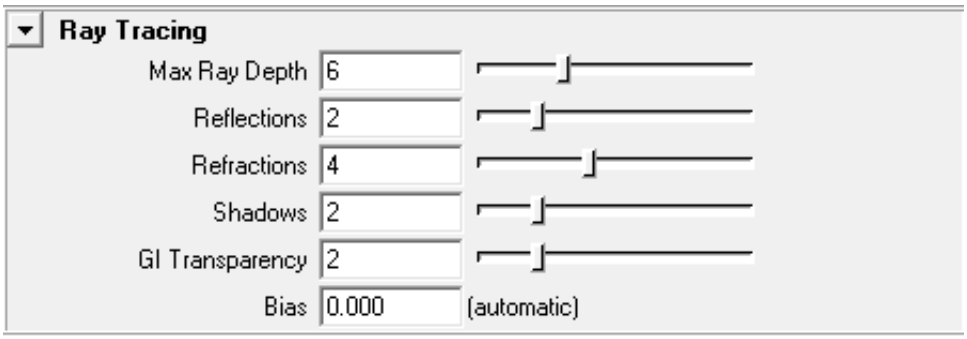

Figure 1.2: Ray tracing settings

Max Ray Depth

The maximum number of ”bounces” a ray can take before being truncated. A bounce can be a reflection or refraction.

Reflections

The maximum number of reflections a ray can take before being truncated. This limits the value set in each material.

Refractions

The maximum number of refractions a ray can take before being truncated. This limits the value set in each material.

Shadows

Controls at what depth a ray can spawn shadow rays. If set to 1, only primary rays will spawn shadow rays. If set to 2, the first bounce will spawn shadow rays as well. This limits the value set in each material.

GI Transparency

The maximum number of transparent surfaces a GI ray can go through, before being truncated. This is used by all gathering light integrators, e.g. Final Gather, Monte Carlo, Occlusion and LUA samplings.

Bias

An error threshold to avoid self intersections. For example, a shadow ray should not intersect the same triangle as the primary ray did, but because of limited numerical precision, this can happen. The bias value moves the intersection point to eliminate this problem. If set to zero this value is computed automatically depending on the scene size.

1.3 Global Illumination

All settings for the global illumination algorithms are found under the Global Illumination tab. This is where you find caustics, photon map, path tracer, final gather and monte carlo settings. Figure 1.3 shows the **Global Illumination** interface in the Maya Render Settings window. The **Enable GI** and **Enable Caustics** switches are master controls for enabling the Global Illumination or Caustics.

Primary and Secondary GI

The Global Illumination system allows you to use two separate algorithms to calculate indirect lighting. You can for instance calculate multiple light bounces with a fast algorithm like the **Path Tracer**, and still calculate the final bounce with **Final Gather** to get a fast high-quality global illumination render. Both subsystems have individual control of **Intensity** and **Saturation** to boost the effects if necessary. Available algorithms include

- **Photon Map**
- **Dynamic Photon Map** (for advanced baking)
- **Path Tracer**
- **Final Gather**
- **Monte Carlo**

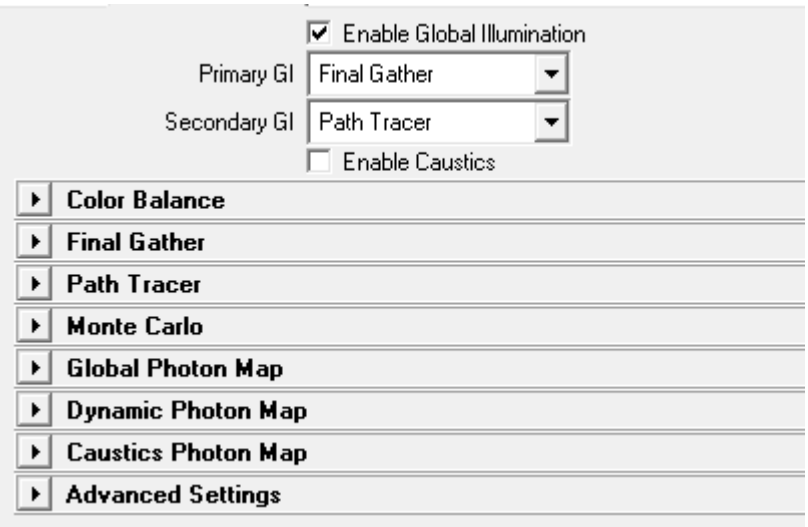

Figure 1.3: Global Illumination settings

There is also the option to use **None** both as primary and secondary GI. For secondary GI, use **None** when you don't need deeper indirect light or if you have a primary GI that cannot use secondary GI. Using **None** for primary GI should be done when baking advanced passes that perform their own gather, e.g. RNM (when not using radiance cache) and the Lua pass. When these passes perform their own gather, only the secondary GI will be sampled, making any primary GI useless.

Caustics

Enables caustics, using caustics photon mapping.

1.3.1 Color Balance

Figure 1.4 shows the **Color Balance** interface for GI.

Primary Intensity

Scales the intensity of the first GI bounce.

Primary Saturation

Scales the saturation of the first GI bounce.

Secondary Intensity

Scales the intensity of secondary GI bounces.

Secondary Saturation

Scales the saturation of secondary GI bounces.

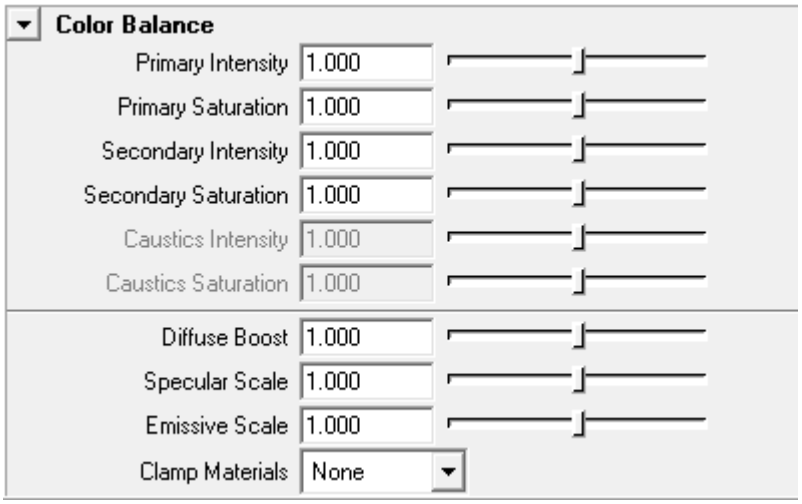

Figure 1.4: Color Balance Settings

Caustics Intensity

Scales the intensity of the caustics photon map.

Caustics Saturation

Scales the saturation of the caustics photon map.

Diffuse Boost

A gain for boosting the diffuse material components when used by GI. Can be used to adjust the amount of indirect diffuse lighting. For example, a scene with very dark textures will not get much indirect lighting, but when boosting all diffuse components the indirect effects are increased.

Specular Scale

Scales the specular material components when used by GI. Can be used to adjust the amount of indirect lighting from specular effects.

Emissive Scale

Scales the emissive material components when used by GI. Can be used to adjust the amount of indirect lighting from emissive materials.

Clamp Materials

Controls if materials should be clamped to preserve physical correctness when used by GI.

- **None**: Disables material clamping.
- **Component**: Clamp each R, G and B component seperatly.

- **Intensity**: Clamp the intensity (HSV Value).

1.3.2 Final Gather

Figure 1.5 shows the **Final Gather** interface in the Maya Render Settings window.

Use Cache

Selects what caching method to use for final gathering.

- **Off (Brute Force)**: disables caching and performs a new final gathering for every shading point. Gives very accurate results but long render times. Can e.g. be used for reference images.
- **Irradiance**: caches the irradiance at selected points in the scene and uses interpolation in between the points. This is the default method.
- **Radiance SH**: caches radiance SH functions at selected points in the scene and uses interpolation in between the points. The radiance cache is useful in some advanced baking passes (e.g. Radiosity Normal Maps), when directional indirect lighting is needed.

Rays

Sets the maximum number of rays to use for each final gather sample point. A higher number gives higher quality, but longer rendering time.

Depth

Sets the number of indirect light bounces calculated by final gather. A value higher than 1 will produce more global illumination effects, but note that it can be quite slow since the number of rays will increase exponentially with the depth. It's often better to use a fast method for secondary GI. If a secondary GI is used the number of set final gather bounces will be calculated first, before the secondary GI is called. So in most cases the depth should be set to 1 if a secondary GI is used.

Contrast Threshold

Controls how sensitive the final gather should be for contrast differences between the points during precalculation. If the contrast difference is above this threshold for neighbouring points, more points will be created in that area. This tells the algorithm to place points where they are really needed, e.g. at shadow boundaries or in areas where the indirect light changes quickly. Hence this threshold controls the number of points created in the scene adaptively. Note that if a low number of final gather rays are used, the points will have high variance and hence a high contrast difference, so in that case you might need to increase the contrast threshold to prevent points from clumping together.

Interpolation Points

Sets the number of final gather points to interpolate between. A higher value will give a smoother result, but can also smooth out details. If light leakage is introduced through walls when this value is increased, checking the sample visibility solves that problem, see Check Sample Visibility below.

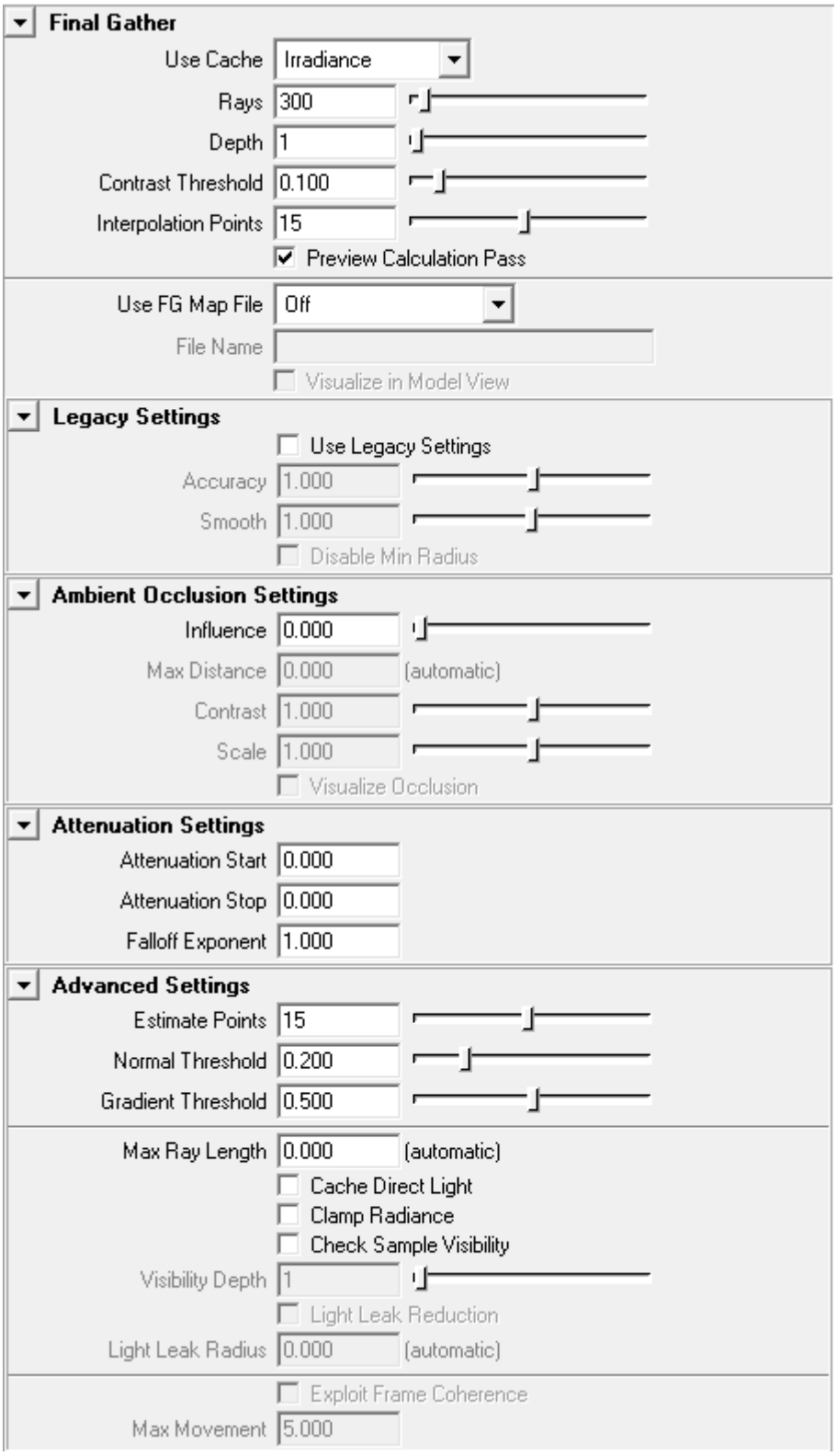

Figure 1.5: Final Gather Settings

Preview Calculation Pass

Turn this on to visualize the final gather prepass. Using the Preview Calculation Pass enables a quick preview of the final image lighting, reducing lighting setup time.

Use FG Map File

Controls usage of final gather map files. By saving the solution to a file, it can be reused for successive renderings. The file will be placed under *turtle/fgMaps* in the current project directory. Note that if lights or geometry are moved, recalculation is necessary.

- **Off**: Disables file usage.
- **Overwrite**: Creates a new map file, overwriting any existing file.
- **Reuse**: Loads the map from a previously created file.
- **Reuse and append**: Loads the map from a file, but also writes back any new samples created during rendering.
- **Precalc. for animation**: Same as Reause and append, but only the FG pre-pass is rendered. This can be used to generate an FG Map file for an animation sequence. First render the animation in this mode, creating the map file, then switch to Reuse before final rendering of the animation. Note that the animation frame step can be set higher than 1 during the precalculation to save time. For instance rendering every other frame, or even as sparse as every 10th frame.

Note that different cache methods (Irradiance / Radiance SH) cannot share the same map file. To solve this problem irradiance cache files will be appended with ".irradiance". If Radiance SH is used with a depth higher than 1, irradiance caching is used for all secondary bounces. So in that case two files will be saved, one Radiance SH cache file named "filename" and one Irradiance cache file named "filename.irradiance".

File Name

Sets the file name of the final gather map file.

Visualize in Model View

Visualize the stored samples as a point cloud in the Maya viewport. You can change point size and color in the attributes of the generated point cloud.

Use Legacy Settings

Enables the older final gather method. With this method points are placed according to the geometry in the scene. More samples are placed near walls and corners of the geometry. The contrast difference of the samples are not considered.

Accuracy (legacy)

Controls the sample point density. Higher accuracy generates more points and a better result at the expense of longer rendering time.

Smooth (legacy)

Applies a filter that reduces noise in the final gather solution. This is much faster than increasing accuracy or sending more rays. 1.0 is the default value. Values below 1.0 gives a sharper and noisier look, values above 1.0 gives a smoother look with less details preserved.

Disable Min Radius (legacy)

When enabled more points will be created in corners of the geometry, which can reduce artifacts. However don't enable this unless you need to, since it will increase the render time.

Ambient Occlusion - Influence

Controls a scaling of Final Gather with Ambient Occlusion which can be used to boost shadowing and get more contrast in you lighting. The value controls how much Ambient Occlusion to blend into the Final Gather solution.

Ambient Occlusion - Max Distance

Max distance for occlusion. Beyond this distance a ray is considered to be unoccluded. Can be used to avoid full occlusion for closed scenes.

Ambient Occlusion - Contrast

Can be used to adjust the contrast for ambient occlusion. Increase this value to make bright surfaces brighter and dark surfaces darker.

Ambient Occlusion - Scale

A scaling of the occlusion values. Can be used to increase or decrease the shadowing effect.

Ambient Occlusion - Visualize Occlusion

When this is enabled a single Ambient Occlusion pass will be rendered, to visualize the AO effect. All other render passes are ignored. This can be usefull while tuning your AO settings.

Attenuation Start

The distance where attenuation is started. There is full intensity before this distance. Note that Attenuation Stop must be set to non-zero to enable attenuation.

Attenuation Stop

By setting this value higher then 0.0, attenuation is enabled. The value controls the distance where intensity goes to zero. Rays that are traced beyond this distance don't contribute to the lighting.

Falloff Exponent

Controls the rate by which lighting falls off by distance. The exponent controls the shape of the function that goes from full intensity (at Attenuation Start) to zero intensity (at Attenuation Stop).

Estimate Points

Sets the minimum number of points that should be used when estimating final gather in the precalculation pass. The impact is that a higher value will create more points all over the scene. The default value 15 rarely needs to be adjusted.

Normal Threshold

Controls how sensitive the final gather should be for differences in the points normals. A lower value will give more points in areas of high curvature.

Gradient Threshold

Controls how the irradiance gradient is used in the interpolation. Each point stores it's irradiance gradient which can be used to improve the interpolation. However in some situations using the gradient can result in white ”halos” and other artifacts. This threshold can be used to reduce those artifacts.

Max Ray Length

The max distance a ray can be traced before it's considered to be a “miss”. This can improve performance in very large scenes. If the value is set to 0.0 the entire scene will be used. The distance is measured in Maya units.

Cache Direct Light

When this is enabled final gather will also cache lighting from light sources. This increases performance since fewer direct light calculations are needed. It gives an approximate result, and hence can affect the quality of the lighting. For instance indirect light bounces from specular highlights might be lost. However this caching is only done for depths higher than 1, so the quality of direct light and shadows in the light map will not be reduced.

Clamp Radiance

Turn this on to clamp the sampled values to [0, 1]. This will reduce low frequency noise when Final Gather is used together with other Global Illumination algorithms.

Check Sample Visibility

Turn this on to reduce light leakage through walls. When points are collected to interpolate between, some of them can be located on the other side of geometry. As a result light will bleed through the geometry. So to prevent this Turtle can reject points that are not visible.

Visibility Depth

Controls for how many bounces the visibility checks should be performed. Since the visibility checks are expensive and increases render time, the default mode is to just check visibility for the first light bounce, Depth = 1. However if you get light leakage in your scene from deeper light bounces, you can increase this value to make sure visibility is checked for deeper bounces as well. Note that this will increase the render time.

Light Leak Reduction

This setting can be used to reduce light leakage through walls when using final gather as primary GI and photon mapping or path tracing as secondary GI. Leakage, which can happen when e.g. the path tracer filters in values on the other side of a wall, is reduced by using final gather as a secondary GI fallback when sampling close to walls or corners. When this is enabled a final gather depth of 3 will be used automatically, but the higher depths will only be used close to walls or corners. Note that this is only useable when photon mapping or path tracing is used as secondary GI.

Light Leak Radius

Controls how far away from walls the final gather will be called again, instead of the secondary GI. If 0.0 is used a value will be calculated by Turtle depending on the secondary GI used. The calculated value is printed in the output window. If you still get leakage you can adjust this by manually typing in a higher value.

Exploit Frame Coherence

When enabled Turtle will save and reuse samples that are far away from animated objects.

Note: If Exploit Frame Coherence is used the points loaded from file will be considered as static (valid throughout the entire animation).

Max Movement

Controls how many final gather points will be saved and how many frames they will be used when ”Exploit Frame Coherence” is enabled. A higher value will lower the amount of points saved and lower the number of frames they are used in. A small value will increase the number of points saved and increase the number off frames they are used in.

Note: If the value is set to low you will get artifacts like the “Lucky Luke effect” (objects being faster than their own shadows).

1.3.3 Path Tracer

The Path Tracer algorithm calculates indirect lighting by tracing a (high) number of rays from the camera into the scene. These rays are bounced and traced further into the scene, depending on the properties of the surfaces they hit. Surfaces with a high diffuse reflectance will have a higher probability to continue the ray paths, hence giving more color bleeding

to other surfaces. The path is only terminated when the probability for absorption is high enough.

Illumination from the ray bounces are stored at selected points in the scene, resulting in a set of cache points. A parameter, Cache Point Spacing, is used to set the distance between the cached points. A smaller distance will generate more cache points, and enable better capturing of details, but also higher memory usage and longer rendering time.

Once the pre-render pass completes, the path tracer cache holds an approximation of the indirect light. The quality of the solution depends on the number of paths used and the density of the cache. If the path tracer solution is used as Secondary GI, with Final Gather or Radiance Cache as Primary GI, the settings can be set quite low and still produce good quality.

The cache points can be saved to a map file and reused for successive renderings. Note that the file option Reuse and Refine will load the map from file and then refine the solution by adding the paths from a new pre-render pass to the cache. By doing this the solution will be better and better for each rendering.

Figure 1.6 shows the **Path Tracer** interface in the Maya Render Settings window.

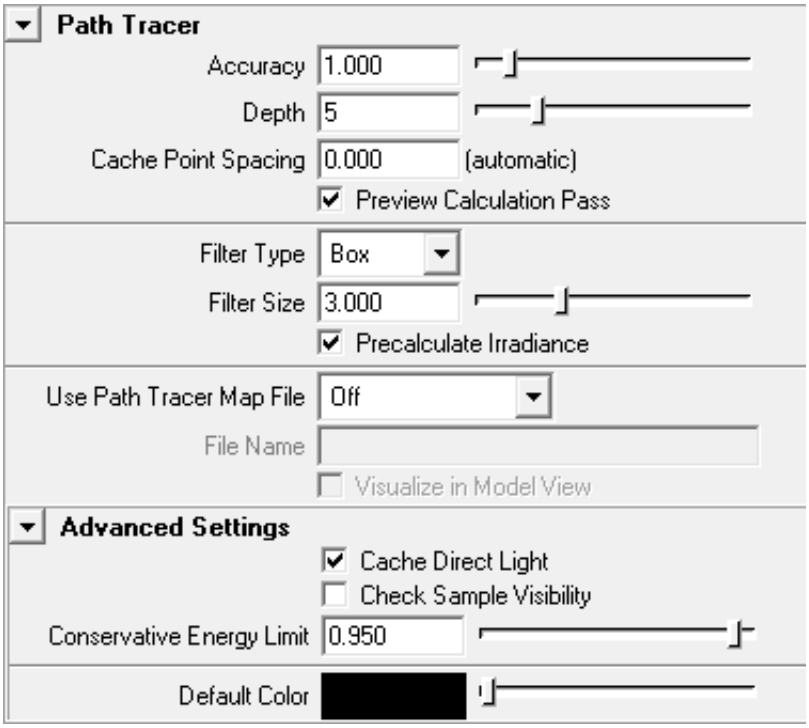

Figure 1.6: Path Tracer settings

Accuracy

Sets the number of paths that are traced for each sample element (pixel, texel or vertex). For preview renderings, you can use a low value like 0.5 or 0.1, which means that half of the pixels or 1/10 of the pixels will generate a path. For production renderings you can use values above 1.0 if needed to get good quality.

Depth

Sets the average number of bounces in the Path Tracer solution. The maximum depth is 40. A lower Depth will result in a faster but noisier solution, but the overall intensities will be the same. How far light actually bounces is much dependent on your materials, brighter materials lets light spread further than dark materials.

Cache Point Spacing

Sets the maximum distance between the points in the path tracer cache. If set to 0.0 a value will be calculated automatically based on the size of the scene. The automatic value will be printed out during rendering, which is a good starting value if the point spacing needs to be adjusted.

Preview Calculation Pass

If enabled the pre-render pass will be visible in the render view.

Filter Type

Selects the filter to use when querying the cache during rendering. None will return the closest cache point (unfiltered).

Filter Size

Sets the size of the filter as a multiplier of the Cache Point Spacing value. For example; a value of 3.0 will use a filter that is three times larges then the cache point spacing. If this value is below 1.0 there is no guarantee that any cache point is found. If no cache point is found the Default Color will be returned instead for that query.

Precalculate Irradiance

Prefilter the cache points before the final pass starts. This can increase the performace using the final render pass and is especially useful when **Check Sample Visibility** is enabled, or the filter kernel is large.

Use Path Tracer Map File

Selects how the cache points are saved or loaded from file. Overwrite will create a new map file. Reuse will load the cache points from the map file and skip the pre-render pass. Reuse and Refine will load the cache points from file but refine the solution by running the pre-render pass again.

File Name

Sets the name of the cache map file. The map file is saved in *turtle/ptMaps* in the current project directory.

Visualize in Model View

If enabled the cache points saved to file will be displayed in Maya's model view after the rendering.

Cache Direct Light

When this is enabled the path tracer will also cache lighting from light sources. This increases performance since fewer direct light calculations are needed. It gives an approximate result, and hence can affect the quality of the lighting. For instance indirect light bounces from specular highlights might be lost.

Check Sample Visibility

Turn this on to reduce light leakage through walls. When points are collected to interpolate between, some of them can be located on the other side of geometry. As a result light will bleed through the geometry. So to prevent this Turtle can reject points that are not visible.

Conservative Energy Limit

The Path Tracer needs to clamp the Maya materials so that they act in a physically accurate way. The Conservative Energy Limit parameter lets you set how tight the clamp is, where 1.0 would mean that a material is allowed to be exactly on the border of being non-physical. The default is 0.95, which means that the material is at most 5% below the critical limit.

Default Color

Sets the color to be returned if no valid cache point is found during a cache query. If no points are found the Filter Kernel needs to be increased. By setting the color to e.g. bright red or green, you can easily see if a larger filter kernel is needed.

1.3.4 Monte Carlo

This is a brute force GI method that calculates GI by sending rays in a hemisphere around the shading points. No caching is used so a new calculation is done for each shading point. An importance sampling method is used for secondary bounces, so a high number of bounces can be used without exploding the render time. This Monte Carlo method is unbiased and can be used for reference images when used as primary GI. It can also be used as secondary GI together with final gather to produce a multi-bounce high quality result without light leakage.

Figure 1.7 shows the **Monte Carlo** interface in the Maya Render Settings window.

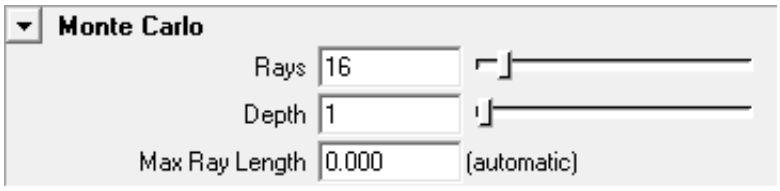

Figure 1.7: Monte Carlo settings

Rays

Sets the number of rays to use for each calculation. A higher number gives higher quality, but longer rendering time.

Depth

Sets the average number of indirect light bounces calculated by monte carlo. A lower number means a noisier but faster solution, although the overall intensities will be the same.

Max Ray Length

The max distance a ray can be traced before it's considered to be a "miss". This can improve performance in very large scenes. If the value is set to 0.0 the entire scene will be used. The distance is measured in Maya units.

1.3.5 Global Photon Map

Figure 1.8 shows the **Global Photon Map** interface in the Maya Render Settings window.

Accuracy

Sets the number of photons to use when calculating global illumination effects. A higher number gives higher quality but takes a longer time to calculate.

Use Photon Map File

Enables usage of photon map files. By saving the photon map to a file, it can be reused in successive renderings. The file will be placed under *turtle/photonMaps* in the current project directory. Note that if lights or geometry is moved, recalculation is necessary. Off disables file usage. Overwrite creates a new photon map file, overwriting any existing file. Reuse loads the photon map from a previously created file.

File Name

Sets the filename of the global photon map file.

Visualize in Model View

Visualize the stored photons as a point cloud in the Maya viewport. You can change point size and color in the attributes of the generated point cloud.

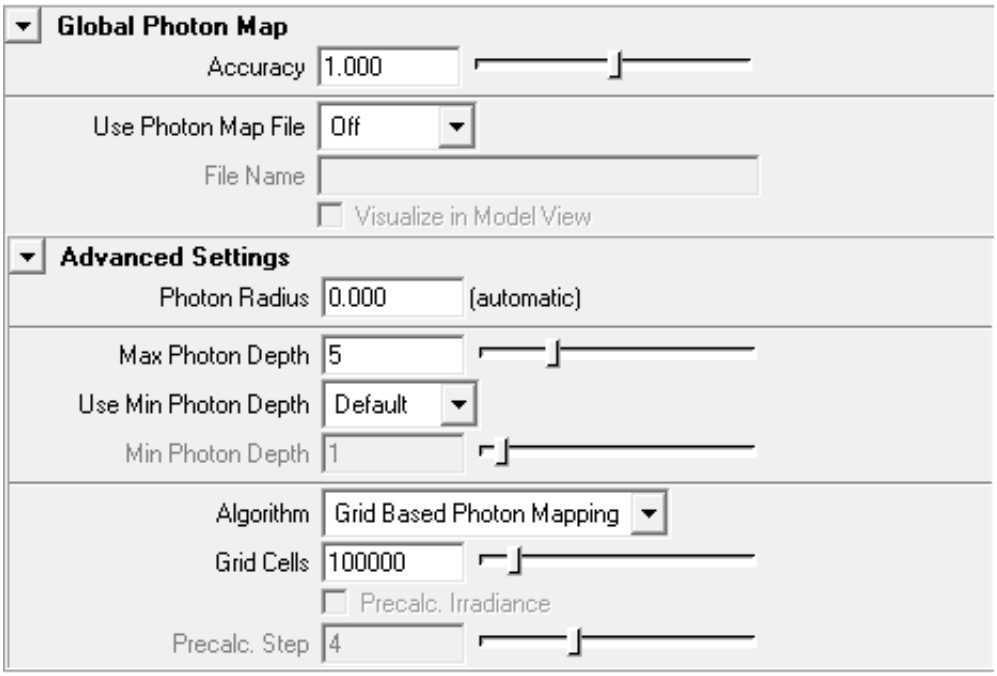

Figure 1.8: Global Photon Map settings

Photon Radius

Sets the maximum search radius to use when searching for photons during lighting calculations for global illumination effects. If the Photon Radius is set to r, photons within a sphere of radius r around the sample point will be used to calculate the light. The radius is given in Maya units. If the Photon Radius is set to 0.0 a value will be calculated automatically based on the size of the scene. The automatic value will be printed out during rendering, which is a good starting value if the point spacing needs to be adjusted.

Max Photon Depth

Sets the maximum number of bounces a global photon can take before it is absorbed.

Use Min Photon Depth

When set to Specified the user can specify a minimum number of bounces a global photon must take before it is stored in the photon map. This can be used to alter the default global photon behavior, which is to store all photons when final gather is used and store all secondary photons when final gather is not used.

Min Photon Depth

Sets the minimum number of bounces a global photon must take before it can be stored in the photon map.

Algorithm

Selects which algorithm to use when calculating global illumination effects. Standard Photon Mapping is more accurate but Grid Based Photon Mapping is much faster.

Grid Cells

Sets the number of grid cells when Grid Based Photon Mapping is used. A higher number gives higher accuracy but longer rendering time. If the scene is very complex, a higher value may be necessary.

Precalc. Irradiance

When enabled, the irradiance will be precalculated at the photon positions before the rendering starts. This gives a much faster rendering at the cost of lower accuracy. If standard photon mapping is used with final gather, it is recommended to enable this feature.

Precalc. Step

Controls how often the precalculation is applied in the photon map. If set to 1 precalculation will be done at all photon positions, if set to 2 at half of the positions, if set to 4 at one quarter of the position, etc. Hence a higher value gives faster rendering but less accuracy.

Note: By excluding the first one or two bounces from being stored in the photon map (using **Min Photon Depth**) one can remove very bright areas, e.g., caused by sunlight through a window.

1.3.6 Dynamic Photon Map

Dynamic Photon Maps are used for specific baking tasks, where light from any possible direction has to be considered. Photons will be emitted towards the scene from every possible direction, and stored using spherical harmonics in a grid structure. Figure 1.9 shows the **Dynamic Photon Map** interface in the Maya Render Settings window.

Total Photons

Number of photons to be emitted into the scene.

Accuracy

Sets the number of photons to use when calculating global illumination effects. A higher number gives higher quality but takes a longer time to calculate.

Use Photon Map File

Enables usage of photon map files. By saving the photon map to a file, it can be reused in successive renderings. The file will be placed under *turtle/photonMaps* in the current project directory. Note that if lights or geometry is moved, recalculation is necessary. Off disables

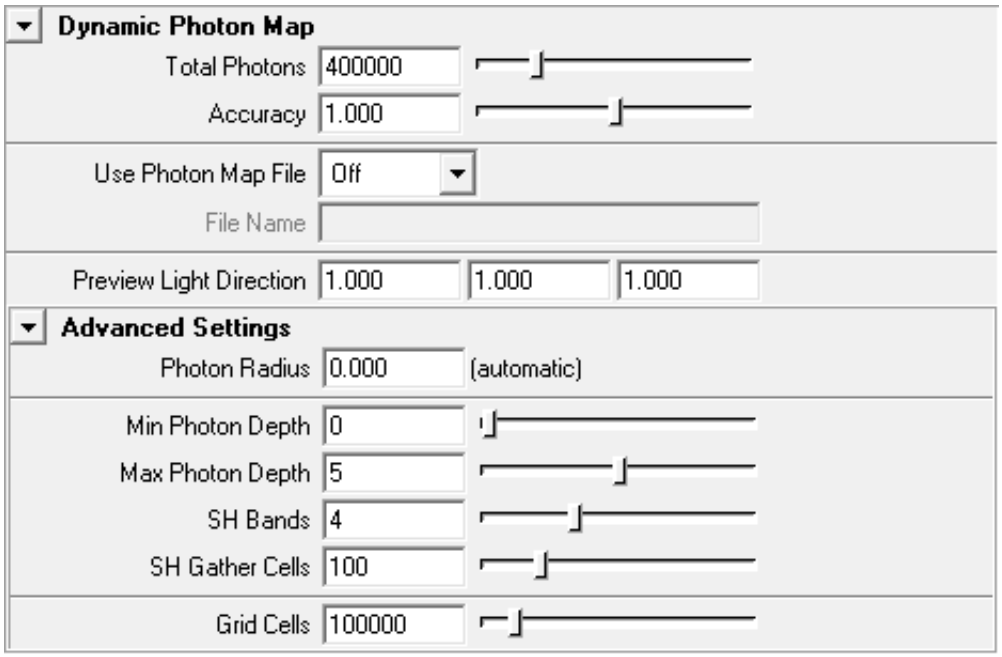

Figure 1.9: Dynamic Photon Map settings

file usage. Overwrite creates a new photon map file, overwriting any existing file. Reuse loads the photon map from a previously created file.

File Name

Sets the filename of the dynamic photon map file.

Preview Light Direction

The preview of the Dynamic Photon Map is only meant to be used to fine tune the intensity of the indirect dynamic light for a baking pass, and will not produce any quality results. In a typical case, you turn off any other lights in the scene, giving you a rough picture of the influence you'll get from the photon map (e.g. when baking color bleeding to a PTM). The Dynamic Photon Map contains photons from any possible direction; the Preview Light Direction specifies one such direction for evaluation of the map.

Photon Radius

Sets the maximum search radius to use when searching for photons during lighting calculations for global illumination effects. If the Photon Radius is set to r, photons within a sphere of radius r around the sample point will be used to calculate the light. The radius is given in Maya units. If the radius is set to 0.0, an automatic value will be used.

Min Photon Depth

Sets the minimum number of bounces a global photon must take before it can be stored in the photon map.

Max Photon Depth

Sets the maximum number of bounces a global photon can take before it is absorbed.

SH Bands

Number of bands to use for the spherical harmonics function in each grid cell. A higher number of bands will allow for more high frequent illumination data.

SH Gather Cells

Incoming photons are stored on their emit direction. This setting determines the number of buckets (each associated with a direction) that will be used for photon collection. Again, a higher number allow for information of higher frequency.

Grid Cells

Sets the number of grid cells when for the Dynamic Photon Map data structure. A higher number gives higher accuracy but longer rendering time. If the scene is very complex, a higher value may be necessary.

1.3.7 Caustics Photon Map

Caustics effects are handled with a special photon map in Turtle. You must explicitly emit caustics photons from light sources. Figure 1.10 shows the **Caustics Photon Map** interface in the Maya Render Settings window.

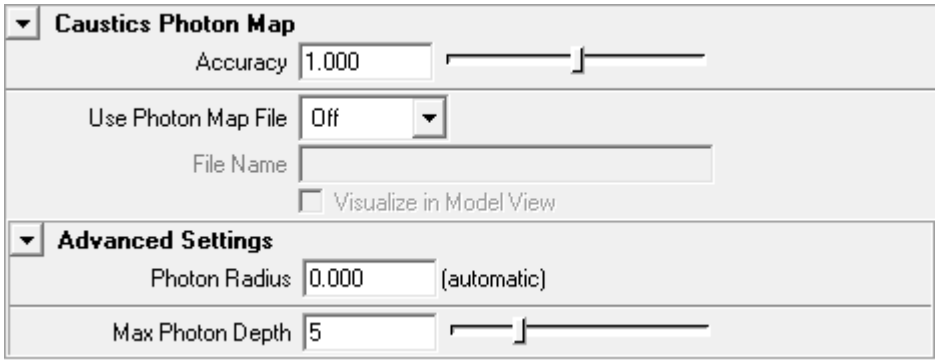

Figure 1.10: Caustics Photon Map settings

Accuracy

Influences the number of photons to use when calculating caustics effects, a higher number take more photons into account. Default value is 1.0.

Use Photon Map File

Enables usage of photon map files. By saving the photon map to a file, it can be reused in successive renderings. The file will be placed under *turtle/photonMaps* in the current project directory. Note that if lights or geometry is moved, recalculation is necessary.

- **Off**: Disables file usage.
- **Overwrite**: Creates a new photon map file, overwriting any existing file.
- **Reuse**: Loads the photon map from a previously created file.

File Name

Sets the filename of the caustics photon map file.

Visualize in Model View

Visualize the stored caustic photons as a point cloud in the Maya viewport. You can change point size and color in the attributes of the generated point cloud.

Photon Radius

Sets the maximum search radius to use when searching for photons during lighting calculations for caustics effects. If the Photon Radius is set to r, photons within a sphere of radius *r* around the sample point will be used to calculate the light. The radius is given in Maya units.

Max Photon Depth

Sets the maximum number of bounces a caustics photon can take before it is absorbed.

1.3.8 Advanced Settings

Advanced GI Settings are controlled from here. Figure 1.11 shows what the interface looks like.

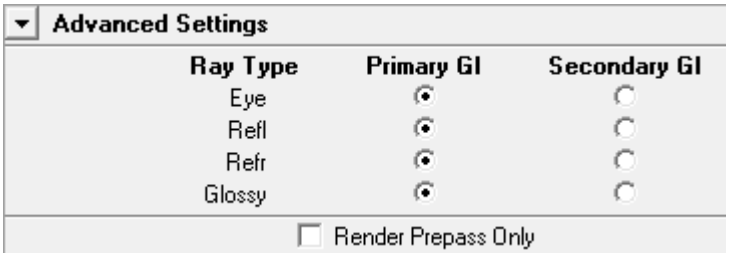

Figure 1.11: Advanced GI settings

Ray Selection

Here you can set which GI solution each ray type should see. This can be used for optimization on ray type. For instance Glossy reflections generally does well with using the photon map directly instead of final gather.

Render Prepass Only

When enabling this, the rendering terminates after the GI prepass is done. This is useful if you want to distribute renderings between different computers. Typically you run through an animation perhaps, each 10th frame, on one computer with "Reuse and append" file mode for the GI. You can then distribute this GI solution to a cluster of render nodes to avoid flickering GI in the animation. It can also be used to avoid seams between tiles when using many computers to render one frame.

1.4 Environment

The environment tab collects all the details for specifying the surrounding environment of your scene. You can set a specific environment for Global Illumination calculations, specify an environment for reflective or refractive objects and the background itself.

The Environment can be set to

- **None** - Black
- **Camera Background** - Use the environment from the current camera
- **Sky Light** - Use a specified color/texture for the entire environment
- **Image Based Lighting** - Use a specified LDR/HDR image as environment

Figure 1.12 shows the **Environment** interface in the Maya Render Settings window.

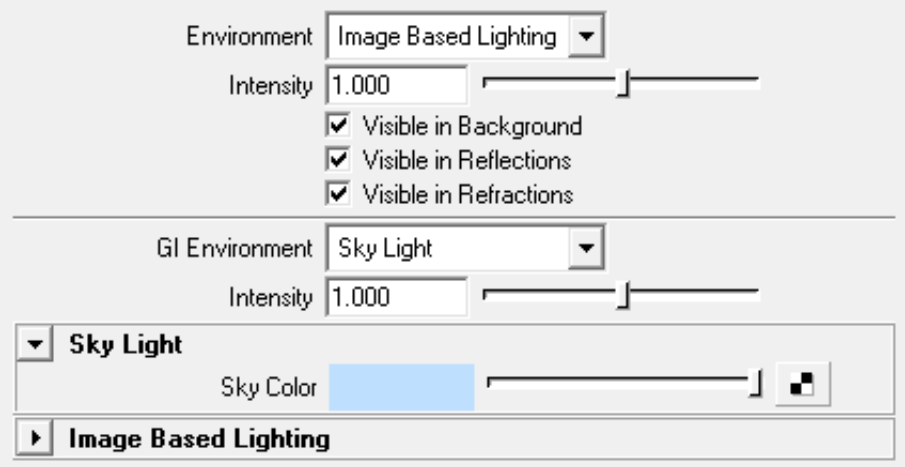

Figure 1.12: Image Based Lighting settings

Intensity

Control the intensity of the specified environment.

Visible in Background

Show the environment in the background of rendered frames. The alpha channel will still store transparent in areas with no geometry coverage.

Visible in Reflections

Show the environment in reflective surfaces.

Visible in Refractions

Show the environment in refractive surfaces.

1.4.1 Sky Light

Sky Color

Specify a color for the Sky Light environment. Suitable to select a constant sky color, or plug-in a procedural sky like a ramp or a more complex shader network.

1.4.2 Image Based Lighting (IBL)

Image Based Lighting is both an algorithm to light a scene with a LDR/HDR image, and a simple environment for images. The IBL is by default set to act as a simple environment. Figure 1.13 shows the **IBL** interface in the Maya Render Settings window.

Image File

The image file to be used. You can use most image types, but you should of course use an HDRI image if you can, like EXR or HDR.

Turn Light Dome

The sphere that the image is projected on can be rotated around the up axis. The amount of rotation is given in degrees.

Swap Y,Z

Swap the up axis. By default Y is up.

Visualize

The HDR image will be visualized for reference as a sphere inside the Maya viewport. This sphere can only be turned around the up axis and is connected to the **Turn Light Dome** option. It is not visible in the rendered image and should be disabled before rendering.

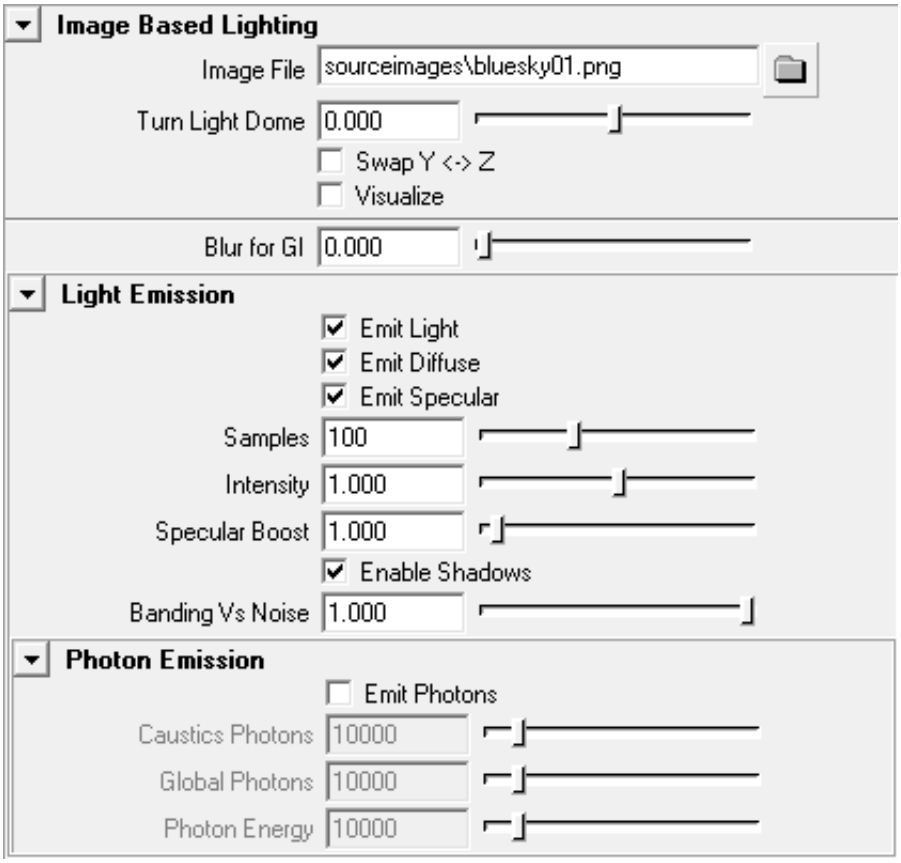

Figure 1.13: Image Based Lighting settings

Blur for GI

Pre-blur the environment image for Global Illumination calculations. Can help a lot to reduce noise and flicker in images rendered with Final Gather. May increase render time as it is blurred at render time. It is always cheaper to pre-blur the image itself in an external application before loading it into Turtle.

Emit Light

Emit direct light from the environment by creating a number of virtual directional lights simulating the environment image. An alternative to this, which often is faster, is to use Final Gather instead so that the final gather rays extract light from the environment map.

Emit Diffuse

To remove diffuse lighting from IBL, disable this check box.

Emit Specular

To remove specular highlights from IBL, disable this check box.

Samples

Sets the number of virtual directional lights created if Emit Light is enabled. This will affect how soft the shadows will be, as well as the general lighting. A higher number of samples, gives better shadows and lighting but increases the render time.

Intensity

Set the intensity of the lighting.

Specular Boost

Further tweak the intensity by boosting the specular component.

Enable Shadows

To create shadows, use this checkbox (also dependent on **Ray Tracing** $\rightarrow$ **Shadows**).

Banding Vs Noise

Control the appearance of the shadows, banded shadows look more aliased, but noisy shadows flicker more in animations.

Emit Photons

Enables photon emission from the environment map.

Caustics Photons

Number of caustics photons to be emitted into the scene.

Global Photons

Number of global photons to be emitted into the scene.

Photon Energy

The total photon energy.

1.5 Options

The **Options** tab hold render settings that you don't need to adjust so often, e.g. settings for memory and performance, framebuffer settings and various overrides.

1.5.1 Memory and Performance

Figure 1.14 shows the **Memory and Performance** interface in the Maya Render Settings window.

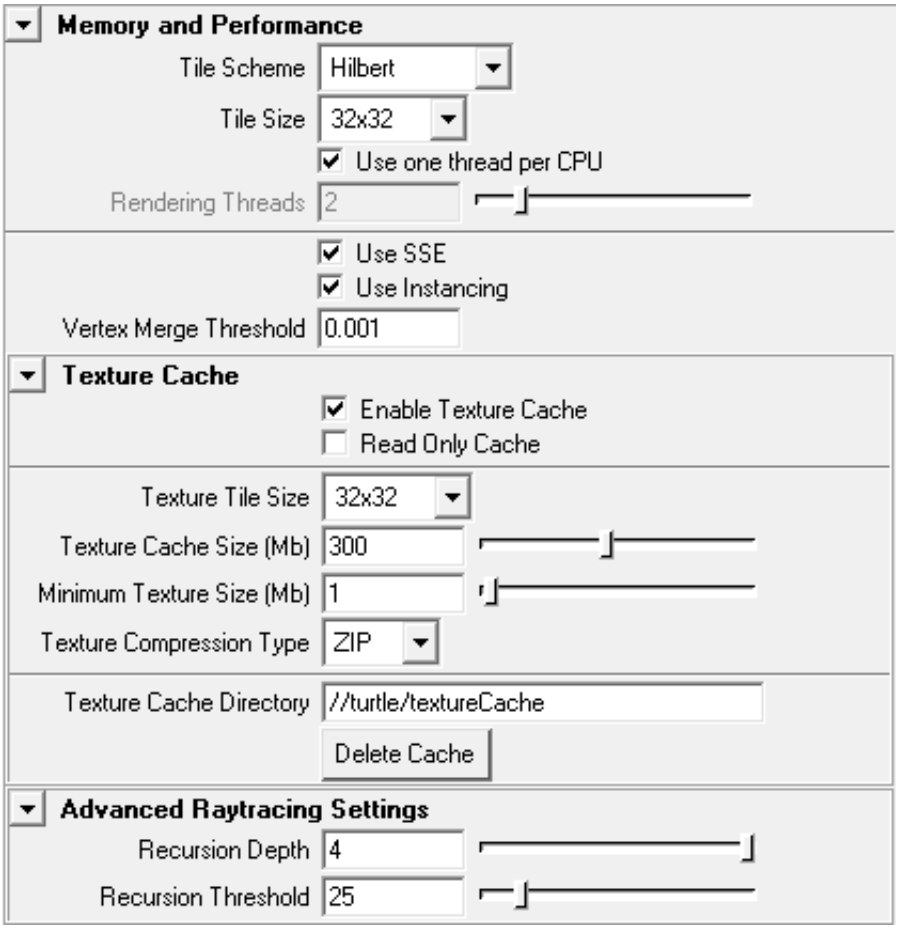

Figure 1.14: Memory and Performance Settings

Tile Scheme

Different ways for Turtle to distribute tiles over the image plane.

- **Hilbert**: A way for Turtle to achieve maximum coherence, e.g. the fastest rendering time possible.
- **Left to Right**: Render from left to right.
- **Concentric**: Starts in the middle and renders outward in a spiral.
- **Random**: A good way to get an early feel for the whole picture without rendering everything.

Tile Size

A smaller tile gives better ray tracing coherence but more inefficient shading. There is no "best setting" for all scenes.

Use one thread per CPU

Let Turtle automatically detect the number of available processors.

Rendering Threads

Manually set the number of rendering threads.

Use SSE

Turtle uses specialized vector instructions to speed up the rendering. The cost is higher memory usage. Turn off SSE/AltiVec if Turtle starts swapping memory.

Use Instancing

Enables use of instanced geometry, which saves memory. If disabled all instanced shapes will get a separate copy of the geometry, which might decrease render time, but increases memory usage.

Vertex Merge Threshold

Sets the threshold used for merging multiple face vertices into single vertices when loading scene geometry. If the difference of vertices is smaller that this threshold they are merged. This is used to optimize meshes data and decrease memory usage. The default value 0.001 should rarely be altered. Can be set to 0.0 to disable mesh optimization.

Enable Texture Cache

Selects whether the texture cache should be used or not. The texture cache works by storing textures in a format supporting random access making it possible to load them more efficiently. In scenes using large textures this will make the memory footprint of textures smaller and the texture loading faster. If disk space is an issue, the texture cache may be a problem since it stores copies of all textures.

Read Only Cache

If this box is checked, textures already in the cache will be used, but modified or added files won't be cached.

Texture Tile Size

Sets the size of tiles in the textures. Smaller tiles will make the cache more memory efficient, but somewhat slower.

Texture Cache Size

Sets the maximum memory usage for the texture cache. Note it's the memory used by the renderer, not physically on the disk.

Minimum Texture Size

Sets the minimum memory usage for a texture for it to be cached. It may be unnecessary to cache small textures that won't use very much memory anyway.

Texture Compression Type

Sets what compression to use for textures stored on disk. The different compression types has different properties:

- None: no compression, gives high performance and large disk usage.
- RLE: Run Length Encoding. Gives good compression on images with large single colored areas.
- PIZ: a good compression type if using large tiles.
- ZIP: a versatile compression giving good results for texture maps in general.

To learn more about the different compression methods, visit the open exr homepage (http://www.openexr.com)

Texture Cache Directory

Sets the directory in which the textures will be stored. A directory starting with "//" will refer to a directory in the project path.

Delete Cache

Pressing this button will delete all files in the texture cache.

Recursion Depth

Decides how deep the Ray Tracing Acceleration Data Structure can recurse.

Recursion Threshold

Decides how many triangles that can reside in a leaf before it is split up. The Recursion Depth has precedence over the threshold. A leaf at max depth will never be split. The Recursion Depth and Recursion Threshold are advanced settings that shouldn't be altered unless Acceleration Data Structures are second nature to you.

1.5.2 Frame Buffer

Figure 1.15 shows the **Framebuffer** interface in the Maya Render Settings window.

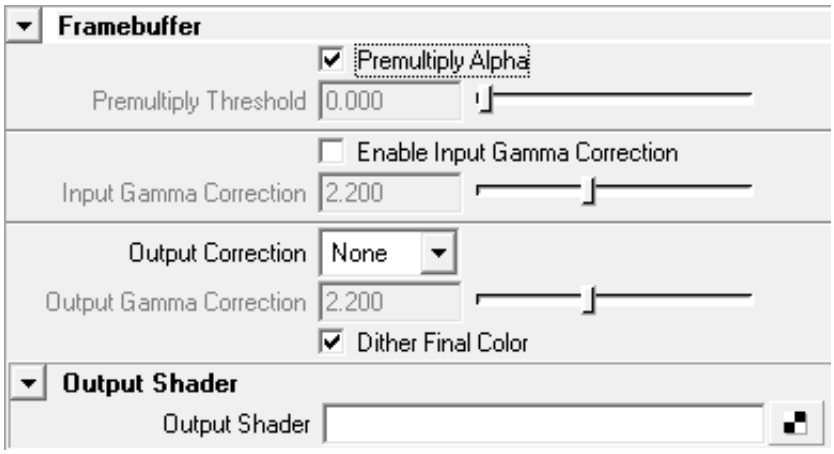

Figure 1.15: Framebuffer Settings

Premultiply Alpha

If this box is checked the alpha channel value is premultiplied into the color channel of the pixel. Note that disabling premultiply alpha gives poor result if used with environment maps and other non constant camera backgrounds. Disabling premultiply alpha can be convenient when compositing images in post.

Premultiply Threshold

This is the alpha threshold for pixels to be considered opaque enough to be "unmultiplied" when using premultiply alpha.

Enable Input Gamma Correction

Turtle renders everything in linear intensity space (not gamma corrected). This means Turtle assumes that all colors selected as inputs on Maya shaders are treated as linear, the same applies to all file textures. The output image produced by Turtle is also in linear space. If this is not desired Gamma Correction can be automatically applied to the input colors and textures, and to the output image. Example, the file textures are in gamma 2.2 and the resulting image should be in gamma 2.0. Setting input gamma to 2.2 will transform the input textures from 2.2 to 1.0 (linear space). Setting output gamma to 2.0 will transform the rendered image from gamma 1.0 (linear) to 2.0.

If some file textures are in linear space then input gamma correction can be individually disabled on the file texture nodes. HDRI file formats such as OpenEXR and Radiance HDR are always treated as linear.

Gamma correction, both input and output, applies to all render modes including Vertex Baking. The exception is Point Cloud Baking where the output file format is always considered linear, and will thus ignore any specified output gamma correction.

Gamma correction is designed to correct colors with intensities between 0.0 and 1.0 between different display devices. Maya allows you to set color values with any possible brightness, but this will lead to very unpredictable results as a relatively limited color value might still be boosted to huge values by the gamma correction. You should always try to keep your colors limited, and use **Intensity** sliders where available, as scalar values will never be gamma corrected. This ensures that the color you see on your monitor will be transferred correctly to linear color space, and still have the boost in intensity that you want.

Input Gamma Correction

Selects the gamma space that input colors are assumed to be in.

Note: HDRI images are always considered linear.

Output Correction

This dropdown selects what kind of correction will be done to the output colors.

- NONE Writes the linear colors directly
- Gamma Takes the linear colors to the gamma specified in Output Gamma Correction
- SRGB Converts the colors to SRGB when showing them. This is similar, but not identical to gamma 2.2

Output Gamma Correction

Selects the gamma space the rendered image is transformed to. This is only considered when Output Correction is set to ”Gamma”.

Note: Point Cloud files are always written out with linear color space.

Dither Final Color

Check this checkbox if you want your images to be dithered when converting them from high dynamic range float to low dynamic range integer colors.

Output Shader

The image rendered can be post processed by applying an output shader. It can only operate on single pixels, one at a time and thus cannot do any advanced operations such as blurring or multi-pixel filtering. It is primarily used for gamma correction or color channel swapping.

To use, just apply a simple color utility to the shading box. When the shading network is finished, the node **ilrOutputShaderBackendNode** must be applied to tell where the shading network should get its color information from. The **ilrOutputShaderBackendNode** is described in section 4.3.1.

A simple example of an output shader could be:

- Connect a gammaCorrectNode to the Output Shader in the Render Globals.
- Connect a ilrOutputShaderBackendNode to the Value of the gammaCorrect.
- Set up the gammaCorrectNode accordingly.
- Render and enjoy your gamma corrected image.

Note: The **ilrOutputShaderBackendNode** must be assigned in the shading network, otherwise Turtle does not insert the original image color correctly. Failure to do so will result in a black image.

1.5.3 Overrides

This section contains settings to globally override local settings. Figure 1.16 shows the **Overrides** interface in the Maya Render Settings window.

Ignore Light Links

Sometimes Turtle will have troubles with light links. This will occur when there is a lot of geometry and a lot of lights and no light links available. Check this box to force Turtle to ignore light links.

Enable Quality Limits

Soft ray traced shadows and glossy reflections tend to increase render times and generally needs a lot of tuning to get a good result at a reasonable render time. Checking this box limits the number of rays for all light sources and glossy shaders in the scene disregarding the actual values on the shaders. This is useful when rendering previews of scenes to see the overall look before starting the final render.

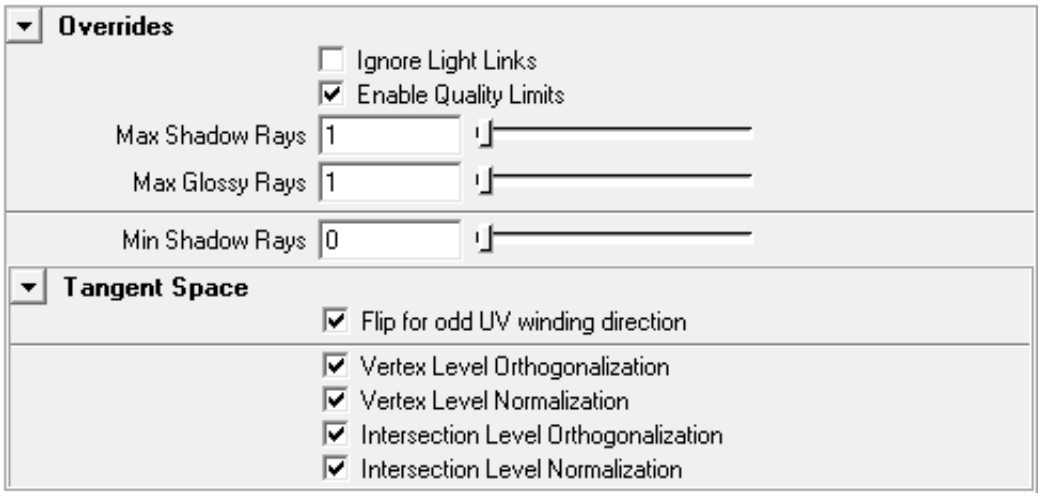

Figure 1.16: Overrides Settings

Note: The Max Shadow Rays setting cannot be used to raise the number of shadow rays above the number set in the Shadow Rays setting for any light source, only to lower the limit. For example if Max Shadow Rays is set to 60 and Shadow Rays for a light source is set to 30, there will not be more than 30 rays per point for that light source. If Max Shadow Rays is set to 15 and Shadow Rays is set to 30, no more than 15 rays will be sent.

Min Shadow Rays

Use this value to ensure that you get enough quality in soft shadows at the price of render times. The setting sets a lower limit on the number of rays sent to determine whether a point is lit by a light source or not. This will raise the minimum number of rays sent for any light sources that have a Min Shadow Rays setting lower than this value.

Note: This setting cannot be used to lower the number if Min Shadow Rays for the light source is set to a higher value. For example if Min Shadow Rays for a light source is set to 10 and this setting is set to 5, there will be at least 10 rays sent, provided that the maximum limit of rays is not reached before that. Setting this to a value higher than Max Shadow Rays under Quality Limits or Shadow Rays for the light source will not send more rays than the minimum value of those two settings for that light source.

Flip for odd UV winding direction

Using this setting will force Turtle to mirror tangent and binormal when UV has odd winding direction.

Vertex Level Orthogonalization

Orthogonalize tangent space basis vectors (tangent, binormal and normal) at every vertex.

Vertex Level Normalization

Normalize tangent space basis vectors (tangent, binormal and normal) at every vertex.

Intersection Level Orthogonalization

Orthogonalize tangent space basis vectors (tangent, binormal and normal) at every intersection point.

Intersection Level Normalization

Normalize tangent space basis vectors (tangent, binormal and normal) at every intersection point.

1.5.4 Output Verbosity

Figure 1.17 shows the **Output Verbosity** interface in the Maya Render Settings window.

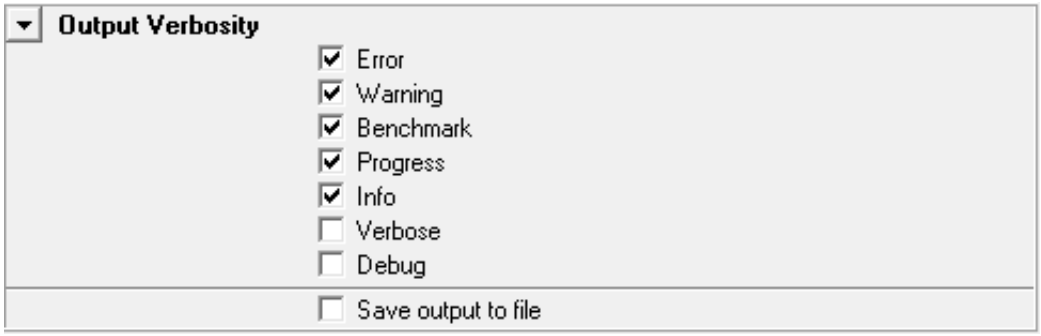

Figure 1.17: Output Verbosity Settings

Error

Prints error messages.

Warning

Prints warning messages.

Benchmark

Prints benchmark information.

Progress

Prints progress information.

Info

Prints info messages.

Verbose

Prints verbose information.

Debug

Prints debug information. Used for development purposes.

Save output to file

If enabled all output will be saved to the file *MAYA_PROJECT_PATH\Turtle\temp\debug.out*.

1.6 Turtle settings on Maya nodes

Turtle can render most standard Maya nodes using their native settings. In many cases those settings are sufficient but for a couple of Maya nodes Turtle also provide the opportunity to add extra settings to further affect their behavior. The nodes that can be complemented with extra Turtle attributes are geometry objects, lights, cameras, surface shaders, displacement shaders, file textures, shading engines and render layers. To add or remove extra Turtle attributes on these nodes simply open up the attribute editor for the selected node. From the menu **TURTLE** select what to add or remove. The extra Turtle settings can then be found in the separate Turtle rollout in attribute editor. See figure 1.18
A **Turtle attribute spread sheet** is also available from the **TURTLE** menu. The spreadsheet lists Turtle attributes for all objects in the current selection. This lets you edit the values of many attributes on many nodes at the same time.

1.7 Geometry Object Settings

See figure 1.19.

1.7.1 Subdivision Settings

To use subdivision, check the Render as subdivision surface box located in the Turtle rollout in the shape attributes on your polygon object.
Turtle supports polygons with a maximum of 5 edges.

Subdivision Depth

Controls the amount of subdivision that occurs. A higher value produces more polygons and a better result, but will also use more memory and increase render times.

Sharp Corners

Do not smooth the corners of an open mesh. A sphere is a closed mesh, a plane is an open mesh.

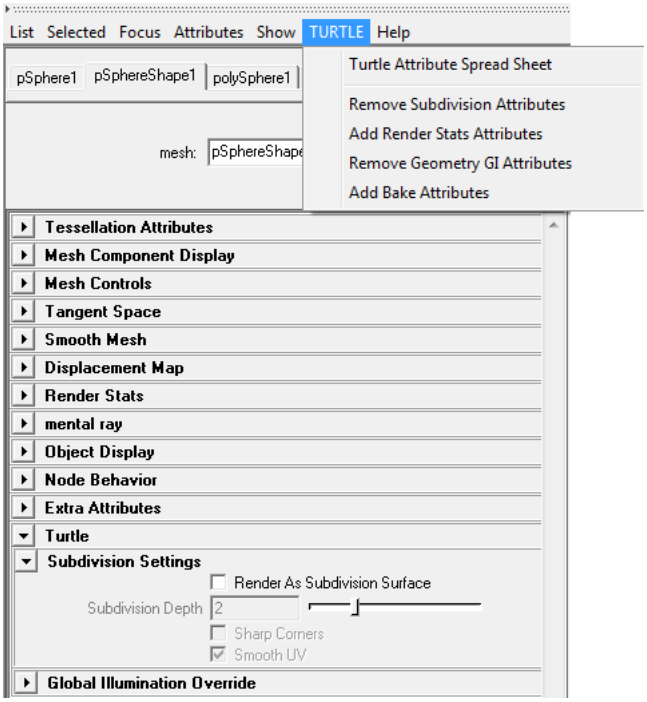

Figure 1.18: Turtle settings in attribute editor

Smooth UV

Checkbox for controlling UV-smoothing. This is enabled per default.

1.7.2 Renderstats

This section sets general parameters for the objects interaction with light.

Smooth Normals

The normals of a mesh can be smoothed at render time. The normals are then calculated as the average of all vertex normals at the same vertex. A threshold can be used to only include normals which have a similar direction. The threshold is given as the maximum angle in degrees, between two normals, under which they will be averaged. This feature is useful in a variety of situations, including:

- When using Pre-Tesselated displacement you often need a very high tesselation setting to get a smooth looking surface. By smoothing the normals on faces that doesn't have too sharp edges, you can get a smooth looking surface with much lower tesselation settings, hence saving both memory and render time.
- When calculating surface transfer from a mesh with face normals you will not get a continuous sampling since the sampling is performed along the normals of the mesh. So by using the Smooth Normals feature you can get a continuous sampling without having to alter the actual normals on the model.

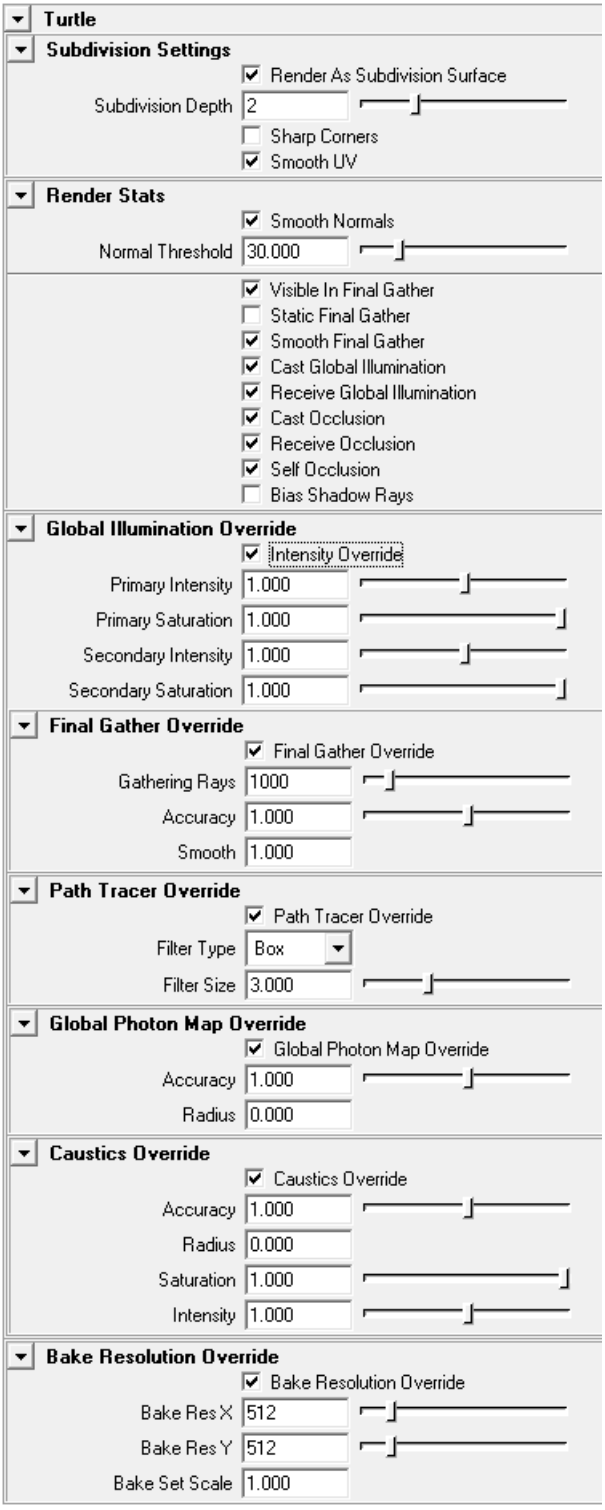

Figure 1.19: Geometry object settings

Visible In Final Gather

Let the object affect final gather calculations.

Static Final Gather

If enabled, the object will keep its final gather samples in RAM-memory between successive frames in an animation.

Smooth Final Gather

If enabled, the final gather solution will be smoothed for this object. If unchecked, the final gather smoothing will be disabled, which might save render time in some situations.

Cast Global Illumination

If enabled, the object will reflect indirect light.

Receive Global Illumination

If enabled, the object will receive indirect light.

Cast Occlusion

If disabled, the object will be invisible for occlusion rays. Therefore, it will not cast any occlusion on other objects when the ilrOccSampler is used.

Receive Occlusion

If disabled, the object will not receive any occlusion from other objects when the ilrOccSampler is used.

Bias Shadow Rays

Enable Bias Shadow Rays to reduce shadow line artifacts on low-resolution geometry, see figure 1.20.

1.7.3 Global Illumination Settings

This section contains override settings for global illumination.

Global Illumination Override

Checking this box allows you to override the settings of the Global Photon Map attributes **Photon Accuracy**, **Photon Radius**, **Saturation** and **Intensity** for a particular object. For a complete explanation of these attributes see **Global Photon Map**, section 1.3.5.

Final Gather Override

Checking this box allows you to override the settings of the final gather attributes **Gathering Rays**, **Accuracy**, **Smooth**, **Saturation** and **Intensity** for a particular object. For a complete explanation of these attributes see **Final Gather**, section 1.3.2.

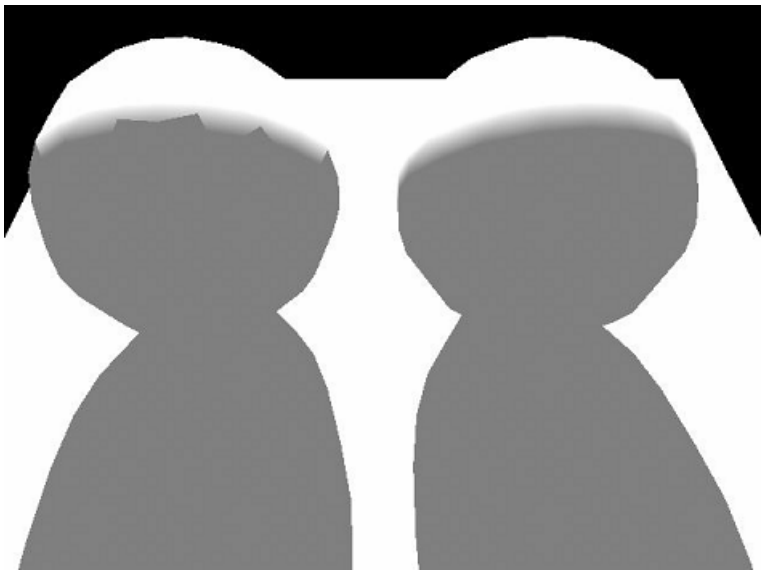

Figure 1.20: Ray-traced shadows without and with Bias Shadow Rays

Caustics Override

Checking this box allows you to override the settings of the caustics attributes **Photon Accuracy**, **Photon Radius**, **Saturation** and **Intensity** for a particular object. For a complete explanation of these attributes see **Caustics**, section 1.3.6.

1.7.4 Bake Resolution Override

Checking this box allows you to override the settings of the texture baking attributes **Bake Res X**, **Bake Res Y** and **Bake Set Scale** for a particular object. For a complete explanation of these attributes see **Baking Reference**, chapter 2.

1.8 Light Settings

1.8.1 Intensity Settings

Primary Intensity

Scales the light intensity for primary rays. Can be used to adjust or turn on/off the amount of light this light source emits on surfaces visible from the camera.

Reflection Intensity

Scales the light intensity for reflection rays. Can be used to adjust or turn on/off the amount of light this light source emits on surfaces visible in reflection.

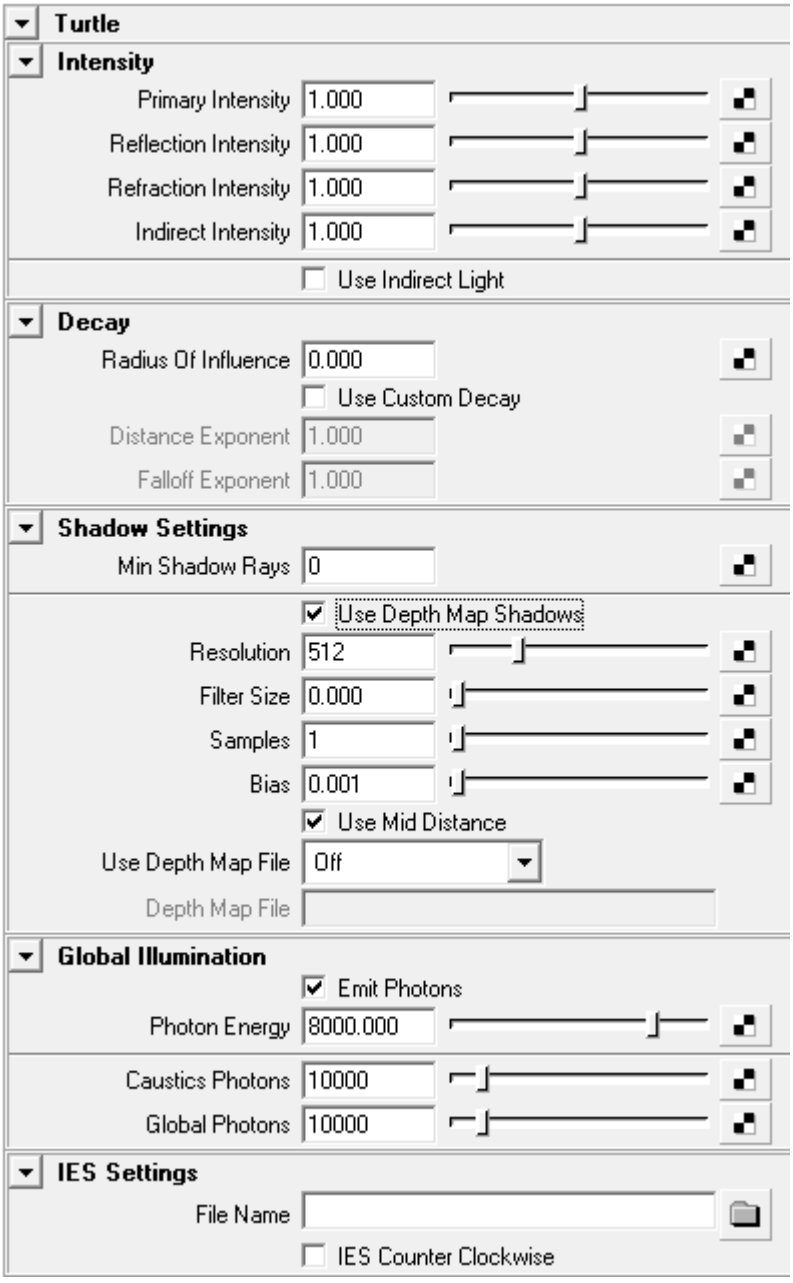

Figure 1.21: Light Settings

Refraction Intensity

Scales the light intensity for refraction rays. Can be used to adjust or turn on/off the amount of light this light source emits on surfaces visible in refractions.

Indirect Intensity

Scales the light intensity for final gather rays. Can be used to adjust or turn on/off the amount of indirect light this light source emits.

Use Indirect Light

Any light source can now be turned into an indirect light. Behaves like a normal light, but instead of the standard light shader it uses the indirect (GI) solution as light shader. This way you can light link the GI, saturate the GI color, shape the GI in form of a spotlight e.t.c. Basically all controls you have on the direct light sources can be used to control the GI solution.

1.8.2 Decay Settings

Radius Of Influence

Sets a radius of influence for this light source. Inside the sphere spanned by the radius light is emitted as usual, but outside the sphere no light is emitted. This will also optimize shadow calculations since no shadow rays will be cast from surfaces outside the sphere.

Use Custom Decay

Enable this if you want a light decay that is a function of the radius of influence. The following formula will then be used to calculate the illumination:

$$illumination = color * intensity * [1.0 - (distance/radius)^d]^f$$

Distance Exponent

Controls the d exponent in the formula above. Can be used to adjust the shape of the light attenuation.

Falloff Exponent

Controls the f exponent in the decay formula above. Can be used to adjust the shape of the light attenuation.

1.8.3 Shadow Settings

Min Shadow Rays

Sets the minimum number of shadow rays used to determine the influence of this light source a any point. Use this setting to ensure that you get enough quality in the soft shadows for this light source at the price of render times. If this is set to zero, Turtle determines a good value for this setting, otherwise the number specified is used. Regardless of the value specified, there will never be more rays sent than specified in the Shadow Rays setting for

the light source or more than the Max Shadow Rays setting specified in the Overrides section of the Render Settings for Turtle.

Use Depth Map Shadows

Enables Turtles Depth Map Shadows. When this checkbox is enabled, Turtle will use the settings below for shadow calculation. If this checkbox is disabled and Maya's Depth Map Shadows are enabled, Turtle will calculate shadows based on Maya's Depth Map Shadows settings.

Resolution

Sets the resolution of the depth map used for calculating shadows. A higher resolution will give less aliasing but consume more memory. If a very soft shadow is desired, set a low resolution and use a large filter size.

Filter Size

The size of the blur filter kernel. A larger value will give a softer shadow. If a low resolution is set, a larger filter size is needed to reduce aliasing.

Samples

Sets the number of samples to use then applying the filter to get soft shadows. A higher value will give less noise but longer rendering time.

Bias

The bias used to reduce self-shadowing artifacts. If Mid Distance is used, this value can often be ignored.

Use Mid Distance

This is used to reduce self-shadowing artifacts. When this is enabled the average distance between the first and the second intersection will be stored in the depth map.

Use Depth Map File

Controls the usage of depth map files. By saving the depth map to a file it can be reused in successive renderings. Note that if the light or shadow casting objects are moved recalculation is necessary. The file is written to the folder *turtle\depthMaps* in the current Maya project directory.

- **Off**: Disables file usage.
- **Overwrite**: Creates a new depth map file, overwriting any existing file. Reuse loads the depth map from a previously created file.
- **Reuse and append**: Loads the depth map from a file, but also writes back new depth map tiles generated during rendering.

Depth Map File

Sets the file name for the depth map file.

1.8.4 Global Illumination Settings

Contains settings for emitting photons.

Emit Photons

Controls whether the light is a photon emitter or not.

Photon Energy

The amount of photon energy emitted from the light.

Caustics Photons

Number of caustic photons emitted from the light.

Global Photons

Number of Global Illumination photons emitted from the light.

1.8.5 IES Settings

Turtle has support for light profiles on Maya point lights. With light profiles, you can easily add real-life lights to your scene for greater realism. A light profile is a light intensity distribution specified by a texture map.

File Name

The file name of the IES Light profile to use. Turtle can read so called IES-files. Many manufacturers of lighting fixtures record photometric light intensity measurements of their products in IES-files and provide these for visualization purposes on their websites. Thus, there is a large number of interesting lights readily available on the internet. Some examples of IESfiles can be found at http://www.daybrite.com/day-brite/photometry.cfm .

IES Counter Clockwise

Enable this to reverse the horizontal angle of the light profile.

1.9 Camera Settings

Environment Camera

Under the Turtle tab on the camera there is an option to make the camera an 'Environment Camera'. When enabled the camera will render an environment map instead of the usual camera view. The environment will be rendered with the camera looking in the positive Z direction. The environment map will be created around the Camera Origin. The 'Type' list box selects one of the following environment map types:

- **Cube Map**: The cube map rendered with Turtle is in 'Vertical Cross' format. It is at the moment not compatible with the Maya 'Env Cube' node since Maya uses six different textures for the sides of the cube map.
- **Ball Map**: The ball map is in the same format as the Maya 'Env Ball' node expects. The ball map is not a very good format for environment maps, since the distribution of samples is very irregular.
- **Spherical Map (LatLong)**: The spherical map (also called Latitude Longitude map) is in the same format as the Maya 'Env Sphere' expects. This is the preferred format for creating and using environment maps with Turtle.

1.10 Material Settings

See figure 1.22.

1.10.1 Glossy Reflections and Refractions

Glossy reflection/refraction simulates non perfect (blurry) reflections and refractions. The Glossy Reflection/Refraction settings are available for the Maya materials Phong, PhongE, Blinn and Anisotropic and are located in the Turtle roll-out in the material settings. The settings differ a bit between the anisotropic and the other shaders.

Glossy Reflections / Glossy Refractions

Enable Glossy Reflections for this material.

Maximum Rays / Maximum Refr. Rays

The maximum number of reflection rays to trace for an intersections point. A higher value gives less noise in the resulting image.

Sharpness / Refr. Blur Angle

How sharp the reflections are. A lower Sharpness value gives blurrier reflections. A higher Refr. Blur Angle value gives blurrier refractions.

Anisotropic / Refr. Anisotropic

Makes the glossy effects anisotropic, spreading out the reflected or refracted image nonuniformly.

Use Maya Shader Angle Settings (Anisotropic Shader only)

Import the rotation of the anisotropic reflection/refraction from the shader settings to align them with the specular highlights.

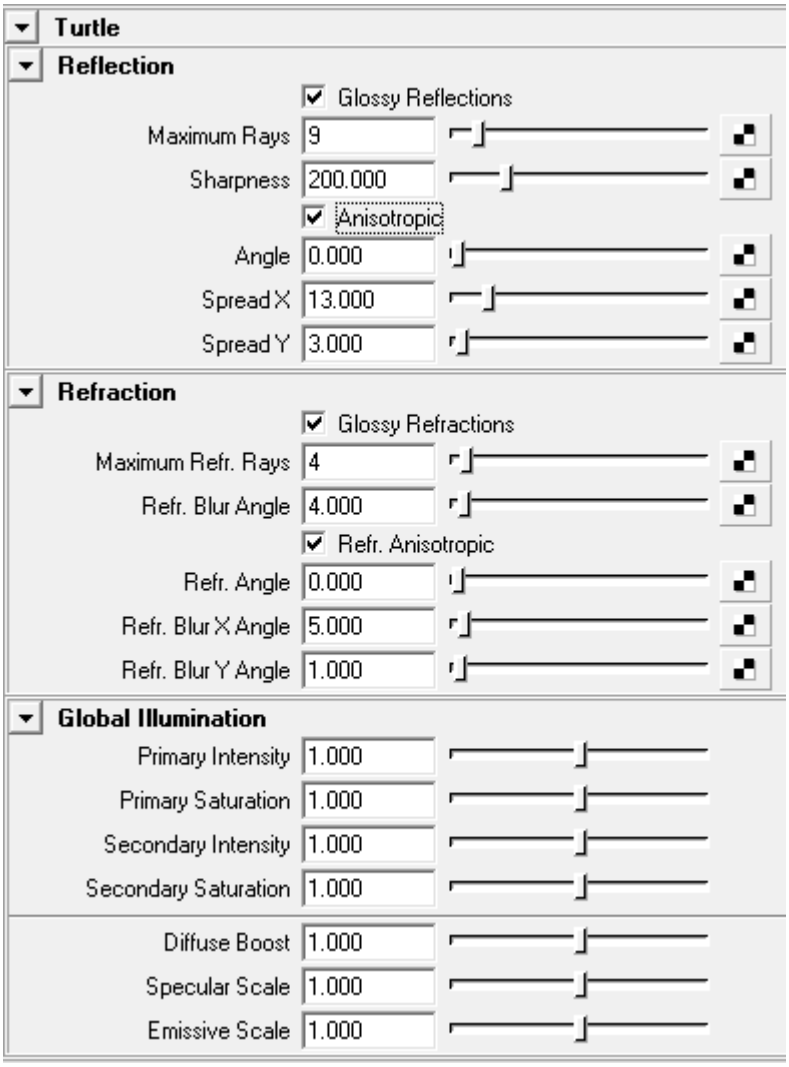

Figure 1.22: Material Settings

Angle / Refr. Angle

Specifies the rotation of anisotropic reflections/refractions, note that it uses 360 degrees for a 180 degrees rotation to make the settings compatible with the angle settings for the anisotropic shader in Maya.

SpreadX / Refr. SpreadX

How much the reflections/refractions will spread out in the tangent direction.

SpreadY / Refr. SpreadY

How much the reflections/refractions will spread out perpendicular to the tangent.

Understanding Sharpness, Blur Angle and Spread

The tangents of the surface defines the direction in which the surface will look brushed when using anisotropic glossy reflections. Having a large value for spreadX will spread out the reflections/refractions along the tangents and a large value of spreadY will spread it out perpendicular to the tangents. If they are the same, the material won't look anisotropic at all. The spread parameters are affected by the sharpness value. This means that it's often a good idea to increase the sharpness vs. decrease the blur angle when turning on anisotropic reflections/refractions since it affects both spreadX and spreadY.

Note: A material with an inconsistent UV mapping will give a poor result.

1.10.2 Global Illumination

Primary Intensity

Scales the intensity of the first GI bounce for this material. Note that there is also a global control for this in Render Settings - Global Illumination - Color Balance.

Primary Saturation

Scales the saturation of the first GI bounce for this material. Note that there is also a global control for this in Render Settings - Global Illumination - Color Balance.

Secondary Intensity

Scales the intensity of secondary GI bounces for this material. Note that there is also a global control for this in Render Settings - Global Illumination - Color Balance.

Secondary Saturation

Scales the saturation of secondary GI bounces for this material. Note that there is also a global control for this in Render Settings - Global Illumination - Color Balance.

Diffuse Boost

If your material has very dark colors/textures, light will fall off very quickly in the Global Illumination. Increasing Diffuse Boost will push up the dark diffuse values to make light spread further. This only affects how the material is seen by the Global Illumination system, direct lighting is not affected. Note that there is also a global control for this in Render Settings - Global Illumination - Color Balance.

Emissive Scale

This setting will scale the emissive contribution (Incandecense) of the material. This only affects how the material is seen by the Global Illumination system, direct lighting is not affected. Note that there is also a global control for this in Render Settings - Global Illumination - Color Balance.

Specular Scale

This setting will scale the specular effects in the indirect light. This only affects how the material is seen by the Global Illumination system, direct lighting is not affected. Note that there is also a global control for this in Render Settings - Global Illumination - Color Balance.

1.11 Displacement Mapping

1.11.1 Common Displacement Settings

Method

There are two different methods for doing displacement mapping in Turtle; **Rendertime Micro Triangles** and **Pre-Tesselation**. Both methods are described below. Which method to use and the settings for each method are found under the Turtle roll-out on the displacement shader (See figure 1.23). The following two settings are used by both methods:

Displacement Scale

Controls the height of the displacement map. A higher value gives more displacement. This attribute is keyable for animation.

Smooth Base Mesh

Adds a smoothing value to the displacement value of a point to improve the smoothness of the base mesh and thus ensure that no base triangle borders are visible on the final render. Smooth Base Mesh works on individual polygons. If a "global" smooth look is desired, render the object as a subdivision surface.

1.11.2 Rendertime Micro Triangles

This method displaces the geometry at render time, and only creates micro triangles as they are needed when a ray hits an object. Hence it's very memory efficient. It also uses a level-of-detail (LOD) approach to scale the amount of tesselation depending on the distance to the camera. This is the default method for displacement mapping.

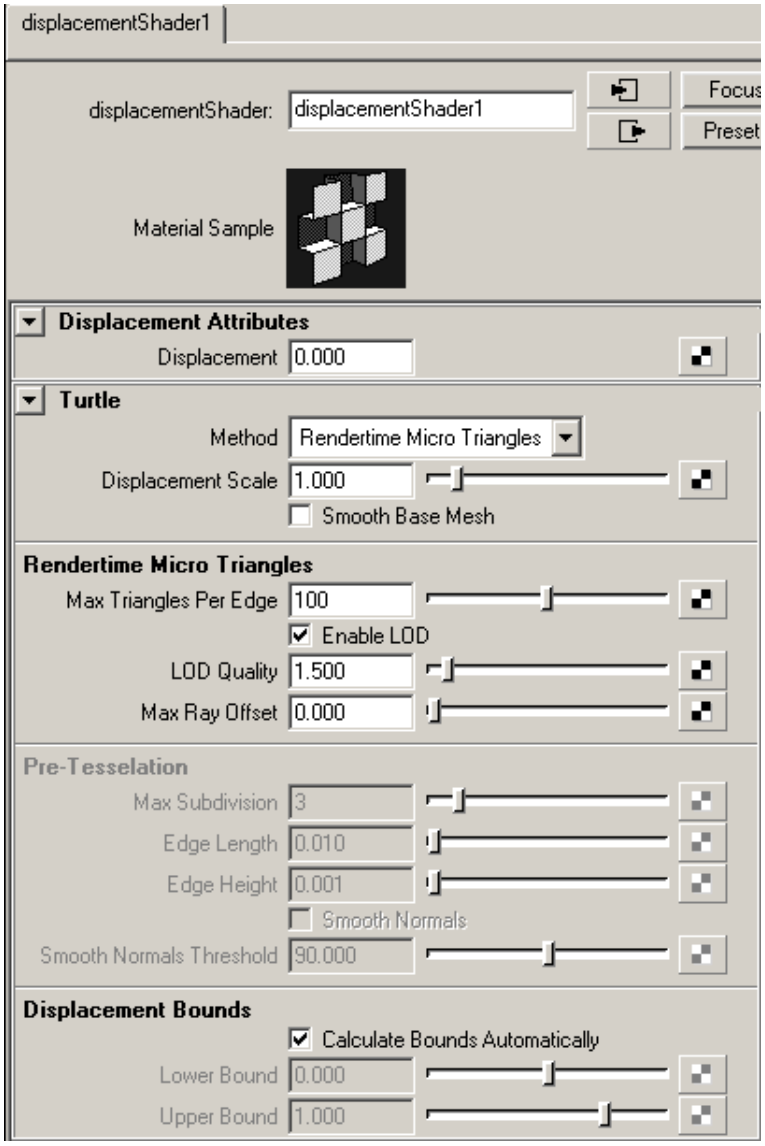

Figure 1.23: The Displacement Shader.

Max Triangles Per Edge

Controls the maximum number of micro triangles a base triangle will be divided into. If Max Triangles Per Edge is set to 10 the base triangle will be divided into 10*10 = 100 micro triangles.

Enable LOD

Enables level of detail on the displaced object. The LOD algorithm will adjust the size of the micro triangles to be less than the size of a pixel. If an object is far away, fewer micro triangles will be generated.

LOD Quality

Forces the LOD to tessellate harder (if larger than one) or more sparse (if less than one). The suggested tessellation from the LOD computation is multiplied with this value. This allows you to trim the LOD feature, if needed.

Max Ray Offset

When combining this method with final gathering and/or occlusion, this parameter has to be adjusted to avoid self-shadowing artifacts. A good rule-of-thumb is to set this value to half the displacement scale value if you are using both positive and negative displacement. If the entire displacement map is positive, leave the ray offset at zero.

1.11.3 Pre-Tesselation

This method tesselates the object into micro triangles before the rendering starts (similar to how Maya and mental ray handles displacement). Since all triangles are kept in memory it can be very memory consuming for highly detailed objects. However the tesselation can be done adaptively to make sure as few triangles as possible are produced.

Max Subdivision

Sets the initial level of subdivision to use when creating micro triangles from a base triangle. The base triangle is split into four new triangles at each level, so you will get 4 to the power of Max Subdiv micro triangles for each base triangle.

Example:

Subdiv $= 3$ gives $4^3 = 64$ micro triangles per base triangle.
Subdiv $= 6$ gives $4^6 = 4096$ micro triangles per base triangle.

As can be seen in the example the micro triangles increase very rapidly with the subdivision parameter, so it should be set as low as possible.

Edge Length

This parameter can be used to stop the subdivision when a specific edge length is reached. The edge length is given as a distance in world space units (cm). If a triangle's edges are shorter then this distance, no more subdivision will be done to that triangle.

Edge Height

This parameter can be used to reduce the number of micro triangles produced. If neighbouring micro triangles have a similar displacement, they can be merged together, forming a larger single triangle. The edge height sets the maximum difference in displacement that triangles can have to be merged.

Smooth Normals

Enable Smooth Normals to interpolate normals for the displaced triangles. Not smoothing the normals may pronounce the displacement effect better.

1.11.4 Displacement Bounds

The bounds for displacement can be calculated automatically by Turtle, or you can set them manually. The default setting is to calculate the bounds automatically. However, sometimes the calculations fails to find the correct bounds which can result in displacement artifacts. By settings the bounds manually you can avoid this problem. Analyzing a shading network to find its bounds can also be quite time consuming, especially for procedural textures. So by using manual bounds you can also save a lot of time in the preprocessing phase. If you use manual bounds, make sure to set them as tight as possible, which will give better performance.

Lower Bound

Controls the minimum displacement value the shader should return. Note that if you have negative displacement this must be a negative value. If the shader returns a lower value than this it will be clamped to this value.

Upper Bound

Controls the maximum displacement value the shader should return. If the shader returns a higher value than this it will be clamped to this value.

Note: When using file textures as displacement maps, use the MipMap filtering option for better performance.

1.12 File Texture Settings

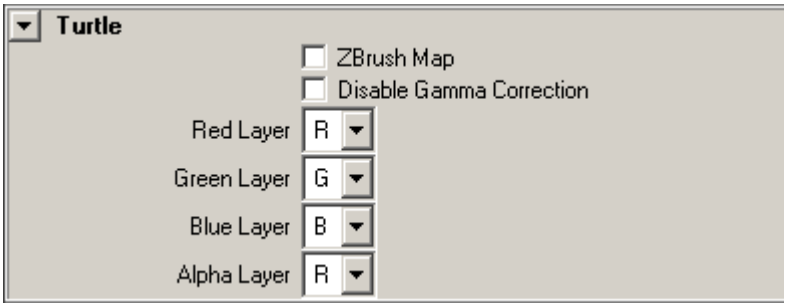

Figure 1.24: File Texture Settings

Turtle adds a few settings to the file texture node. See figure 1.24.

ZBrush Map

If loading a ZBrush map, enable this check box to ensure that the data is correctly formatted to be used in Maya.

Disable Gamma Correction

Force Turtle to disable gamma correction for the current image.

Red/Green/Blue/Alpha Layer

If the currently chosen file is an OpenEXR file, use these attributes to choose which layers to use as output.

1.13 Shading Engine Settings

Turtle adds a few settings to the shading engine. See figure 1.25.

Photon Shader

Here you can connect a photon shader for the material.

Surface Shader Override

Here you can override the shader for the material. Note that only Turtle cares about this setting so if you for instance have a CG shader which can't be rendered with Turtle, you can override what Turtle renders by connecting a shader here.

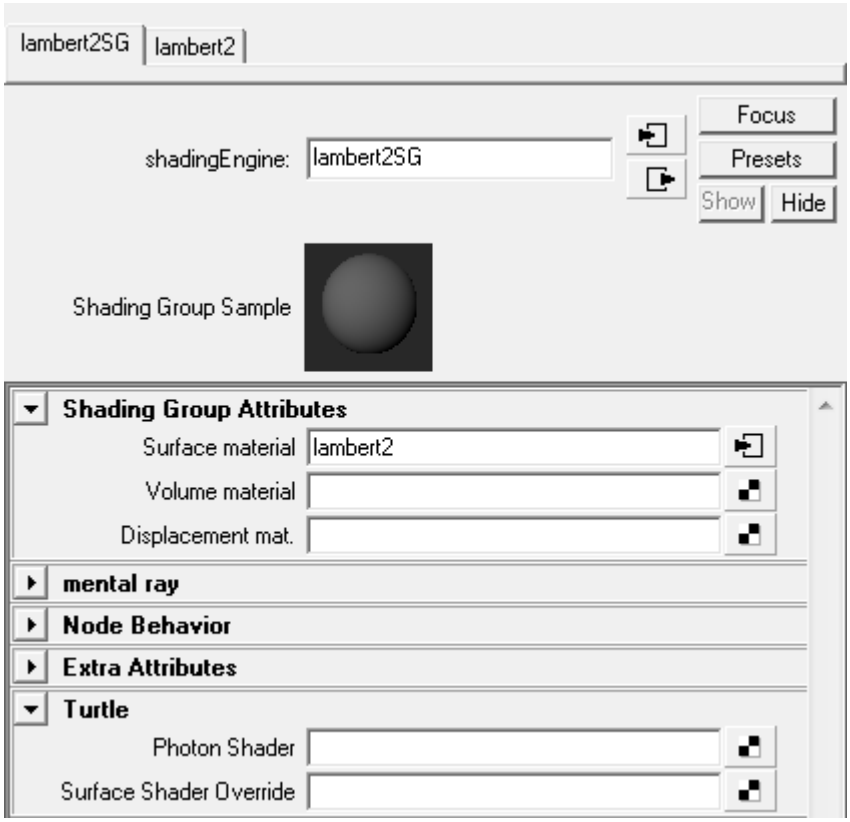

Figure 1.25: Shading Engine Overrides

1.14 Render Layer Settings

Turtle can add a number of custom render passes for Render Layers

- Illumination
- Indirect Illumination
- Albedo
- Diffuse
- Specular
- Ambient
- Incandescence
- SSS
- Reflections
- Refractions

1.15 Nurbs Tessellation

When tessellating nurbs surfaces, Turtle tries to match Maya's render tessellation. In some cases the normals can differ over a surface seam, resulting in shading artifacts. If this happens you can enable Smooth Normals on the object. See section 1.7.2 **Smooth Normals** for a description of this feature.

1.16 IBL Light Rig Editor

The IBL Light Rig Editor is used to create a spherical rig of directional lights to represent environment lighting. It takes an environment shader, like for example an envBall, as input to calculate the colors and distribution of the lights. The settings in IBL Light Rig Editor has a lot in common with Image Based Lighting in the Environment settings.

1.16.1 IBL Light Rig Editor Options

Light Rig Name

The name of the new light rig to be created.

Environment Shader

The environment shader to be used like for example an evnBall or envCube.

Samples

Sets the number of directional lights to be created. This will affect how soft the shadows will be, as well as the general lighting. A higher number of samples, gives better shadows and lighting but increases the render time.

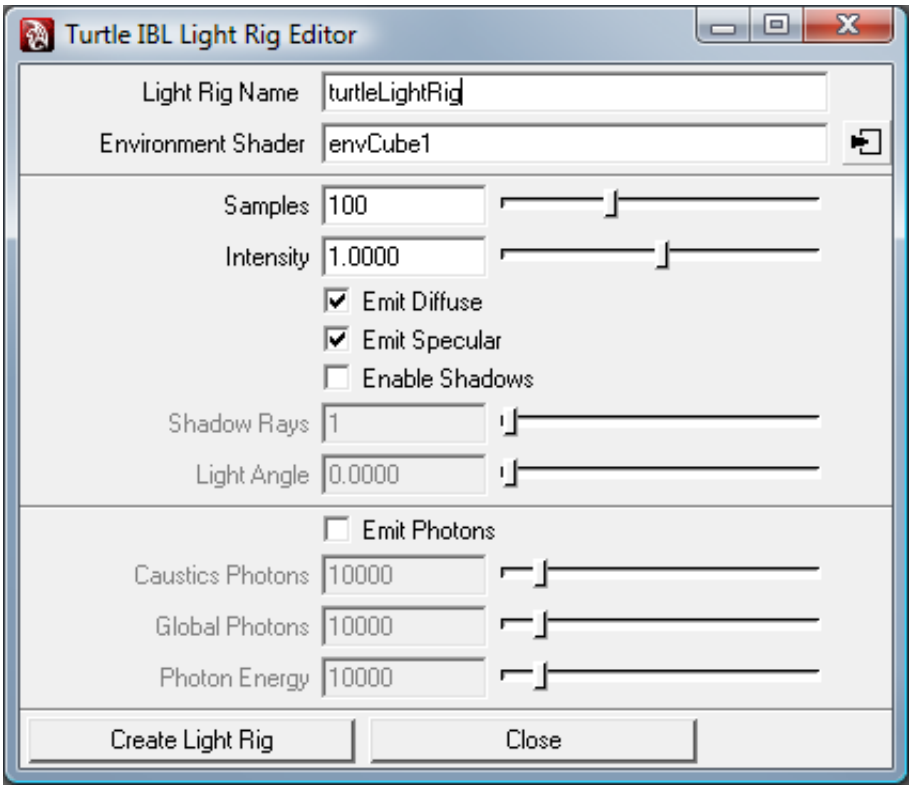

Figure 1.26: IBL Light Rig Editor

Intensity

Set the intensity of the lighting.

Emit Diffuse

To remove diffuse lighting from the lights, disable this check box.

Emit Specular

To remove specular highlights from the lights, disable this check box.

Enable Shadows

Enables raytraced shadows on the created lights.

Shadow Rays

Sets the number of shadow rays for raytraced shadows.

Light Angle

Sets the light angle for raytraced shadows.

Emit Photons

Enables photon emission from the lights in the rig.

Caustics Photons

Number of caustics photons to be emitted into the scene.

Global Photons

Number of global photons to be emitted into the scene.

Photon Energy

The total photon energy.

1.17 Programmable Shading

Turtle 3 introduced some programmable shading functionality through use of the Lua Node. The Lua Node takes a script file written in the language Lua as input and executes it for each fragment. This section will show how Lua is used in Turtle as well as give a few examples on how to use it. A reference on the programming language Lua can be found at www.lua.org. Note that indexing of arrays and vectors starts from 1 in Lua. From version 4, Turtle's types and functions have been changed to follow this convention of Lua.

1.17.1 The Lua Node

Script File

This field specifies the source file of the Lua script. Similar to the Maya File Texture Node, it can be prefixed by \\ in order to make it project-relative. One can for example specify \\turtle\scripts\phong.lua to specify a script located in the turtle part of the project directory.

Input 1-5

These are four color inputs that can be connected to other shading nodes. They are accessible from the Lua script.

output1

This is the result of the Lua script.

1.17.2 Turtle extensions to Lua

This is a reference of the Turtle extensions to Lua.

Data types:

vec3: A vector of three float values. It can be constructed with the function vec3(x,y,z).

Functions for retrieving implicit data:

vec3 getPoint(): Returns the world-space intersection point of the fragment.

vec3 getNormal(): Returns the normal of the fragment.

vec3 getTangentU(): Returns the tangent of the fragment.

vec3 getTangentV(): Returns the bitangent of the fragment.

vec3 getViewDir(): Returns the vector from the fragment to the eye in world space.

vec3 getUV(): Returns the texture coordinates of the fragment. Uses the first two components of vec3.

vec3 getdTdX(): Returns the projection of the x axis of the screen space pixel in texture space. Uses the first two components of vec3.

vec3 getdTdY(): Returns the projection of the y axis of the screen space pixel in texture space. Uses the first two components of vec3.

float getT(): Returns the distance from ray origin to current fragment.

vec3 getInput(i): Returns the i:th input from the Lua Node. It is indexed from 1 to 5.

Functions for lighting:

int getLights(): Returns the number of lights in the scene.

bool getLightName(i): Returns the name of the light.

vec3 getLightDir(i): Returns the direction from the i:th light to the fragment in world space.

float getLightDist(i): Returns the distance from the i:th light to the fragment in world space.

vec3 getLightCol(i): Returns the color of the i:th light.

vec3 getLightColUnshadowed(i): Returns the color of the i:th light, not taking shadows into account.

bool getLightAmb(i): Returns true if the light is ambient.

bool getLightCastShadows(i): Returns true if the light cast shadows.

Functions for ray tracing:

vec3 shootRay(vec3 o, vec3 d): Shoots a ray with origin=o and direction=d with shading. Returns the color of the intersection.

float shootOccRay(vec3 o, vec3 d, [float tmax]): Shoots a ray with origin=o and direction=d without shading. Returns the distance to the closest intersection or -1.0f if no intersection is found. The maximum distance to intersection can be specified in tmax if desired. The occlusion ray ignores the transparency of the occluder.

Utility functions:

float length(vec3): Returns the length of the vector.

float smoothstep(min, max, x): Returns the smoothstep function.

float step(min, max, x): Returns the step function.

float dot3(vec3 x, vec3 y): Returns the scalar product x.y.

vec3 cross3(vec3 x, vec3 y): Returns the cross product $x \times y$.

float add3(vec3 x, y): Returns x+y. y could be float or vec3.

float sub3(vec3 x, y): Returns x-y. y could be float or vec3.

float neg3(vec3 x): Returns -x. x is vec3.

float mul3(vec3 x, y): Returns x*y. y could be float or vec3. If y is vec3 the multiplication is element-wise.

float div3(vec3 x, float y): Returns x/y.

vec3 reflect(vec3 v, vec3 normal): Returns v reflected around the normal.

vec3 refract(vec3 v, vec3 normal, float inIor, float outIor): Returns v refracted around the normal. $inIor$ is index of refraction for the material the ray is leaving, $outIor$ the index of refraction for the material the ray is entering.

vec3 normalize(vec3 v): Returns v normalized.

Operations on vec3:

+: This is aliased to add3.

- (binary): This is aliased to sub3.

- (unary): This is aliased to neg3.

*: This is aliased to mul3.

/: This is aliased to div3.

[i]: Returns the i:th component of a vec3.

Special global variables:

output: What is written to this will be available on the output1 attribute of the Lua Node.

1.17.3 A note on using the Lua node as an Output Shader

It is indeed possible to use a Lua node as an Output Shader. This is achieved by connecting a Lua node to the Output Shader in the Turtle Render Globals. An Output Shader backend node can then be connected to one of the Lua inputs. Example usage could be deferred shading or adjusting the levels of the output image. Please note that no implicit data such as lights or fragment information except for the inputs are available when the Lua node is not connected to a shader.

1.17.4 Examples on Lua shaders

yellow.lua

Simplest possible example: a yellow shader.

```
-- Yellow shader
output = vec3(1,1,0)
```

input.lua

Even simpler: shading with the first input.

```
-- Input shader
output = getInput(1)
```

normal.lua

Slightly more complicated: a normal shader.

```
-- Normal shader
output = getNormal()
```

lambert.lua

A lambert shader:

```
-- Lambert shader
-- Input 1 is diffuse color

c1 = getInput(1)
lights = getLights()
normal = getNormal()

outcolor = vec3(0,0,0)

for i=1,lights do
    lcol = getLightCol(i)
    if (getLightAmb(i)) then
        outcolor = outcolor + lcol
    else
        dval = dot3(normal, getLightDir(i))
outcolor = outcolor + lcol * c1 * dval
    end
end
output = outcolor
```

phong.lua

A phong shader:

```
-- Phong shader
-- Input 1 is diffuse color
-- Input 2 is specular color
-- Input 3.x is the phong exponent

c1 = getInput(1)
c2 = getInput(2)
i3 = getInput(3)

phongexp = i3[1]
lights = getLights()
normal = getNormal()
viewdir = getViewDir()

color = vec3(0,0,0)

for i=1,lights do
    lcol = getLightCol(i)
    if (getLightAmb(i)) then
        color = color + lcol
    else
        lightdir = getLightDir(i)
        halfvector = normalize(lightdir + viewdir)
        sval = dot3(normal, halfvector)
        dval = dot3(normal, lightdir)
        if (sval < 0) then sval = 0 end
        sval = math.pow(sval, phongexp)
        color = color + lcol * (c1 * dval + c2 * sval)
    end
end
output = color
```

reflect.lua

A shader that shows simple raytraced reflections.

```
-- Reflection shader

output = shootRay(getPoint(), reflect(getViewDir(), getNormal()))
```

occlusion.lua

A shader that returns red if there is occlusion in the reflection direction and black otherwise.

```
-- Occlusion shader

occ = shootOccRay(getPoint(), reflect(getViewDir(), getNormal()))
if (occ > 0) then
    output = vec3(1,0,0)
else
    output = vec3(0,0,0)
```

```
end
```

output.lua

An output shader which inverts the image:

```
-- Output Shader
-- Input 1 is connected to the Output Shader backend node

output = - getInput(1) + 1
```

Chapter 2

Baking Reference

Baking is the common term for saving shading/lighting on objects in a scene. Many different aspects of the lighting can be baked and the result can be output to texture maps or as colors per vertices. The settings for baking is found under the **Baking** tab in the Render Settings window. Most settings under the other tabs will also affect baking. You can for instance enable Turtle's anti-aliasing and filtering to bake high quality textures, or enable Global Illumination to bake down light maps with indirect lighting.

To bake something you first need to set the **Render Type** to Baking. If you press the render button in Maya Turtle will then perform baking instead of ordinary frame rendering. During Texture Baking the result will be displayed in Maya's Render View. Render Region is supported so you can mark a smaller area in the Render View to bake specific regions of the texture map.

2.1 Bake Layers

Turtle is using a layer system to setup baking. For each bake operation Turtle operates on the objects in a specific bake layer. Each bake layer can hold different settings, so you can separate different baking setups into different bake layers. All bake layers with their respective settings will be saved to the scene file.

When working inside Maya all bake operations are performed on the currently active bake layer. The current bake layer is selected with the **Bake Layer** control under the Baking tab. The settings displayed under the Baking tab are consistent with the settings in the current bake layer, so if you change a settings it will only affect the currently active bake layer.

To perform baking on an object it must be assigned to a bake layer. You can create bake layers and assign objects to bake layers using the Lighting/Shading menu in the Rendering context, **Lighting/Shading** → **Assign New Bake Layer (TURTLE)** or **Lighting/Shading** → **Assign Existing Bake Layer (TURTLE)**. These menu selections will assign the selected objects to new or existing bake layers. You can also use the **Bake Layer Editor** (see below) to organize bake layers and bake layer objects.

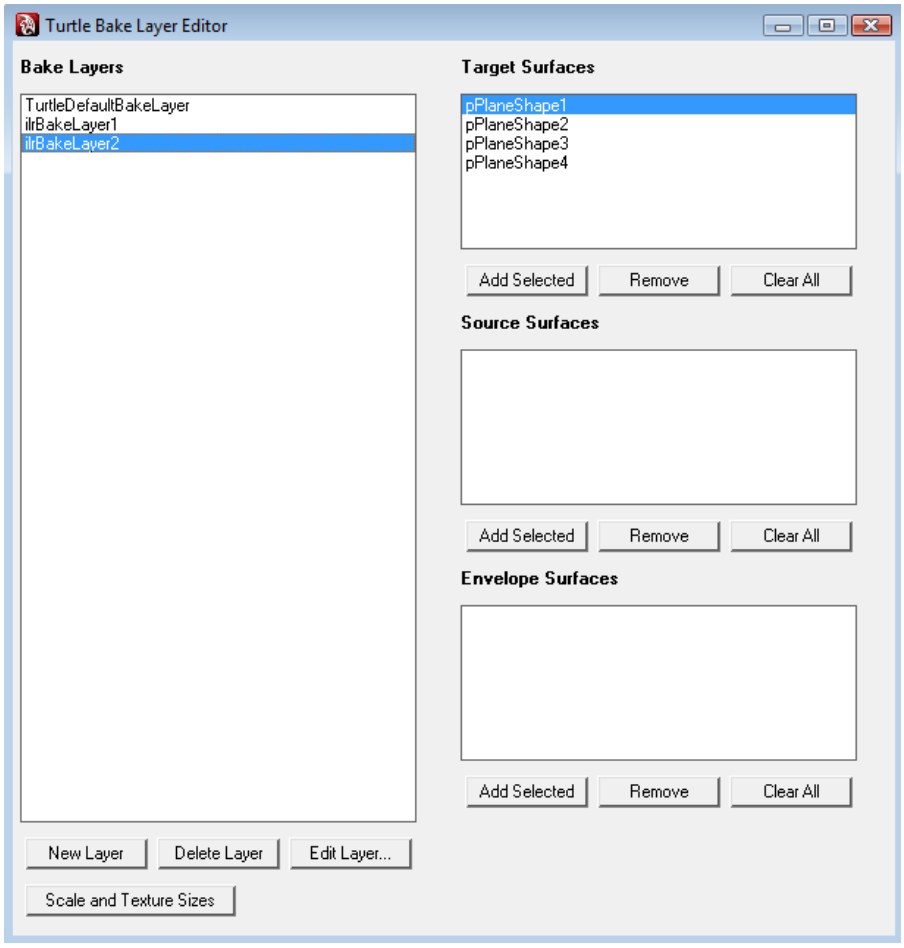

Figure 2.1: Bake Layer Editor

2.1.1 Bake Layer Editor

The Bake Layer Editor can be used to organize the bake layers in your scene. You open it by selecting **Lighting/Shading** → **Bake Layer Editor (TURTLE)** or **Windows** → **Render Editors** → **TURTLE** → **Bake Layer Editor** or you can press the Bake Layer icon in the Turtle shelf.

The left list (**Bake Layers**) displays all bake layers in the scene. When selecting a layer in the list, it will be made active, and it's assigned objects are displayed in the right list (**Target Surfaces**). You can create new layers and delete or edit the selected layer using the buttons below the layer list. You assign object to a layer by selecting the layer (making it the active layer), select the objects in the model view and press **Add Selected** under the Target Surfaces list. To remove objects from a layer, select the objects in the Target Surfaces list and press **Remove** or press **Clear All** to remove all objects at once.

Note: You can select multiple items in the surfaces lists when you do add, remove and select actions.

The **Source Surfaces** list is used if you want to bake something on one object and transfer the result onto another object (aka Surface Transfer). The typical example is when you transfer lighting or shading information from a highly detailed mesh onto a low detailed version of the same mesh. The resulting texture maps (e.g. normal maps, displacement maps or light maps) can then be used on the low resolution mesh to make it look more detailed than it really is. See section 2.2.1 for more information on this.

By double-clicking on a layer in the layer list, it's assigned objects will be put on the global selection list (**Right Click** → **Select Targets** has the same effect). The Target / Source / Envelope lists also has a right-click-menu which can be used to add or remove objects from the global selection list.

Scale and Texture Sizes

Click the **Scale and Texture Sizes** button to open the UV Size Assignment editor. This editor automatically assigns a texture size to each object in the bake layer. The object with the largest area in the layer is found and all other texture sizes are scaled in relation to the current area / max area quote. An object can never have a smaller texture size than the **Min Resolution** and never a larger resolution than the **Max Resolution**.

The **Quadratic Textures** forces Turtle to take the largest dimension and make the texture size quadratic. The **Create Power of 2 Textures** checkbox makes the dimensions into power of 2 numbers (64, 128, 256...). The dimension is always rounded up, for example 129 is rounded up into 256.

Shared UV-set

The shared UV-set functionality basically does the same thing as the **Texture Sizes**, but instead of calculating resolutions it calculates relative UV-spaces. The **UV Padding** determines the space between different UV patches.

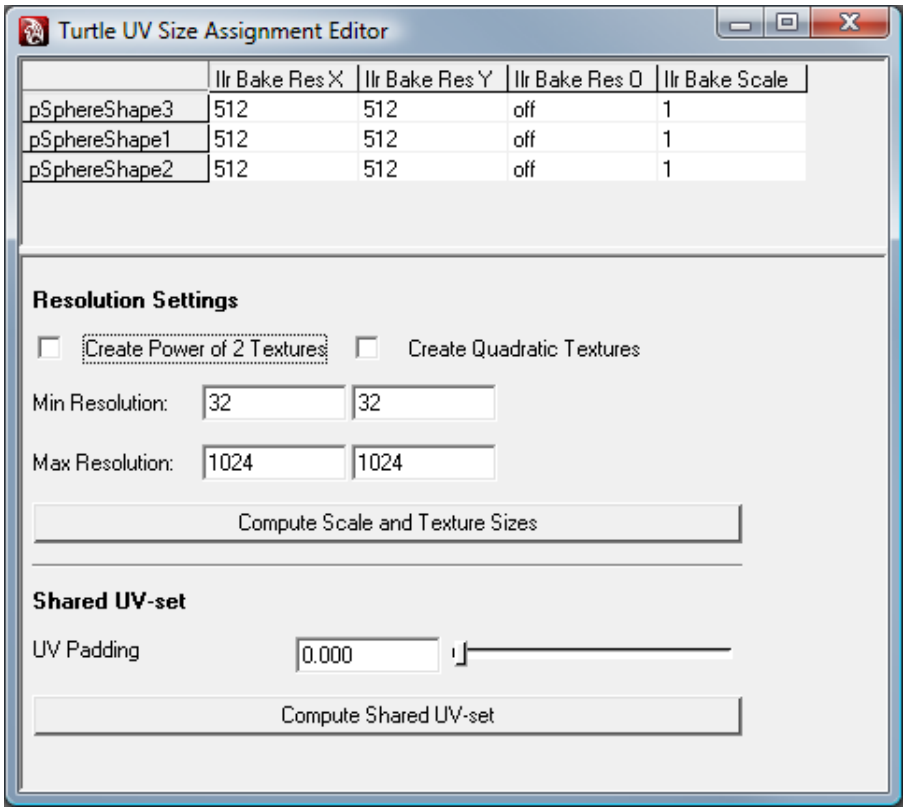

Figure 2.2: UV Size Assignment Editor

2.2 Bake Layer Settings

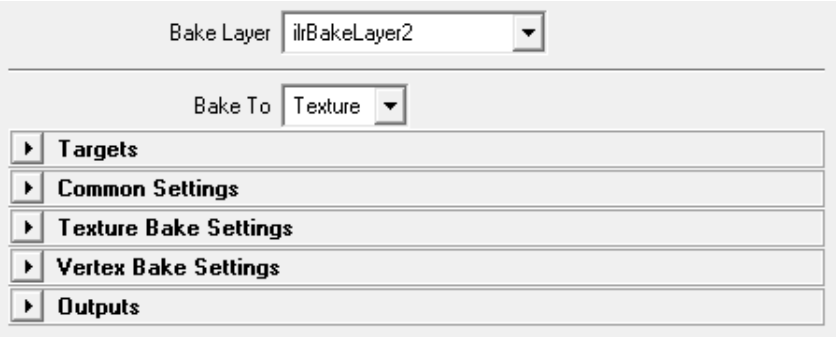

Figure 2.3: Bake Layer Settings

When a bake layer is activated, by selecting it in the Bake Layer drop-down or selecting it in the Bake Layer Editor, it's settings are shown in the Baking tab. The first thing to select is which bake type to use (what the output should be): Texture or Vertices. This is selected with the **Bake To** drop-down. When one bake type is enabled all settings that don't apply to that bake type will be grayed out.

Note: The settings shown in the Baking tab in the Render Settings window is the settings for the currently active bake layer. When another bake layer is activated Turtle will switch to show the settings for that layer. However, all settings for each bake layer is saved in the layer, so settings are never lost.

2.2.1 Targets

Under the **Targets** tab you specify which objects you want to bake to for the current layer. Figure 2.4 shows the **Targets** interface in the Maya Render Settings window.

You can also specify that you want to bake from one object to another, so called **Surface Transfer**. Surface Transfer means baking surface properties from highly detailed surfaces to low detail surfaces. The UV-set of the target surfaces are used to create the texture maps. The source surfaces do not need to be UV-mapped. Almost anything can be baked down with Surface Transfer. Normal maps, displacement maps and various illumination passes are easy to bake down, and you have some highly specialized passes for even more advanced uses. This feature also works well for vertex baking, e.g. to bake vertex colors from different LOD levels of a mesh.

Surface Transfer Links are used to control which highres surfaces to consider for each lowres surface. Each lowres surface can have its own list of links to highres surfaces. This makes it easier to avoid invalid intersections that may occur if the highres surfaces overlap each other. The same kind of links are used to specify envelope surfaces. The **Source Surface** and **Envelope Surface** lists are used to setup these links, see below.

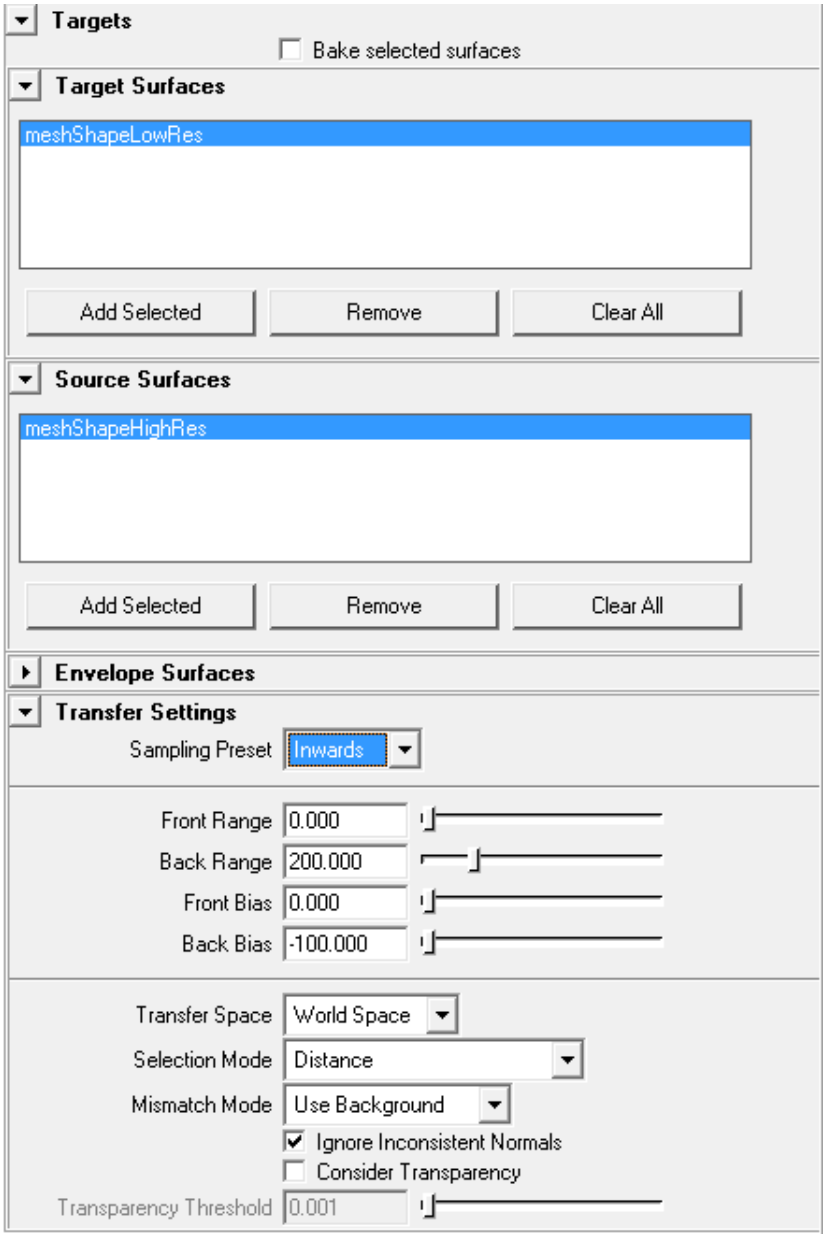

Figure 2.4: Bake Targets

Bake selected objects

When this is enabled, you will override the objects setup in the bake layer and instead bake the objects selected in the model view. Multiple objects can be selected and baked at the same time. The bake settings in the current bake layer will be used for the baking. Surface Transfer cannot be done in this mode.

Target Surfaces

This is where you add the surfaces you want to bake to. Click **Add Selected** to add surfaces to the list. Click **Remove Selected** to remove surfaces from the list. Click **Clear All** to clear the list.

Source Surfaces

If Surface Transfer is requested, this is where you add links to source surfaces for each target surface. To make a new link to a source surface from a target surface, first mark the target surface in the Target Surfaces list above. Then select the source surfaces and click **Add Selected**. Click **Remove Selected** to remove links from the list, and click **Clear All** to clear all links.

Envelope Surfaces

When using Surface Transfer without an envelope mesh, the sampling rays are sent from the low res surface to the high res surface. Sometimes this process fails since the high res mesh and the low res mesh is too different. This problem can be alleviated by using an envelope mesh. The envelope mesh works as the ray origin mesh and the best intersection on the high res surface is decided according to the Envelope Mode selection. The hit is then transferred to the low res surface. Envelope Surfaces are specified by creating links in the same way as for source surfaces (see above).

Envelope Mode

There are four different envelope modes:

- **Closest** finds the closest point on the high res surface.
- **Inside Only** considers hit points on the high res surface that are inside the envelope mesh.
- **Outside Only** considers hit points on the outside of the envelope mesh.
- **Inside First** considers hit points on the inside of the envelope. If it cannot find a valid hit inside, it will search outside as well.
- **Outside First** is the same but it searches the outside first.

Front Range

Sets the max distance that will be traced in front of the target surface when searching for source surfaces. If surfaces are found both in front of and behind, the closest one will be used.

Back Range

Sets the max distance that will be traced behind the target surface when searching for source surfaces. If surfaces are found both in front of and behind, the closest one will be used.

Front/Back Bias

Sets an offset distance for the origin of the front/back probe ray when searching for source surfaces. Can also be negative to make an offset in the opposite direction.

Transfer Space

Selects whether the objects should be sampled from world space or object space. If the objects are in world space, they must be placed in the same position, as opposed to object space, where target and source objects can be positioned side by side for instance, as long as the respective object spaces match.

Selection Mode

Selects how Turtle chooses what surface to sample.

- **Distance** chooses the closest surface in front or in the back.
- **Best Normal** chooses the surface with the same normal as the sample point.
- **Distance with flipped normal** chooses the closest surface and flips the normal if the direction is backwards.

Mismatch Mode

Selects how Turtle handles the case when no source surface is found.

- **Use Background** returns the background color.
- **Use Target Surface** shades and returns the color of the lowres intersection.

Ignore Inconsistent Normals

With this option you can choose to ignore intersections where the normals of the highres and lowres surfaces are inconsistent (pointing in opposite directions). The search will then continue until a surface with consistent normal is found (or no surface is found). This will help you to pick the right surface intersection in areas with both front facing and back facing surfaces, e.g. around the ears or in the armpits of a character. This is enabled by default.

Consider Transparency

Turtle defaults to only sampling the first surface hit when making a surface transfer. If you have a set of transparent objects, that should be accumulated, you need to enable **Consider Transparency**. The **Transparency Threshold** decides when Turtle will continue sampling through a semi-transparent surface.

2.2.2 Common Settings

In this section you find settings that apply for both texture and vertex baking.

Camera

Controls which camera to use when baking. This affects the **Bake View Dependent** and **Normal Direction** settings.

Normal Direction

Controls in which direction the normals of the objects to be baked should be pointing. If the normal faces away from the camera the baking will be completely black.

- **Surface Front** uses the original surface normal.
- **Surface Back** flips all normals.
- **Facing Camera** flips all normals that faces away from the camera.

Orthogonal Reflections

Controls whether secondary rays are sent in directions according to the camera or according to a virtual, orthogonal camera. The virtual orthogonal camera is placed directly above and directly facing the point to be shaded.

Bake Shadows

Enables baking of shadows.

Bake Alpha

Controls if the alpha channel should be saved.

Bake View Dependent

When enabled, Turtle will only bake texels that are visible from the camera.

Background Color

Sets the background color. All unsampled texels or vertices will receive this color.

2.2.3 Texture Bake Settings

Here you find settings that are only valid for texture baking.

Width / Height

Sets the resolution of the texture map.

Conservative Rasterization

Enable this to solves baking problems for triangles thinner than 1 pixel.

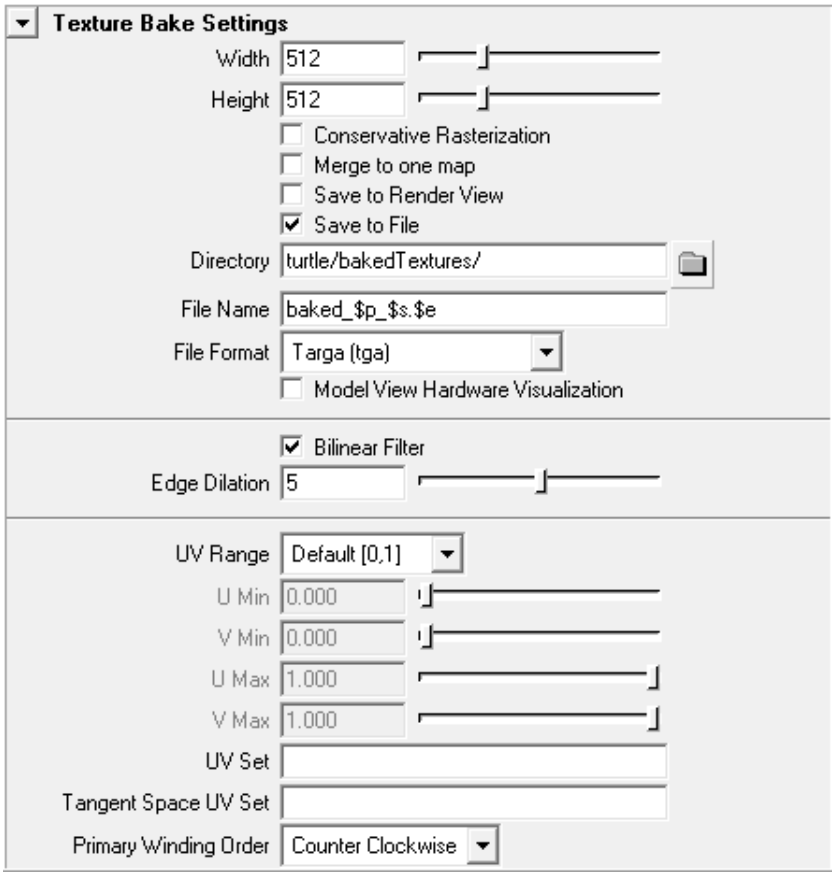

Figure 2.5: Texture Bake Settings

Merge to one map

Controls how to handle the case when you have multiple objects in one bake layer. If enabled all objects will be baked into the same map (it is the users responsibility to ensure that the objects do not contain overlapping UV's). If disabled the objects will be baked into separate maps.

Save to Render View

Store the image in the render view, it will be kept in the set of saved images in the drag bar.

Save to File

Store the image in the directory specified by **Directory** and **File Name**.

Directory

The directory where the baked textures will be stored.

File Name

The name of the baked texture. The name is put together by the user with the help of the following predefined variables:

$m − material name (shading group)
$s − shape name
$t − full path to shape
$u − uvset name
$p − output pass name
$r − render layer name
$b − bake layer name
$e − file format extension
$f − frame number

When these variables are found in the file name they are replaced with the corresponding text string.

Example:

"lightmap_$m_$s.$e" translates to "lightmap_lambert1SG_meshShape1.tif"
"lightmap_$t.$f.$e" translates to "lightmap_mesh1_meshShape1.42.tif"
"normalmap_$s_$u.$e" translates to "normalmap_meshShape1_uvSet1.tif"

File Format

Sets in which format the baked texture will be stored. Supported formats are TGA, OpenEXR, TIFF, TIFF16, TIFF32, IFF, OpenEXR MultiLayer, Windows Bitmap and PNG.

Model View Hardware Visualization

Enable to automatically generate an ilrHwBakeVisualizer node (section 4.3.9) and connect to the target object's surface shader **Hardware Shader** attribute after bake is completed. A file texture node will be generated for each baked texture and connected to the corresponding input of the ilrHwBakeVisualizer node. In order to get proper output from the hardware shader, make sure that the option **Shading** → **Hardware Texturing** in the viewport menu has been enabled, and that **Renderer** → **Default Quality Rendering** is set.

Note: Note that the auto assignment of this node will only work if the target object's surface shader has a **Hardware Shader** input.

Note: By editing the top lines of the MEL file **ilrBake.mel** in the Turtle **plug-in** folder it is possible to modify how Turtle connects baked textures to the ilrHwBakeVisualizer node.

Bilinear Filter

Dilates just enough to cover the pixels that are missed due to bilinear filtering.

Edge Dilation

Sets the number of pixels to stretch texture edges. This can be used to eliminate bleeding problems, if filtering is used on the baked textures. It will also fill out holes in the texture, which might occur when baking surface transfer from displacement mapped surfaces. See figure 2.6.

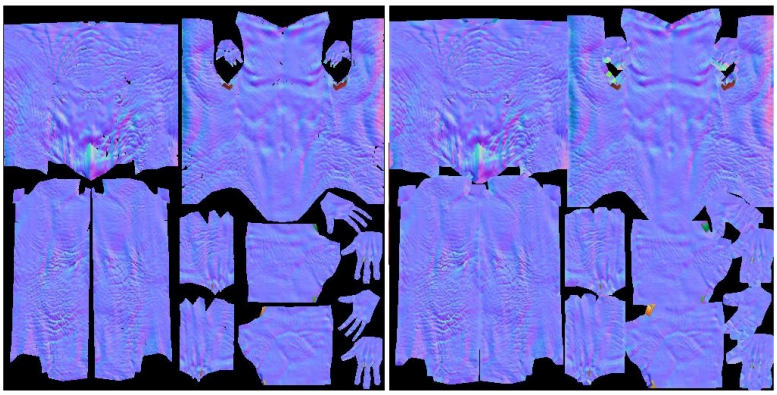

Figure 2.6: Without and with edge dilation filter.

UV Range

Controls the amount of the object to be sampled in texture UV space.

- **Default [0,1]** bakes the object in the range U [0,1] V [0,1].
- **Entire Range** bakes the entire object in UV-space.

- **User Specified** bakes the object in the UV range defined by the user using the U/V min/max sliders.

UV Set

The name of the UV layout to use when baking. If this is not specified Turtle will use the current UV-set. The current UV-set will also be used when the specified UV set cannot be found.

Tangent Space UV Set

The name of the UV layout to use for calculating tangent space. This can be specified for baking into one UV-set but with a tangens space from another UV-set. If this is not specified Turtle will use the UV-set specified above, or the default UV-set.

Primary Winding Order

Controls which polygon to bake if there are polygons with overlapping UV's. This can for example be used to control how baking polygons with mirrored UV's should be handled.

2.2.4 Vertex Bake Settings

Vertex baking is a technique often used in games and other real-time applications, where illumination is saved to each vertex in the mesh. Here you find settings that are only valid for vertex baking.

Sampling Mode

Selects what sampling mode to use for vertex baking.

- Per Vertex. Takes one sample per polygon vertex.
- Triangle Subdiv. Takes a number of samples over each triangle and accumulates the results to the vertices. The number of samples taken is adaptive and depends on the size of the triangle.

Min Samples

Sets the minimum number of samples to take for each triangle if **Triangle Subdiv** is selected as sampling mode.

Max Samples

Sets the maximum number of samples to take for each triangle if **Triangle Subdiv** is selected as sampling mode.

Vertex Bias

Used to move the sample points a small amount for each vertex color. The sample points are moved from the vertex towards the center of the polygon. If set to 0.0 the sample point is exactly at the vertex, if set to 1.0 the sample point is at the polygon center. This is very useful to alleviate Final Gather artifacts when baking to vertices.

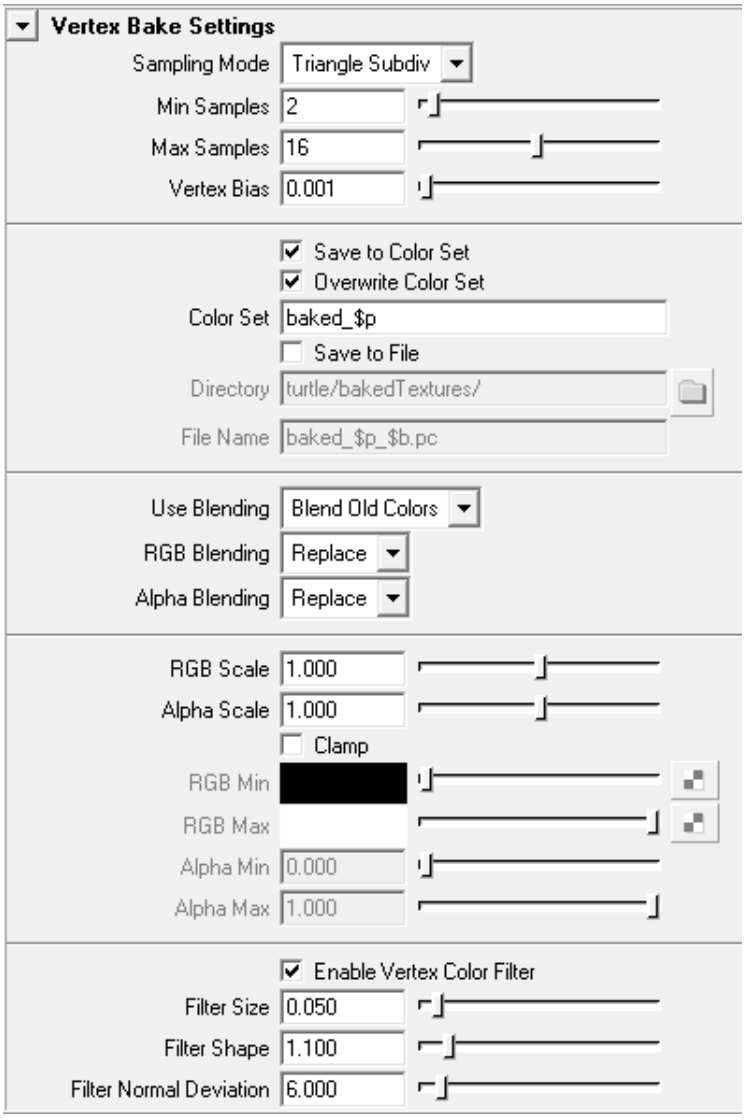

Figure 2.7: Vertex Bake Settings

Save to Color Set

When enabled the baked vertex colors will be save to a color set in the mesh.

Overwrite Color Set

When enabled the given color set will be overwritten if it exists, otherwise a new color set will be created.

Color Set

The name of the vertex color set to use when baking. If this is not specified Turtle will use the current color set. If the specified color set cannot be found, a new color set will be created. The name is put together with the help of the following predefined variables:

$s − shape name
$p − output pass name
$r − render layer name
$b − bake layer name

When these variables are found in the name they are replaced with the corresponding text string.

Example:

"baked_$p_$s" translates to "baked_tpIllumination_meshShape1"
"$b_$p" translates to "ilrBakeLayer1_tpIndirectIllumination"

Save to File

If enabled the baked vertex colors will be save to a point cloud file. Vertex color point clouds can then later be imported an assigned to a mesh with the command **ilrImportVertexColorsCmd**, see section 5.2.3. This is useful for e.g. when batching vertex bake jobs on multiple machines.

Directory

The directory where the files will be stored.

File Name

The file name pattern. The name is put together with the help of the following predefined variables:

$s − shape name
$p − output pass name
$r − render layer name
$b − bake layer name

When these variables are found in the name they are replaced with the corresponding text

string.

Example:

"baked_$p_$b.pc" translates to "baked_tpIllumination_bakeLayer1.pc"

If ".xml" is used as extension the file is save in the point cloud XML format. Otherwise the file is saved in the point cloud binary format.

Use Blending

Controls if and how blending should be used.

- Disabled. Disables blending and new colors will replace any existing colors.
- Blend Old Colors. Blends the new colors with old colors, if there exists a color set with the same name.
- Blend Passes. If used together with multiple output passes, it blends between the passes, creating a single new color set.

RGB Blending / Alpha Blending

Controls which blending mode to use.

- Replace. Replaces the old color with the new one.
- Add. Adds the new color to the old one.
- Subtract. Subtracts the new color from the old. If the result is less than zero, the result is clamped to zero.
- Multiply. Multiplies the new value with the old.
- Divide. Divides the new value with the old.
- Average. Averages the new and the old color.
- None. Do nothing. Useful if you want to rebake the color but not the alpha.

RGB Scale / Alpha Scale

Scales the RGB and Alpha values with the given scaling factor.

Clamp

When enabled the RGB and Alpha values will be clamped to the given min/max values.

Enable Vertex Color Filter

Enables filtering of the vertex values to give a smoother appearance.

Filter Size

Sets the size of the filter kernel, given in fraction of object size. 0.0 means no filtering, 1.0 means ”filter size” = ”object size”.

Filter Shape

Sets the filter shape, which affects the filtering appearance. A higher value gives a wider filter.

Filter Normal Deviation

Sets how much a vertex’s normal can deviate before the vertex is ignored by the filter. Given in degrees.

2.2.5 Outputs

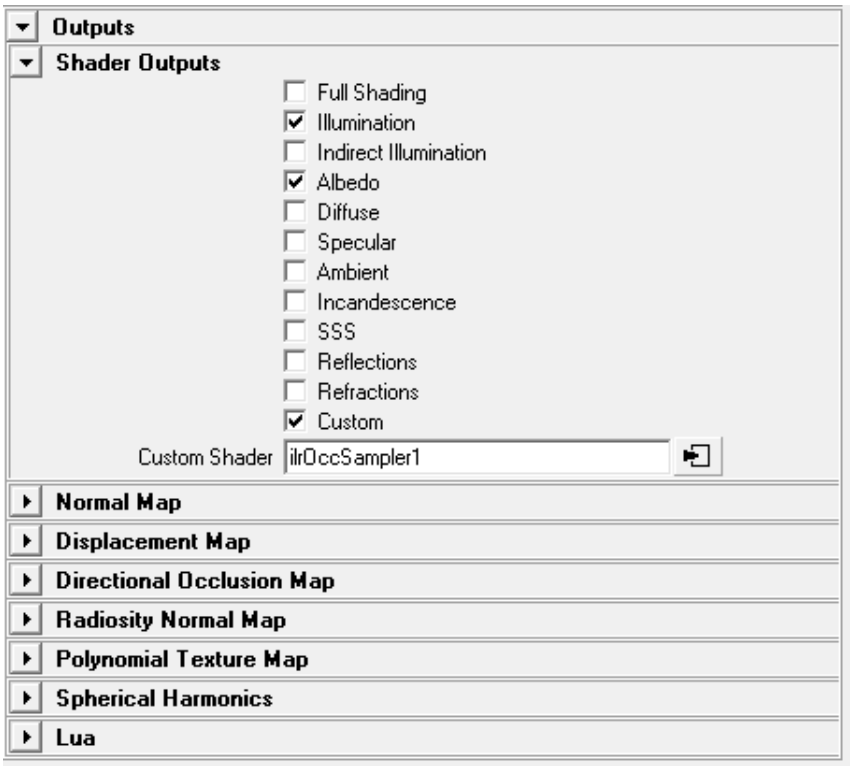

Figure 2.8: Bake Outputs

Here you control which bake passes Turtle should output. If several passes is checked, Turtle will render several textures (or vertex color sets).

Shader Outputs

Can be used to output various components from a shader to separate passes, see figure 2.8. **Full Shading** will give the complete lit and shaded result. **Illumination** will give the incident illumination (including indirect light) and **Indirect Illumination** will give the incident indirect illumination only (**incident** here means the shader's color/texture will not be included). The rest of the output passes will give the corresponding shading component.

Custom

Can be used to create a custom pass by connecting a shader to the **Custom Shader** slot. For example connecting an ilrOccSampler node will give an occlusion pass and connecting an ilrShadowMask node will give a shadow mask pass.

Normal Map

Check this box if you want to bake a normal map.

Coordinate System

Sets the coordinate system for normals when creating normal maps.

- Tangent Space. The resulting map will be in tangent space. The tangent space created can be changed by altering the tangent overrides in the Options tab in Render Settings window.
- Object Space. The resulting map will be in object space.
- World Space. The resulting map will be in world space.

Modify Channels

Defines how the normal components are mapped to the color components.

- Default. X, Y, and Z components are mapping directly to R, G and B components.
- Invert Red. Inverting the red channel.
- Invert Green. Inverting the green channel.
- Flip Red-Green. Switch places for red and green channels. Mapping X to G and Y to R.

Use Face Tangents

By checking this box, Turtle will use the face tangents of the object instead of the vertex tangents.

Include Bump Maps

Specifies if bump mapping should be included when creating normal maps.

Stencil in Alpha Channel

By checking this box, Turtle will store a stencil (hit-or-miss) in the alpha channel.

Displacement in Alpha Channel

By checking this box, Turtle will store the displacement in the alpha channel. Can be used to create a map for parallax mapping. Use the displacement scale and displacement offset described below, to normalize the result to a valid range.

Displacement Map

By checking this box, Turtle will generate a displacement map.

Fit to range[0,1]

Scale the displacement to [0,1] range.

Scale

Scales the sampled displacement value. Can be used to set the displacement map values to a valid range.

Offset

Adds an offset to the sampled displacement value. Can be used to set the displacement map values to a valid range.

The other more advanced baking outputs, like Radiosity Normal Maps, Polynomial Texture Mapping, Directional Occlusion and Spherical Harmonics, are covered in the chapter **Advanced Baking**, section 2.3. There is also the possibility to program your own advanced output pass using Lua. This is covered in the chapter **Programmable Baking**, section 2.4.

2.3 Advanced Baking

In Turtle there are a number of advanced output passes for baking. These are all described below, along with a few external links with more information and detail on the techniques.

2.3.1 Radiosity Normal Maps

Baking a Radiosity Normal Map (RNM) will result in three textures, each containing information about incoming light computed based on one of the three RNM normals. The RNM can either be sampled using brute force or by using **Final Gather** with Cache Type **Radiance Cache**. Using this cache usually gives faster render times and better quality.

- **Adjust Intensity**: If enabled, Turtle tries to match the intensity calculated for the surface normal by the RNM with the intensity value one would get using a standard light map. This function is used when different tangent spaces on the mesh produces seams in the RNM.

- **Sample Type**: Type of illumination to gather.
- **Samples**: The minimum number of samples to use when gathering illumination. This value is only used in brute force mode.

External Links

- www2.ati.com/developer/gdc/D3DTutorial10_Half-Life2_Shading.pdf

2.3.2 Directional Occlusion

This is a technique developed by Chris Green at Valve® which allows for a fast dynamic occlusion, needing only the four channels of an 8-bit RGBA texture for storage.

- **Samples**: The minimum number of samples to use when gathering illumination.
- **Exponent**: The exponent is used during computations to define the 'sharpness' of the occlusion values. Higher values may produce a more pronounced effect.
- **Basis Vector 1-4**: Basis vectors to use for the four different channels.

For this technique we define four vectors in tangent space. Each one is used by one color channel of the resulting texture as a surface normal vector to compute an occlusion value. To light using the directional occlusion texture, do the following:

Assume the light direction $\bar{l}$ in tangent space. The weights used for each component of the texture is computed by dot product between $\bar{l}$ and each of the basis vectors $\bar{v}_i$, $1 \leq i \leq 4$, giving the scalar weights w_i. The final light contribution c is then given by

$$c = \frac{\sum_1^4 d_i w_i}{\sum_1^4 w_i},$$

where d_i are the sampled values from the directional occlusion texture. Note the possibility to reduce these computations to simple scalar products.

2.3.3 Polynomial Texture Maps

Baking a Polynomial Texture Map (PTM) will generate a number of coefficients for a second degree bivariate polynomial, which is used to approximate the incoming light function over the hemisphere. When baking a PTM, the output file format must be set to a format capable of storing float values (OpenEXR, OpenEXR MultiLayer or TIFF32). Also make sure clamping of output is disabled in the Anti-Aliasing settings.

- **Samples**: The minimum number of samples to use when gathering illumination.
- **Sample Type**: Type of illumination to gather.
- **Output**: Whether to use the intensity of the gathered illumination to generate one PTM or generate one PTM for each color channel.

External Links

- www.hpl.hp.com/ptm.

2.3.4 Spherical Harmonics

This option will let you generate textures containing spherical harmonics coefficients. Since the values of these coefficients can lie outside the range $[0, 1]$, the same recommendations as for PTMs apply: use float textures (OpenEXR, OpenEXR MultiLayer or TIFF32) and disable any clamping.

- **Bands**: The number of bands to use for the spherical harmonics basis function. The number of actual basis functions used will equal the number of bands squared ($nBands^2$).
- **Samples**: The minimum number of samples to use when gathering illumination.
- **Sample Type**: Type of illumination to gather.
- **Space Type**: Space to gather information in.
- **Output**: Whether to use the intensity of the gathered illumination to generate one set of SH coefficients or to generate one set of SH coefficients for each color channel.

External Links

- www.research.scea.com/gdc2003/spherical-harmonic-lighting.pdf
- msdn.microsoft.com/directx, Microsoft DirectX SDK contains examples of how to use spherical harmonics.

2.4 Programmable Baking

You have the possibility to script and customize your own baking output using Lua. Note that there is a difference between the scripts used for the ilrLuaNode (section 1.17) and the scripts used for the Lua baking pass.

2.4.1 The Lua bake script

The Lua bake script should have the following structure:

```
function setup()
-- settings for baking
end

function basis(x, y, z)
-- define function basis to fit data to
-- optional
end

function bake()
-- compute the data which is output to texture or point cloud
-- optional
end
```

2.4.2 Setup function

This function is executed once before baking and is used to set the different baking settings.

Available functions:

bset(⟨string⟩ setting, ⟨-⟩ value) Give a setting the specified value.

- **gather.useRadianceCache**, ⟨ bool ⟩ Uses the radiance cache instead of brute force sampling. All gather functionality is controlled via the final gather settings. **basis.type** should be set to ”sh”.
- **gather.sampletype**, ⟨ string ⟩
  - “none”, “n”
  - “indirect illumination”, ”ii”
  - “dynamic illumination”, “dyni”
  - “direct dynamic illumination”, “ddyni”
  - “indirect dynamic illumination”, “idyni”

  Default value is “direct dynamic illumination”. This settings determines the type of light to gather. If set to **none**, the gather before each execution of the basis function is disabled. This is useful when e.g. only baking direct static light, and no gather is needed.
- **gather.space**, ⟨ string ⟩
  - “world space”, “ws”
  - “object space”, “os”
  - “tangent space”, “ts”

  Default value is “tangent space”. Specify the space in which to perform the gather.
- **gather.distribution**, ⟨ string ⟩
  - “cosine weighted”, “cw”
  - “equiareal”, “ea”

  Default value is “equiareal”. Specifies the distribution type to use for gather.
- **gather.minsamples**, ⟨ int ⟩ $[1, \infty[$

  Default value is 128. Specifies minimum number of samples used for gather. This number is used to determine the number of sample cells to use in the gather distribution. Note that the actual number of sample cells used not necessarily equals this number (though it will be roughly the same).
- **gather.maxsamples**, ⟨ int ⟩ $[1, \infty[$

  Default value is 256. Specifies maximum number of samples used for gather, is used to set a upper limit for super sampling in the gather distribution.
- **gather.coneangle**, ⟨ int ⟩ [0, 360]

  Default value is 180. Cone angle to use for gather distribution. Should be set to 180 or 360 in most cases.

- **gather.clamp**, ⟨ boolean ⟩

  Default value is false. Specify whether or not to clamp the values sampled during gather.

- **gather.cosweighdynamic**, ⟨ boolean ⟩

  Default value is false. Specify whether gathered dynamic light should be cosine weighed on normal.

- **output.size**, ⟨ int ⟩ [1, ∞[

  Default value is 4. Specify the size of the array returned by the bake function.

- **basis.type**, ⟨ string ⟩
  - "none", "n"
  - "ptm"
  - "sh"
  - "custom", "c"

  Default value is "none". Determines the type of function basis to fit gathered data to. If set to other than none, the bake function can be omitted from the script, and the coefficients for the specified function basis will be generated automatically. If set to custom, the basis function must be defined.

- **basis.rgb**, ⟨ boolean ⟩

  Default value is false. This setting should be taken into consideration when baking function coefficients. If the bake function returns one set of coefficients for each color channel, this setting should be set to true, and the bake function should return the coefficients for all three channels as one concatenated array. If generating coefficients for the intensity of gathered data, this settings should be set to false.

- **basis.sh.bands**, ⟨ int ⟩ [1, 10]

  Default value is 2. Specify the number of bands to use if generating SH coefficients.

2.4.3 Basis function

This function defines the values of the function basis you want to fit the gathered data to. It only needs to be defined if output has been set to custom basis (basis.type is set to custom). An array with the values must be returned. The function is called once for every direction used in the gather before baking. The following example shows how a PTM would be defined:

```
function basis(x, y, z)
    return {x*x, y*y, x*y, x, y, 1}
end
```

2.4.4 Bake function

The bake function is executed once for every pixel (or super sample if super sampling is enabled). It must return an array with the data that should be output to the texture or point cloud. It is also important that the size of the returned array is the same as specified in the setup function (output.size). In the bake function you can use all the function that are available for ilrLuaNode scripts (see section 1.17), except for the getInput() function. In addition, there are a number of utility functions specific for Lua baking available:

Available functions:

⟨**int**⟩ **getSampleCount()**

- Return value: Returns the number of sample cells used in the distribution. Note that the number of sample cells not necessarily equals the minimum number of samples set in the setup function.

⟨**vec3**⟩ **getSampleValue(**⟨**int**⟩**)**

- Argument 1: index of sample cell to query for value.
- Return value: value (color) as vec3.

⟨**vec3**⟩ **getSampleDirection(**⟨**int**⟩**,** ⟨**boolean**⟩**)**

- Argument 1: index of sample cell to query for direction.
- Argument 2: transform direction vector, false: gather space, true: world space
- Return value: direction as vec3.

⟨**vec3**⟩ **getSampleMean()**

- Return value: mean value of all sample cells as vec3 (unweighed)

⟨**vec3**⟩ **getSampleProjectedSum(**⟨**vec3**⟩**,** ⟨**boolean**⟩**)**

- Argument 1: projection vector.
- Argument 2: include values with negative weight.
- Return value: sum of all sample cell values as vec3, weighed using the scalar product between sample cell direction and given vector.

⟨**vec3**⟩ **getGatherSpaceVector(**⟨**int**⟩**)**

- Argument 1: [1,3] index of gather space vector.
- Return value: a gather space vector in world space

⟨**array**⟩ **getBasisCoefficients(**⟨**int**⟩**)**

- Argument 1: [0,3] generate basis coefficients for chosen basis from following data
  - 0: color intensity
  - 1-3: corresponding color channel
- Return value: array (table) of coefficients

⟨**array**⟩ **evalSHBasis(**⟨**vec3**⟩**,** ⟨**int**⟩**)**

- Argument 1: direction to evaluate SH function for.
- Argument 2: [1,10] number of SH bands to use.
- Return value: array (table) of SH basis values, size of array is (number of bands)^2

⟨**vec3**⟩ **gatherToWorld(**⟨**vec3**⟩**)**
⟨**vec3**⟩ **worldToGather(**⟨**vec3**⟩**)**

- Argument 1: vector to transform.
- Return value: transformed vector.

2.4.5 Using the Radiance Cache with Lua

Baking with Lua can become quite slow since the gather is executed once per sample. You can work around this to some extent by using the radiance cache. To enable this you need to set **gather.useRadianceCache** to true and **basis.type** to ”sh” in the setup function. GI with **Final Gather** with **Radiance SH** needs to be enabled as well.
There are some drawbacks with using the cache:

- The cache only contains 3 band SH. If you are trying to bake higher orders, the cache will act as a low pass filter.
- **getSampleCount** will always return 1 when using the cache.
- **getSampleValue** cannot be used with the cache.
- **getSampleDirection** cannot be used with the cache.
- **getSampleMean** cannot be used with the cache.

External Links

- www.lua.org

2.5 Point Cloud Baking

This type of baking will output a point cloud file, containing custom type of information for a set of arbitrary points in a scene. These points could represent vertex points for a shape or any other set of points. The **Point Cloud Bake Editor** is located under **Window** → **Rendering Editors** → **TURTLE** → **Point Cloud Bake Editor**, and on the Turtle shelf.

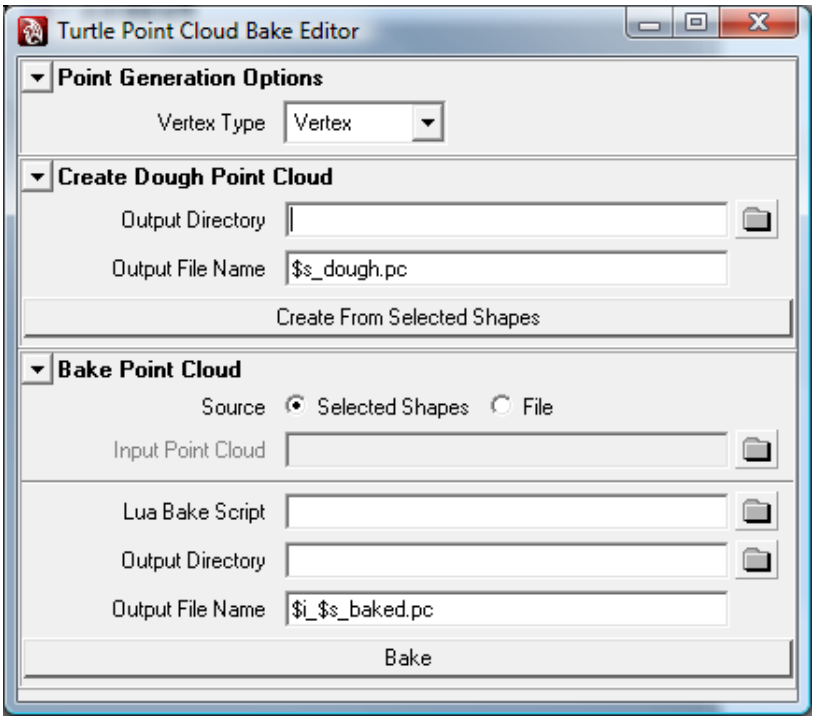

Figure 2.9: The Point Cloud Bake Editor.

2.5.1 Create Dough Point Cloud

This section of the editor lets you create a 'dough' point cloud from a shape's vertices or face vertices. The output file can be used as a file source for point cloud baking, or as a template for creating custom point cloud files.

- **Output Directory**: Directory to store generated point cloud file in.
- **Output File Name**: The name of the point cloud file. **$s** in the filename will be replaced with the name of the shape used to generate the points. If the extension of the file is set to be **.xml**, an xml file will be generated, otherwise the result will be a file in binary format.
- **Create From Selected Shapes**: Click this button to generate the point cloud file(s).

2.5.2 Bake Point Cloud

This part of the editor lets you specify the settings for baking point clouds.

- **Source**: Whether to bake from a 'dough' point cloud, or from selected shapes.
- **Input Point Cloud**: If **Source** is set to **file**, specify the 'dough' point cloud file to use as source.
- **Lua Bake Script**: Lua script file to execute for baking.
- **Output Directory**: Directory to put generated point cloud in.

- **Output File Name**: File name of the generated file. **$s** will be replaced with the shape name, **$i** will be replaced with the source file name. If extension is **.xml**, an xml file will be generated, otherwise a binary file will be generated.

2.5.3 Point cloud files

Point cloud files can be written in two different formats. If the file name specified has given the extension **.xml**, the file will be written in xml format. In all other cases, the file will be written in a binary, non-public format. The binary format should be used for e.g. photon map files, where speed and size are important factors. If you want to use the data for other purposes, outside of Turtle, the xml format is recommended. Note that point cloud points are stored in object space when generated from a shape. The object to world transformation matrix is stored in the point cloud file header.

2.6 Texture Resampling

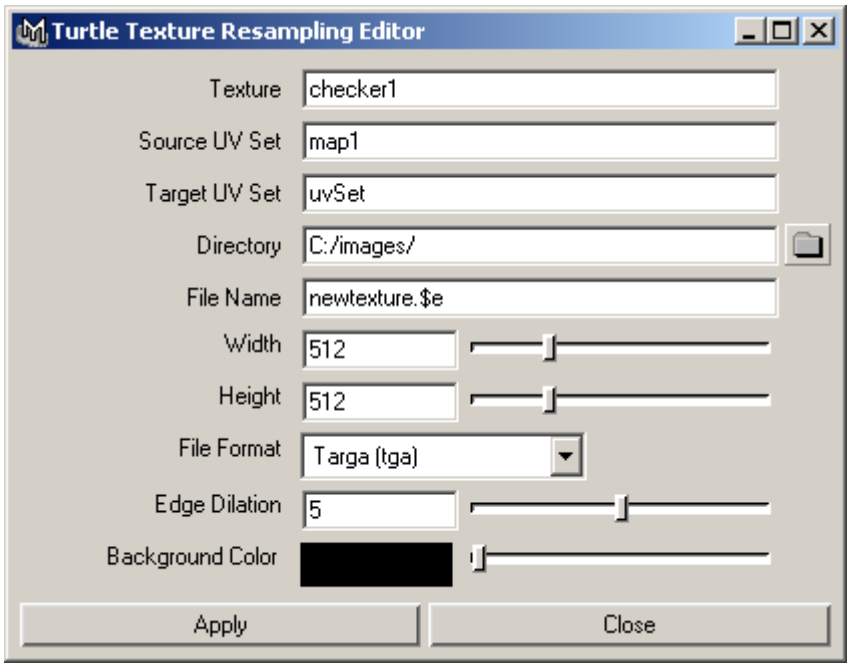

Figure 2.10: The Texture Resampling Editor.

Texture Resampling is a special form of baking in which you resample an already baked texture, e.g. from one uv-set to another. Texture resampling is controlled by the Texture Resampling Editor.

Using the Texture Resampling Editor (figure 2.10), you can bake a texture by resampling an already existing texture. Maybe you have baked a texture using one uv-set, but want to convert it to another uv-set. Instead of rebaking the texture again, you can simply resample the texture to the new uv-set, an operation that potentially is faster.

A typical example of this is if you have created a model and textures using ZBrush, but want to use an UV layout generated by Maya. Then you specify the ZBrush layout as source

UV set, and the Maya UV layout as the target UV set in the Texture Resampling Editor. The resampled texture will then correspond to the Maya UV layout.

Another example is resampling a texture from one resolution to another.

The Texture Resampling Editor is located under **Window** → **Rendering Editors** → **TURTLE** → **Texture Resampling Editor**.

Texture

The name of the texture that will be resampled.

Source UV Set

The name of the UV layout that corresponds to the texture to be resampled.

Target UV Set

The name of the UV layout that the texture should be resampled to.

Directory

The directory where the resampled texture will be stored.

File name

The name of the resampled texture. The name is put together by the user with the help of the following predefined variables:

$m − material name (shading group)
$s − shape name
$t − full path to shape
$u − uvset name
$e − file format extension
$f − frame number

When these variables are found in the file name they are replaced with the corresponding text string.

Example:

"lightmap_$m_$s.$e" translates to "lightmap_lambert1SG_meshShape1.tif"
"lightmap_$t.$f.$e" translates to "lightmap_mesh1_meshShape1.42.tif"
"normalmap_$s_$u.$e" translates to "normalmap_meshShape1_uvSet1.tif"

Width / Height

Sets the resolution of the resampled texture map.

File Format

Sets in which format the resampled texture will be stored.

Edge Dilation

Sets the number of pixels to stretch texture edges. This can be used to eliminate bleeding problems, if filtering is used on the resampled textures.

Background Color

Sets the texture background color. All unsampled texels will receive this color.

Chapter 3

Batch Rendering Reference

Turtle, like Maya, has a command line renderer instead of a network rendering distribution tool. The reason is that many studios prefer to use their own network rendering distribution software or one available from a third party developer. Turtle is built to be easy to use with independent distribution software through the command line renderer.

When rendering from the command line, all of Turtle's render globals can be accessed. Light, Camera and Material attributes cannot be changed from the command line.

3.1 Command Line Frame Rendering

Frame rendering from command line is done by using Maya's program Render.exe with the flag "-renderer turtle". All settings to both programs are given through argument flags. Below is tables of all flags and their types. The tables are only meant to be a reference. Additional setting information can be found under the **Render Reference**, chapter 1.

Usage

```
Render -proj "c:/MyProj" -renderer turtle <flags> <filename>
```

General Purpose Flags

Frame naming/numbering options:		
-rd	Directory in which to store image files.	STRING
-im	Image file output name.	STRING
-of	Output image file format. See the Render Settings window to find available formats.	STRING
-s	Starting frame for an animation sequence.	FLOAT
-e	End frame for an animation sequence.	FLOAT
-b	By frame (or step) for an animation sequence.	FLOAT
-pad	Number of digits in the output image frame file name extension.	INTEGER
-rfs	Renumber Frame Start: number for the first image when renumbering frames.	INTEGER
-rfb	Renumber Frame By (or step) used for renumbering frames.	INTEGER
-fnc	File Name Convention: any of name, name.ext, ... See the Render Settings window to find available options. Use namec and namec.ext for Multi Frame Concatenated formats. As a shortcut, numbers 1, 2, ... can also be used.	INTEGER

Camera options:		
-cam	Specify which camera to be rendered.	STRING
-rgb	Turn RGB output on or off.	STRING
-alpha	Turn Alpha output on or off.	BOOLEAN
-depth	Turn Depth output on or off.	BOOLEAN

Resolution options:		
-x	Set X resolution of the final image.	BOOLEAN
-y	Set Y resolution of the final image.	BOOLEAN
-region	Renders a region of the image. -region left bottom right top	INTEGER[4]
-percentRes	Renders the image using percent of the resolution.	FLOAT
-ard	Device aspect ratio for the rendered image.	FLOAT

Render Layers and Passes:		
-rl	Render each render layer separately.	BOOLEAN|STRING
-rp	Render passes separately. 'all' will render all passes.	BOOLEAN|STRING

Mel callbacks:		
-preRender	Mel code executed before rendering	STRING
-postRender	Mel code executed after rendering	STRING
-preLayer	Mel code executed before each render layer	STRING
-postLayer	Mel code executed after each render layer	STRING
-preFrame	Mel code executed before each frame	STRING
-postFrame	Mel code executed after each frame	STRING

Verbosity:		
-errorMsg	Enables error messages.	BOOLEAN
-warningMsg	Enables warning messages.	BOOLEAN
-debugMsg	Enables debug messages.	BOOLEAN
-verboseMsg	Enables verbose messages.	BOOLEAN
-infoMsg	Enables info messages.	BOOLEAN
-progressMsg	Enables progress messages.	BOOLEAN
-benchmarkMsg	Enables benchmark messages.	BOOLEAN
-saveMsg	Saves all output messages to a log file in *project_path\turtle\temp\debug.out*	BOOLEAN

Render option flags

Memory and Performance:		
-disableSSE	Disables usage of SSE instructions.	BOOLEAN
-tileSize	Sets the render tile size as a power of two (4=16x16, 5=32x32, 5=64x64, ...)	INTEGER
-renderThreads	Sets the number of threads to use for rendering.	INTEGER
-textureCache	Enable the texture cache.	BOOLEAN
-readOnlyCache	Put the texture cache into read only mode.	BOOLEAN
-textureTileSize	Set the texture cache tile size (16=16x16, 32=32x32, 64=64x64, ...).	INTEGER
-textureCacheSize	Sets the total texture cache size (in MB).	INTEGER
-minimumTextureSize	Sets the min size of a texture to be cached (in MB). A smaller texture will not be put into the cache.	INTEGER
-textureCompressionType	Set the compression type to use in the cache.	INTEGER
-textureCacheDirectory	Set the directory to store texture cache files.	STRING

Framebuffer:		
-preMultiply	Enables alpha premultiplication.	BOOLEAN
-preMultiplyThreshold	Sets the premultiply threshold.	FLOAT
-inputGammaCorrection	Enables gamma correction.	BOOLEAN
-gammaInput	Sets the input gamma.	FLOAT
-outGammaCorrection	Enables output correction. (0=None, 1=Gamma, 2=sRGB)	INTEGER
-gammaOutput	Sets the output gamma.	FLOAT

Overrides:		
-ignoreLightLinks	Disables all light linking.	BOOLEAN
-rtQualityLimits	Enables ray tracing quality limits.	BOOLEAN
-rtMaxShadowRays	Sets the max shadow rays if rtQualityLimits is enabled.	INTEGER
-rtMaxGlossyRays	Sets the max glossy rays if rtQualityLimits is enabled.	INTEGER

3.2 Command Line Texture Baking

Texture baking from command line is done by using Maya's program Render.exe with the flag `-renderer "turtlebake"`. All settings are given through argument flags. Below are tables of all flags and their types. The tables are only meant to be a reference. Additional setting information can be found in the Texture Baking section.

Usage

```
Render -proj "c:/MyProj" -renderer turtlebake <flags> <filename>
```

All the flags of the command line frame rendering apply to command line texture baking as well.

Note: The -bakeLayer flag must always be set.

Texture Bake option flags

Bake layers and objects:		
`-bakeLayer`	Specifies a bake layer to bake. Use 'all' to bake all layers.	`STRING`
`-include`	Includes additional object to the previous specified bake layer. If no bake layer is specified a new empty one will be created.	`STRING`
`-exclude`	Excludes objects from the previous specified bake layer.	`STRING`

Bake layer override settings:		
-override	If enabled the override settings will be used instead of the bake layer settings.	BOOLEAN
-width	The width of the texture.	INTEGER
-height	The height of the texture.	INTEGER
-directory	Sets the directory where the textures will be saved.	STRING
-fileName	Sets the file name of the texture.	STRING
-imageFormat	Sets which image format to use when saving the texture.	INTEGER
-uvRange	Controls which uv range to bake.	INTEGER
-uMin	Sets the minimum u-value to bake.	FLOAT
-uMax	Sets the maximum u-value to bake.	FLOAT
-vMin	Sets the minimum v-value to bake.	FLOAT
-vMax	Sets the maximum v-value to bake.	FLOAT
-uvSet	Sets which uv-set to use when baking.	STRING

-lua	Enables a lua scripted output pass.	BOOLEAN
-luaFile	Sets the lua script file to use.	STRING

Chapter 4

Turtle Nodes Reference

Turtle adds several nodes to Maya. These nodes are Turtle specific. Rendering a file with Turtle nodes in Maya or Mental Ray will not work.

4.1 Surface Shader Nodes

4.1.1 ilrOccSampler

The ilrOccSampler Material performs occlusion sampling.

Type

The type of occlusion. It can be set to either **Ambient Occlusion** or **Reflective Occlusion**.

Output

The output from the node. The possible settings are:

- **Occlusion**: The sampler outputs the occlusion value in the Color channel.
- **Vectors**: The sampler outputs the bent normal (if **Bend Normals** is checked) in the Color channel and the occlusion value in the alpha channel.
- **Vectors and Occlusion**: The sampler outputs the normal in the color channel and the occlusion value in the alpha channel.
- **Environment * Occlusion**: The sampler outputs the occlusion value multiplied with the environment map in the color channel. If **Reflective Occlusion** is used, the reflection direction is used instead of the surface normal as the Vector.

Coordinate System

The coordinate system to use when saving vector information to file.

- **World Space**: Use World Space.
- **Tangent Space**: Use Tangent Space.

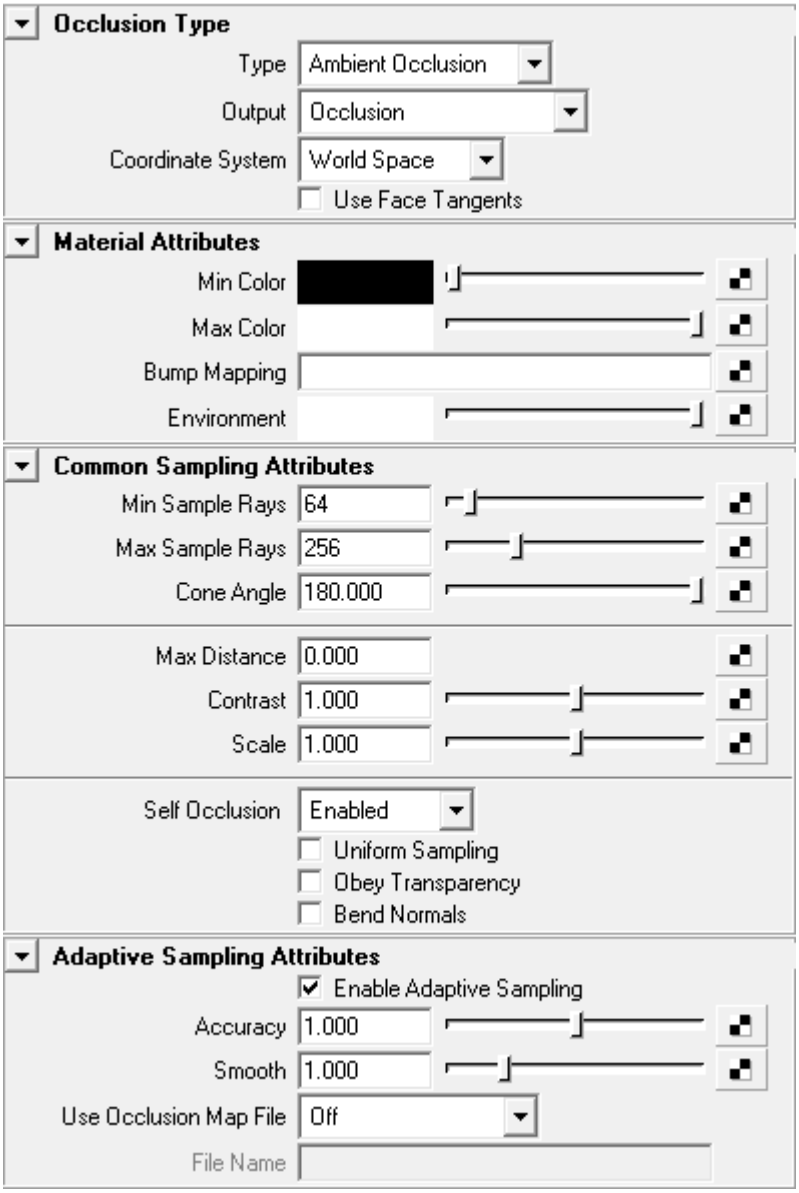

Figure 4.1: ilrOccSampler settings

Use Face Tangents

If Tangent Space is chosen, the Use Face Tangents checkbox tells Turtle to calculate face tangents instead of using the interpolated vertex tangents.

Min Color / Max Color

The sampler interpolates between the two colors according to the occlusion value.

Bump Mapping

The bump channel on the occlusion sampler.

Environment Map

The environment map channel.

Min Sample Rays

The minimum amount of rays that will be sent by the sampler.

Max Sample Rays

The maximum amount of rays that will be sent by the sampler.

Cone Angle

Decides how much of the hemisphere that is to be sampled. 180 degrees covers the entire hemisphere.

Max Distance

Sets the max occlusion distance. Beyond this distance there is full visibility. Hence it also controls the maximum distance an occlusion ray is traced. If set to 0.0 the value is calculated according to the size of the scene. For closed scenes this value needs to be set according to some fraction of the scene size, otherwise there will be full occlusion everywhere.

Contrast

Can be used to adjust the contrast for ambient occlusion. Increasing this value will make dark surfaces darker and bright surfaces brighter.

Scale

A scaling of the occlusion values which can be used to increase or decrease the shadowing effect. By increasing this value you will boost the shadowing effect.

Self Occlusion

Sets how to handle self occlusion, i.e occlusion rays intersecting the same object as they are emitted from. It has the following options:

- **Disabled**: Objects does not cast occlusion on itself.
- **Environment**: Hits on the same objects are treated as misses, i.e the ray is terminated and the environment is set.
- **Enabled**: Objects occludes themselves normally. This is the default method.

Uniform Sampling

Sets whether uniform sampling should be used. If disabled cosine weighted sampling is used, where more rays are concentrated near the normal of the surface. If uniform sampling is used all surfaces around the shading point gives an equal amount of occlusion. If cosine weighted sampling is used, surfaces in front of the shading point gives more occlusion than surfaces to the side.

Obey Transparency

If enabled, the sampler will consider the transparency of the surface hit by the occlusion ray and continue if it is transparent.

Bend Normals

If enabled, Turtle will calculate the most unoccluded direction. Bend Normals does not work with Adaptive Sampling.

Enable Adaptive Sampling

If Adaptive Sampling is enabled, the occlusion sampler will not sample every point. Instead it will interpolate neighboring points much like final gather.

Accuracy

Controls number of samples created. Increase to force more samples and get higher quality.

Smooth

Radius of smoothing filter.

Use Occlusion Map File

Enables Turtle to save occlusion samples to file and reuse them in subsequent renders.

File Name

Name of file to save samples to.

4.1.2 ilrBssrdfShader, Subsurface Scattering

Subsurface scattering simulates light that penetrates the material and scatters around many times before bouncing back out. Most non-metallic materials will exhibit some amount of sub surface scattering. Some examples of materials with high sub surface scattering include wax, marble, milk and skin.

The diffuse and specular part of the sss shader works just as a standard lambert / blinn shader.

Material Presets

Some preset parameters for common real life materials.

Diffuse Reflectance

Diffuse surface reflectance for the material. If presets are used, only the Scatter Length needs to be adjusted in most cases.

Scatter Length

Sets the average distance the light travels inside the material before any scattering. The value is given for each channel/wavelength (RGB).

Scatter Length Scale

Scaling value of scatter length values.

Intensity

A multiplier that can be used to scale the subsurface scattering light. The remainder will be shaded in accordance to the Lambert model. (If set to 0.8, 80% will be shaded as an SSS material and 20% will be shaded as a Lambert material.

Distribution Type

Allows you to select if subsurface scattering samples are distributed in triangle space or in the UV space of the mesh. You most often want to use triangle space, otherwise the subsurface scattering effect will be distorted if the mesh is badly UV mapped.

Approximation

Sets the approximation level to use when calculating the subsurface scattering light. A lower value gives better results but longer render time.

Sample Density

This attribute controls the number of sample points created. The Diffuse Reflectance and Scatter Length is used to calculate the number of sample points needed. However sometimes more samples are needed to get a good result. A higher value gives more sample points.

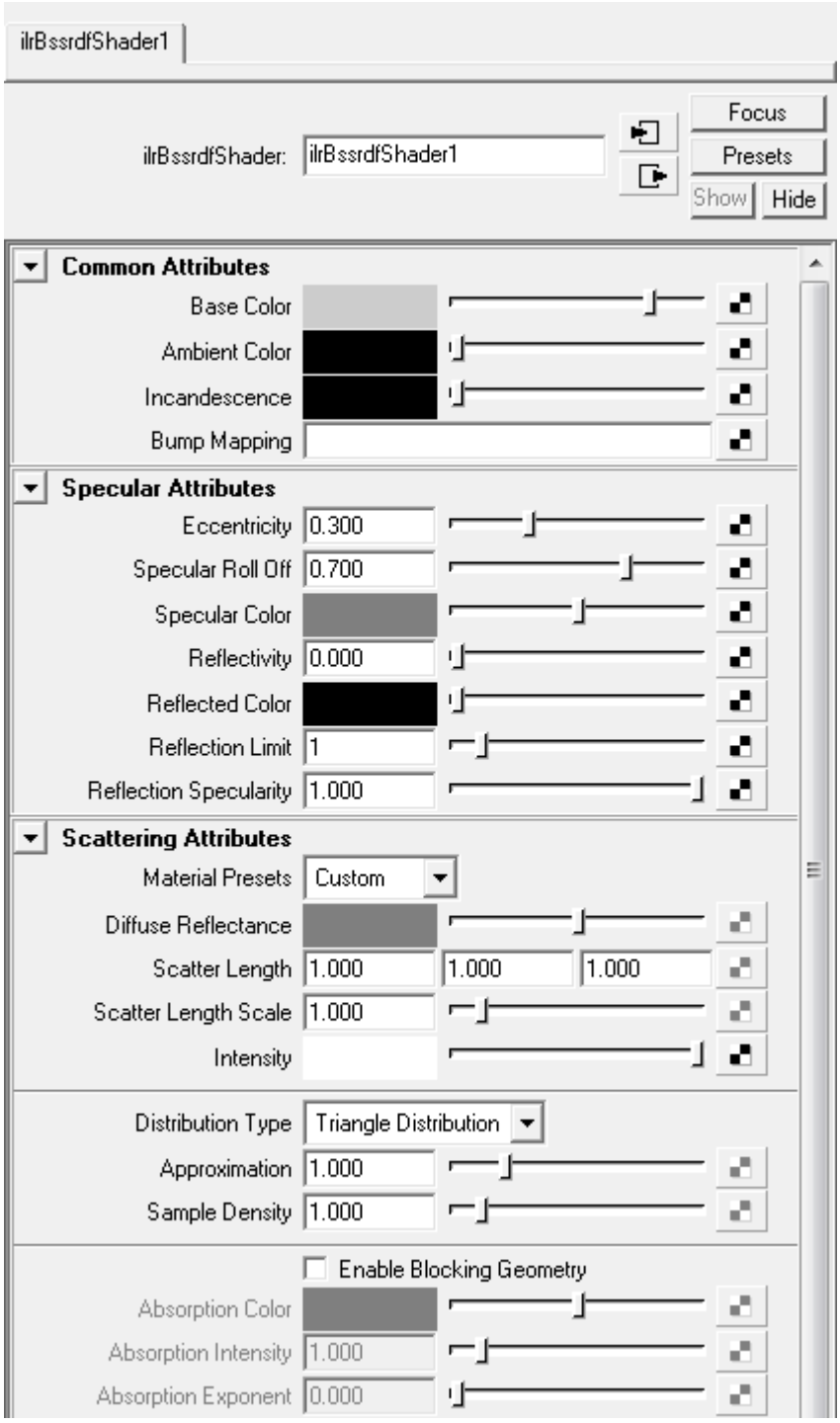

Figure 4.2: Subsurface Scattering Settings

Enable Blocking Geometry

Can be used to simulate opaque objects inside a translucent object, e.g., skeleton inside a hand. All objects that are inside the SSS object will be regarded as blockers.

Absorption Color

Sets the color that the blocker will absorb, i.e. if a red color is given the blocker will absorb the red light component.

Absorption Intensity

Sets the amount of absorption. A higher value will give more absorption. Note that a negative value will generate light instead, which means the blocker can be turned into a light emitter.

Absorption Exponent

Sets a decay value for the blocker. Use this to control the blocker effect. A high value will make the blocker less visible.

4.1.3 ilrOrenNayar Shader

The Oren-Nayar shader implements the reflectance model proposed by Michael Oren and Shree K. Nayar in their Siggraph94 paper *"Generalization of Lambert's Reflectance Model"*.

As opposed to a Lambertian surface, which is perfectly flat, the Oren-Nayar model assumes a rough surface composed of many Lambertian microfacets. The Oren-Nayar model also accounts for complex masking, shadowing and inter-reflection effects between the facets.

Thus, you apply the Oren-Nayar shader to objects in your scene that have rough, diffuse surfaces to give them a more realistic appearance. Figure 4.3 illustrate the difference between the Lambert reflectance model and the Oren-Nayar model. In accordance with a real paper cup, the Oren-Nayar cup has a more flat appearance, with almost constant brightness over the entire surface.

The Oren-Nayar shader has two settings that are specific to this shader.

Material

Select from over 60 different, common materials with predefined roughness values. These real-life materials have been taken from a database compiled by researchers at Colombia University and Utrecht University. The researchers have measured the reflectance of these materials and matched the Oren-Nayar model to the measurements. The database is publicly available at their website.

Roughness

A large roughness value corresponds to a very rough surface. A roughness of 0.0, on the other hand, means a perfectly flat surface. With the roughness set to 0.0, the Oren-Nayar shader will produce the same result as a Lambert shader.

Figure 4.3: Two paper cups, one with Oren-Nayar shader (left) and one with a standard Lambert shader (right)

The **Common Materials Attributes** work as they do for a standard Maya shader.

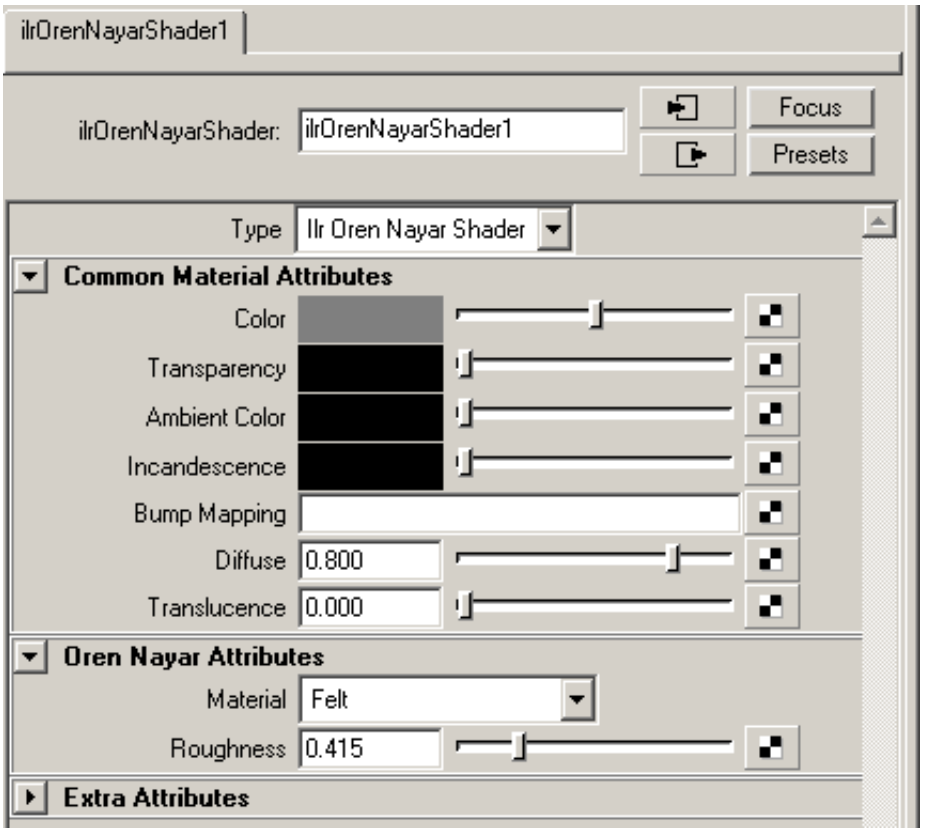

Figure 4.4: Oren Nayar settings

4.1.4 ilrAshikiminShader

The Ashikimin surface shader is especially suited for anisotropic glossy metallic objects. The main difference between the Ashikimin shader and the standard Maya anisotropic shader is that the specular highlights and glossy reflection always match up in the Ashikimin shader.

Diffuse

The diffuse color of the shader.

Bump Mapping

The bump map.

Specular

The specular color of the shader.

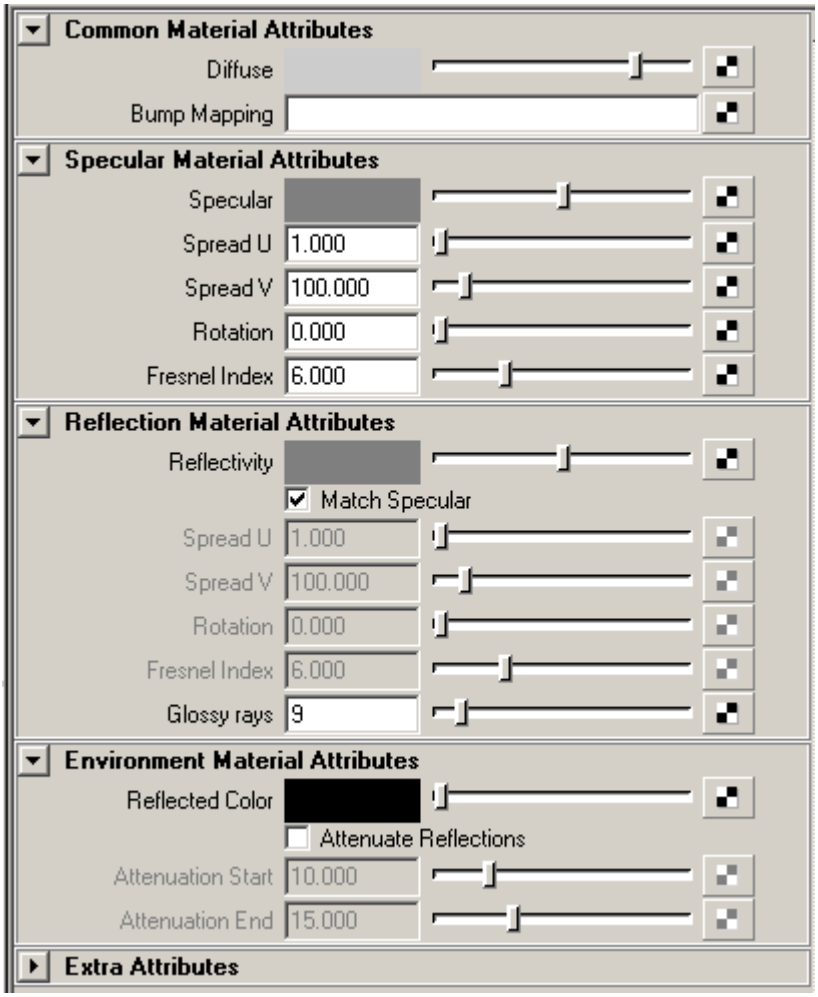

Figure 4.5: ilrAshikiminShader Settings

Spread U

Controls how much the specular highlight will spread in the U direction. A high value will result in a large spread.

Spread V

Controls how much the specular highlight will spread in the V direction. A high value will result in a large spread.

Rotation

Rotates the specular highlight.

Fresnel Index

Controls the effect of the specular highlight. A low value will suppress the highlights except for grazing angles. A high value will make the highlight appear at all angles.

Reflectivity

Controls the reflectivity of the shader.

Match Specular

If enabled, the glossy reflections will match the specular highlight. If it is disabled, the reflection Spread U, Spread V, Rotation and Fresnel Index need to be set.

Glossy Rays

The number of glossy rays to send. A high value will yield nice result, but will take a long time to render.

Reflected Color

Controls the reflected color of the shader.

Attenuate Reflections

This will attenuate the reflections beginning from the Attenuation Start and finish at the Attenuation End. At distances lower than the start value, the proper glossy reflection will be used. At distances greater than the end value, the reflected color will be used. At distances in-between, Turtle will cross fade between the correct reflections and the reflected color. If nothing is connected to the reflected color, the camera environment or the Turtle IBL image will be used. Just altering the reflected color will not have any effect, an environment node must be connected to achieve the effect. The attenuation mode is used to reduce the noise that comes from very unfocused glossy reflections.

Attenuation Start

The start distance in world units.

Attenuation End

The end distance in world units.

4.1.5 ilrDielectricShader

The ilrDielectricShader is used to simulate materials like glass, gem stones and various liquids such as water. This shader uses the Fresnel formulas to calculate reflectance and transmittance, e.i., the fraction of light that is reflected and transmitted in a point on a surface. Generally, the Fresnel formulas specify that at normal incidence almost all light is transmitted through the interface into the material and only a small fraction of light is reflected. On the other hand, if an incident beam of light lies almost in the tangent plane of a surface point, a much larger fraction of the incoming light will be reflected. Thus, a surface with an ilrDielectricShader applied becomes more and more reflective as the angle of view increases. This is called the Fresnel effect.

The ilrDielectricShader represents an interface between two different materials. Thus,

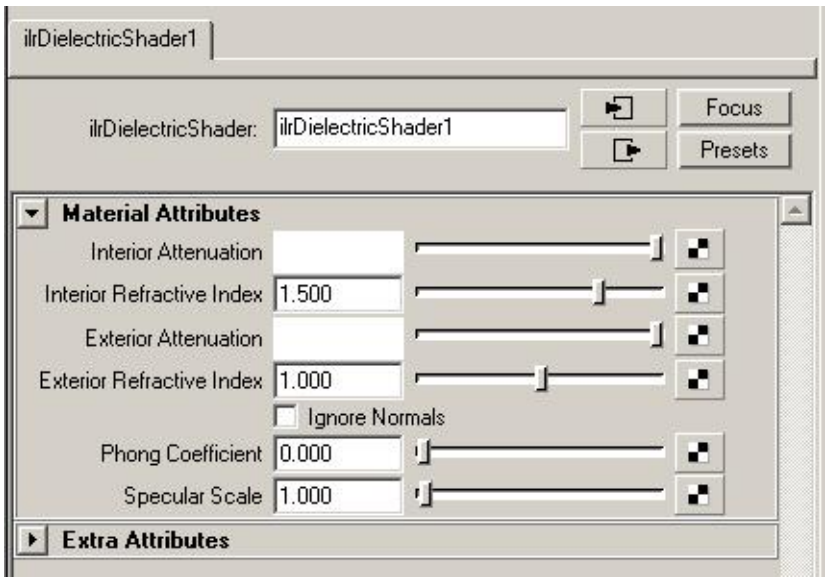

Figure 4.6: ilrDielectricShader settings

you can specify, e.g., the index of refraction for both the exterior and the interior side of the surface interface, see figure 4.6. Per default, the exterior material is the one into which the surface normal points. It is important to model your object correctly and apply ilrDielectricShaders with the correct settings on different parts of an object. For example, if you model a glass of cognac, you need to work with three different types of ilrDielectricShaders: a glass-air interface, a glass-cognac interface and a cognac-air interface.

Interior- and Exterior Refractive Index

The refractive index of the exterior- and interior side of the interface, respectively.

Interior- and Exterior Attenuation

This shader uses Beer's law of attenuation to achieve colored materials. In this model, the material color is specified by an attenuation constant. This specifies the fraction of the incoming light that is remaining after traversing one unit length of the material. Thus, an attenuation constant of (1, 1, 1) represents a perfectly transparent material and (0, 0, 0) a totally opaque material. The attributes **Interior Attenuation** and **Exterior Attenuation** specifies the attenuation constant for the interior- and exterior side of a surface interface, respectively.

Ignore Normals

When a ray encounters an ilrDielectricShader interface, it checks the normal in the intersection point to figure out which side of the interface that represents the exterior and the interior side, respectively. Per default, the exterior side is the one into which the surface normal points. If the **Ignore Normals** box is checked, however, the surface normal is ignored. Instead, whether or not a ray traverses the interior of a dielectric material is decided by counting the number of ilrDielectricShaders that the ray has passed so far. Check this box for geometry that have normals that do not specify "the outside" of a surface in a unified way.

Phong Coefficient

If this value is larger than 0.0, a specular high-light is produced.

Specular Scale

This value is a multiplier that can be used to amplify or diminish the intensity of the specular hightlight.

4.2 Photon Shader Nodes

4.2.1 ilrBasicPhotonShader

Diffuse

The chance that the photon performs a diffuse reflection. The photon energy is also scaled with the value. A value higher than 1 will always spawn a diffuse reflection and it will also increase the total energy.

Specular

The chance that the photon performs a specular reflection. The photon energy is also scaled with the value. A value higher than 1 will always spawn a specular reflection and it will also increase the total energy.

Transparency

The chance that the photon will transmit. The photon energy is also scaled with the value. A value higher than 1 will always spawn a transmission and it will also increase the total energy.

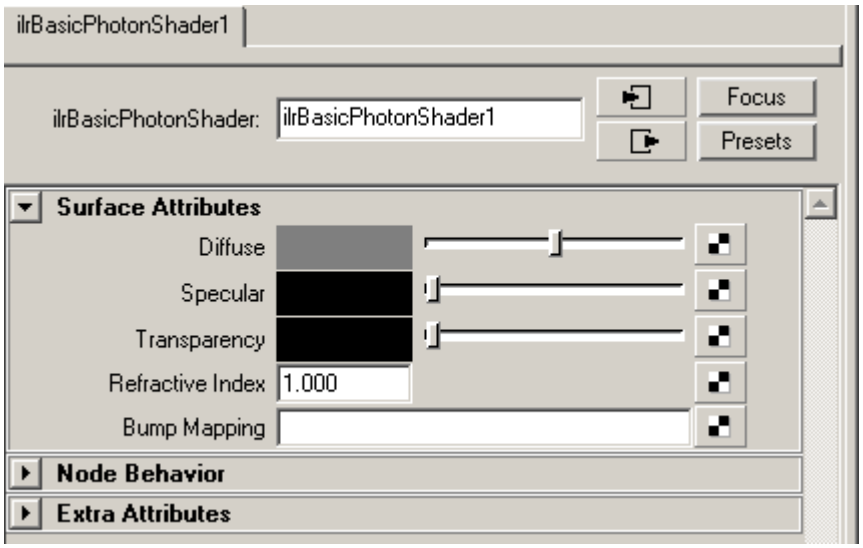

Figure 4.7: ilrBasicPhotonShader Settings

Refractive Index

The refraction index to be used when transmitting a photon.

Bump Mapping

The bump map to be used.

4.2.2 ilrPhysicPhotonShader

The ilrPhysicPhoton shader looks exactly as the Basic Photon shader. The difference is that this shader gives a better photon energy distribution. If any of the values of the attributes diffuse, specular or transparency is higher than 1.0, the ilrBasicPhotonShader will boost the photon energy, while the ilrPhysicPhotonShader gives a more physically correct energy distribution with lower variance.

Diffuse

The chance that the photon performs a diffuse reflection. The photon energy is also scaled with the value.

Specular

The chance that the photon performs a specular reflection. The photon energy is also scaled with the value.

Transparency

The chance that the photon will transmit. The photon energy is also scaled with the value.

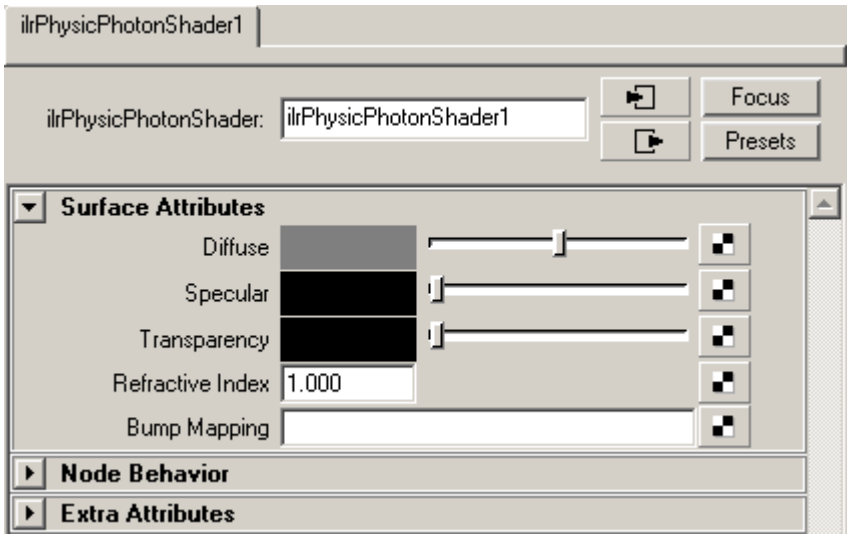

Figure 4.8: ilrPhysicsPhotonShader settings

Refractive Index

The refraction index to be used when transmitting a photon.

Bump Mapping

The bump map to be used.

4.2.3 ilrDielectricPhotonShader

Figure 4.9 shows the attributes of this shader as they appear in the Maya Attribute Editor. These attributes work exactly as for the ilrDielectricShader described below. See ilrDielectricShader, section 4.1.5 for a complete explanation. The difference is that this shader handles photon scattering from a dielectric surface interface.

4.3 Utility Nodes

4.3.1 ilrOutputShaderBackendNode

The **ilrOutputShaderBackendNode** is a utility node only used in output shaders. It has no inputs, only two outputs provided for shading networks.

If an output shading network has not included this node, it cannot know the original color to work on (from the source image) and the resulting color will probably be black (depending on the shading network).

4.3.2 ilrUVMappingVisualizer

The **ilrUVMappingVisualizer** is a utility node designed to help decide an appropriate texture resolution for the given object. The **Min Texels** and **Max Texels** tells the node how

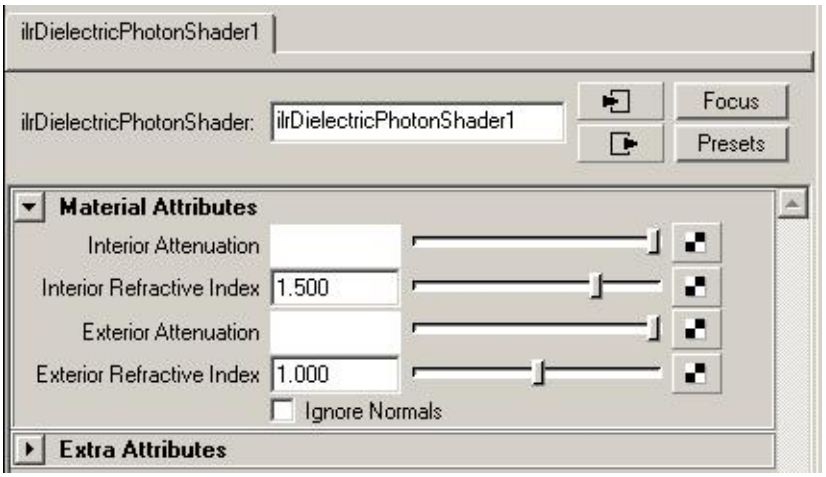

Figure 4.9: ilrDielectricPhotonShader settings

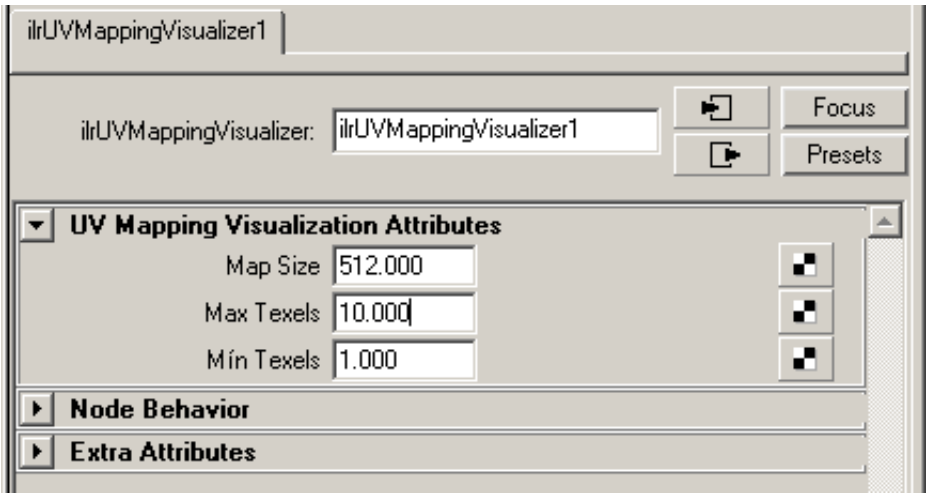

Figure 4.10: ilrUVMappingVisualizer settings

many texels per world space unit that is wanted.
The Max Texel is a subjective measure that sets the quality of the final render. A higher value makes it possible to move closer to the object in an animation without magnifying the texture. The cost is texture memory. The Min Texels sets the minimum bound. The **Map Size** attribute is the desired texture map resolution (as a quadratic texture). The shader outputs black if the calculated texture coverage is insufficient, and white if it is sufficient. In most cases the Min and Max Texels are set once for an entire scene. This means that there are only two ways to fix an object with insufficient coverage. Either scale up the resolution or change the UV-layout.

4.3.3 ilrOccData

The ilrOccData node is used to plug in occlusion passes / textures into a shading network.

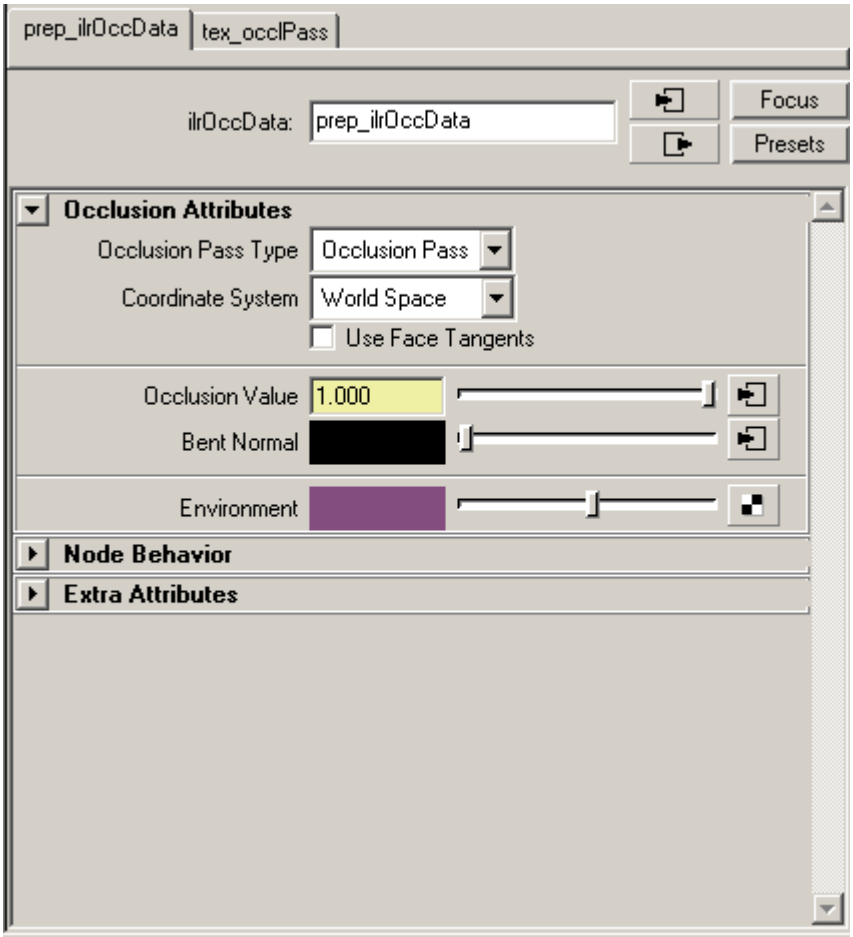

Figure 4.11: ilrOccData Settings

Occlusion Data Type

If a pass is plugged in the Occlusion Data Type should be set to **Occlusion Pass**, if a texture is plugged in the Occlusion Data Type should be set to **Occlusion Map**.

Coordinate System

The coordinate system to use when reading vector information from file.

- **World Space**: Use World Space.
- **Tangent Space**: Use Tangent Space.

Use Face Tangents

If Tangent Space is chosen, the Use Face Tangents checkbox tells Turtle to calculate face tangents instead of using the interpolated vertex tangents.

Occlusion Value

The occlusion value from the pass / texture (the greyscale image).

Bent Normal

The normal value from the pass / texture (the colourful image).

Environment

The environment map for the occlusion pass.

4.3.4 ilrRaySampler

The ilrRaySampler is a material that makes it possible to sample properties on a high definition surface (normals, color etc.) and apply them on a low definition surface.

Below is a quick step by step to get started with the ilrRaySampler.

- Open up the high definition model and place it at (0,0,0).
- Create a low definition model resembling the high definition surface. For example, if you have a high definition head, create a low definition version of the same head with only the most basic features as modeled geometry.
- Place the low definition surface at (0,0,0).
- Assign an ilrRaySampler node to the low definition surface. This is done by assigning a surface shader to the low definition surface and connecting an **ilrSurfaceSampler** to its color node. If you want to sample the high definition normals you need alter the standard connection. Bring up the Hyper Shader and connect the ilrRaySamplers out normal to the color of the surface shader.
- Alter the render stats on the high definition surface. Turn off “Primary Visibility” and make sure that “Visible in Reflections” and “Visible in Refractions” is turned on.

- Render. You should now see the low definition mesh rendered with the normals of the high definition surface as its color.

You can now bake the result to a texture as you would with a standard surface material.

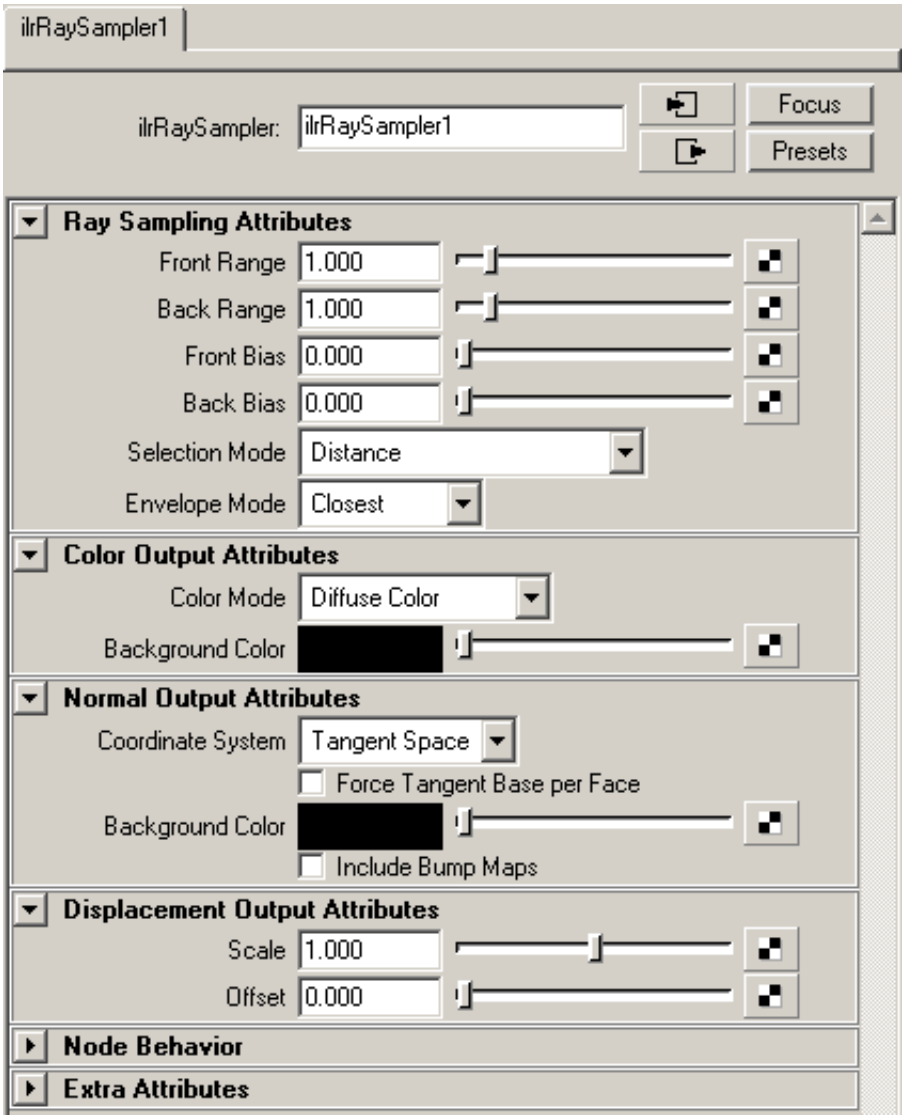

Figure 4.12: ilrRaySampler Settings

Front Range

The maximum range for a ray to travel from the low definition surface before it is considered to have missed the high definition surface. Front means that it is sent in the direction of the low resolution geometry's normal.

Back Range

Same as front range but the ray travels in the opposite direction.

Front Bias

Offsets the ray's starting point slightly along the ray to avoid self intersecting.

Back Bias

Same as front bias but for the back ray.

Selection Mode

Selects how Turtle chooses what surface to sample.

- **Distance**: Chooses the closest surface in front or in the back.
- **Best Normal**: Chooses the surface with the same normal as the sample point.
- **Distance with flipped Normal**: Chooses the closest normal, and flips the normal if the direction is backwards.

Color Mode

When sampling final shading from the high resolution geometry, this drop down controls what elements that are to be transferred.

- **Albedo**: Samples the albedo color from the high resolution geometry.
- **Full Shading**: Samples the complete shading model on the high resolution geometry.
- **Illumination**: Samples the incoming direct illumination on the high resolution geometry without applying the high resolution geometry's material color etc.
- **Indirect Illumination**: Samples the incoming indirect illumination (from FG, GI and AO) illumination on the high resolution geometry without applying the high resolution geometry's material color etc.

Background Color

The color the shader defaults to when missing the high resolution geometry.

Coordinate System

The coordinate system to store the sampled high resolution normal in.

- Tangent Space. The resulting normals will be in tangent space. The tangent space created can be changed by altering the tangent overrides in the Options tab in Render Settings window.
- Object Space. The resulting normals will be in object space.
- World Space. The resulting normals will be in world space.

Modify Channels

Defines how the normal components are mapped to the color components.

- Default. X, Y, and Z components are mapping directly to R, G and B components.
- Invert Red. Inverting the red channel.
- Invert Green. Inverting the green channel.
- Flip Red-Green. Switch places for red and green channels. Mapping X to G and Y to R.

Use Face Tangents

By checking this box, Turtle will use the face tangents of the object instead of the vertex tangents.

Include Bump Maps

Enable this to include bump maps that are applied on the high definition geometry.

Background Color

The color the ray sampler defaults to when missing the high definition geometry.

Displacement Scale

A scale factor to multiply the sampled displacement with.

Displacement Offset

A factor to offset the sampled displacement with.

Background Value

A default displacement value to set when missing high resolution geometry.

4.3.5 ilrNormalMap

The ilrNormalMapNode is a utility node that remaps file texture normal maps into camera space. Plug it into the BumpMapping (normalCamera) attribute of a surface shader.

Note: The ilrNormalMap node should be mapped directly to the normalCamera attribute. You should NOT use a Bump2D node in between. Use the hypershade to map it.

Coordinate System

The coordinate system the normal map was created in.

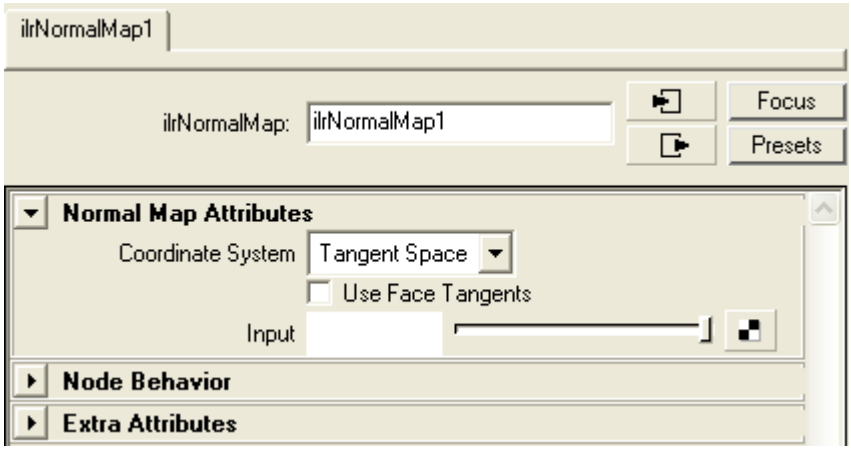

Figure 4.13: ilrNormalMap settings

Use Face Tangents

Uses a fixed tangent space per face instead of interpolating the vertex tangent spaces.

Input

The input file. Map a file node containing the file texture normal map to this attribute.

4.3.6 ilrSurfaceThickness

The ilrSurfaceThickness node is a utility node that samples the distance to occluding geometry behind the sample point. This value could be used for simple sub surface scattering effects or for any purpose in any general shading network. There are a number of ways to sample the thickness, either with one ray to get a single measurement, or to send a number of rays in a cone, to get a weighted measurement.

Number of Rays

Defines how many rays to send to sample the thickness.

Cone Angle

Defines how large cone to sample in. 0 degrees is a single ray, 180 degrees is a full hemisphere.

Maximum Rays Distance

Defines how far a ray is allowed to go before it is considered to be infinite.

Send Inwards

Flips the direction of the sample rays. If the normals are pointing inwards, this check box must be unchecked.

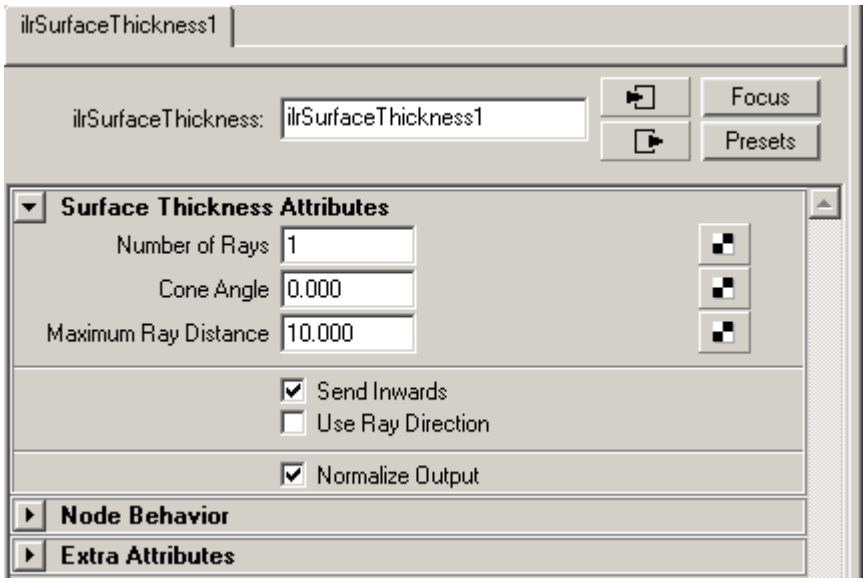

Figure 4.14: ilrSurfaceThickness Settings

Use Ray Direction

Normally the samples are sent along the surface normal, but if this check box is enabled, the sample rays are sent in the incoming ray direction.

Normalize output

Scales the output with Maximum Ray Distance.

4.3.7 ilrShadowMask

The **ilrShadowMask** is a utility node designed to help creating shadow mask passes. Used with the default settings it will give the same output as the standard Shadow pass in the Turtle pass system. The major drawbacks with the default shadow pass which the Shadow-Mask solves is the following;

Back facing polygons are not handled by shooting shadow rays since they are shaded black anyway in standard shaders. This leads to the standard shadow pass returning a fully lit result for a pixel that actually is in shadow.

Jagged shadow borders due to low-tesselated polygons. This is a by-product of ray traced shadows normally covered up by the shading.

The ilrShadowMask fixes those two issues when using **Mask Back Facing Surfaces**. It will shadow everything correctly so back faced surfaces will be black, and by tweaking the **Start Drop Off** and **Stop Drop Off**. The ilrShadowMask emulates the soft shadow border by applying the lambert shading model in between the values.

Min Color

Color to be used when completely in shadow.

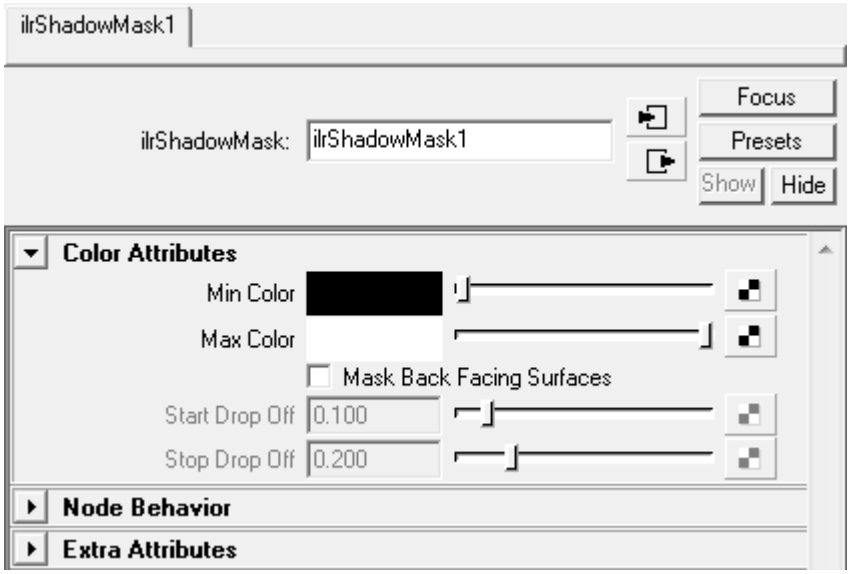

Figure 4.15: ilrShadowMask settings

Max Color

Color to be used when completely lit.

Mask Back Facing Surfaces

Mask the back facing surfaces by applying a simple lambert shading algorithm.

Start Drop Off

Specifies from where the shadow should start when masking hard, jagged shadow edges.

Stop Drop Off

Specifies from where the shadow should end when masking hard, jagged shadow edges.

4.3.8 ilrLuaNode

The ilrLuaNode allows for custom shaders through the use of Lua scripts. For thorough documentation on this node, see section 1.17.

4.3.9 ilrHwBakeVisualizer

The **ilrHwBakeVisualizer** node is meant first and foremost as a hardware visualization aid when baking textures. If the visualization option is enabled (see section 2.2.3), an instance of this node will be automatically generated and assigned to the 'Hardware Shader' input of the target object's surface shader after a bake is completed. The shader supports up to three different UV-sets, Turtle will automatically connect the appropriate UV-set when baking. If multiple objects share the same material, switch nodes will be connected by Turtle to select

specific textures for each baked object. The advanced techniques available in this shader are described in more detail in section 2.3.

This node will not produce any meaningful results if used in a software rendering, and is currently not supported on the Linux and Mac platforms.

A few attributes on this node are worth some closer attention:

Height Map

To visualize height maps ilrHwBakeVisualizer uses parallax mapping with offset limiting.

- **Height Map**: the height map to use for parallax mapping.
- **Offset**: offset of the height map, subtracted from height map value before scaling is applied.
- **Scale**: scaling of the height map.

Spherical Harmonics (Vertex Lighting)

Enabling this effect will use the spherical harmonics from a point cloud file to light the shape per-vertex. Only directional lights will be taken into account.

- **Coordinate System**: whether to evaluate spherical harmonics in object space or world space.
- **Input File**: point cloud file to use for pre-vertex lighting.
- **Reload**: reload the specified point cloud file.

Hardware Shader Features

In this rollout you can specify which features should be enabled in the visualization. Changing any of these values will force the hardware shader to be recompiled with the current feature set. If you experience problems with the shader output, a likely reason is hardware limitations. When this is the case, try reducing the number of features or the number of light sources.

- **Alpha**: toggle the use of transparency color/texture.
- **Ambient**: toggle the use of ambient color/texture.
- **Incandescence**: toggle the use of incandescence color/texture.
- **Light Map**: toggle the use of a light map.
- **Specular Color**: toggle the use of a specular component.
- **Height Map**: toggle parallax mapping.
- **Normal Map**: toggle the use of a normal map.
- **Directional Occlusion**: toggle the use of directional occlusion maps.
- **Radiosity Normal Map**: toggle the use of radiosity normal maps.
- **Spherical Harmonics (Vertex Lighting)**: toggle normal mapping.

- **PTM**: toggle computation of diffuse component using PTM input.
- **RGB**: toggle PTM RGB mode (full color) instead of LRGB (intensity).
- **Extract Normal**: enable if normal should be extracted from PTM. If enabled, the shader will ignore the Normal Map setting and use only the normal extracted from the PTM. Only in effect when PTM is enabled.

Hardware Texture Cache

All textures used by Turtle hardware shaders are handled by a common texture manager. In this roll out you have control over updating textures in the cache. Updating textures may be needed when textures have been modified in an external editor.

- **Update Current Textures**: update (reload from file) all textures connected to the current shader.
- **Update All Textures**: update all textures in cache.
- **Clear All Textures**: remove all textures from cache. This will clear the cache and force a reload of all textures during the next viewport refresh.

4.3.10 ilrPolyColorPerVertex

Used to render out objects with vertex colors. Turtle will only use the currrent color set when rendering.

4.4 Geometry Nodes

4.4.1 ilrPointCloudShape

The point cloud shape is used to visualize a point cloud file generated by Turtle. It is used by photon maps and final gather points, as well as point clouds generated from baking. A point cloud shape is created either via the Turtle shelf, or with the following command in the script editor:

```
createNode ilrPointCloudShape;
```

Input File

The point cloud file to visualize.

Point Cloud Type

This field displays the type of the loaded point cloud. Depending on the type different options are available. For Irradiance and Radiance point clouds additional settings can be found under Final Gather Options.

Reload

Reload specified point cloud file. May be necessary if the file has been changed externally.

Color

The color to use for the point visualization.

Use Point Colors

If this setting is enabled, the color information from the point cloud file will be use instead of the color specified above. This can be useful e.g. for visualizing the intensity of a final gather cloud.

Display Gamma

View point cloud with selected gamma inside Maya. Note that point cloud files are always stored in linear gamma (1.0).

Point Size

Sets the size of the points to draw.

Displace Along Normal

If the point cloud contains information about normals, this setting can be used to displace the points along their normals in the viewport. This may be useful when using smooth shading in the viewport to avoid intersection (depth culling) of points lying on surfaces.

Normal Length

Used to set the length of the visualized point normal.

Direction Length

If points contain directions (e.g. incident direction for photons), this setting is used to set the length of the visualization of the direction.

Final Gather Options

Depth Uses the depth property of points to determine if they should be visualized. Useful if you want to visualize the result of Final Gather at a certain depth.

Max Movement and Time Only show points that given this Max Movement are valid at the time specified by the Time attribute. Has not effect if Max Movement is zero.

Float Array Options

Num Elements The number of float elements per point in the loaded point cloud.

Interpretation If the loaded point cloud is a Float Array Point Cloud, how should the values be interpreted? All available choices are shown in this dropdown. Once an interpretation is selected, additional settings can be made under Spherical Harmonics Options or Radiosity Normal Options.

For colored SH point clouds the visualizer assumes that all R-coefficients come first, then all the G-cofficients and then the B-cofficients. An eventual A-channel is ignored.

For colored RNM point clouds the visualizer assumes that all the coefficients for the first basis vector comes first (RGB), then all coefficients for the second vector etc. If an extra fourth basis vector is available it is assumed to encode the color in the normal direction.

Coordinate System Specific for spherical harmonics and radiosity normal clouds. Sets the space to evaluate the SH/RNM function in (usually the space it was baked in).

Visualize as Spheres If this settings is enabled, SH-points and RMM-points are visualized using spheres. If disabled, directional lights will be used to specify the evaluation direction of the SH/RNM-function.

Chapter 5

Turtle Commands Reference

Everything you do in Turtle is executed through MEL commands. This way it's easy to customize more or less anything you want in Turtle using MEL scripting. For instance you might want to invoke some part of Turtle's functionality into an existing rendering pipeline. This chapter is a reference for all the commands Turtle adds to Maya, including all argument flags they have.

5.1 Render Commands

5.1.1 ilrRenderCmd

Performs single frame camera renderings. All render settings are taken from the node **TurtleRenderOptions**.

```
Synopsis: ilrRenderCmd [flags]
Flags:
 -c -camera      String           The camera to render from
 -r -resolution  Int Int          The resolution to render
 -reg -region    Int Int Int Int  Specifies a region to render
 -f -frame       Float            The animation frame time
 -l -layer       String           Sets a specific render layer
```

5.1.2 ilrBatchRenderCmd

Renders animation sequences. All render settings are taken from the node **TurtleRenderOptions**.

```
Synopsis: ilrBatchRenderCmd [flags]
Flags:
   -l -layer   String
   -r -region  UnsignedInt UnsignedInt UnsignedInt UnsignedInt
```

5.2 Bake Commands

5.2.1 ilrTextureBakeCmd

Performs texture baking. All texture bake specific options are taken as argument flags. All other render settings are taken from the node **TurtleRenderOptions**.

```
Synopsis: ilrTextureBakeCmd [flags]
Flags:
 -alb -albedo                     on|off
 -alp -alpha                      on|off
 -amb -ambient                    on|off
  -bb -backBias                   Float
 -bgc -backgroundColor            Float Float Float
  -bl -bakeLayer                  String
  -br -backRange                  Float
   -c -camera                     String
  -cr -conservative               on|off
  -cu -custom                     on|off
 -cus -customShader               String
   -d -displacement               on|off
 -dia -displacementAlpha          on|off
 -dif -diffuse                    on|off
  -do -displacementOffset         Float
 -drm -displacementRemap          on|off
  -ds -displacementScale          Float
   -e -envelope                   String (multi-use)
  -ed -edgeDilation               UnsignedInt
  -em -envelopeMode               UnsignedInt
   -f -frame                      Float
  -fb -frontBias                  Float
  -ff -fileFormat                 UnsignedInt
 -fls -fullShading                on|off
  -fn -fileName                   String
  -fr -frontRange                 Float
   -h -height                     UnsignedInt
 -iil -indirectIllumination       on|off
 -iin -ignoreInconsistentNormals  on|off
  -il -illumination               on|off
 -inc -incandescence              on|off
   -l -layer                      String
  -mm -mismatchMode               UnsignedInt
 -mrg -merge                      UnsignedInt
   -n -normals                    on|off
 -ncs -normalsCoordSys            UnsignedInt
  -nd -normalDirection            UnsignedInt
 -nfc -normalsFlipChannel         on|off
 -nft -normalsFaceTangents        on|off
 -nub -normalsUseBump             on|off
  -or -orthoRefl                  on|off
```

```
 -po -ptmOutput                 UnsignedInt
 -ps -ptmSamples                UnsignedInt
-pst -ptmSampleType             UnsignedInt
  -r -region                    UnsignedInt UnsignedInt UnsignedInt UnsignedInt
-rai -rnmAdjustIntensity        UnsignedInt
-rfl -reflections               on|off
-rfr -refractions               on|off
-rst -rnmSampleType             UnsignedInt
  -s -source                    String (multi-use)
 -sf -saveToFile                on|off
 -sh -shadows                   on|off
 -sm -selectionMode             UnsignedInt
-spc -specular                  on|off
-srv -saveToRenderView          on|off
-stb -stencilBake               on|off
  -t -target                    String (multi-use)
-trs -transferSpace             UnsignedInt
-umn -uMin                      Float
-umx -uMax                      Float
-urv -useRenderView             UnsignedInt
-uvr -uvRange                   UnsignedInt
-uvs -uvSet                     String
 -vd -viewDependent             on|off
-viz -visualize                 on|off
-vmn -vMin                      Float
-vmx -vMax                      Float
  -w -width                     UnsignedInt
 -wo -windingOrder              UnsignedInt
```

5.2.2 ilrVertexBakeCmd

Performs vertex baking. All vertex bake specific options are taken as argument flags. All other render settings are taken from the node **TurtleRenderOptions**.

```
Synopsis: ilrVertexBakeCmd [flags]
Flags:
  -ab -alphaBlend               UnsignedInt
 -alb -albedo                   on|off
 -alp -alpha                    on|off
 -amb -ambient                  on|off
 -amn -alphaMax                 Float
 -amx -alphaMin                 Float
  -as -alphaScale               Float
  -bb -backBias                 Float
 -bgc -backgroundColor          Float Float Float
  -bl -bakeLayer                String
  -br -backRange                Float
   -c -camera                   String
  -cl -clamp                    on|off
  -cs -colorSet                 String
```

```
 -cu -custom                    on|off
-cus -customShader              String
  -d -directory                 String
-dif -diffuse                   on|off
  -e -envelope                  String (multi-use)
 -em -envelopeMode              UnsignedInt
  -f -frame                     Float
 -fb -frontBias                 Float
 -fl -filter                    on|off
-fln -filterNormalDev           Float
-fls -fullShading               on|off
-flz -filterSize                Float
 -fn -fileName                  String
 -fr -frontRange                Float
-iil -indirectIllumination      on|off
-iin -ignoreInconsistentNormals on|off
 -il -illumination              on|off
-inc -incandescence             on|off
  -l -layer                     String
 -mm -mismatchMode              UnsignedInt
-mns -minSamples                UnsignedInt
-mxs -maxSamples                UnsignedInt
 -nd -normalDirection           UnsignedInt
-ocs -overwriteColorSet         on|off
 -or -orthoRefl                 on|off
-rai -rnmAdjustIntensity        UnsignedInt
 -rb -rgbBlend                  UnsignedInt
-rfl -reflections               on|off
-rfr -refractions               on|off
-rmn -rgbMin                    Float Float Float
-rmx -rgbMax                    Float Float Float
 -rs -rgbScale                  Float
-rst -rnmSampleType             UnsignedInt
  -s -source                    String (multi-use)
-scs -saveToColorSet            on|off
 -sf -saveToFile                on|off
 -sh -shadows                   on|off
 -sm -samplingMode              UnsignedInt
-spc -specular                  on|off
  -t -target                    String (multi-use)
-trs -transferSpace             UnsignedInt
 -vb -vertexBias                UnsignedInt
 -vd -viewDependent             on|off
```

5.2.3 ilrImportVertexColorsCmd

Assigns baked vertex colors to meshes. Reads vertex color data from a point cloud file and assigns the colors to the specified meshes. This can be used for importing vertex colors that has been generated with batch baking.

NOTE1: If DirectMapping is used (-importMethod 0) the assignment is done in the order the meshes are listed, so make sure to list them in the same order as they were listed when the point cloud was created. If the point cloud was generated from a bake layer, it's easy to list the shapes in the right order using: sets -query myBakeLayer.

NOTE2: If Nearest or Filtering is used (-importMethod 1,2) the -searchRadius controls the maxradius/filtersize and -maxNormalError controls the maximum allowed normal deviation. For Nearest the searchRadius defaults to the size of the whole cloud. For Filtering it defaults to 10 percent of the size of the current shape. The maxNormalError defaults to 0.1 for both methods (about 6 degrees deviation). If the flags are not specified the default values are used.

```
Synopsis: ilrImportVertexColorsCmd [flags] String...
Flags:
  -ab -alphaBlend        Unsigned (0=Repl|1=Add|2=Sub|3=Mult|4=Div|5=Avrg|6=None)
 -amn -alphaMax          Float
 -amx -alphaMin          Float
  -as -alphaScale        Float
  -bl -bakeLayer         String
  -cl -clamp             on|off
  -cs -colorSet          String
  -ia -importAlpha       on|off
  -im -importMethod      Unsigned (0=DirectMapping|1=Nearest|2=Filter)
 -mne -maxNormalError    Float
 -ocs -overwriteColorSet String
 -pcf -pointCloudFile    String
  -rb -rgbBlend          Unsigned (0=Repl|1=Add|2=Sub|3=Mult|4=Div|5=Avrg|6=None)
  -rl -renderLayer       String
 -rmn -rgbMin            Float Float Float
 -rmx -rgbMax            Float Float Float
  -rp -renderPass        String
  -rs -rgbScale          Float
  -sr -searchRadius      Float
```

Example:

ilrImportVertexColorsCmd -pcf "d:/myfolder/mypointcloud.pc" myShape1 myShape2 myShape3
ilrImportVertexColorsCmd -pcf "d:/myfolder/mypointcloud.pc" ('sets -query myBakeLayer')
ilrImportVertexColorsCmd -pcf "d:/myfolder/mypointcloud.pc" -im 1 myShape1
ilrImportVertexColorsCmd -pcf "d:/myfolder/mypointcloud.pc" -im 2 -sr 12.5 -mne 0.15 myShape1

5.2.4 ilrPointCloudCmd

This command performs various tasks with point clouds.

```
Synopsis: ilrPointCloudCmd [flags]
Flags:
   -b -bake             Bakes point cloud data using a LUA bake script.
                        Input points can be taken from point cloud files
```

```
                        or polygon meshes.
   -c -create           Create empty point clouds from given input
                        polygon meshes.
   -i -info             Prints info about the given input point cloud files.
   -m -merge            Merge multiple input files to a single output file.
   -r -rebuild          Rebuilds the given point cloud shapes.
  -in -inputFile        String (multi-use)
  -is -inputShape       String (multi-use)
 -lua -luaFile          String
  -od -outputDirectory  String
 -out -outputFile       String
   -t -type             Int    Output point cloud type.
                               For bake - kFloatArrayPoint=7 kRGBArrayPoint=8
                               For create - kIrradiancePoint=2 kDoughPoint=6
```

5.2.5 ilrUVSACmd

Calculates a texture size for each object in the specified bake layer, according to the world space size of the objects, see 2.1.1 for more information on this.

```
Synopsis: ilrUVSACmd [flags]
Flags:
  -bl -bakelayer  String
```

5.2.6 ilrCreateSharedUVCmd

Calculates a shared UV layout for the objects in the specified bake layer. The UV shells for each object is scaled according to the world space size of the objects, see 2.1.1 for more information on this.

```
Synopsis: ilrCreateSharedUVCmd [flags]
Flags:
  -bl -bakelayer  String
```

5.3 Other Commands

5.3.1 ilrDisplacementToPolyCmd

Converts displacement maps into polygons for the given meshes. If no meshes are given the current selection list is used. New polygon shapes are created with triangles according to the displacement map. The Pre-Tesselation method of the displacement shader is used, so make sure that method is enabled in the shader.

```
Synopsis: ilrDisplacementToPolyCmd [String...]
No Flags.
```

5.3.2 ilrIBLCmd

Generates a dome of directional lights from an environment map. Can also be set to emit photons to give global illumination. An environment shader must be given as input using -shader “shaderName”.

```
Synopsis: ilrIBLCmd [flags]
Flags:
  -cp -causticsPhotons  Int
  -ed -emitDiffuse      on|off
  -ep -emitPhotons      on|off
  -es -emitSpecular     on|off
  -gp -globalPhotons    Int
 -int -intensity        Float
  -pe -photonEnergy     Float
   -s -shader           String
  -sa -shadowAngle      Float
 -sap -samples          Int
 -set -setName          String
  -sh -shadows          on|off
  -sr -shadowRays       Int
```

5.3.3 ilrGetFileLayersCmd

Returns the names of the layers in a MultiLayer EXR image file. Takes the path to the file as input.

```
Synopsis: ilrGetFileLayersCmd String
No Flags.
```

Chapter 6

Turtle Administration

6.1 Removing Turtle dependencies from a scene

If for some reason you need to work with a scene on a Maya installation without Turtle, you may be a need to clear the scene from Turtle dependencies. With the scene loaded, run the **ilrClearScene** script in the script editor (the script is available from the Turtle shelf). This will clear out all Turtle nodes from the scene. Open the plug-in manager, unload the Turtle plug-in and finally save your scene. Now the scene will have no Turtle dependencies, and will open on a Maya installation without Turtle.

If Turtle already is uninstalled run the mel commands

```
lockNode -lock off `ls -type unknown`; and
delete `ls -type unknown`;
```

These commands will first unlock all locked unknown nodes and then delete them. Turtle cannot be loaded when you do this since then the nodes aren't of the "unknown" type. Since these commands deletes all unknown nodes in the scene and aren't undoable, making a backup is strongly advised.

Chapter 7

Appendix A: Copyrights

7.1 OpenEXR

Turtle supports OpenEXR. See copyright notice below.

Copyright (c) 2002, Industrial Light & Magic, a division of Lucas Digital Ltd. LLC All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Industrial Light & Magic nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7.2 The Radiance Software License, Version 1.0

Turtle has in some ways utilised “Radiance” code. We are grateful for their work. See quoted software license below.

Copyright (c) 1990 - 2002 The Regents of the University of California, through Lawrence Berkeley National Laboratory. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

The end-user documentation included with the redistribution, if any, must include the following acknowledgment: ”This product includes Radiance software (http://radsite.lbl.gov/) developed by the Lawrence Berkeley National Laboratory (http://www.lbl.gov/).” Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

The names ”Radiance,” ”Lawrence Berkeley National Laboratory” and ”The Regents of the University of California” must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact radiance@radsite.lbl.gov.

Products derived from this software may not be called ”Radiance”, nor may ”Radiance” appear in their name, without prior written permission of Lawrence Berkeley National Laboratory.

THIS SOFTWARE IS PROVIDED “AS IS” AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL Lawrence Berkeley National Laboratory OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of Lawrence Berkeley National Laboratory. For more information on Lawrence Berkeley National Laboratory, please see <http://www.lbl.gov/>.

7.3 libJPEG

Turtle has in some ways utilised IJG code. We are grateful for their work. See quoted legal text below. Turtle DOES NOT utilise ansi2knr.c, script "configure", arithmetic coding or GIF files mentioned in this legal text.

LEGAL ISSUES
============

In plain English:

1. We don't promise that this software works. (But if you find any bugs, please let us know!)
2. You can use this software for whatever you want. You don't have to pay us.
3. You may not pretend that you wrote this software. If you use it in a program, you must acknowledge somewhere in your documentation that you've used the IJG code.

In legalese:

The authors make NO WARRANTY or representation, either express or implied, with respect to this software, its quality, accuracy, merchantability, or fitness for a particular purpose. This software is provided "AS IS", and you, its user, assume the entire risk as to its quality and accuracy.

This software is copyright (C) 1991-1998, Thomas G. Lane. All Rights Reserved except as specified below.

Permission is hereby granted to use, copy, modify, and distribute this software (or portions thereof) for any purpose, without fee, subject to these conditions:
(1) If any part of the source code for this software is distributed, then this README file must be included, with this copyright and no-warranty notice unaltered; and any additions, deletions, or changes to the original files must be clearly indicated in accompanying documentation.
(2) If only executable code is distributed, then the accompanying documentation must state that "this software is based in part on the work of the Independent JPEG Group".
(3) Permission for use of this software is granted only if the user accepts full responsibility for any undesirable consequences; the authors accept NO LIABILITY for damages of any kind.

These conditions apply to any software derived from or based on the IJG code, not just to the unmodified library. If you use our work, you ought to acknowledge us.

Permission is NOT granted for the use of any IJG author's name or company name in advertising or publicity relating to this software or products derived from it. This software may be referred to only as "the Independent JPEG Group's software".

We specifically permit and encourage the use of this software as the basis of commercial products, provided that all warranty or liability claims are assumed by the product vendor.

ansi2knr.c is included in this distribution by permission of L. Peter Deutsch, sole proprietor

of its copyright holder, Aladdin Enterprises of Menlo Park, CA. ansi2knr.c is NOT covered by the above copyright and conditions, but instead by the usual distribution terms of the Free Software Foundation; principally, that you must include source code if you redistribute it. (See the file ansi2knr.c for full details.) However, since ansi2knr.c is not needed as part of any program generated from the IJG code, this does not limit you more than the foregoing paragraphs do.

The Unix configuration script "configure" was produced with GNU Autoconf. It is copyright by the Free Software Foundation but is freely distributable. The same holds for its supporting scripts (config.guess, config.sub, ltconfig, ltmain.sh). Another support script, install-sh, is copyright by M.I.T. but is also freely distributable.

It appears that the arithmetic coding option of the JPEG spec is covered by patents owned by IBM, AT&T, and Mitsubishi. Hence arithmetic coding cannot legally be used without obtaining one or more licenses. For this reason, support for arithmetic coding has been removed from the free JPEG software. (Since arithmetic coding provides only a marginal gain over the unpatented Huffman mode, it is unlikely that very many implementations will support it.) So far as we are aware, there are no patent restrictions on the remaining code.

The IJG distribution formerly included code to read and write GIF files. To avoid entanglement with the Unisys LZW patent, GIF reading support has been removed altogether, and the GIF writer has been simplified to produce "uncompressed GIF's". This technique does not use the LZW algorithm; the resulting GIF files are larger than usual, but are readable by all standard GIF decoders.

We are required to state that "The Graphics Interchange Format(c) is the Copyright property of CompuServe Incorporated. GIF(sm) is a Service Mark property of CompuServe Incorporated."

7.4 libtiff

Turtle has in some ways utilised "libtiff" code. We are grateful for their work. See quoted software license below.

Copyright (c) 1988-1997 Sam Leffler Copyright (c) 1991-1997 Silicon Graphics, Inc.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that (i) the above copyright notices and this permission notice appear in all copies of the software and related documentation, and (ii) the names of Sam Leffler and Silicon Graphics may not be used in any advertising or publicity relating to the software without the specific, prior written permission of Sam Leffler and Silicon Graphics.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER

OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

7.5 Lua

Turtle has in some ways utilised "Lua" code. We are grateful for their work. See quoted software license below.

Lua is licensed under the terms of the MIT license reproduced below. This means that Lua is free software and can be used for both academic and commercial purposes at absolutely no cost.

For details and rationale, see http://www.lua.org/license.html.

Copyright (C) 1994-2006 Lua.org, PUC-Rio.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

7.6 libXML 2

Except where otherwise noted in the source code (e.g. the files hash.c, list.c and the trio files, which are covered by a similar licence but with different Copyright notices) all the files are:

Copyright (C) 1998-2003 Daniel Veillard. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is fur- nished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or

substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

7.7 libpng

Turtle has in some ways utilised "libpng" code. We are grateful for their work. See quoted software license below.

libpng version 1.2.6, December 3, 2004, is Copyright (c) 2004 Glenn Randers-Pehrson, and is distributed according to the same disclaimer and license as libpng-1.2.5 with the following individual added to the list of Contributing Authors

Cosmin Truta

libpng versions 1.0.7, July 1, 2000, through 1.2.5 - October 3, 2002, are Copyright (c) 2000-2002 Glenn Randers-Pehrson, and are distributed according to the same disclaimer and license as libpng-1.0.6 with the following individuals added to the list of Contributing Authors

Simon-Pierre Cadieux
Eric S. Raymond
Gilles Vollant

and with the following additions to the disclaimer:

There is no warranty against interference with your enjoyment of the library or against infringement. There is no warranty that our efforts or the library will fulfill any of your particular purposes or needs. This library is provided with all faults, and the entire risk of satisfactory quality, performance, accuracy, and effort is with the user.

libpng versions 0.97, January 1998, through 1.0.6, March 20, 2000, are Copyright (c) 1998, 1999 Glenn Randers-Pehrson, and are distributed according to the same disclaimer and license as libpng-0.96, with the following individuals added to the list of Contributing Authors:

Tom Lane
Glenn Randers-Pehrson
Willem van Schaik

libpng versions 0.89, June 1996, through 0.96, May 1997, are Copyright (c) 1996, 1997 Andreas Dilger Distributed according to the same disclaimer and license as libpng-0.88, with the following individuals added to the list of Contributing Authors:

John Bowler
Kevin Bracey
Sam Bushell
Magnus Holmgren
Greg Roelofs
Tom Tanner

libpng versions 0.5, May 1995, through 0.88, January 1996, are Copyright (c) 1995, 1996 Guy Eric Schalnat, Group 42, Inc.

For the purposes of this copyright and license, "Contributing Authors" is defined as the following set of individuals:

Andreas Dilger
Dave Martindale
Guy Eric Schalnat
Paul Schmidt
Tim Wegner

The PNG Reference Library is supplied "AS IS". The Contributing Authors and Group 42, Inc. disclaim all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The Contributing Authors and Group 42, Inc. assume no liability for direct, indirect, incidental, special, exemplary, or consequential damages, which may result from the use of the PNG Reference Library, even if advised of the possibility of such damage.

Permission is hereby granted to use, copy, modify, and distribute this source code, or portions hereof, for any purpose, without fee, subject to the following restrictions:

1. The origin of this source code must not be misrepresented.

2. Altered versions must be plainly marked as such and must not be misrepresented as being the original source.

3. This Copyright notice may not be removed or altered from any source or altered source distribution.

The Contributing Authors and Group 42, Inc. specifically permit, without fee, and encourage the use of this source code as a component to supporting the PNG file format in commercial products. If you use this source code in a product, acknowledgment is not required but would be appreciated.

7.8 zlib

Turtle has in some ways utilised "zlib" code. We are grateful for their work. See quoted software license below.

Copyright notice:

(C) 1995-2004 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly
jloup@gzip.org

Mark Adler
madler@alumni.caltech.edu

If you use the zlib library in a product, we would appreciate *not* receiving lengthy legal documents to sign. The sources are provided for free but without warranty of any kind. The library has been entirely written by Jean-loup Gailly and Mark Adler; it does not include third-party code.

If you redistribute modified sources, we would appreciate that you include in the file ChangeLog history information documenting your changes. Please read the FAQ for more information on the distribution of modified source versions.

7.9 GLFW

Turtle has in some ways utilised GLFW code. We are grateful for their work. See quoted software license below.

Copyright © 2002-2006 Camilla Berglund

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented

as being the original software.
3. This notice may not be removed or altered from any source distribution.

7.10 CG

Turtle has in some ways utilised CG code. See quoted software license below.

Copyright (c) 2002-2004, NVIDIA Corporation.

NVIDIA Corporation("NVIDIA") supplies this software to you in consideration of your agreement to the following terms, and your use, installation, modification or redistribution of this NVIDIA software constitutes acceptance of these terms. If you do not agree with these terms, please do not use, install, modify or redistribute this NVIDIA software.

In consideration of your agreement to abide by the following terms, and subject to these terms, NVIDIA grants you a personal, non-exclusive license, under NVIDIAs copyrights in this original NVIDIA software (the "NVIDIA Software"), to use, reproduce, modify and redistribute the NVIDIA Software, with or without modifications, in source and/or binary forms; provided that if you redistribute the NVIDIA Software, you must retain the copyright notice of NVIDIA, this notice and the following text and disclaimers in all such redistributions of the NVIDIA Software. Neither the name, trademarks, service marks nor logos of NVIDIA Corporation may be used to endorse or promote products derived from the NVIDIA Software without specific prior written permission from NVIDIA. Except as expressly stated in this notice, no other rights or licenses express or implied, are granted by NVIDIA herein, including but not limited to any patent rights that may be infringed by your derivative works or by other works in which the NVIDIA Software may be incorporated. No hardware is licensed hereunder.

THE NVIDIA SOFTWARE IS BEING PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF TITLE, NONINFRINGEMENT, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR ITS USE AND OPERATION EITHER ALONE OR IN COMBINATION WITH OTHER PRODUCTS.

IN NO EVENT SHALL NVIDIA BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, EXEMPLARY, CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, LOST PROFITS; PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) OR ARISING IN ANY WAY OUT OF THE USE, REPRODUCTION, MODIFICATION AND/OR DISTRIBUTION OF THE NVIDIA SOFTWARE, HOWEVER CAUSED AND WHETHER UNDER THEORY OF CONTRACT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY OR OTHERWISE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.