Kray 2.x manual

English

Contents

Articles

References

Kray User Manual

Installation

Language ›

Important

- Kray will not run on LW version older than 7.5.
- It is advisable that you put Kray.ls LightWave's Lscripts dir.
- We recommend that you remove all old Kray plugins and restart LightWave before installing new one.

General

The Kray package contains different plugins for different platforms. Inside the package you will find the following folders:

- **Common**: contains all GUI plugins that will appear in LightWave Master plugins list. This are platform independant.
- **MacUB**: Contains main Mac Universal Binary Kray plugin.
- **Win32**: Contains main Kray plugin for Windows LighWave 32bit.
- **Win64**: Contains main Kray plugin for Windows LighWave 64bit.

Windows

To install Kray on Windows follow this steps:

1. Unpack the archive and copy contents of **Common dir** to directory where LW holds Lscript files
   - in LW7: *[LW_DIR]/Lscripts*
   - in LW8 or LW9+ *[LW_DIR]/Plugins/Lscripts*

2. Installing Kray.p:
   - For 32bit LW copy **kray.p** from **Win32** folder to LW's plugins dir *[LW_DIR]/Plugins*.
   - For 64bit LW copy **kray.p** from **Win64** folder to LW's plugins dir *[LW_DIR]/Plugins*.

Note: if you have older CPU without support [1] please use Win32/NoSSE kray.p instead.

Mac

Note: The Mac version of Kray is only available for the Universal Binary (UB) version of LightWave 9.5.1+ and above. Currently there is no PowerPC (CFM) version.

To install Kray on Mac follow these steps:

1. Unpack the archive and copy contents of **Common dir** to directory where LW holds Lscript files.

- in LW9.5.1+ *[LW_DIR]/SharedSupport/Plugins/Lscripts* or *Users/~/Library/Application Support/LightWave3D/Plugins/Lscripts* .

Note: If a *Lscript* folder does not exist within the *Plugins* folder feel free to create it.

2. Copy **Kray.plugin** from **mac32ub** folder to LW's plugins dir *[LW_DIR]/SharedSupport/Plugins* or *Users/~/Library/Application Support/LightWave3D/Plugins/*

Adding plugins to LightWave

Start LW and add all Kray *.lsc files from Lscripts and Kray.p from plugins to LW plugins by going to Utilities > Add Plugins.

Note: If you get a **"line 8 Script type 'master' differs from architecture" error** when adding the *Kray.ls* try adding the via Master Plugins: LScript

You can connect **Kray Options**, **Kray Frame** and **Kray Anim** to Layout buttons or assign hot keys. You can also reach Kray Options through "Utilities > Master plugins > Add Layout or Scene Master".

References

[1] http://en.wikipedia.org/wiki/SSE2#CPUs_supporting_SSE2|SSE2

Basics of global illumination

Basics of global illumination

Global illumination is a general name for a group of algorithms used in 3d computer graphics that are meant to add more realistic lighting to 3D scenes. Such algorithms take into account not only the light which comes directly from a light source (*direct illumination*), but also subsequent cases in which light rays from the same source are reflected by other surfaces in the scene (*indirect illumination*).

Images rendered using global illumination algorithms often appear more photorealistic than images rendered using only direct illumination algorithms. However, such images are computationally more expensive and consequently much slower to generate. One common approach is to compute the global illumination of a scene and store that information with the geometry, i.e., radiosity. That stored data can then be used to generate images from different viewpoints for generating walkthroughs of a scene without having to go through expensive lighting calculations repeatedly.

Radiosity, ray tracing, beam tracing, cone tracing, path tracing, metropolis light transport, ambient occlusion, and photon mapping are all examples of algorithms used in global illumination, some of which may be used together to yield results that are fast, but accurate.

(source: http://en.wikipedia.org/wiki/Global_illumination)

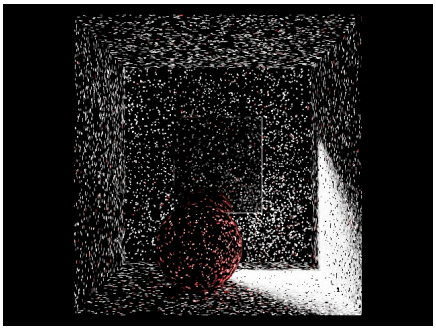
Global illumination techniques in Kray

Kray can use several of these techniques – Raytracing, Photon/Light mapping and Path tracing. Raytracing computes only direct illumination while Photon mapping, Light mapping and Path tracing compute full global illumination in a scene.

Photon mapping

Photon mapping is one of the techniques which help speed up the calculation of GI dramatically. It consists of two steps: shooting photon and final gathering. In the first step a specified number of light packets called photons are fired from a light source and bounced around the scene. Whenever a photon intersects with a surface, the intersection point, incoming direction, and energy of the photon are stored in a cache called the photon map.

You can see photons in Kray by using **Photon estimate mode Global unfiltered**.



Global unfiltered

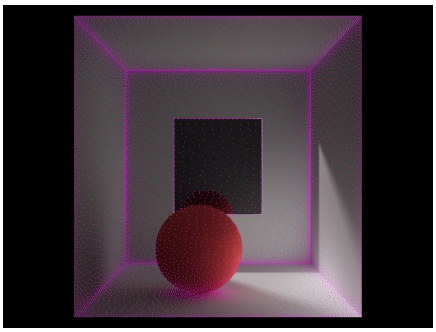
Photon mapping is only a rough approximation of a light at any given point. Therefore we need another step to get more accurate result.

Final Gathering

The final step in rendering that Kray does is called Final Gathering (or FG).

This technique computes a more accurate illumination at a sampled point. It uses the photon map that was computed in the previous step to help speed up the process. Computing irradiance for every point in space may still be slow therefore it uses another technique called **irradiance caching** to speed up the process. What **irradiance caching** does is, it only computes irradiance at points where light is most likely to change - for example in corners, in place of shadows, where two surfaces are near each other, etc.). Where surface is large and flat it uses less points to save a time. Kray finds these points by doing a pre pass before final render called prerender.

You can see where Kray is computing FG by selecting `Photon mapping` mode with `cache irradiance` and turn on `show samples` in `FG` tab.



You can also turn `irradiance caching` off in the main tab by unchecking cache irradiance button. This might be useful in difficult light conditions where cached irradiance gives a lot of errors (splotches) which don't go away even by increasing number of FG rays.

Photon mapping and irradiance caching can be saved to a file and be reused for another frame. So in an animation you only have to compute FG for those points that have not yet been computed (that have not been seen by camera in previous frames).

Links for more in depth knowledge:

http://en.wikipedia.org/wiki/Photon_mapping

http://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/photon_mapping/PhotonMapping.html

Path Tracing

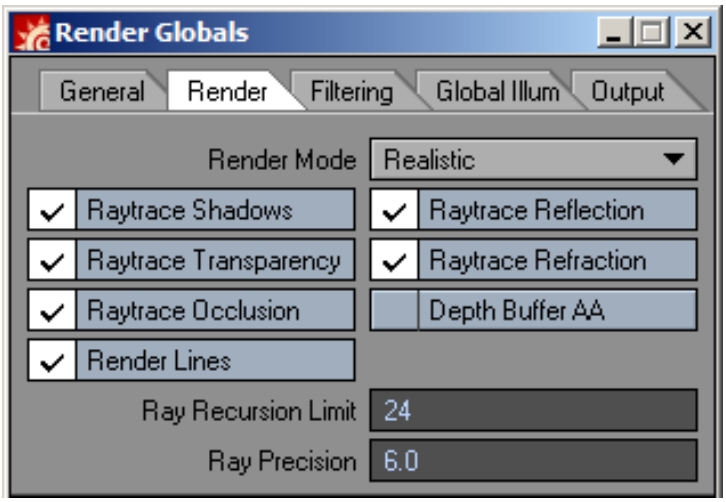
Path tracing is a form of ray tracing whereby each ray is recursively traced along a path until it reaches a light emitting source where the light contribution along the path is calculated. This recursive tracing helps for solving the lighting equation more accurately than conventional ray tracing.

Setting up Lightwave

Environment

Since Kray is relying on real world physics to compute global illumination it is important to set up a scene properly.

The first step is to turn on all raytracing flags in LightWave render options.



Ray recursion limit tells Kray how many times the light should bounce. In contrast to other render engines, a high number of light bounces doesn't increase the render time significantly.

The number of bounces can be overridden by adding the following command to the **tailer** line in Kray GUI: **recurse 100;** This will tell Kray to use 100 bounces of light.

Lights

The lighting of a scene should be set up realistically.

This means turning off ambient light and setting it to 0% because it makes renders look washed out and unnatural, and also, all lights should use **Inverse Distance Square** falloff which is closer to what happens in reality.

If LightWave is not set-up this way Kray will give a warning saying **Physically incorrect light model in a scene with global illumination**. This is just a warning and will not stop Kray from rendering.

Background can also act as a light source in global illumination scene when using Light mapping. Photons tracing does not support background lighting since there is no point from where photons can be emitted. You can overcome this by using a luminous dome instead of background light.

Since CG is not an exact science - it is all good as long as it looks good. For example sometimes you will want your lights to have linear falloff, to lighten up certain parts more and avoid over burned areas.

There is a switch in Kray GUI which converts Spotlights into area lights. This is especially handy when you want soft sun shadows. Using LightWave's "**soft edge angle**" you control how soft the shadows are.

Tip: For a nice realistic Sun light use **Spotlight** with **soft edge angle** to **0.55**. Put sun 100km away (that's easiest done with **Sun spot** motion modifier) and set Light to have **Inverse distance^2 falloff** of **100km**.

Models

Since Kray does many different computations to avoid long render times it is necessary to obey some rules when building 3d models.

Here are some guidelines to follow:

- Try to model clean and if possible "watertight". This means that you should check that your points are merged together and polygons share edges. For example: if polygons don't share edges in room corner in a room, you may get light leaks, because Kray will sample points from both sides of polygon instead only from inner side.
- UV's can sometimes cause problems. So be take care when unwrapping your models.
- Try to avoid very long and thin polygons. This may cause extremely large memory consumption and hence long render times. If you run into this problem then just slice the object up in smaller more regular pieces.

For more help take a look into troubleshooting section of this manual.

Surfaces

Since Kray uses physically accurate computation to render the scene there are some guidelines that you should try to follow when setting up your surfaces.

Light and material models used in 3d packages (such as LightWave) are not always physically correct. For example in LightWave you can create materials that have diffuse 100% and reflection 100%. Diffuse + reflection = 200% and that means the material reflects (it reflects all light diffusely and reflectively) more light than it receives. Physical incorrectness in a non-GI scene helps to hide lack of global illumination, but in a GI scene such surfaces can look very unnatural.

Kray does not force user to use physically correct models. They are sometimes useful to make rendered images look more dramatic, it is important to know when the scene matches the laws of physics and what will happened if we break them.

The general balance rule that should be followed to avoid unrealistic materials is :

```
Color*Diffuse+Reflection+Translucency<100%
```

or a more complex version including transparency would be:

```
Color*Diffuse(100%-Transparency)+Transparency+Reflection+Translucency < 100%
```

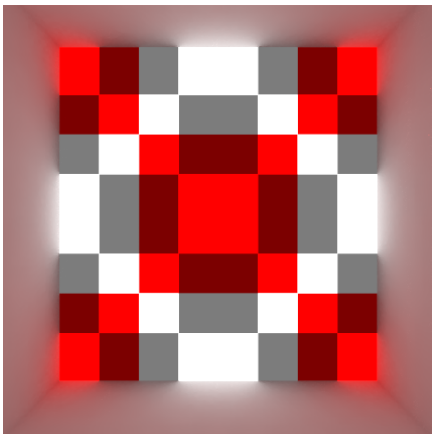
Notice that it is "less than" not "less or equal than". In reality no material reflects all the light that hits its surface. The same rule is valid if we use gradients or textures for every gradient angle and texture pixel.

Balance rule does not affect Luminosity. Feel free to use 300% Luminosity and 95% Diffuse for example.

Luminosity

Objects with surface material Luminosity > 0%, emit light. On a GI enabled scene they can be used to illuminate the scene. Howerver you can also use luminous object to light a scene in Diffuse model Raytrace only mode but you need to switch to **Compute as direct** Luminosity model in the quality tab of the Kray settings which will make them emit light even without GI.

Generally it's not suggested to use luminosity to tweak the look of a material that normally doesn't emit light. Same as light intensity, you are not limited to 100% luminosity and you can use values over 100%. Unlike regular lights, you can also have them textured to emit multicolored light and with variable intensity.



Luminous Geometry with Mapped color and intensity

Specularity

Specularity is used to fake light reflections. It simulates how the hotspot of a light would be reflected on a surface. This is used because rendering blurry reflections was often very slow. It has important limitation though: it doesn't render light reflected from other surfaces.

In Kray however, you don't often need to use specularity because blurry reflections work very fast and look much more realistic. So instead of using specularity and glossiness you just have to add reflection and some blurring to get nice glossy surface. However there are still uses for specularity - in cases where you want to see light reflected in a surface and you don't have any physical model of the light (sun for example).

Kray Interface

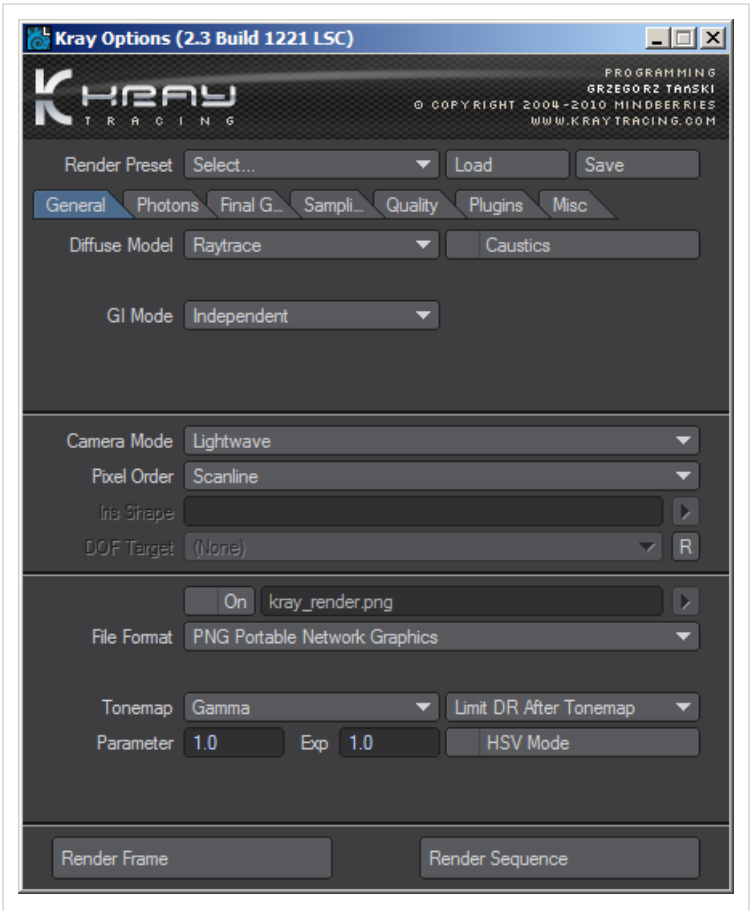
Kray GUI is your main interface to control Kray renderer parameters. You will spend most of the time here so get familiar with the controls and what they do. It is best to start with reading the sections below and then experiment with the values to see the effect.

Sections

- General tab
- Photons tab
- FG tab
- Sampling tab
- Quality tab

GUI - General tab

General tab



Render preset

This selection list has all the predefined presets available to Kray. When a preset is chosen the corresponding values are used. For example: if you select Medium it will set “medium” preset for photons, FG, Quality and all the other “sub-presets”.

You can select between various presets we have pre-defined to make your work easier. Please keep in mind that while there was a lot of effort put into making the presets as good as possible, and they often are a good starting point, they just won't work best in every situation.

Save/Load

Save buttons lets you save Kray settings into a file which can than be recalled in any scene by using Load button.

Diffuse model

This is the main render mode selection button. It lets you choose between various render methods which are explained below.

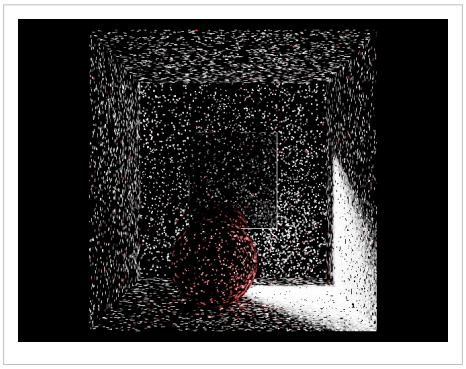
Raytrace

This mode will render the scene only with raytracing. This means that no global illumination will be computed. This mode is useful mainly for testing purposes. Make sure your raytrace flags are ON in LightWave render globals to make it work properly.

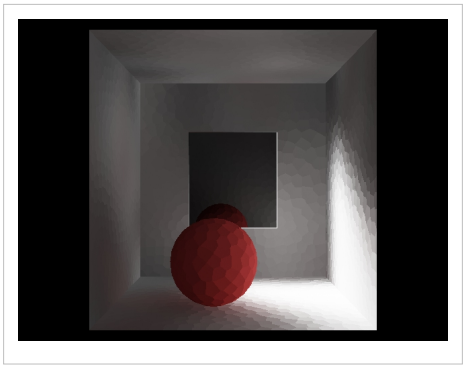
Photon estimate

Photon estimate lets you choose between several modes that render out a quick representation of global illumination in the scene. In this modes no final gathering is used so rendering is very fast.

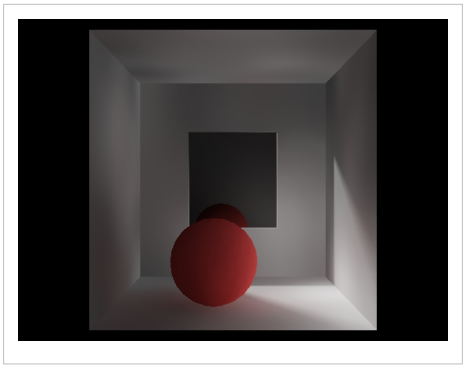
Global filtered - mode renders photons that are fired into the scene and filters them.



Global unfiltered - mode renders representation of photons in the scene. You will sometimes want to see photon density in your scene and this mode is best for showing that.



Precomputed - will render out representation of irradiance cells. It mimics global illumination in a scene quite well. It's often used to see how big the cells in the scene are and if irradiance is smooth enough or not.



Precomputed filtered – is very similar to precomputed. The only difference is that filtered mode filters (blurs) cells together. It renders close representation of global illumination in the scene very fast and is therefore recommended mode to use when tweaking light or surfaces. This mode also offers option to ray trace shadows and render caustics which makes it even better representation of final render.

Photon mapping

This is the the main render mode in Kray render engine. It uses photon mapping with cached irradiance to render out full global illumination images fast.

Cache irradiance

When this check box is turned on, Kray will use irradiance cache to speed-up rendering. Read more about irradiance caching here. When this check box is turned off, Kray will not use any irradiance caching and will evaluate each pixel separately. This gives the best quality shadows at the expense of render time.

Path tracing

Path tracing is a form of ray tracing whereby each ray is recursively traced along a path until it reaches a light emitting source where the light contribution along the path is calculated. This recursive tracing helps for solving the lighting equation more accurately than conventional ray tracing. The drawback is that with higher number of bounces it can get quite slower than Photon mapping with irradiance caching. Path tracing however does not require shooting of photons.

Caustics

The Caustics check box will enable rendering of caustics in the scene. So whenever you want caustics to show up, don't forget to turn this on.

GI mode

To speed up rendering of image sequences Kray offers 3 different modes.

Independent

This mode will turn off GI sharing. It's used for testing or when you're rendering a single still.

Time interpolation

This mode can be used whenever there are moving objects and/or light sources in a scene. Kray will use illumination data from the current frame and from the previously rendered frames to speed up rendering. There are 2 basic ways of blending frames together. One is to take N (named **Frames** parameter in GUI) from previous frames and count its average illumination.

```
y0=(x0+x1+x2+...+xn)/N
```

where *y0* is interpolation result for current, frame *x0* is irradiance computed for current frame, *x1,x2,...xn* are irradiances from previous frames. Number of frames *N* is controlled by **Frames** parameter in GUI.

Another way is to take interpolated solution from the previous frame and blend it with the current frame with given proportions.

```
y0=y1*e+(1-e)*x0
```

y1 is result of interpolation from previous frame, parameter *e* in equation is labeled **Extinction** in GUI.

You can combine both methods. Then interpolation equation will look as follows:

```
y0=y1*e+(1-e)*(x0+x1+x2+...+xn)/N
```

Interpolation can be applied to **Precached map** (photon/light map after precaching) and **Irradiance cache** (final gather cache) or both.

Shared for all frames

If only camera is moving in your scene you should select this mode. You should specify a filename that Kray will use to save GI data into. If you don't specify any filename Kray will render into LightWave Image Viewer (only supported in LightWave 9.6+).

There are 3 different modes available:

Load: will load up the Gi data in the specified file. If the GI file doesn't contain enough data to accurately render global illumination it will compute new samples. This mode is often used when testing FG settings since it will only load save data and not save any new one into the file.

Save: will save computed GI data into the specified file when render frame is done.

Update: will load any existing GI data from file and on render end save new GI data into the same file. This mode speeds up rendering of animations dramatically. First frame will take some time to render, but all the consecutive frames will render much faster since they will compute GI only for the part of the scene that hasn't been calculated already. You should be careful with this mode when stopping and restarting rendering of animations. You should always use the same GI file to render complete animation. If you fail to do so you risk having sudden jump of light because every GI computation is a little different. So for example you rendered animation from frame 1-30 with one GI file and then frames 31-60 with another you will see a sudden jump of light in frame 31.

Reset File: Will clear the GI file on your disk so you don't have to go and manualy delete it.

Allow animation: will allow objects to move in the animation. Note however that GI will be computed only for the first frame. Other frames will use existing GI. This may cause artefacts around moving objects because indirect shadows will stay in place even though object moves. However direct shadows from lights will work just fine.

If you want to use full GI solution on animatable items use Time Interpolation mode

Bake only: In this mode final raytracing step will be skipped. This mode is useful if you are preparing scene for network rendering and want to prerender only GI to speed up the process.

Camera

You can choose between using LightWave camera or any of the built in cameras.

LightWave: Will use currently selected LightWave camera for rendering.

Spherical: Will render panoramic render mode, also called longitude/altitude. This is good for rendering out 360° panoramas.

Fisheye: panoramic render mode, also called light probe.

Texture baker: this mode allows to bake lighting on textures. Some new options are available when this mode is selected.

- **Object** allows you to pick the object you want to bake
- **UV map** allows you to pick UV texture you want to bake
- **Edge extend** will extend the baked texture over the edge to prevent black lines on borders

Stereo: Allows you to render out strip of several images which work as autostereogram.

- **Eye separation** distance between camera points.
- **No. of Images** Number of images in a strip being rendered.

Pixel order

Pixel order allows user to select order of pixels appearing on screen. This option have influence on quality and rendering time if combined with Irradiance caching. It should be noted that the pixel order greatly affects the placement of FG samples.

Scanline: will render pixels line by line starting at top.

Scancolumn: will render each row of pixels starting from left.

Random: will render pixels randomly.

Progressive: will progressively render the frame, updating image with better detail in each pass. This mode quickly shows a general preview of the lighting in the scene.

RenderWorm: will crawl through your image like a worm, revealing pixels underneath.

Frost: similar to RenderWorm but with a "frost" like effect.

Note: that Pixelorder **Progressive** render slower than **Scanline** or **Scancolumn** while **Random** gives the best estimate of render time remaining

Iris Shape

When DOF in LightWave Camera properties is enabled you can use any image to represent Lens aperture shape. This can be a simple black and white image of even color image so R, G and B rays all can have different iris shapes.

DOF target

DOF target will let you choose object which Kray will use as a focus point. This option overrides LightWave camera Focal distance. DOF target will work only with LightWave Classic camera.

Note: to see DOF in render you must choose FSAA or Random Antialiasing mode on Sampling tab.

Output files

Lets you specify name of an output file where Kray will write the image to.

On: If this checkbox is turned OFF the rendered image will be displayed inside LightWave Image viewer. If the option is on you can specify file format to save image directly to disk.

Format: You can choose between several standard file formats. HDR lets you render into high dynamic range image which enables you to tonemap in post. PNG, TGA and TIFF formats support alpha channel.

You can use special symbols in filename to change its format. See renderinfo command for details.

Tone map

Tone mapping is used to apply color transformations on the final image colors. Sometimes an image can contain a higher range of colors that can be displayed on a computer screen. Color mapping has the task of re-mapping the image values to be suitable for display purposes.

Kray can use the following methods of tone mapping:

Linear: Will not apply and correction to the image (it's the same as using Gamma with parameter : 1 and exposure : 1). This may produce too dark images since images need some gamma correction to be displayed correctly on monitors. It's also what is used by LightWave and Fprime by default.

Gamma: This is a standard correction. The amount of Gamma correction is defined by Parameter field. Most monitors use gamma correction of 2.2 but this may produce washed out images. Generally speaking values between 1.4 and 1.8 work best with Kray (if you want to use Linear Workflow, you need to add Kray Quick Linear workflow plugin), but feel free to experiment.

Generally you can think of the **parameter** value of something that affects contrast. Low values (for example 1) will have a lot of contrast and the dark areas will be really dark and possibly losing a lot of detail. A high value (say 2 or 3) will make the image have far less contrast and the dark areas will be more visible.

Exposure affects light intensity in general. It works similar to exposing photos. Higher number means longer exposure thus brighter image, lower mean shorter and therefore darker image.

Another way to think of parameter and exposure is Exposure being the White point of the image and parameter the black point.

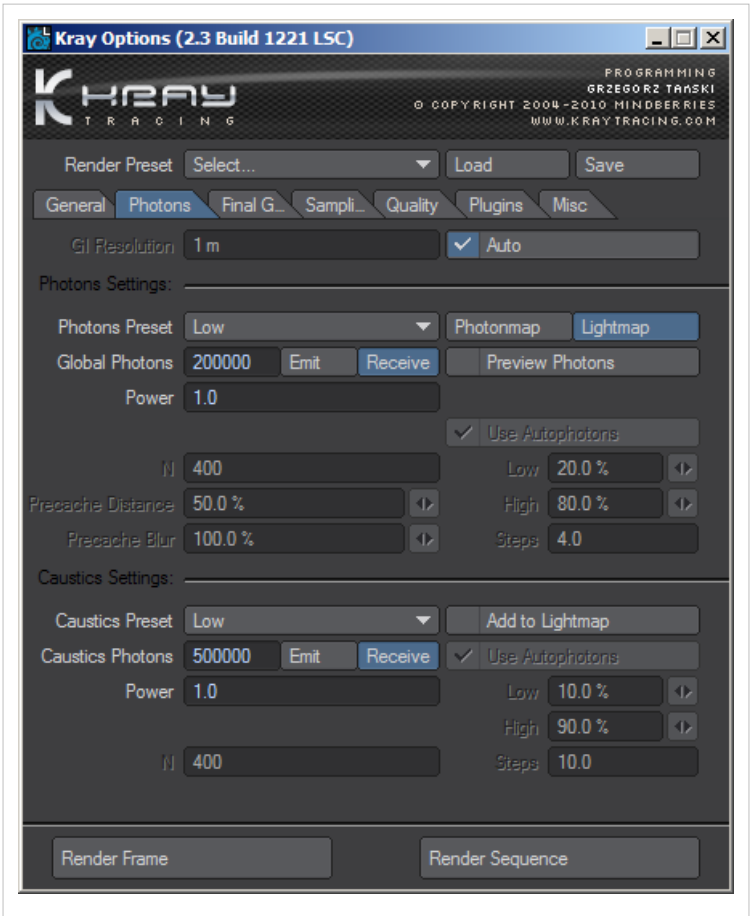
Exponential: will reduce the brightness of overexposed areas to pure white. Limits dynamic range of primary rays (those fired from camera). If a RGB component of a camera ray is bigger then value it is truncated. This is useful to make sure the over bright parts of an image are antialiased properly, for example the edges of polygons with Luminosity which is much higher than 100%.

LimitDR: by default limitdr is set to OFF (no dynamic range limit). LimitDR is done per camera ray, not per pixel. Therefore LimitDR command acts differently depending on when it's used. When its used before tonemap, Limitdr is applied before tone mapping and therefore any tonemaping like exposure or gamma can influence on how pixels are sampled (anti-aliased, etc.). When it's applied after tone mapping it will use current tonemapping settings for pixel sampling. *Note that for Low Dynamic Range (LDR) images, it's dynamic range is always limited to output exposure.*

HSV mode: when using tone mapping colors may loose their saturation. When this option is turned on Kray will try to keep saturation of colors instead of washing them out.

GUI - Photons tab

About



This section controls the first stage of rendering. The stage consists of two stages: Computing global PM (Photon or Light maps) and Precomputing irradiances. In the first stage photons are shot into the scene and evaluated. In the second stage the photons are filtered into irradiance map. For more information refer to Global illumination techniques in Kray.

GI resolution

GI resolution field is common for Photons and FG tabs. This allows user to easily rescale all GI parameters by changing GI resolution. For example if precache distance is set to 50% and GI resolution is 150mm that means that precache distance will be 50% of 150mm = 75mm.

Generally you can leave this on Auto unless in rare cases (where the scene is very big and the camera views a small detail of it) that an improper value is chosen by Kray.

Photon settings

Preview photons

Will show you a preview of photon shooting phase. This can take quite a bit longer than with Preview Photons OFF so turn it off once you don't need it anymore.

Photon preset

This selection list lets you choose predefined photon presets. When you choose Custom from the preset list advanced settings are unlocked.

Photon map/Light map

This let's you choose the mode for shooting photons. Photon map will shoot photons from light sources while Light map will shoot photons from camera. Generally Lightmaps make more efficient use photons, since photons are shot primarily to areas seen by the camera. To achieve similar efficiency with photonmaps you will have to use light portals. Photon maps do not support light from background but you can use a luminous dome instead.

Global photons

Controls the number of photons shot into the scene. The higher the number the longer it will take for Kray to shoot photons but the light estimation of the scene will be better. Usually the values vary between 100.000 and 1.000.000 photons.

Emitted/Received: since the number of photons that are shot into the scene and those that actually hit something may differ a lot Kray let's you choose preferred method. In emitted mode Kray will keep shooting photons until it has shot the number of photons specified by global photons setting. When received mode is selected Kray will only count those photons that actually hit objects in your scene.

Power

This value let's you increase or decrease the brightness of indirect light. Higher values make brighter indirect light while lower values will make it darker.

N

This value defines how many photons will Kray blur together to produce smoother irradiance.

Precache distance

This is the size of irradiance cells in the scene. The bigger the scene, the lower the accuracy of indirect light. Smaller precache distance will produce more cells and thus more accurate indirect light.

The value is a percentage of the GI Resolution. So if you have GI resolution of 1 meter and precache distance 50% the precache distance in 500mm.

Precache blur

Has similar effect as N value – it blurs irradiance, thus removing the irradiance splotches. It works faster than N but it may not produce as accurate irradiance because N depends on photon density, while precache blur depends on size of GI resolution. It is usually a good idea to keep a good balance between both values.

Use Autophotons

(Note: this is advanced setting which probably never needs to be changed)

Usually photons density is not even across the scene. It is high in some parts of a scene and low in the others. In such situation the constant size of a photon filter does not work well. Therefore it is better solution to adapt filter size to photons density.

When Autophotons setting is off you can manually control the filter size. Kray will use appropriate radius size between min and max values to find the number of photons specified by the N setting.

When Steps is bigger then 1 Kray tries to find the size of a filter radius that contains N photons. It will start with Min radius and increase filter radius Steps times unless it reaches Max radius.

When Autophotons is turned on, Kray will analyze photon maps before actual render and set all rendering critical parameters automatically.

Low and High percentages are equivalents of Min and Max radius, but they are expressed relatively to real photons density in the scene. If Low value is to big, regions in scene with high photons density (low radius) may contain to few (less then N) photons. If High value is to small, areas with low photons density (big radius) will contain much more photons then N.

Dynamic is equivalent of Steps, but it is automatically adapted to real distance between regions of high and low photons density. That means if for example Dynamic is set to 10 and density of photons is even on the scene, autophoton system will set Step to 1. But in the case when photons density vary in different parts of the scene, Steps value may be higher, but not bigger then 10. Too low Dynamic value may appear as noise on caustics. Too high value on the other hand may slow down rendering process.

Caustics settings

Caustics preset

This selection list lets you choose predefined caustics presets. When you choose Custom from the preset list advanced settings are unlocked.

Add to lightmap

Add to lightmap adds the light information of the caustics to the lightmap and it makes the caustics light contribute to the general lighting during FG process. This option is only available in lightmap mode, because Photonmap always includes caustics light.

Caustics photons

controls the number of caustics photons shot into the scene. The higher the number the longer it will take for Kray to shoot photons but the light estimation of the scene will be better.

Emitted/Received: since the number of photons that are shot into the scene and those that actually hit something may differ a lot Kray let's you choose preferred method. In emitted mode Kray will keep shooting photons until it has shot the number of photons specified by global photons setting. When received mode is selected Kray will only count those photons that actually hit objects visible in your scene.

Power

This value let's you increase or decrease the brightness of caustics. Higher values make brighter caustics while lower values will makes them darker.

N

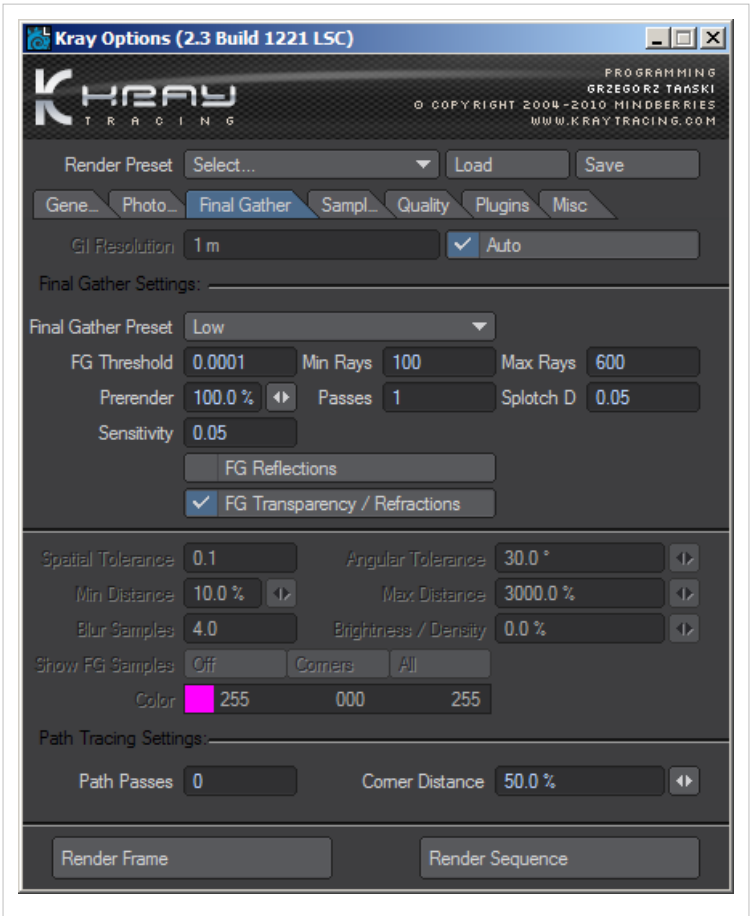
This value defines how many photons will Kray blur together to produce smoother irradiance.

Use Autophotons

Same as Use Autophotons setting above.

GUI - FG tab

FG tab



Here is where you will spend most of the time in Kray GUI. This tab controls the most important stage of rendering – final gathering (FG).

GI resolution

See GI Resolution for more information.

FG preset

This selection list lets you choose predefined FG preset. When you choose Custom from the preset list advanced settings are unlocked.

FG threshold, min rays, max rays

Whenever you see threshold in Kray interface it always means the same.

Kray starts sampling point by shooting min number of rays into the scene. Threshold defines the how much difference between two pixels is "allowed". If difference is greater than that defined by threshold, Kray will increase rays, until the difference gets smaller than the threshold. It will also stop adding rays if it reaches max rays before reaching the threshold value.

The FG threshold should be usually left at default 0.0001. Increasing this number may cause splotches in the scene but may reduce render time.

Min rays define the least amount of rays each sampled point will fire. This value is usually kept low – around 100-300, but may be increased if there are artifacts in the scene that do not disappear when increasing max rays value.

Max is the maximum rays Kray will use for any sampled point. If this value is too low you splotches will appear. Increasing this value means longer render time. By increasing this number render time increases approximately linear. For example if you increase max rays from 500 to 1000 render time will double.

Prerender

Prerender is the first step of FG which helps in defining which parts of the scene should be evaluated more thoroughly. When prerender value is 50% it means that it will render use 50% of the final resolution. You should also note that increasing prerender will cause longer prerender pass but shorter final render pass.

This value is best kept between 50%-100%.

> *It is important to understand how Kray chooses location of samples. To optimize rendering as much as possible samples are placed at "strategical" places. This are usually places where surfaces are close to each other or places of sudden light change. On large flat areas there is no need to compute samples for every pixel so Kray will only place only very few samples there. This way speed of rendering can be greatly improved. More samples means longer render time. Therefore it's important to find the right balance between sample density and render quality.'*

There are several settings which control placement of samples in a scene:

Passes

Passes defines the number of passes Kray will render to find new sample points. In each pass kray will find more sample points and add them to solution. The number of new sample points found is displayed in render window. More passes used will make much cleaner global illumination and produce less splotches. Usual values are 1-3, more passes mean longer render time.

Note: when rendering animations don't use passes > 1 because this may cause very slow rendering times.

Splotch detection

Splotch detection controls how sensitive is Kray to detecting artifacts in global illumination solution. Increasing this number will make Kray less sensitive while making render times shorter. Values around 0.05 work well with 2-3 steps.

Sensitivity

Sensitivity is another control that controls how sensitive Kray is to detect artifacts in global illumination solution. It compares neighbouring irradiance gradients and if the difference between them is higher than specified then it will add more samples. Values around 0.05 usually work well with 2-3 steps.

Spatial tolerance

In a simplified terms, spatial tolerance defines the smallest distance between two sampled points in space. When this number is too large the shadows will look blurred and undefined. If this number is too small then the render times can get much longer than needed. Good values range around 0.05-0.1 for typical scenes. I also depends on the scale of your scene. Smaller scene will require smaller Spatial tolerance.

Angle tolerance

Angle tolerance defines the placement of sample points on curved surfaces. The value defines how much can surface angle change before new sample is added.

Distance min / distance max

In simplified terms this parameters control the minimum and maximum distance allowed between any two sample points.

The true math behind it is a little more complex. Minimum distance is acctualy the shortest lenght of the ray fired from the sample point. Rays shorter than this will be treated as if they were 'min distance' long. The opposite is true for the maximum distance. Rays that travel longer than 'max distance' will be treated as if they are 'max distance' long.

B/D

Brightness/Density will add more samples in places of high light contrast. It's not very useful any more and is left here mostly for the legacy purposes. To get better density of samples in high contrast areas use more prerender steps as described above.

Blur

Will blur neighbouring samples and thus reduce artefacts such as splotches. The drawback is that it may also blur some very fine indirect shadows. Useful values are 0-5.

Show samples

This option lets you visualize placement of FG samples. This is primarily used for testing and troubleshooting.

Off

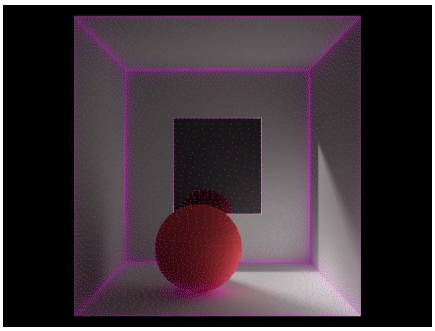
Will not show any samples.

Corners

Will show only samples in corners.

All

Will show all samples in scene.



Using Show All FG samples

FG reflections

Will turn on or off tracing of FG rays on reflective surfaces. Turning this option off may speed up render times but it may also cause inaccurate reflections.

- Note: When using material nodes changing FG reflections/refractions will not have any effect. The only way is to exclude shader from FG is by using Kray Indirect Rays shared and multiplier 0.

FG transparency/refractions

Will turn on or off tracing of FG rays on refractive or transparent surfaces. Turning this option off may speed up render times but it may also cause inaccurate calculation of GI through transparent/refractive surfaces.

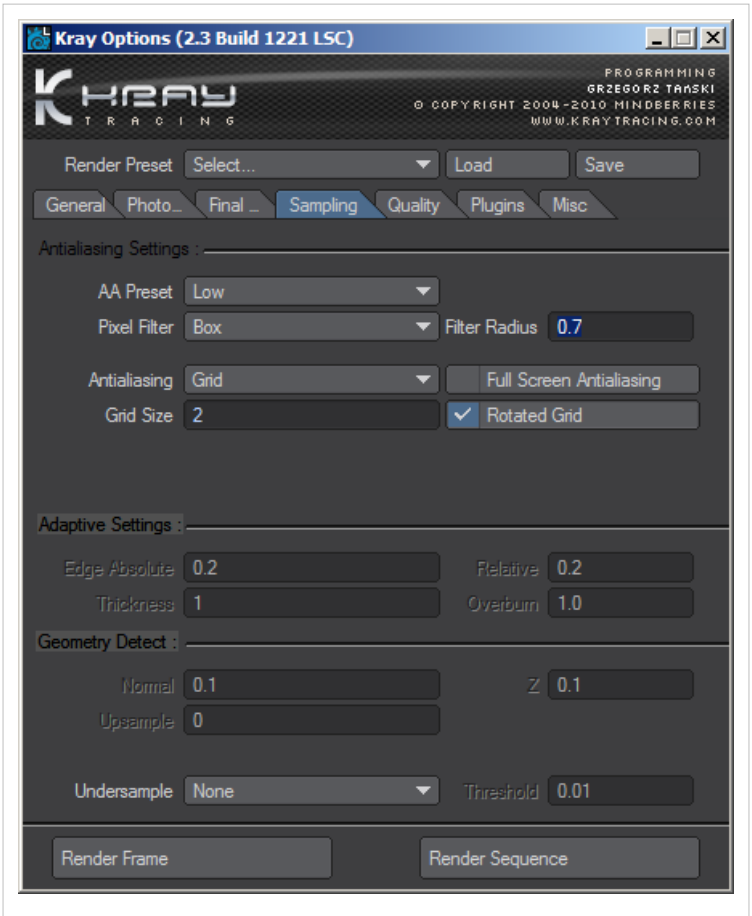
- Warning: Your windows may block FG rays if you turn this flag off. To work around put them on a separate layer and turn ON "unseen by radiostiy" in the object properties.
- Note: When using material nodes changing FG reflections/refractions will not have any effect. The only way is to exclude shader from FG is by using Kray Indirect Rays shared and multiplier 0.

Corner distance/Paths

When Paths is set to > 0 Kray will turn to path tracing when rendering corners. This may reduce splotches or light leaks. Distance specify how far from a corner paths rendering will be used.

GUI - Sampling tab

Sampling tab



AA preset

This selection list lets you choose predefined Anti-aliasing preset. When you choose Custom from the preset list advanced settings are unlocked.

Antialiasing

Antialiasing is used to smooth out jagged edges. You can choose among several antialiasing methods from this drop down list.

Grid

Grid size Controls the subdivision of anti aliasing grid. Kray will subdivide each pixel into a grid of the specified subdivision. So Grid 3 for example means each pixel will be subdivided into sub pixels.

Rotate Grid

This method helps with occasions when near horizontal or vertical lines don't get smoothed enough by using ordinary grid anti-aliasing.

FSAA

Kray has two methods of choosing which pixel to anti-alias and which not. One is adaptive (see section 6.4.2.6) and will only anti alias pixels that it considers to have bad AA. The other method is Full screen anti-aliasing or FSAA. When this option is turned on Kray will anti-alias every pixel in a render (instead of only those that suffer from bad AA). This will significantly extend render times but it's worth it if your scene contains lot's of small, fine details.

This mode is useful whenever you are rendering motion blur and DOF (depth of field).

Quasi-random

Is another method of anti aliasing that will fire specified number of rays from each pixel marked for anti aliasing. Rays are not shot in completely random way. More rays are shot in the direction where Kray thinks it contributes more to brightness of the pixel – hence Quasi-random. This is why this is very fast anti-aliasing method.

Kray starts sampling point by shooting min number of rays into the scene. Threshold defines how much difference between two pixels is "allowed". If difference is greater than that defined by threshold, Kray will increase rays, until the difference gets smaller than the threshold. It will also stop adding rays if it reaches max rays before reaching the threshold value.

Random FSAA

This works similar to new motion blur in LW 9.2. Each subframe is splited in time and different pixels from the same subframe can be rendered for different time. That changes motion blur banding into noise. It will use random method of firing rays and is therefore quite slow compared to the other methods, but it supports motion blur (MB) subframes and can be used if you want to render the scene with motion blur.

Rays will specify number of rays to be used.

This mode is useful whenever you are rendering motion blur and DOF (depth of field).

Pixel filter

You can choose between several different methods for filtering of pixels. Each pixel filter will look a little bit different. Some produce sharper and some softer images. Some are better when dealing with moire patterns.

Filter radius specifies the amount of “blur”. When radius is large image will look more blurry. With low filter radius images will look sharper.

Adaptive anti-aliasing settings

The following options control which pixels will Kray choose for anti aliasing when FSAA is turned off.

Edge absolute is a difference between two pixels. If it is higher that than value both pixels are on the edge and will not be marked for AA.

Relative is a relation between brightness of two pixels (brightness of 1st pixel / brightness of 2nd pixel). Where edge absolute often misses edges in dark areas (not enough brightness difference) relative does not.

Normal and Z is normal angle and distance to camera. Normal will catch geometry edges and also normal differences produced by bump maps. Z will catch differences in depth so even if you have black square on black

background it will be marked as edge if z is > zero.

Upsampling Z and normal can be detected to an upsampled image. For example: if you have 640x480 rendering and some details are sub pixel size, they will be skipped in edge detection. If you use upsample Kray will detect normal and Z edges as if you were rendering a larger image and won't miss those details.

Thickness is a edge thickness in around antialiased pixel that will also be marked for antialiasing.

Overburn is used to control weather Kray should anti-alias edges on overexposed areas.

```
Note: DOF (depth of field) and motion blur will not work in adaptive mode.
```

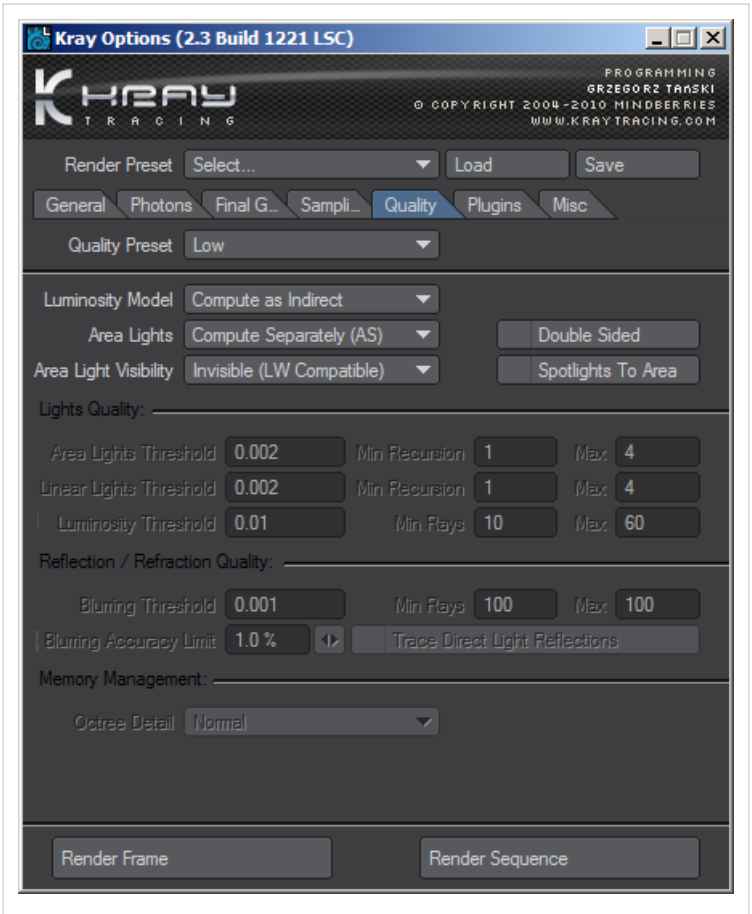
Undersample and threshold

Under sampling is a technique that can save some CPU time needed to perform rendering. This works as follows: In the first pass the image is sampled with lower resolution (Undersample factor defines how low), and if difference between neighbouring pixel values exceeds threshold image is locally sampled with higher resolution.

GUI - Quality tab

Quality tab

This selection list contains the available predefined presets.



Luminosity model

Luminosity model tells Kray how Luminosity should be computed. In LightWave luminosity is just a color. A surface like that does not illuminate anything if the radiosity is disabled. In Kray you have a choice:

- **Luminosity model: Compute as direct** Your luminosity surface can really illuminate other objects even if GI is disabled
- **Luminosity model: Compute as indirect** will illuminate other objects only if GI is on.
- **Luminosity model: Auto** Surfaces with luminosity higher than the one set by the level value will emit light directly while lower luminosity surfaces will emit light indirectly.

If the luminosity light is large and not very bright (cloudy sky dome for example) faster rendering will be achieved by using the Indirect computation method. If, however, the luminosity light source is small and very bright using the direct method is preferable.

Area lights

Area lights can be computed in two ways.

Compute separately by the use of the optimized algorithm for rectangular lights, in which case every light is computed separately

Compute with luminosity or computing them with luminosity lights. *Note that Area lights must be set to "Visible" for this option to work.*

The former is best for scenes with not so many area lights. If there are numerous area lights in a scene it can be more effective to compute the area lights with luminosity.

Double sided

This check box allows you to make area light single or double sided. If your area lights illuminate the scene from one side only, make them single sided (uncheck this option) and in so doing save many precious photons in you GI simulation.

Area lights visibility

Area lights visibility allows you to see a light. In LightWave area lights illuminates surfaces, but you cannot see them directly. This option enables you to make lights visible directly.

Note that Area lights must have inv^2 falloff to be able to be seen by Kray.

Spotlight to area

This option lets you convert all spotlights in the scene to area lights. Spotlight's soft edge angle will be used to define how soft the shadow edge should be.

Note: this option will not work if you have no falloff on your lights.

Lights Quality

Area lights

Area lights sets quality for area lights. It has an effect only when Area lights is set to Compute separately. Threshold defines quality of sampling (the lower the better). Recurse min and Recurse max sets minimum and maximum number of recursions of adaptation. Higher values give better quality (less noise) but longer rendering time.

Linear lights

Linear lights sets quality of linear lights. Meaning of fields is similar to Area lights above.

Luminosity lights

Luminosity lights sets quality of sampling illumination of objects which surface has non-zero Luminosity. If Area lights on general tab is set to Compute with luminosity these values also sets quality of area lights. Note that to enable luminosity lights at all, you need to select Luminosity model Compute as direct or Automatic.

Sampling starts with Rays min and continues unless noise level is lower then Luminosity lights threshold or Rays max is reached.

Reflection/refraction Quality

Blurring threshold

Reflection/Refraction blurring samples defines number of samples and thus quality of reflection and refraction blurring. Higher values give better quality (less noise). Sampling starts with Rays min and continues unless noise level is lower then Luminosity lights threshold or Rays max is reached.

Blurring Accuracy Limit

Blurring accuracy limit speeds up rendering of blurry reflections/refractions. Kray will use photon estimate solution instead of FG to render surfaces that have blurring higher than the value specified by blurring accuracy limit.

Trace direct lights

Allows point and linear lights to be visible in blurred reflections (like specularity), also area and lumi lights are sampled different way in this mode (like direct lights with the same pros/cons).

Memory management

Octree detail

Octree detail controls how much RAM Kray will use for rendering. Selecting **Low** will use less RAM at the expense of longer render time. **High** setting will use more RAM but you will gain some render time there. **Normal** setting is the best compromise between speed and RAM usage.

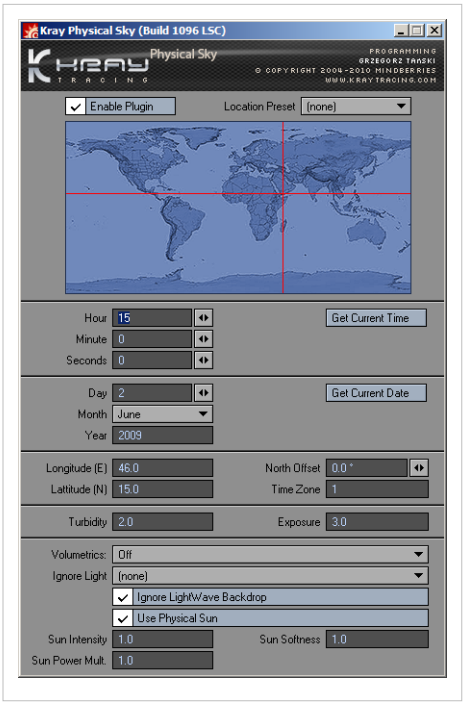
This settings will be most noticeable in scenes with lots of polygons. Simple scenes will not benefit much from this settings.

Kray plugins

Kray Master Plugins

Master plugins can be accessed by going to **Utilities** tab > **Master plugins** and from the top listbox **Add layout or scene master** choosing the appropriate Kray plugin.

Kray PhySky plugin



This plugin create physicaly correct sun and sky for easy lighting of your scene. The map allows you to select the location on earth by clicking on it. You can also select from the predefined list of cities. The position of sun will be automatically adjusted based on location and time of day/year.

Most of the settings should be self explanatory.

Nort Offset

Default north orientation is +Z axis. If you want to rotate position of North you can do so here.

Turbidity

This setting controls the color of the sky and sun. Lower values produce more clear Blue sky while higher produce more foggy, yellow sky and sun.

Exposure

Controls the brightness of the sky and sun. Increasing this value will make your scene brighter.

Ignore Light

You can select a light from the scene that will be ignored by Kray. This is handy for example if you have a Sun light in a scene which you use for LightWave or Fprime rendering but you want to ignore it in Kray.

Volumetrics

Volumetric effect adds a haze like effect to the scene. This can be applied to backdrop only or to full scene. **Affect Backdrop**: will apply volumetric effect only to backdrop. **Affect Shadows**: will apply volumetric effect to backdrop and shadows. **Affect All**: will apply volumetric effect to backdrop, shadows and geometry. This may increase rendering time.

Ignore LightWave backdrop

You can check this box if you want to ignore LightWave backdrop completely.

Use Physical Sun

When this option is off PhySky will only render light coming from sky. You need to turn this option on to get light and shadows from Sun.

Sun Intensity

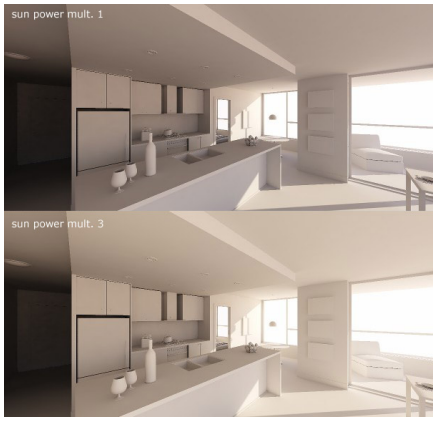
This setting controls Sun intensity separately from the exposure control above. The exposure control controls the overall brightness of the sky while Sun Intensity controls only the sun.

Sun Softness

This setting controls softness of sun shadow. Increasing this number will make Sun shadow softer.

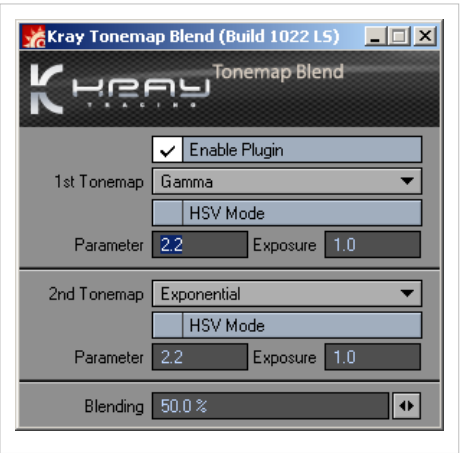
Sun Power Mult.

This setting controls power of indirect light coming fron physical sun. Increasing this number will make sun's bounced light brighter.



Sun Power Mult. Example

Kray Tonemap Blending plugin



This plugin allows you to mix two tone mappers together.

Note: Make sure you turn tonemapping in Kray General tab to "Linear". Otherwise this plugin will be adding tone mapper on top of the one from the general tab.

1st / 2nd Tonemap

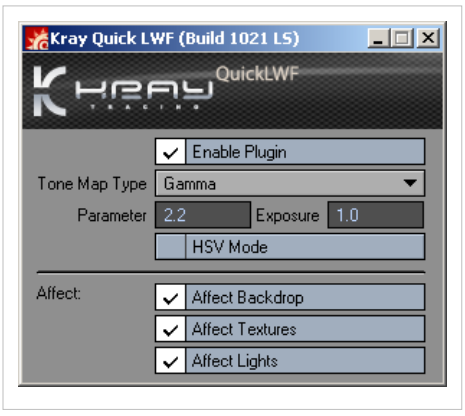
Pick the first of the two tonemappers you want to mix.

HSV : This parameter tells Kray to tonemap image by keeping saturation of colors close to original. **Parameter** : Adjust value of the tonemapper. Higher values produce less contrast while low more. **Exposure** : If you want to increase or decrease the exposure of your image enter appropriate value here.

Blending

With Blending you control how much of each tonemapper the final image contains. Value of 100% means that only 1st tonemapper will be used while 0% means only 2nd tonemapper will be used. Values in between mix 1st tonemapper with the second.

Quick Linear Workflow plugin



If you ever played with gamma values in Kray GUI you have probably noticed that increasing gamma can bring to washed out looks and changing color tones in textures. This plugin will help you get better and more accurate colors in your renders. The plugin tonemaps the incoming colors and textures so that they are calculated properly within Kray. Parameters:

Input Tone Map

with this settings you choose the tonemap of your input images and colors. Usualy this setting should be kept at Gamma.

Parameter

This is the value of your input tonemap. Usually images on computers have gamma value of 2.2 so this is what you enter here.

Exposure

If you want to increase or decrease the exposure of your input colors/textures enter appropriate value here

HSV

This parameter tells Kray that input images are tonemapped in HSV mode.

Affect backdrop

will use quick LWF settings for backdrop

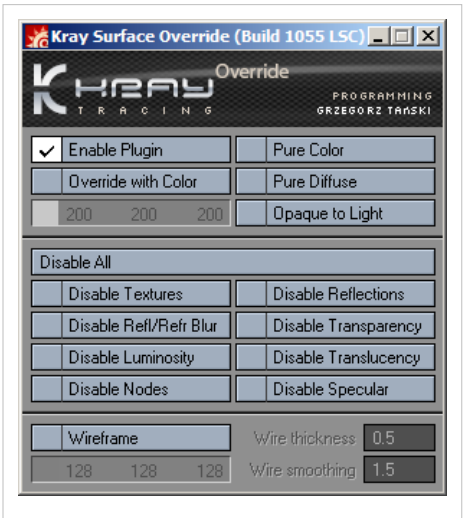
Affect textures

will use quick LWF settings for textures

Affect lights

will use quick LWF settings for light colors

Kray Override - Surface Override GUI



This is a master plugin that will help you control surface override settings.

Enable Override

Will let you quickly enable/disable override in Kray.

Override with color

Will render the object in a specified color while keeping all other channels intact (diffuse, reflection, transparency, ...).

Opaque to light

Will prevent light from passing through transparent surfaces.

Pure color

Will render surfaces in pure matte color (no textures, no shading).

Pure diffuse

Will render surfaces as a pure diffuse without any textures, colors, reflections, etc.

Disable ...

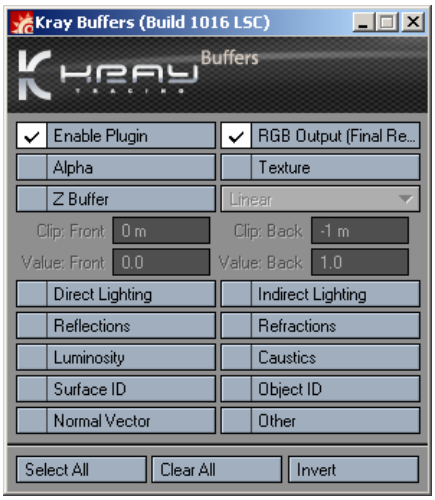
Disable Textures will turn off all textures. **Disable Refl/Refr blurring** will turn off all reflection and refraction blurring. **Disable Luminosity** will turn off all luminosity. **Disable reflections** will not render any reflections. **Disable transparency** will render transparent surfaces as opaque. **Disable translucency** will not render translucency.

Wireframe

This option will render out polygon edges in wireframe overlay. You can control the thickness, the smoothing and color of the wireframe.

Here's an example of different override options:

Kray Buffers



Kray Buffers GUI will lets you choose which buffers you want to output from Kray. Kray will create a separate file with a buffer name in the filename.

The settings are pretty self explanatory so we will not get in depth about each buffer here.

Z buffer

Z buffer mode

- linear: linear gradient from black to white
- inverse: inverted linear gradient from white to black
- log: logarithmic gradient from black to white

Clip: distance of front and back clip plane. Values above and this will be clipped. If you want to turn clipping of enter -1 for Back. **Value:** here you can define value of front and back pixels. The number is float so you can use it with HDR Z buffer.

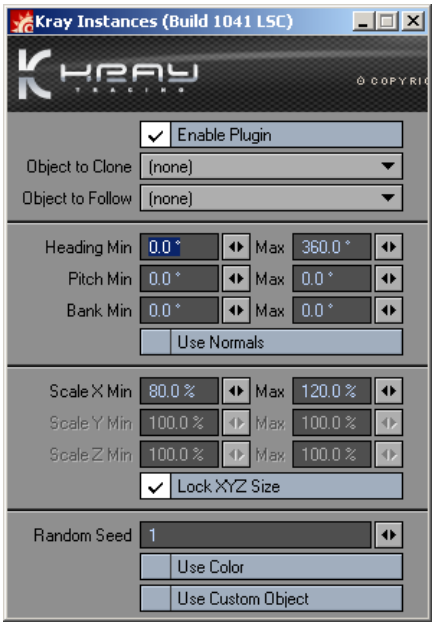
The buffers can be comped together with this formula:

```
final image=tone_mapping_operator((caus+dire+indi)*txtr+othr+lumi+refl+refr)
```

Kray Custom Object plugins

Custom object plugins can be added to object by going to **object properties** ***and on the*** *Geometry tab* ***clicking on*** *Add custom object*' list.

Kray Instances



Kray instances plugin allows you to easily place thousands of instances into your scene. The plugin allows random scale and rotation and some other useful options. It's a Custom Object plugin that should be applied to the object (template) that defines position of your instances. The template can be either polygon mesh, 1 point polygons or pure points (no polys). Pivot point of the object will be used as the center of the instance so make sure it is properly positioned otherwise you may get unpredictable results.

Object to clone

Choose object that will be instanced

Object to Follow

Here you can choose object that instances will “follow”. Instances will use it’s rotation and scale parameters.

Rotation

Here you define random rotation settings.

Use normals

Turning this ON will align instances to template mesh normals (only works on polygons!).

Scaling

Here you define random scaling settings.

Lock XYZ Size

Turning this ON will lock X, Y, Z scale parameters so object will be resized equally on all axis.

Use Color

Turning this ON will use color from the template object and color filter the instances.

Use custom object

You can use special object to place your instances. The default object is triangle 1mx1mx1m in size with a 2 point polyline defining it's axis. This object defines position, orientation and scale of the instances. Note: you can still apply random rotation/scale values on top.

Example scene

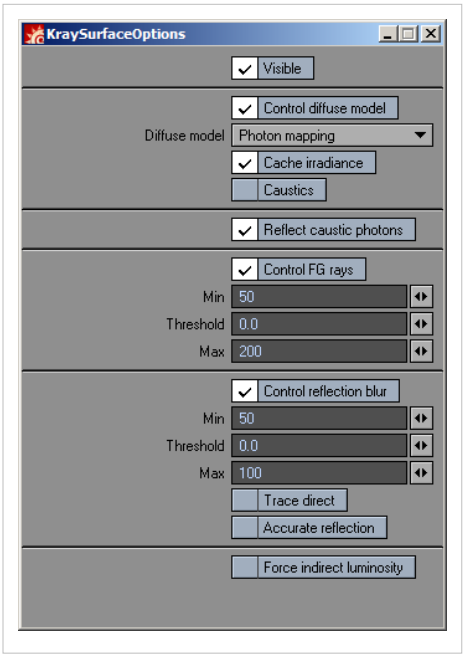
Download an example scene here: Kray Instancing example scene [1]

Kray Surface Shaders

Surface shaders can be applied to surfaces by opening **Surface editor** and on the **Shaders** tab clicking on **Add shader** list.

Kray Surface Options

The KraySurfaceOptions Shader can be used to change various Kray settings for each surface.



Visible

Turns the visibility (from everything, camera, rays, reflection etc) of the surface on/off.

Control diffuse model

Enables you to change the GI model for this surface only. The available models are all that are normally available to Kray : Raytrace, Photon Filtered, Photon Filtered+Raytrace Direct, Photon Mapping, Path Tracing.

Cache Irradiance

(only available with Photon Mapping and Path Tracing) Allows you to turn on or off Irradiance Caching for this surface.

Caustics

Allows you to turn on or off caustics for this surface.

Reflect Caustic photons

In a scene that has Caustics on, unchecking this will disable the production of *reflection* caustic photons from this surface. If the scene has Caustics off this setting will do nothing.

Control FG rays

Allows you to use different number of rays for this surface, so you can use less rays for areas

Control Reflection Blur

Allows you to use different number of reflection rays per surface, so you can for example use more rays for the floor surface and less for the rest of the scene.

Trace direct

Same as trace direct light reflection in the quality tab.

Accurate reflection

Allows you to override the Blurring accuracy limit setting in Kray rendering options and use the FG solution instead of photon filtered for reflection of this surface.

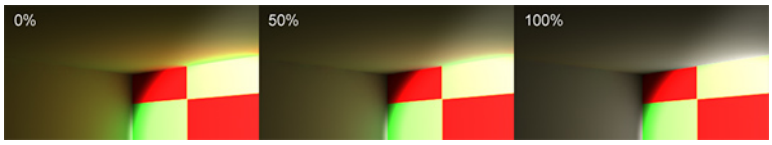
Force indirect luminosity

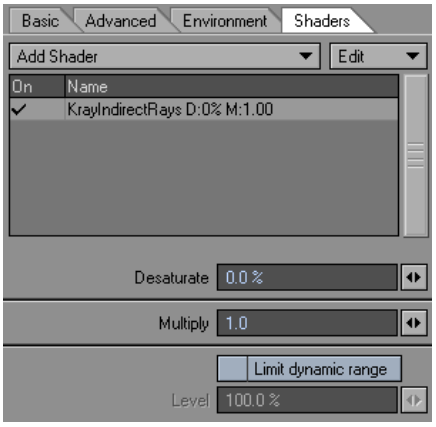
Allows you to make a luminous polygon emit light indirectly regardless of the settings in the Quality tab of the Kray rendering options.

Kray Indirect Rays

Desaturate

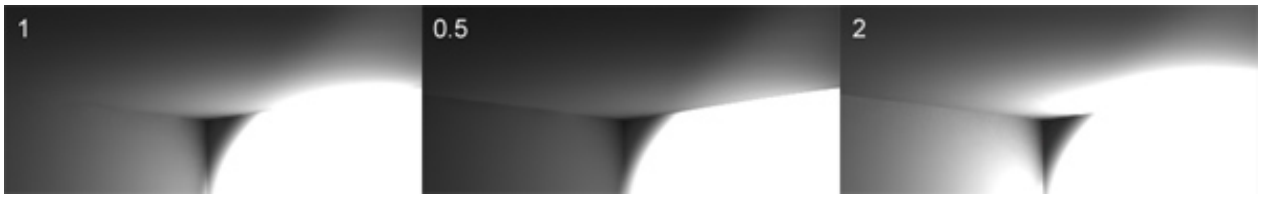
This can be used to control color bleeding. The percentage represents how much the light that bounces of the surface will be desaturated.





Multiply

Multiply can increase or decrease the amount of light that bounces off the surface.



Limit Dynamic Range

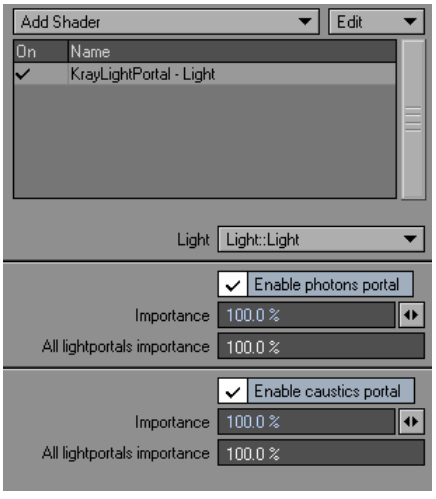
Limits the dynamic range of the indirect rays that leave the surface. Can be used on luminous geometry that emit light indirectly. The emitted light is clamped to limit dynamic range value eliminating over bright values which effectively reduces possible splotches (but also the power of the emitted light gets reduced as well), while the surface itself appears to be very bright. This is an unrealistic function but it can be a useful tool too when you want to improve your result without caring too much for realism.

Kray Light Portal

See also Using Lightportals tutorial.

The KrayLightPortal can be used to make some surfaces act as Light Portals. Light Portals are special surfaces that will guide photons into places where you need them. This helps you get better photon distribution in places where you need it. Remember that the Light portal surface normal should be facing towards the light source you want it to attract photons from. A tutorial on how to use this shader can be found here: Using Lightportals.

This shader is useful only in Photonmap mode since Lightmap already distributes photons in based on camera view.



Light

This Drop down list enables you to link this light portal to a specific light. If you want to link multiple lights to a specific light portal you can add the KrayLightPortal shader multiple times on the same surfaces.

Enable Photons Portal

Ticking this will make this surface act as a Light portal

Importance: This percentage amount to the percentage of photons that will be aimed at this surface.

All lightportals importance: This shows you the added importance of all light portals. If the sum of all light portals exceeds 100% the percentage of the *other* light portals will be reduced so that the sum is 100% once you exit the light portals panel.

Enable Caustics Portal

Importance: This percentage amount to the percentage of caustic photons that will be aimed at this surface.

All lightportals importance: This shows you the added importance of all light portals. If the sum of all light portals exceeds 100% the percentage of the other light portals will be reduced so that the sum is 100% once you exit the light portals panel.

KrayPhotonMultiplier Surface Shader

With this shader you can control the amount and intensity(power) of photons leaving the surface. This is similar to power multiplier setting in the Kray GUI - Photons tab but it work on per surface basis.

Luminosity number multiplier

You can increase the number of photons emitted from Luminous surface by increasing this number.

Luminosity power multiplier

You can increase the power (intensity) of the photons emitted from Luminous surface by increasing this number.

Diffuse number multiplier

You can increase the number of the photons bouncing from diffuse surface by increasing this number.

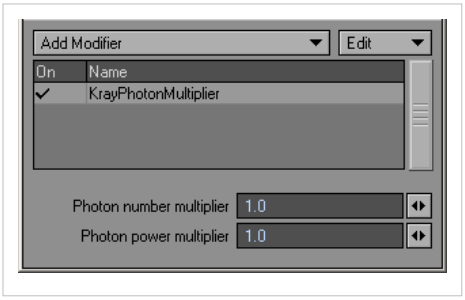
Diffuse power multiplier

You can increase the power (intensity) of the photons bouncing from diffuse surface by increasing this number. This can be helpful for example in a room with dark floor where bounced photons would be few. Increasing this setting will make the room lighter as if the floor was brighter.

The Kray motion modifier

The motion modifier plugins can be added to the object/light by pressing 'm' key or going to **Window > Motion options**. In the bottom of **IK and modifiers tab** the **Add modifier** list contains the list of plugins available.

KrayPhotonMultiplier



KrayPhotonMultiplier modifier controls how much photons a light source emits. You can apply this modifier to a Light or an object containing luminous polygons.

You should assign this shader to a light source that is not emitting enough photons or if you want the power of light to be stronger.

Photon Number multiplier

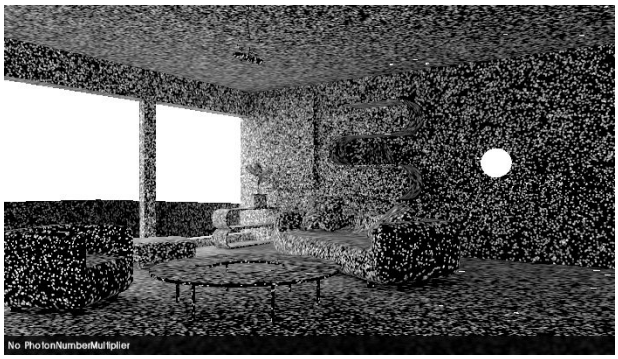
This setting will increase the number of photons the light emits.

Photon Power multiplier

This setting will increase the power of the photons the light emits. Note that the intensity of the direct light will not be stronger. Only bounced light will be brighter.

You must experiment to find the right value but that's usually quite easy: just render in **Photon estimate - Global unfiltered** or **Precomputed** mode and find the right balance.

Here's an example of a render **without using PhotonNumberMultiplier:**



Unfiltered: No PhotonNumberMultiplier

Precomputed: No PhotonNumberMultiplier

And here's an example of a render **with PhotonNumberMultiplier 100000.0:**

Unfiltered: With PhotonNumberMultiplier

Precomputed: With PhotonNumberMultiplier

Modeler Plugins

KrayAutoParts

When using direct luminosity model, Kray will only recognize 1 light per luminous surface/object. But when you assign different part names then Kray will treat each part as a separate light source.

This plugin automatically assigns part names to selected polygons. Each group of polygons that is connected will get one part name.

KrayAutoParts.p is available as a separate download from the Kray website. [2]

References

[1] http://www.kraytracing.com/resources/instancing_example.zip
[2] http://www.kraytracing.com

Advanced features

About Commands

Kray has a built in script language for scene definition. It allows user to define the scene in a text file. LightWave's version of Kray uses script to pass parameters from GUI to Kray render core (you can see it if you open .lws file with Kray settings in a text editor). Some Kray options are not exposed in GUI and are available only from Kray script.

You can run Kray script commands from Kray GUI level. On the misc Tab there are 2 fields: **Header commands** and **Tailer commands**. They correspond to Kray script commands fired before importing LW scene (Header) and after import (Tailer). Some commands should be fired before importing LW's scene, some after and some works no matter where they are placed.

Header commands

var __blendss

Determines how polygons casts shadows (for direct illumination).

value = 0 - polygon cast shadow no matter if it is front sided or back sided to the light source.

value = 1 - only front side of polygon casts shadow (like in LW).

value = 2 - only back side of polygon casts shadow.

```
Example:var __blendss,0; // default value
```

var __undersample_edge

Sets undersampling mode.

value = 1 means edge undersampling (slower, but more accurate).

value = 0 is corner undersampling.

```
Example:var __undersample_edge,1; // default value
```

var __importflags,__importflags|64;

Controls how shadows are rendered in objects that have dissolve applied. Default is solid shadows. To turn lw compatible mode use:

```
var __importflags,__importflags|64;
```

Tailer commands

animmode

Will force Kray to shoot photons across camera path on screamernet by reading frame range from .lws.

```
 Example:  animmode 1;
```

edgedetectglobals

edgedetectglobals <edge thickness>,<edges upsampling>;

Controls global (slot independent) settings for edge detector.

edgedetectslot

edgedetectorslot <slot index>,<edge absolute>,<edge relative>,<normal detection>,<z detection>,<overburn level>;

Edge detection settings slot. **<slot index>**=0 hold default settings for a surface. Slot index used by *surface* can be controlled in *surface options*. This allows to use different edge detection settings per surface.

octree

octree <depth>,<object per octree leaf>;

<depth> - octree depth the same as "Max octree depth in GUI"

<object per octree leaf> - how many objects will octree leaf will include. Default value is 14. Resonable values are 4-50. Higher means slower rendering, but lower octree memory usage

```
Example: octree 60,20;
```

octsmallcube

Usage: octsmallcube <ratio>;

<ratio> Additional parameter for octree generation. It is a ratio between size of polygon and smallest octree leaf. For advanced users.

```
Example: octsmallcube 2;
```

octbuild

octbuild <cost>,<separation>,<force_split>;

<cost> - parameter that controls how much ram per polygon will be used by octree. Default is 40. Reasonable values are 10 (low memory usage, slower raytracing), 200 (more mem, faster render)

<separation> - another trade of between ram and speed. Default 0.1. Values between 0-1. Resonable values are <0.5

<force_split> - 0 or 1. If zero Kray will not divide scene where it looks that this will not give much (speed). If 1 it will divide it anyway and hopes next division will give some gain.

```
Example: octbuild 60,0.1,0;
```

octshow 2

Shows octree. Green boxes means fine division. Yellow not so good, but good enough. Red poor and white bad. If there are some small white or red boxes but most of they are green its ok. If white or red boxes dominates or they are big, this means raytracing performance will be poor.

Note: LW Classic camera must be used.

```
Example: octshow 2;
```

recurse

Sets ray recursion depth and overrides LightWave's setting. Allows to set higher recursion depth then available in LW.

```
Example: recurse 100;
```

Other commands

Commands below can be used in either header or tailer.

about

Displays info about rendering engine.

```
Example: about;
```

accurateshadows

Control shadow casting routine. Objects with smoothing applied will render smooth shadows even if evaluated polygon is not seen direcly by the light source. In some cases this can produce light leaks (that is the conseqence of "smoothing"). To turn on accurate shadows use *accurateshadows 1;*. This will prevent leaks, but parts of object in shadow are shadowed and this is percived as "smoothing error"). Default is *accurateshadows 1;* (LightWave compatible mode).

```
Example: accurateshadows 1; // Turns on accurate shadows.
```

cam_singleside

If *mode* = 1 camera rays intersects front side of polygon only. When *mode* = 0 camera rays intersects both sides of polygon.

```
Example:cam_singleside 1; // default value
```

debug "debug_flags"

Displays debug infos in rendering log. *debug_flags* determines what information will be displayed.

debug_flags	**Meaning**
0	No debug info
1	Luminosity lights
2	Objects import
4	Rendering finalization
8	Log to file (Default filename is *kray_log.txt*. It can be changed with logfile)

You can sum *debug_flags*. For example *debug_flags* = 2+8 will show object import info and save render log to a file. By default *debug_flags* is set to 0.

```
Example: debug 1+2+4+8;
```

echo "text"

Displays *text* on Kray console.

```
Example: echo "This text will be displayed in Kray console.";
```

finishclose

Forces Kray windows to close automatically after rendering (useful when you are rendering with LightWaves's Render-Q)

```
Example: finishclose;
```

globalpmblurmultiplier

Controls power compensation of blurred reflections/refractions, it should be reverse of photon power multiplier i.e. photon power=5 -> globalpmblurmultiplier 1/5;)

```
Example: globalpmblurmultiplier 1/5;
```

importancegammamultiplier

Controls quality of reflection blur. 1 is default. Values >1 - lower quality faster render <1 slower, more rays used in recursive reflections.

```
Example: importancegammamultiplier 1.5;
```

include "filename"

Allows to load set of Kray script commands from external file.

```
Example: include "myfile.txt";
```

lmimportance "importance"

Controls how accurate direct light and caustics are computed during lightmap building. *importance* = 1 computes direct light and caustics with best quality (slowest). *importance* = 0 gives fastest, but most inaccurate result.

```
Example: lmimportance 0.5;
         lmimportance 0; // default value
```

logfile "filename"

Enables loging to a file and sets log *filename*. See debug for more loging options.

```
Example: logfile 'render_log1.txt';
```

lwblurdoubleside

Controls how refraction blur is computed. When set to 1, refraction blur is applied on both incoming and outgoing rays of glass surface. When set to 0 (default, LW compatible setting) refractin blur is applied only for incoming rays.

```
Example: lwblurdoubleside 1; // enable blur on both sides
```

lwo2rayslimit

Can prevent problems with LW carpaint material. **lwo2rayslimit 1,0.9;** works as follows: rays above recursion level 1 of node evaluation will be fired only in 90% (0.9) tries. It prevents for very depth recursion (which can happend with car paint material)

```
Example: lwo2rayslimit 1,0.9;
```

lwo2selfinstance

lwo2selfinstance 1; will use old, but faster clip map code (but may cause incompatibility with LW in certain cases when)

lwo2selfinstance 3; will use new clip map code (slower, but more LW compatible)

```
Example: lwo2selfinstance 3;
```

lwo2unseenbyrays_affectsgi

lwo2unseenbyrays_affectsgi 0; When you add this to tailer, object can affect gi, but will be unseen by rays (reflection and refraction) and camera. This is not LW compatible, but allows to hide object and still affect GI. By default (LW compatible mode) unseen by rays implies unseen by gi.

```
Example: lwo2unseenbyrays_affectsgi 0;
```

lwtransoptions sideness_mode,color

sideness_mode determines how transparency/refraction works.

sideness_mode = 0 transparency/refraction ray leaves object before next intersection

sideness_mode = 1 transparency/refraction ray intersects both sides of a polygon

sideness_mode = 2 transparency/refraction ray intersects front hit front side of polygon only

If *color* = 1 transparency and refraction amount is computed separately for r,g,b components if there is a refraction/transparency texture. *color* = 0 is LW compatible setting, transparency and refraction uses grayscaled refraction/transparency texture.

```
Example: lwtransoptions 1,1; // default values
```

lwvolumetrics

Enables or disables calculation of FG on volumetrics.

lwvolumetrics 1; Enable rendering of Volumetrics lwvolumetrics 2; Affect FG lwvolumetrics 4; Affect Shadows

```
Example: lwvolumetrics 1+2+4; // enables FG and shadow calculation on volumetrics.
```

limitdr

Limits dynamic range of primary rays (those fired from camera). If a RGB component of a camera ray is bigger then *value* it is truncated. This is useful to make sure the overbright parts of an image are antialiased properly, for example the edges of polygons with Luminosity which is much higher than 100%.

By default **limitdr** is set to zero (no dynamic range limit). Note that for LDR images, image dynamic range is always limited to output exposure. The only difference with **limitdr** is that this is done per camera ray, not per pixel. Limitdr command can be header or tailer, but it acts different depending on where placed. When its header, limitdr is applied before tone mapping. When its tailer, its applied after tone mapping.

```
Example:limitdr 0; // no dynamic range limit
        limitdr 1;
```

mipmap "filter_size_shift,additional_samples,minsamples,maxsamples"

Controls anisotropic filtering of textures (texture antialiasing).

With *filter_size_shift* you can control texture blurring. Negative values decreases blurring, positive increases. Typical values are between -1 and 1. Values < -1 can cause texture aliasing. Values > 1 can look too blurred.

additional_samples,minsamples and *maxsamples* controls number of samples Kray uses for anisotropic filtering. More samples - slower rendering, but better quality of texture filtering. Kray computes number of samples from texture anisotropy level and than adds *additional_samples* to this value. *minsamples,maxsamples* are minimum and maximum limits.

```
Example: mipmap 0.2,4,5,30;mipmap 0,0,0,30; // default values
```

octstats

Displays information about octree used for rays intersection optimization. Information for advanced users and debugging purposes.

```
Example: octstats;
```

octsmallcube ratio

Additional parameter for octree generation. It is a *ratio* between size of polygon and smallest octree leaf. For advanced users.

```
Example: octsmallcube 2; // default value
```

overridesurface

Override surface options let's you controll the way object is rendered. To make your life easier use KrayOverride.ls master plugin which can be found in the downloads section of the webpage.

Meaning of bits of flag is below:

- all surfaces cast shadow 1 (even transparent surfaces will cast shadow)
- disable textures 2 (will use only base color)
- disable color 4 (overrides all object with specified color)
- disable shading 8 (disables all shading and shadows)
- disable reflection/refraction blur: 16
- disable luminostiy: 32
- disable specular: 64
- disable reflection: 128
- disable transparency: 256
- disable translucency: 512
- diffuse only 1024 (renders objects as pure diffuse with no textures, reflections, transparency etc.)
- disable lw material node 2048 (turns off material nodes for all surfaces)
- disable color wireframe mode 4096 (turns off color and replaces with wireframe shader)

```
Example: overridesurface 1+2+4+1024; is old surface override behaviour (all diffuse including reflection and refraction)
```

```
Example: overridesurface 2+16+64; does not use Override surface color, but disables only textures (surface color is used).
```

photons_singleside

Enables front side intersection only for photons fired from light sources. By default photons hits both sides of polygon.

```
Example: photons_singleside;
```

postprocess

Adds post processing filter to Kray's native post processing system.

Note: Some examples are multiline. You can use multiline syntax in Kray script files (see include [1] command), or write a command in single line and directly place in Header/Tailer field.

postprocess add,(r,g,b)

Adds *r*,*g*,*b* values to every pixel of rendered image.

```
Example: postprocess add,(0.5,0.5,0.5);
```

postprocess ca,strength,aspect_ratio,distance_influence,samples

Chromatic aberration postprocessing filter.

Strength controls how much the effect is visible.

Aspect_ratio is the pixel aspect ratio.

Distance_influence controls how abberation strength changes with a distance from center of an image.

Samples is numer of samples. Controls quality/speed ratio. Minimum numer of samples is 3 (one sample per r,g,b channel)

```
Example: postprocess ca,0.1,1,2,100;
```

postprocess autoexp,strength

Compensates exposure of an image. *Strength* is a value between 0 and 1. *Strength*=0 no compensation. *Strength*=1 full compensation.

```
Example: postprocess autoexp,1;
```

postprocess filter3

3x3 convolution filter.

```
Example 1: postprocess filter3,-1,-1,0,-1,0 ,1,0 ,1 ,1;
```

```
Example 2: limitdr 0;postprocess filter3,-1,-1,0,-1,0,1,0,1,1;postprocess add,(0.5,0.5,0.5);
```

postprocess filter5

5x5 convolution filter.

```
Example: postprocess filter5,0 ,-1,-1,0 ,0,-1,-1,-1,0 ,0,-1,-1,0 ,1 ,1,0 ,0 ,1 ,1 ,1,0 ,0 ,1 ,1 ,0;
```

postprocess gblur,blur radius,blend

Gaussian blur blended with original image. *Blur radius* controls amount of gaussian blurring. *Blend* is a balance between original image and blurred image. *Blend*=0 , only original image (filter does nothing). *Blend*=1 , only blurring, no blending with original. *Blend*=0.5 , 50% blurred image 50% original image. This filter works nice with overblown areas on the images. See example below and compare it with **limitdr** examples.

```
Example: limitdr 0;postprocess gblur,15,0.4;
```

postprocess mult,(r,g,b)

Multiplies every pixel of the render image by *r*,*g*,*b* values.

```
Example: postprocess mult,(0.5,0.5,0.5);
```

postprocess desaturate

v is value between 0 and 1, 0 - no change 1 - grayscale

```
Example: postprocess desaturate,0.9; will make almost grayscale render.
```

postprocess gamma,(v)

```
Example: postprocess gamma,0.8
```

postprocess erode

This command is good when baking a shadow or any kind of UV. It will expand the borders by so many pixels depending on which value you put in (0 to 5)

```
Example : postprocess erode,5; (in header or tailer commands)
```

postprocess saveimage

Saves current state of image to a file. Note that used with bitmapprocess allows to save *bitmap* object to a file.

```
Example : postprocess saveimage,"filename.png",png;
          bitmapprocess img,saveimage,"test.jpg",jpg,10;  // last param is quality of jpg image
```

previewsize

Usage: previwesize width,height;

This command enables you to specify the size of render preview window.

```
Example: previewsize 2000,1500;
```

rem comment

Comment in Kray script. Not very useful in **Header commands** and **Tailer commands** fields, but can be handy in Kray script files (see include).

```
Example: rem This is comment. Kray will ignore it.
```

renderinfo info text

Adds information bar to rendered image. Info text can contain following symbols:

Symbol	Meaning
%kray%	Kray logo.
%ver%	Kray version number.
%build%	Kray build time.
%arch%	System architecture (bus width 32/64 and endianess BE/LE/ME).
%time%	Rendering time in ***<hours>*h *<minutes>*min *<seconds>*sec** format.
%ptime%	Rendering time in ***<hours>*:*<minutes>*:*<seconds>*:*<centiseconds>*** format.
%days%	Render time - days.
%hours%	Render time - hours.
%mins%	Render time - minutes.
%secs%	Render time - seconds.
%centisecs%	Render time - 1/10 seconds.
%sectime%	Total render time in seconds.
%now%	Current time and date.
%file%	Output image filename.
%fileformat%	Output file format.
%width%	Image width.
%height%	Image height.
%fgraysmin%	FG tab/Min rays value.
%fgraysmax%	FG tab/Max rays value.
%threads%	Number of render threads.
%frame%	Current frame.
%buffer%	Buffer name (if rendering buffers other then RGB, see needbuffers).

You can also add custom symbols from Kray script level. For example:

```
var test1,5;
string test2,"test_string";
renderinfo "value is %test1% name is %test2%";
```

will produce **value is 5 name is test_string**. The same symbols can be also used in output render filename.

```
Example: renderinfo "%kray% %ver% | Rendering time : %time%";
```

renderinfosize text scale,border size in pixels

Scales render info font size and sets border size.

```
Example: renderinfosize 0.66,22; // default setting
```

showcornersamples (r,g,b)

Shows irradiance samples in corners with desired color (those who matches **Thin geometry distance**). Works only if irradiance caching is on.

```
Example: showcornersamples (10,0,0);
```

showphstats

Shows photon statistics.

```
Example: showphstats;
```

smoothprecachescale ratio

Sets the *ratio* between **precached** and **precached filtered** search range. **Precached filtered** interpolates between neighbour irradiance samples and require bigger search range (**precached filtered** needs at least 2 samples when for **precached** one sample is enough). High values gives better filtering, but needs more computing power.

```
Example: smoothprecachescale 1.2; // default value
```

surface_flags flags

Set flags for diffuse surfaces. Flags allow to enable/disable direct lighting computation and texturing (and output irradiance).

flags	**Meaning**
1	Enables direct lighting computation
2	Enables textures

Surface flags can be added. *flags* = 1+2 means that both direct lights and textures must be computed (default setting). Disabling textures and direct lighting is very useful combined with texture baker. This allows to bake irradiance maps in low resolution without loosing details of texturing and shadows.

```
Example: surface_flags 1+2; // default value
```

lwbasenormalgrab

In LW9.2-9.5 smoothing normals are computed wrongly. lwbasenormalgrab 1; can be used to overcome this problem.

outputformat

This will output file with Alpha channel. You can use the following commands: *:outputformat pnga;*

outputformat bmpa;

outputformat tifa;

outputformat tgaa;

```
example: outputformat pnga;
```

outputtolw

Kray buffers cannot be saved by default when Lightwave image saver is used. To save Kray Buffers and still use Lightwave Image saver do this: set regular Kray output filename and add outputtolw 1; to the list of commands.

```
example: outputtolw 1;
```

gradients_flags

+1 - turns on irradiance gradients (higher quality, but slower interpolation)

+2 - fg samples are computed in rendering phase

+4 - fg samples are computed in antialiasing phase

When +2 or +4 is missing, value from "Precomputed filtered" is used instead of black color.

```
Example: gradients_flags 0+1+2+4;
```

needbuffers

Outputs different channels each to a separate file.

+0x1 *RGB output (final render)* (on by default)

+0x2 *alpha channel* (filename suffix **_alph**)

+0x4 *texture* (suffix **_txtr)**

+0x8 *antialiased z buffer* (suffix **_zbuf**) NOTE: you can save non antialiased z buffer with *postprocess savez,'filename.hdr';*

+0x10 *direct lighting* (suffix **_dire**)

+0x40 *indirect lighting* (suffix **_indi**)

+0x100 *caustics* (suffix **_caus**)

+0x400 *reflections* (suffix **_refl**)

+0x1000 *refractions* (suffix **_refr**)

+0x4000 *luminosity* (suffix **_lumi**)

+0x10000 *other* (suffix **_othr**) these include backdrop, specularity and everything not included in one of above

+0x40000 *surface id* (suffix **_sfid**) random color for each surface

+0x100000 *object id* (suffix **_obid**) random color for each object

The buffers can be comped together with this formula:

```
final_image=tone_mapping_operator((caus+dire+indi)*txtr+othr+lumi+refl+refr)
```

Note: You can use *outputformat pnga;* or *outputformat tgaa;* tailer commands to force Kray save 32bit (RGBA) PNG or TGA with alpha channel. Otherwise alpha channel will be saved in a separate image.

```
Example: needbuffers 0+0x1+0x2+0x4+0x8; will output final render, alpha, texture and Z buffer
```

lwo2airpolys

lwo2airpolys 0;

no need to use air polygons on glass.

lwo2airpolys 1;

99,9% compatible with LW <9.0. Need to use air polys on glass.

lwo2rayslimit

It prevents for very deep ray recursion which can be very slow sometimes with LW nodes. This command helps in those situations.

Example: rays above recursion level 1 of node evaluation will be fired only in 90% (0.9) tries.

```
lwo2rayslimit 1,0.9;
```

local oversample

to enable this type in tailer commands:

prerender <prerender>,<steps>,0;

where prerender overrides prerender from GUI (values from 0 to 1) <steps> - number of prerendering steps (FG refinement steps) additional command controls splotch detection

```
Example: prerender 1,2,0.01; will render 100% prerender in 2 steps with 0.01 tolerance.
```

gradients_neighbour

gradients_neighbour <sensitivity>;

values from 0 to 1. default is 0.5

```
Example: gradients_neighbour 0.01;
```

Photon received minimum hit ratio

Limits hit ratio (so sometimes photons will stop firing if the hit ratio is lower). Default value is 1% (0.01).

```
Example: photonreceivedminhitratio 0.01;
```

Z- buffer tonemapper

Clips front and back of the Z-buffer to the specified values.

Mode: <linearlinversellog>

white_clip: Distance of front clipping plane **black_clip:** Distance of back clipping plane **white_value:** Color value of front clipping plane **black_value:** Color of back clipping plane

```
ztonemapper linear,0,100,1,0;
```

Custom GUI plugins

If you are familiar with Lscript or C programming, you may write custom GUI for Kray commands. If you open .lws file with Kray settings in a text editor you can find Kray commands exported from *Kray.ls* plugin. They are organized in sections. Each section starts with

```
KrayScriptLWSInlined <section order>;
```

and ends with

```
end;
```

Any plugin that can write settings to *.lws* file can add new sections and they will be parsed by Kray. **Section order** is an integer number that tells Kray when this section should be parsed. Sections with lower numbers are parsed first. Sections with negative numbers are parsed before loading LightWave scene (like Header_commands). Sections with positive numbers are parsed after scene is imported to Kray (Tailer commands). *Kray.ls* uses section numbers -2000 (variables settings) -1000 (header commands) and 1000 (tailer commands).

KrayBuffers (source code archive [2]) is a graphical interface to needbuffers command and example Lscript code of how to write custom GUI plugins for Kray.

References

[1] http://www.kraytracing.com/manual/kray_commands/index.php#include
[2] http://www.kraytracing.com/pub/scripts_ls.zip

Article Sources and Contributors

Kray User Manual *Source*: http://www.kraytracing.com/wiki/index.php?oldid=1766 *Contributors*: AcidArrow, Admin, Erwin zwart, Jure, 2 anonymous edits

Installation *Source*: http://www.kraytracing.com/wiki/index.php?oldid=1718 *Contributors*: AcidArrow, Admin, Erwin zwart, Haven1000, Jure, Khan973

Basics of global illumination *Source*: http://www.kraytracing.com/wiki/index.php?oldid=1554 *Contributors*: AcidArrow, Admin, Jure, Matt gorner

Setting up Lightwave *Source*: http://www.kraytracing.com/wiki/index.php?oldid=1461 *Contributors*: AcidArrow, Admin, Erwin zwart, Jhansen33@gmail.com, Jure, Noble

Kray Interface *Source*: http://www.kraytracing.com/wiki/index.php?oldid=1462 *Contributors*: Admin, Jure

GUI - General tab *Source*: http://www.kraytracing.com/wiki/index.php?oldid=1463 *Contributors*: AcidArrow, Admin, Gtanski, Jure, Limbus, Noble, 2 anonymous edits

GUI - Photons tab *Source*: http://www.kraytracing.com/wiki/index.php?oldid=1464 *Contributors*: Admin, Jure, Talats, 2 anonymous edits

GUI - FG tab *Source*: http://www.kraytracing.com/wiki/index.php?oldid=1465 *Contributors*: Admin, Jure, Talats, 2 anonymous edits

GUI - Sampling tab *Source*: http://www.kraytracing.com/wiki/index.php?oldid=1466 *Contributors*: AcidArrow, Admin, Jure, 4 anonymous edits

GUI - Quality tab *Source*: http://www.kraytracing.com/wiki/index.php?oldid=1467 *Contributors*: Admin, Jure, 5 anonymous edits

Kray plugins *Source*: http://www.kraytracing.com/wiki/index.php?oldid=1470 *Contributors*: AcidArrow, Admin, Gtanski, Haven1000, Jure

Advanced features *Source*: http://www.kraytracing.com/wiki/index.php?oldid=1471 *Contributors*: Admin, ErwinZwart, Gtanski, Jure, Marky, 4 anonymous edits

Image Sources, Licenses and Contributors

Image:unfiltered.jpg *Source*: http://www.kraytracing.com/wiki/index.php?title=File:Unfiltered.jpg *License*: unknown *Contributors*: Admin
Image:Samples.jpg *Source*: http://www.kraytracing.com/wiki/index.php?title=File:Samples.jpg *License*: unknown *Contributors*: Admin
Image:RaytraceFlags.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:RaytraceFlags.png *License*: unknown *Contributors*: AcidArrow
Image:TexturedLuminosity.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:TexturedLuminosity.png *License*: unknown *Contributors*: AcidArrow
Image:KrayRenderOptions.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:KrayRenderOptions.png *License*: unknown *Contributors*: AcidArrow, Admin, Jure
Image:Precomputed.jpg *Source*: http://www.kraytracing.com/wiki/index.php?title=File:Precomputed.jpg *License*: unknown *Contributors*: Admin
Image:Precomp_filtered.jpg *Source*: http://www.kraytracing.com/wiki/index.php?title=File:Precomp_filtered.jpg *License*: unknown *Contributors*: Admin
Image:PhotonsTab.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:PhotonsTab.png *License*: unknown *Contributors*: AcidArrow, Admin, Jure
Image:FGTab.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:FGTab.png *License*: unknown *Contributors*: AcidArrow, Admin, Joreldraw, Jure
Image:samples.jpg *Source*: http://www.kraytracing.com/wiki/index.php?title=File:Samples.jpg *License*: unknown *Contributors*: Admin
Image:SamplingTab.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:SamplingTab.png *License*: unknown *Contributors*: AcidArrow, Admin, Jure
Image:QualityTab.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:QualityTab.png *License*: unknown *Contributors*: AcidArrow, Admin, Jure
Image:KrayPhySky.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:KrayPhySky.png *License*: unknown *Contributors*: Admin, Jure
Image:Sunpowermult.jpg *Source*: http://www.kraytracing.com/wiki/index.php?title=File:Sunpowermult.jpg *License*: unknown *Contributors*: Jure
Image:KrayTonemapBlend.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:KrayTonemapBlend.png *License*: unknown *Contributors*: Admin
Image:QLWF.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:QLWF.png *License*: unknown *Contributors*: Admin
Image:KrayOverride.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:KrayOverride.png *License*: unknown *Contributors*: Admin, Jure
Image:Override.gif *Source*: http://www.kraytracing.com/wiki/index.php?title=File:Override.gif *License*: unknown *Contributors*: Admin
Image:KrayBuffers.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:KrayBuffers.png *License*: unknown *Contributors*: Admin
Image:krayinstances.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:Krayinstances.png *License*: unknown *Contributors*: Admin
Image:KraySurfaceOptions.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:KraySurfaceOptions.png *License*: unknown *Contributors*: AcidArrow
Image:KrayIndirectRays.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:KrayIndirectRays.png *License*: unknown *Contributors*: AcidArrow
Image:KrayIndirectRaysDesaturate.jpg *Source*: http://www.kraytracing.com/wiki/index.php?title=File:KrayIndirectRaysDesaturate.jpg *License*: unknown *Contributors*: AcidArrow
Image:KrayIndirectRaysMultiplier.jpg *Source*: http://www.kraytracing.com/wiki/index.php?title=File:KrayIndirectRaysMultiplier.jpg *License*: unknown *Contributors*: AcidArrow
Image:LightportalGUI.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:LightportalGUI.png *License*: unknown *Contributors*: Admin
Image:KrayPhotonMultiplier.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:KrayPhotonMultiplier.png *License*: unknown *Contributors*: Admin
Image:Photonmultiplier modifier.png *Source*: http://www.kraytracing.com/wiki/index.php?title=File:Photonmultiplier_modifier.png *License*: unknown *Contributors*: Admin
Image:No multiplier-unfiltered.jpg *Source*: http://www.kraytracing.com/wiki/index.php?title=File:No_multiplier-unfiltered.jpg *License*: unknown *Contributors*: Admin
Image:No multiplier.jpg *Source*: http://www.kraytracing.com/wiki/index.php?title=File:No_multiplier.jpg *License*: unknown *Contributors*: Admin
Image:With multiplier-unfiltered.jpg *Source*: http://www.kraytracing.com/wiki/index.php?title=File:With_multiplier-unfiltered.jpg *License*: unknown *Contributors*: Admin
Image:With multiplier.jpg *Source*: http://www.kraytracing.com/wiki/index.php?title=File:With_multiplier.jpg *License*: unknown *Contributors*: Admin