Image by StoneAge

YAF(A)RAY
user's guide

About this document:
Document revision 3.
Written by Alvaro Luna Bautista.

INDEX

1. What Yaf(a)Ray is

Yaf(a)Ray is Yet Another Free Raytracer; nobody knows for sure what the **(a)** stands for. It is a *raytracing* render engine based on a new source code, different than the used for the **YafRay 0.0.x** series. Rewriting the source code from scratch was necessary because the old design was exhausted and did not allow for more changes or additions. There are several conceptual differences to **YafRay 009**. As a consequence of these depth changes, renders will completely differ from **YafRay 009** ones. You will find additional information about **Yaf(a)ray** in the link below:

http://wiki.yafray.org/bin/view.pl/UserDoc/YafaRay

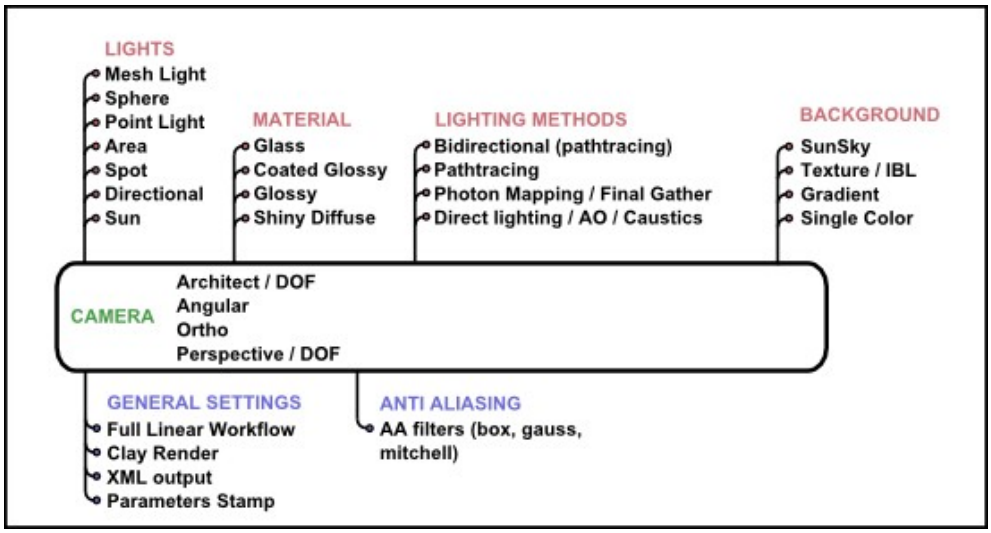
YafRay has been an open source project closely related to **Blender** from the start. For many years it was the only choice **Blender** users had apart from **Blender Internal**. In fact, **YafRay 009** and previous releases of **YafRay** used an exporter plug-in that was 'hacked' into the official **Blender** source code. This *plug-in* provided a good integration of **YafRay** in **Blender**, but on the hand it shaped a lot its development. There were some **YafRay** features that did never make the **Blender** panels because it meant more contamination of the Blender source code. On the other hand, many times **YafRay** development efforts were focused on supporting **Blender** new features instead of exploring independent *raytracing* routes.

Integrating **Yaf(a)Ray** into Blender like the old **YafRay 009** would mean a complete redesign of the exporter *plug-in* and adding more 'alien' code to Blender. Besides, at the end of the day **YafRay** and **Blender** are separated F.O.S.S. projects. Taking into account that there exist now other external render engines for **Blender** users apart from **Yaf(a)Ray**, that kind of integration does not make sense anymore. Ideally, a *render API* should exist in **Blender** for equal and seamless integration of any *external engine*.

Until that *render API* arrives, **Yaf(a)Ray** uses:

1. A python-coded **User Interface (UI)** to use in a **Blender** window to configure materials, lights and render settings and to launch the scene rendering.

2. A separate graphical **Render Output Window** where **Yaf(a)Ray** renders are shown and saved.



Overview of Yaf(a)Ray most important features

2. Yaf(a)Ray installation notes (Windows)

Yaf(a)Ray does not need any **Blender** special compilation to work, any official release from **blender.org** will work for instance**.** There are two components needed to run **Yaf(a)Ray**, which are:

1. **Yaf(a)Ray** binaries. Default installation proposed for **Yaf(a)Ray** binaries is **C:\Program Files\YafaRay** or the equivalent in your respective OS language.

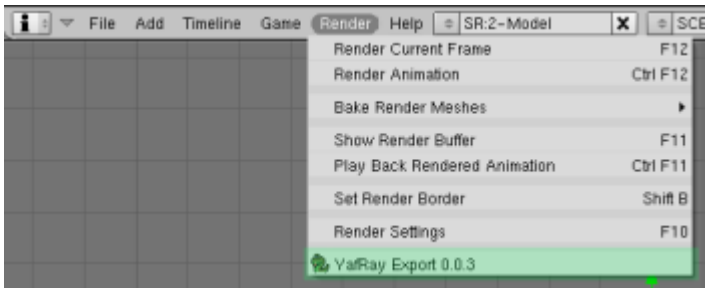
   In the same installation process, some Y**af(a)Ray** python scripts should be installed in the **Blender** scripts folder. The installer will look for it and tell you where it is found. Alternatively you can point the installer to the **Blender** executable folder, usually in **C:\Program Files\Blender Foundation\Blender** or the equivalent in your respective language.

2. A separate installer for the **Render Output Window** libraries. They must be installed in the Blender executable folder, usually in **C:\Program Files\Blender Foundation\Blender** or the equivalent in your respective OS language.

> **Note:** It is necessary to have Python 2.5 installed to run Yaf(a)Ray scripts. You can find it here: http://www.python.org/download/

3. Yaf(a)ray UI and general workflow

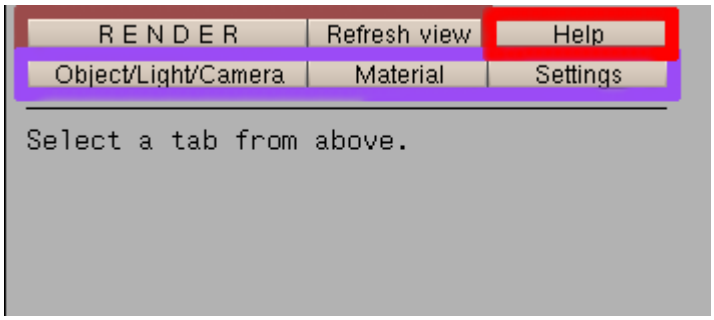
Once everything is *installed,* a special entry is added in the **Blender Render** menu, called **YafaRay Export 0.0.3**, which automatically executes the **Yaf(a)Ray** User Interface (**UI**) inside a **Blender 3DWindows**. Split your **3DWindows** and click in **YafaRay Export 0.0.3**



> **Note:** Here you can learn how to split the Blender 3DWindows:
>
> http://wiki.blender.org/index.php/Manual/PartI/Interface/Window_system

The **Yaf(a)Ray User Interface (UI)** will be shown, which is divided into:

1. Three main sections of settings which are **Object/Light/Camera**, **Material**, and **Settings.**
2. A **Help** text.
3. Two function buttons which are **Refresh view** and **Render**.



- **Settings**: This section is used to choose a lighting method and to configure render general parameters, render anti-aliasing and background settings.
- **Material**: The script takes **Blender** defined materials and applies them custom **Yaf(a)Ray** material properties.
- **Object/Light/Camera**: The script takes the current **Blender** *selected and active* object and applies it custom **Yaf(a)Ray** properties, depending on the object selected (camera, light or object). Lights and camera settings in **Yaf(a)Ray UI** script replace almost all **Blender** and **YafRay 0.0.9** settings, except from texturing mapping settings.
- **Help**: displays a brief help text.
- **Refresh View:** The **User Interface** must automatically refresh to show specific settings for the kind of object selected in the **Blender** *3DWindows*. Use the **Refresh** button if the **UI** is not self refreshed when a new object is selected. A right mouse button click on a empty area of the **User Interface** will refresh it too.
- **RENDER:** Launches the scene rendering and a separate **Render Output Window** in which the render progress can be seen. Use the **Render Output Window** settings to save your image. Once the render is finished or stopped, the **Render Output Window** must be closed before coming back to **Blender**. Don't forget to save your render.

4. Render Output Window

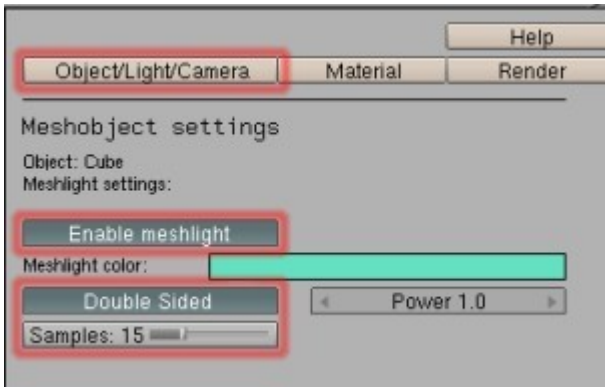
(to complete as the development of this part progresses, at the moment a few simple buttons)

5. Object, Lights, and Camera settings

5.1 Object

With this feature, objects in the scene can act as area light sources. Arealight is a light type that can produce soft shadows and its shape can be seen in reflective surfaces. The soft shadows need to be sampled several times and later interpolated to reduce noise. This type of light takes more time to render in contrast with point light types such as **spot** and **point**.

First of all, you must select the mesh you want it to act as a light source. Next you must click on the **Object/Light/Camera** button. Finally you must activate the **Enable Meshlight** button. To change the light color, just click on the rectangle next to 'Meshlight color:' to open a Color Picker.



- **Meshlight color** rectangle: opens a Color Picker.
- **Power**: intensity multiplier for meshlight color.
- **Double Sided**: considers both sides of the mesh as arealight sources.
- **Samples**: defines the amount of samples taken to simulate the soft shadows. The more samples, the less noisy the shadows but the longer it will take to render.

5.2 Lights

Lights settings are mostly controlled by the **Yaf(a)Ray UI**. In **Blender**, the only actions required are placing lights and choosing a Blender light type. Arealight size and spot beam parameters are controlled in **Blender** panels though. In **Yaf(a)Ray** , lighting power is controlled by a couple of buttons (**Power** and **Color**) available in the python **UI** for every light type, therefore **Blender Distance** and **Energy** buttons haven't any effect in **Yaf(a)Ray**.

Once a light is placed and its type defined in **Blender**, select it and press **Object/Light/Camera** main button. The python **UI** will automatically change to show specific settings for the type of light selected.

5.2.1 Point and Sphere

When you create o select a **Lamp** light in the **Blender 3DWindows**, the python **UI** will let you choose between two options to configure this kind of light, **Point** and **Sphere**.

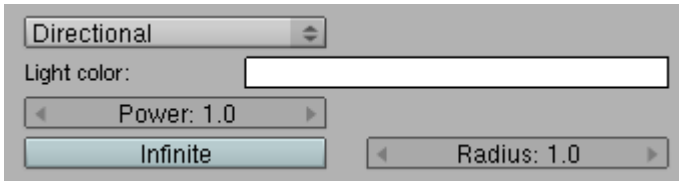
A **Point** light is a typical *omni directional* point light source as in **Blender** Internal with hard shadows, while a **Sphere** light is a spherical area light source which can produce soft shadows.

- **Light color** rectangle: opens a Color Picker.
- **Power:** intensity multiplier for light color.
- **Radius**: sets the radius of the **Sphere** light in Blender units.
- **Make light visible:** area light gets rendered visibly.
- **Samples**: defines the amount of samples taken to simulate soft shadows. The more samples, the less noisy the shadows but the longer it will take to render.

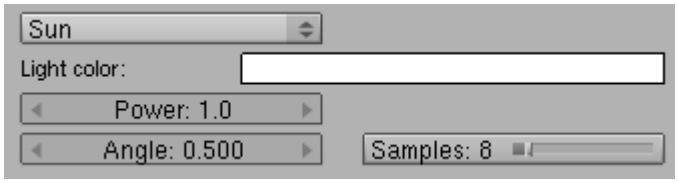
5.2.2 Directional and Sun

When you create o select a **Sun** light in the **Blender 3DWindows**, the **Yaf(a)Ray UI** will let you choose between two kind of lights to do this kind of job, **Directional** and **Sun**.

Directional light is a traditional **sun** light model which produces parallel rays and hard shadows.



The new **Yaf(a)Ray Sun** light is a more advanced concept and will help us to get blurred edges in shadows when the shadow itself gets further from the casting object, as in the real life. The angle button sets the visible area of the sun. Real Sun is visible in a cone angle of about 0,5º. A bigger angle mean a bigger sun, as well as softer shadows, which could be interesting for dawn or sunset scenes and for a sunlight filtered by an overcast sky.



- **Light color** rectangle: opens a Color Picker.
- **Power:** intensity multiplier for light color.
- **Infinite:** if enabled, area covered by the **directiona**l light is infinite. If disabled, light fills a semi-infinite cylinder.
- **Radius:** if **infinite** is disabled, radius of the semi-infinite cylinder for **directional** light.

- **Angle:** visible size of the s**un** light. Affects shadows.
- **Samples:** it defines the amount of samples taken to simulate the soft shadows, when **Direct Lighting** or **Pathtracing** is used. The more samples, the less noisy the shadows but the longer it will take to render.

5.2.3 Area

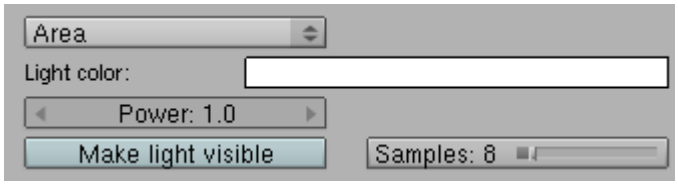
Arealight is a *area light* type that can produce soft shadows and its shape can be seen in surfaces with reflective properties. The soft shadows need to be sampled several times and later interpolated to reduce noise. This type of light takes more time to be computed in contrast with point light types such as **spot** and **point**.

In regard to **Arealight**, the most important change is the **Make light visible** button and the general workflow to get 'visible' arealights in reflective surfaces. In the old workflow, when photons were enabled, we needed to place 'emit' planes just behind every area light to make arealights 'visible' and to add lighting power to the scene.

Now, when **Make light visible** is enabled, a rectangle in the size of the area light is generated, so that the area light gets rendered visibly. Therefore, the 'emit' plane is no longer needed. More lighting power is added when **Make light visible** is on.

When **photons** is enabled, arealights cast direct light and photons. The **Make visible** option really only affects reflections in surfaces.

When **Pathtracing** is used, the **Make light visible** option also creates *caustics*, although there exist an option to not trace caustics with path tracing as they tend to be extremely noisy (**'none'** option in **Pathtracing** settings, see **5.1.1**).



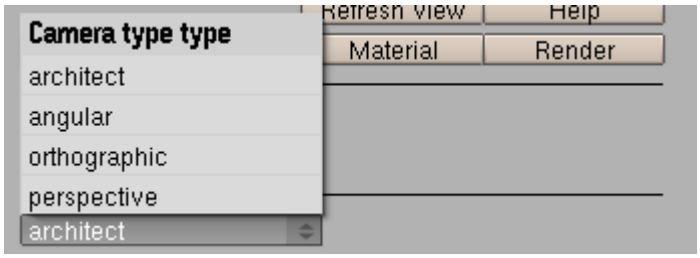
- **Light color**: opens a Color Picker.
- **Power:** intensity multiplier for arealight color.
- **Make light visible:** area light gets rendered visibly.
- **Samples:** defines the amount of samples taken to simulate the soft shadows, when **Direct Lighting** or **Pathtracing** is used. The more samples, the less noisy the shadows but the longer it will take to render.

5.2.4 Spot

Spot is a common point light with directional properties. Beam properties are defined in **Blender** panels (**SpotSi** and **SpotBl** sliders).

5.3 Camera settings

First a camera should be created or exist in your scene. **Lens angle** must be configured in the **Blender** camera panels. Then select the camera and press **Object/Light/Camera** main button. The python **UI** will automatically change to show specific camera settings. There are four camera types in **Yaf(a)Ray: Architect, Angular, Orthogonal** and **Perspective.**



5.3.1 Architect

This camera type works like a **Perspective** camera type, the only difference is that the vertical component of the perspective effect is neglected, so scene vertical lines are not convergent. **DOF** settings are available for this camera type; they are explained in the **5.3.4** paragraph.

5.3.2 Angular

A camera type useful to produce *light probe* images. *(to complete)*

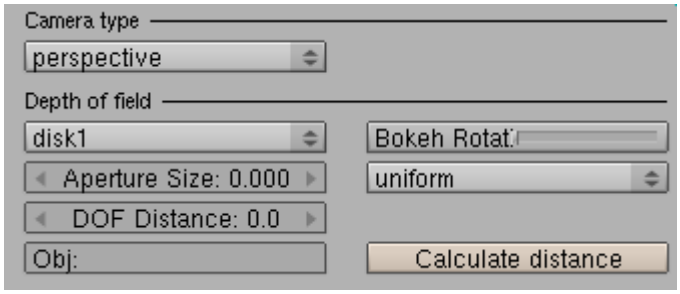
5.3.3 Orthogonal

A camera type that renders a orthographic (perpendicular) projection of the scene, without perspective effects.

- **Scale:** Specify the scale of the ortho camera, to control camera zoom.

5.3.4 Perspective

Perspective is the standard camera mode that simulates a lenses photographic camera, with perspective effects. All settings available for this camera type are used to enable and configure the *depth of field* (**DOF**) effect. **DOF** is the distance in front of and behind the subject which appears to be out of focus.



- **Bokeh type:** controls the shape of out of focus points when rendering with depth of field enabled (blur disk). This is mostly visible on very out of focus highlights in the image. There

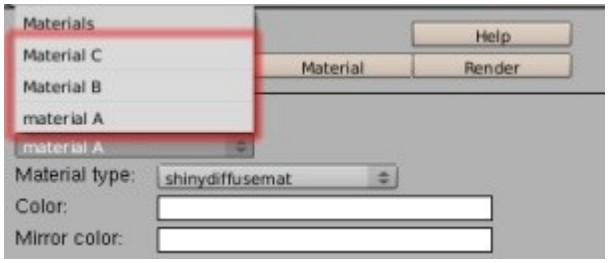
are currently seven types to choose from.

- **Bokeh rotation**: rotation of the blur disk.
- **Aperture**: The size of the aperture determines how blurred the out-of-focus objects will be. A rule of thumb is to keep it between 0.100 and 0.500 (0 disables **DOF**).
- **Bokeh bias:** controls the accentuation of the blur disk. Three types available, uniform, center or edge, with uniform the default.
- **DOF distance:** set the focal point in which objects will be in focus.
- **Obj:** Enter the name of the Blender object that should be the focal point.
- **Calculate distance:** it calculates the distance between the object entered in **Obj:** and the camera, and writes this value in the **DOF distance** button.

The **DOF** effect depends also on the render *anti aliasing* settings to get a nice blurred effect. First of all it is recommended to lower **'AA threshold'** a bit, but not set it to totally zero. Setting a high number of **'AA passes'** is also not really going to make all that much difference, the main smoothness factor that makes the most difference is really the amount of **'AA samples'**. A single pass with a high number of samples may be sufficient.

6. Material settings.

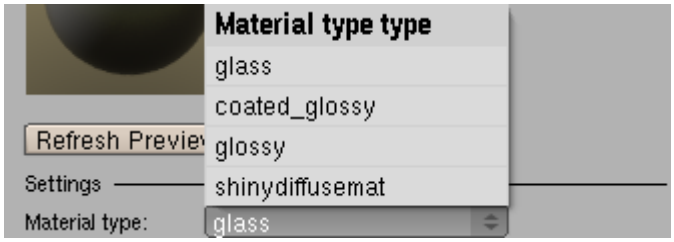
In contrast with the light settings, the **UI** script just takes Blender list of existing materials and applies them **Yaf(a)Ray** custom properties. For instance, in the scene below there is a list of three materials configured in Blender:



(insert paragraph here about mat preview)

Settings in the **Blender/YafRay** '**Material**' panel are replaced by **Yaf(a)Ray UI** settings. **Ramps** are not supported. **Blender** *Multimaterial* (more than one material in an object) is supported.

There are four material types in **Yaf(a)Ray**, with many possibilities for each of them to achieve advanced properties. They are **glass**, **coated_glossy**, **glossy** and **shynnydiffusemat.** This is a brief list of what **Yaf(a)Ray** materials can be useful for:



Glass: glass, water, fake glass.

Glossy: all kind of plastics, clean and polished metal, clean rough metal, car paint, finished wood, lacquered surfaces, painted surfaces, varnished wood, glaze and organic surfaces, materials with anisotropic reflections.

Coated_glossy: car paint, lacquered surfaces.

Shnydiffusemat: stone, rusted metal, concrete, fabric, paper, rough wood, curtains, emit surfaces, perfect mirror, materials with a basic transparency and *alpha mapping* with shadows, etc.

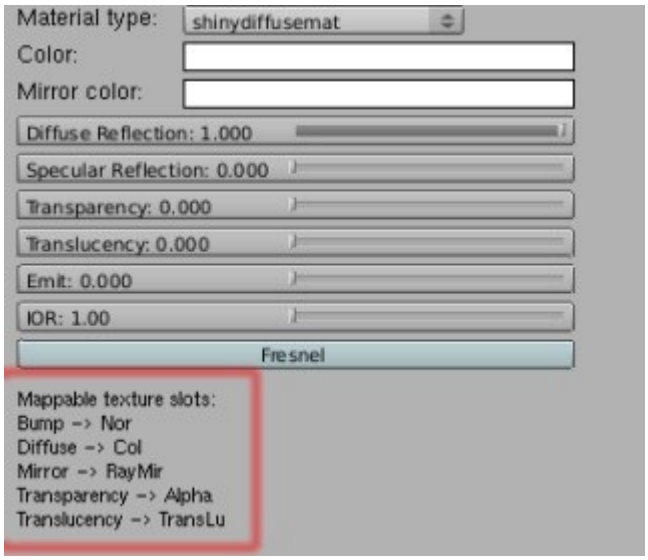
6.1 Mapping notes.

(this section needs a big overhaul and needs to be much more complete. I have prepared a lot of stuff to do it)

All the properties configured in the **Shaders** and **Mirror Transp Blender/YafRay 009** panels are replaced with the **Yaf(a)ray UI** settings. Only **Blender** mapping settings from **Texture**, **Map input** and **Map to** panels (F9) are taken into account.

(insert pic about Blender panels and buttons supported)

Map to texture *modulation modes* are partially supported. In the image below for instance, the **shinydiffuse** material will process only five Blender **Map to** modulation modes, which are **Nor**, **Col**, **Raymir**, **Alpha**, and **Translu**. In the first case for instance, the Blender **Nor** mode will affect bump mapping in Y**af(a)Ray.** *(to complete)*

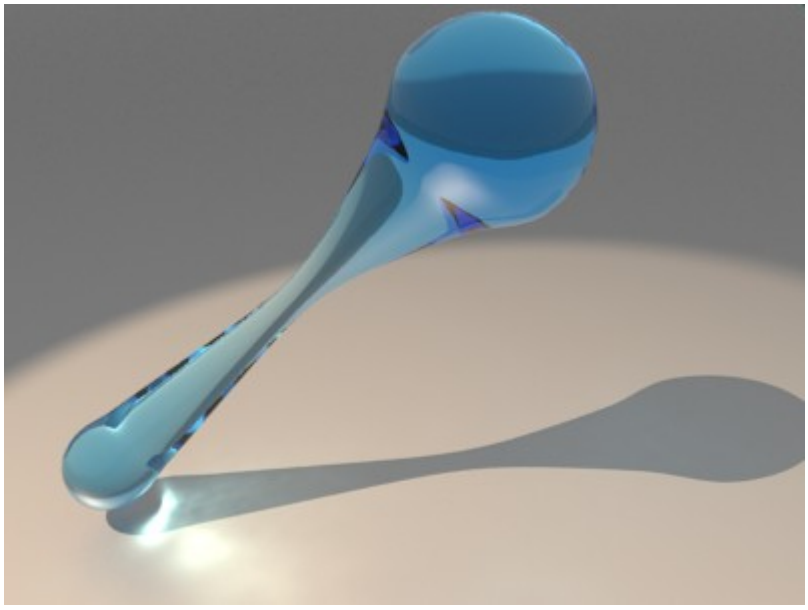


6.2 Glass

This kind of material can be useful to render realistic glass, water and any kind of transparent medium with *index of refraction*. Use this material to get refractive *caustics* and light reflection (mirror) depending on the incident angle.

This material can be also useful to get glass with fake transparent shadows, when caustic rays are not powerful or uniform enough to lit the scene behind a transparent object.

Absorption is the process by which light is absorbed by a medium, although not all light get absorbed, some is *reflected* or *refracted* instead. The more distance light had to get thought a medium, the more it gets absorbed. In a glass with different sections, the glass color will get darker if the section is bigger. By using **Absorption** we also get tinted caustics, depending on **Absorp. Color.**

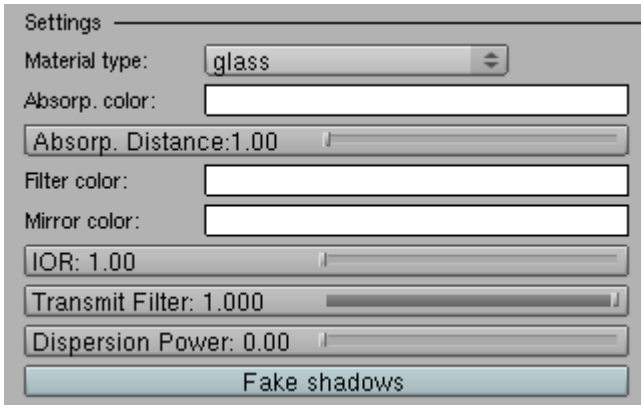


Example of the absorption effect with tinted refractive caustics.

Filter settings not only defines the color of the refracted light, but it also affects color of the transparent medium, like **Absorption color** does.

Use **Absorption** if you want to get the "absorption" effect (as the object section gets bigger, the object color gets darker). Use **Filter** if the absorption effect is not important to you and you want more control over the refracted color. **Filter color** also defines color of the transparent shadows when the **Fake Shadows** button is enabled.

Not all *light tracing* methods are optimized to render lighting effects after a refraction event. **Path tracing** (without *post-caching*) is not well suited for such a case. **Pathtracing** *caustics* tend to be very noisy and a very big amount of samples is needed to get a smooth result. To get transparent shadows in such a material you have got two options: working out a caustic photon map with enough visual consistency, or activate both the **Fake Shadows** button in the **Material** section and the **Transparent Shadows** button in the **Render** section to get fake glass shadows.



- **Absorp. color:** opens a Color Picker. It defines the color of of the non-absorbed light, therefore it sets the color of the glass. White disables absorption.
- **Absorp. Distance:** transmission of light through the material.
- **Filter color:** opens a Color Picker. It defines the color of the refracted light.
- **Mirror color:** opens a Color Picker. It defines the color of the reflected light.
- **IOR**: Light index of refraction, it produces either *refraction* or *reflection*, depending on the

incident angle.

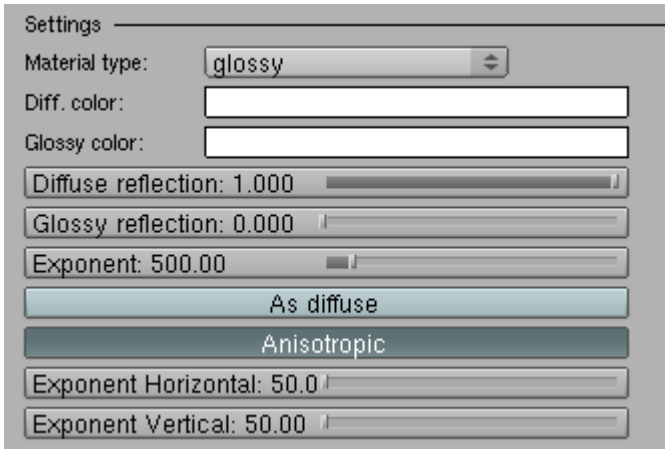
- **Transmit Filter:** strength of the refracted light when a **Filter color** is used.
- **Dispersion power:** strength of the *dispersion* effect, disabled when it is 0. When **Path tracing** is used, noise depends on path tracing **samples**, the more the samples the lesser the noise.
- **Fake Shadows:** When enabled, light rays that get through this object are traced without refraction index. Use **Filter color** and **Transmit Filter** to tint these rays. The **Transparent Shadow**s button in the **Render** section must be enabled too for this feature to work.

6.3 Glossy

A glossy reflection means that tiny random bumps on the surface of the material cause the reflection to be blurry. In fact there is a wide range of materials with such a reflection. **Yaf(a)ray** glossy material can be useful for all kind finished surfaces such as plastics, polished metal, car paint, finished wood, lacquered surfaces, painted surfaces, varnished wood, glaze and organic materials, etc. **Yaf(a)ray** glossy reflections have got by default *Fresnel effect*. Below there is an example of a glossy reflection.

Example of Glossy reflection

This material will be useful to get *anisotropic* reflections too. An *anisotropic* reflection means that reflection is not equal in all directions. This kind of reflection happens when a defect in a reflective surface repeats quite regularly and in the same direction. When **Anisotropic** is enabled, the **exponent** value is divided into a **vertical** and an **horizontal** component. If you want to get an anisotropic reflection, use a different value for each one. As a result the reflection will take an oval shape. The *anisotropic* effect can be reinforced by using a suitable bump map. When **Anisotropic** is enabled, two new buttons will show up, and the **exponent** will have no effect. **Horizontal** and **Vertical** components of the exponent depend on *UV mapping* coordinates.



- **Diff. color:** opens a Color Picker. It defines the color of the *diffuse reflection*.
- **Glossy color:** opens a Color Picker. It defines the color of the *glossy reflection*.
- **Diffuse reflection:** amount of diffuse reflection (**Diffuse color** brightness multiplier).
- **Glossy reflection:** amount of glossy reflection.
- **Exponent**: Blur of the glossy reflection; the higher the exponent the sharper the reflection. Use values between 0.5 and 200 for plastics and higher values for metallic surfaces.
- **As diffuse:** Treat glossy component as diffuse, faster when using *photon mapping*.
- **Anisotropic:** Enables anisotropic reflection (disables *isotropic* **Exponent**)
- **Exponent Horizontal:** exponent blur in the U direction (UV mapping coordinates)
- **Exponent Vertical:** exponent blur in the V direction (UV mapping coordinates)

6.4 Coated Glossy

Coated Glossy is basically a glossy material (see the previous paragraph) with some kind of reflective coating layer on top. **IOR** is the setting that controls reflectivity of the coating top layer.

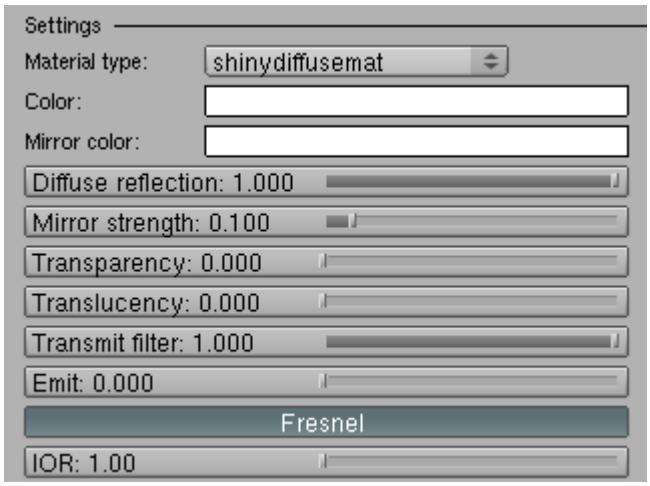
6.5 Shinydiffuse

Shinnydiffuse is a model of diffuse shader without 'viewer' properties (apart from Fresnel). It can be useful to get:

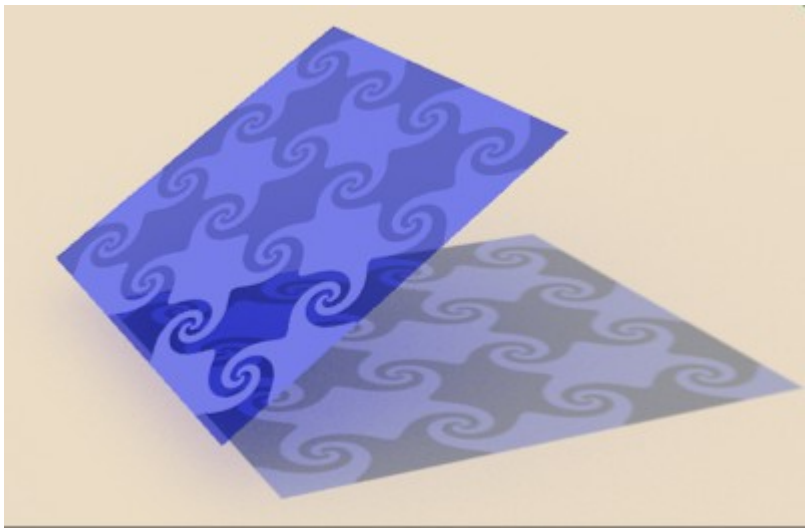
- *Diffuse* materials without any specular (glossy) component.
- Perfect mirror reflection without or with *Fresnel* effect (for *reflective caustics*).
- *Translucency* with *color filtering* and *alpha mapping* with *shadows calculation*.
- Transparency with *color filtering* and *alpha mapping* with *shadows calculation*.
- Diffuse **Emit** surfaces.

For instance, this material can be used for rough stone, rusted metal, concrete, fabric, paper, rough wood, chrome balls, car paint, curtains, billboards, etc**.** Shadows calculation from *alpha mapping* is supported.

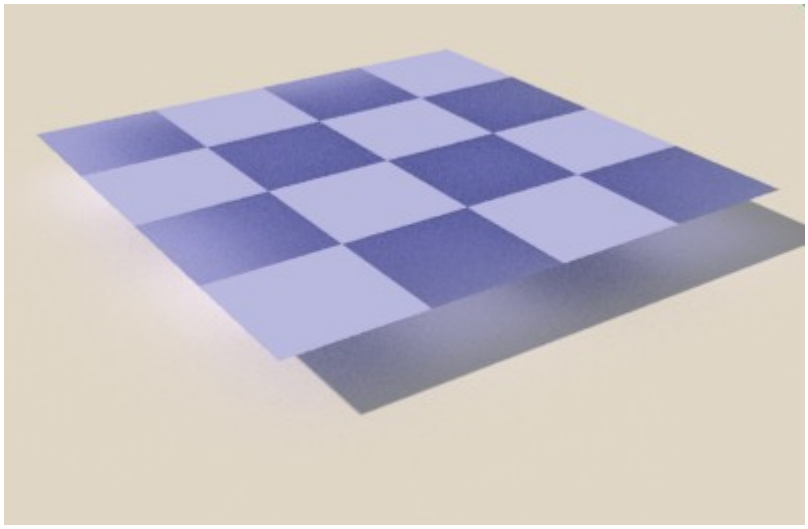
When **Mirror strength** is higher than 0, a new option appears to enable **Fresnel** reflection. If **Fresnel** is enabled, a new button appears to set an index of refraction (**IOR**) for the **Fresnel** reflection: the higher the index, the higher the reflection. Use this option to get *reflective caustics.*



- **Diffuse color:** opens a Color Picker. It defines the color of the diffuse reflection.
- **Mirror color:** opens a Color Picker. It defines the color of the mirror reflection.
- **Diffuse reflection:** amount of diffuse reflection (**Diffuse color** brightness multiplier).
- **Mirror strength:** amount of mirror reflection (enables **Fresnel**)
- **Transparency**: Basic transparency effect without refraction.
- **Translucency:** *Diffuse* transmission of light in the shadowed part of the object.
- **Transmit filter:** amount of tinting of light for **Transparency** and **Translucency**.
- **Emit:** Amount of diffuse light the material emits. If **Direct Lighting** is used, only surface properties can be seen. To use it as a lighting source you will have to use **path tracing**. It does not emit photons.
- **Fresnel:** Enables *Fresnel* Reflection.
- **IOR: I**ndex **o**f **R**efraction for the Fresnel effect, produces *reflective caustics*.



Example of a mapped transparency with color filtering and shadows calculation from alpha mapping.

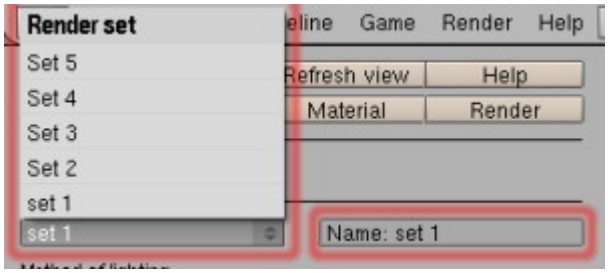


Example of a mapped translucency with color filtering and shadows calculation from alpha mapping.

Example of reflective caustics using shinydiffuse Fresnel reflection.

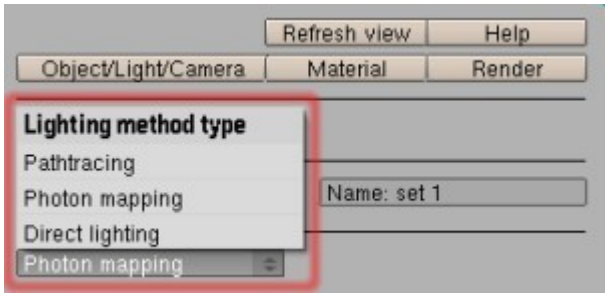
7. Render settings

Render Set can be used to configure as many as five different set of render settings. You can use this feature to compare rendering *performance* between different sets. Render sets can be renamed by using the **Name:** text button.



7.1 Lighting Methods

There are currently four lighting methods available in **Yaf(a)Ray**. **Bidirectional, Pathtracing** and **Photon Mapping** perform *global illumination* (direct light + indirect light), while **Direct lighting** only takes into account direct light from emitting sources without *indirect light* contribution. Caustic *photon mapping* and Ambient Occlusion can be rendered when **Direct lighting** is used.



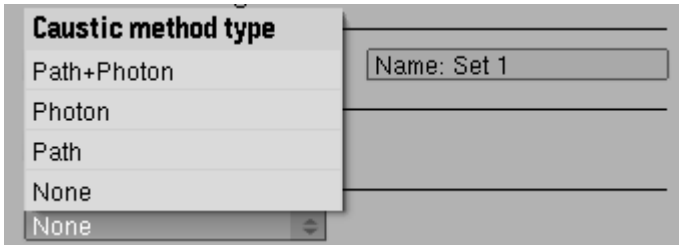
7.1.1 Bidirectional

*(to complete. In beta stage. Use a high number of **AA passes** and **AA inc. samples** and **AA thresold=0** to reduce noises in every pass)*

7.1.2 Path Tracing

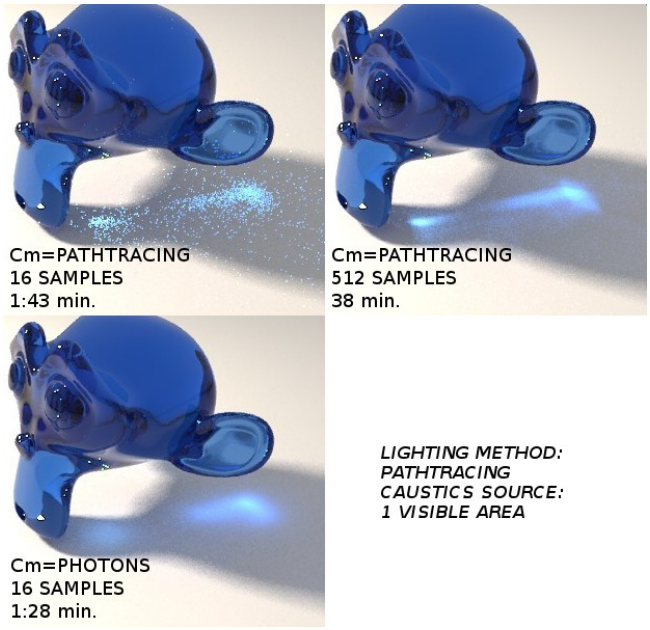
Path tracing is a relatively old **GI** *biased method* in which each ray is recursively traced from the camera along a path until it reaches a light source. When a light source is found, the light contribution along the path is calculated back to the camera taking into account surface properties. Many samples are need to be taken and interpolated for each camera pixel to get a smooth result. A light source can be either a light, the scene background or both. Scenes with relatively small light sources and with a high contrast between light sources and its surrounding areas will need more samples to remove noise. Therefore **pathtracing** is a **GI** solution more suited for outdoor scenes and for daylight indoor scenes with big windows and a regular distribution of light.

Path tracing (without post-caching) is not well suited for caustic effects. Pathtracing caustics tend to be very noisy and a very big amount of samples is needed to get a smooth result. In **Yaf(a)Ray** we have alternative methods to render the caustics component caustics when **pathtracing** is used:



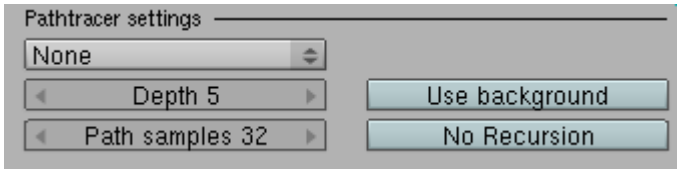
- **Path+Photon:** a mix of a caustic photon map and path tracing caustic rays are used to get caustics.
- **Photon:** a fast photon map is used to render caustics. Path tracing caustics rays are not rendered.
- **Path:** Path tracing caustics rays are rendered.
- **None:** the caustic component is not rendered.

This is an example about how methods for the caustics component work in **pathtracing**. In the first image (upper left), **Path** is used to get *caustics*, which are very noisy when a low number of samples is used (16). In the second render, 512 pathtracing samples are used to improve *caustics* but it takes much more time to render (38 minutes). In the third example, photons are used to produce the *caustics* component and the render time is the lowest of them all (**Cm** stands for **Caustic method**):



Comparison between methods used for the caustic component in pathtracing

The other components of the *global illumination* model are rendered as usual. *Area light* types (sphere and area) with the **Make Light visible** option enabled are needed to produce caustics in **pathtracing**. More information about the *caustic* photon map settings can be found in the **5.1.3 Direct Lighting** paragraph (they are the same). The other **pathtracing** settings are:



- **Depth:** defines the number of rays bounces in order to find the light sources, the more bounces the better quality and the less noise but the longer it will take to render.
- **Samples:** This setting is the equivalent to **GI Quality** in **YafRay 0.0.9**, and it defines the number of samples to take per camera pixel, the more samples the better render quality and the less noise, but the longer it will take to render as well. A **Depth** of 3 to 5 and 32-256 **samples** should do okay for almost every scene. For this and for other 'Samples' settings, it is a good practice to increase & decrease your samples in base 2 steps (2-4-8-16-32-64... etc)
- **Use background:** Makes use of the background color or texture as a light source.
- **No recursion:** only lighting is calculated without reflective or refractive events.

7.1.3 Photon Mapping and Final Gather

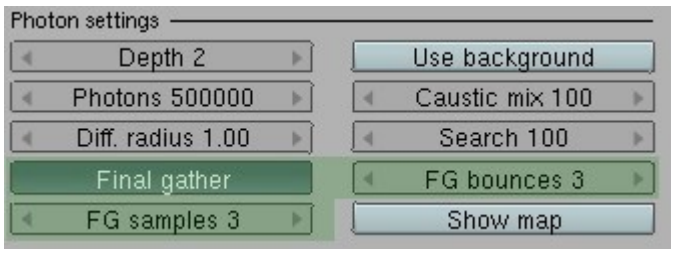
Yaf(a)ray settings on photons are more or less the same than in other raytracers. Photon mapping is a *Global Illumination* algorithm that calculates the flow of light energy throughout a scene. With photon mapping, light packets called photons are sent out into the scene from the light sources. Whenever 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.

Once the actual rendering has started, the correct appearance of a surface in a given point is estimated by examining and interpolating photons in the neighborhood. This is in fact the slowest part of the algorithm. **Diff. Radius** defines the radius of this search, while **Search** defines the maximum number of photons to mix. Photon mapping is a technique meant to be used in closed or almost closed environments. HDRI backgrounds can't emit photons.

- **Depth:** maximum number for *reflections* (bounces) and *refractions*.
- **Photons:** Number of photons to trace. the more photons, the more information to generate the photon map from.
- **Diff. Radius:** Radius to search for non-caustic photons.
- **Caustic Mix:** maximum number of caustic photons to mix (blur).
- **Search:** maximum number of non-caustic photons to mix.
- **Use background:** *(to complete)*

Final gather is a caching technique to improve and 'complete' photon mapping by gathering, after photon tracing, an approximation of the local *irradiance* using several illumination bounces. This information is used at render time for further interpolation with the obvious advantage of requiring a less accurate, therefore faster, **GI** simulation and still have a physically correct simulation.

To reduce noise in renders when using **FG** it is recommended to increase **FG samples** and to use relatively high *anti aliasing* settings.



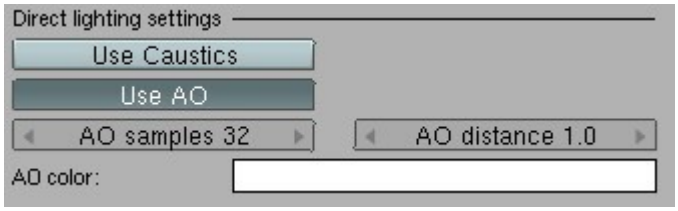
- **FG bounces:** in a precomputed phase, determines the number of bounces for Final Gather rays.
- **FG samples:** Final Gathering samples for interpolation, the more the better, but the longer it will take to render.
- **Show map:** *(to complete)*

7.1.4 Direct Lighting and Ambient Occlusion.

When **Direct Lighting** is used, only lighting provided from light sources is considered, without indirect contribution from other surfaces (light bounce). Since *caustics* photon mapping does not need post caching techniques like **Final Gathering** for blurring/interpolating photons, an option is provided to produce *caustic* photon maps when **Direct lighting** is used. Lights will be the *caustic* photons sources.

- **Photons:** Number of caustic photons to shoot.
- **Caustic Depth:** Number of refraction events for caustic photons
- **Caustic Mix:** number of photons to mix (blur)
- **Caustic Radius:** amount of caustic blurring. The result also depends on **Caustic Mix**.

Ambient Occlusion is a shading method that takes into account attenuation of light due to occlusion. Ambient occlusion is most often calculated by casting rays in every direction from the surface. Rays which reach the background or “sky” increase the brightness of the surface, whereas a ray which hits any other object contributes no illumination. As a result, points surrounded by a large amount of geometry are rendered dark, whereas points with little geometry on the visible hemisphere appear light.



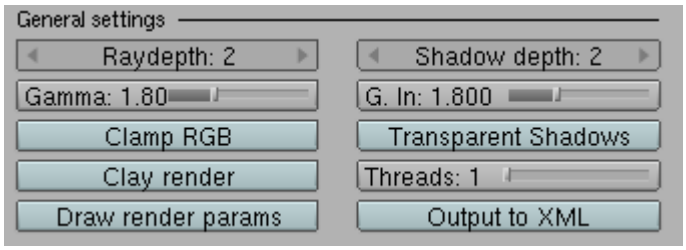
- **AO Samples:** The number of rays used to detect if an object is occluded. Higher numbers of samples give smoother and more accurate results, at the expense of slower render times
- **AO Distance:** The length of the occlusion rays. The longer this distance, the greater impact that far away geometry will have on the occlusion effect. A high Dist value also means that the renderer has to search a greater area for geometry that occludes, so render time can be

optimized by making this distance as short as possible, for the visual effect that you want.

- **AO Color:** Color for ambient occlusion rays, also useful to test lighting AO power.

7.2 General Settings.

Under this section are grouped several general *raytracing* settings:

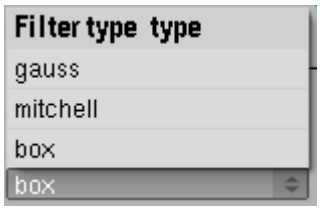


- **Raydepth:** Maximum depth (bounces) for *recursive raytracing*. Increase this setting to get deeper into successive reflective and refractive events in the scene.
- **Shadow depth:** Number of bounces for shadow rays when glass **fake shadows** is enabled. A sort of 'fake' raydepth.
- **Gamma:** *Gamma correction* in renders, to use in the *linear work flow*. It changes the response to light of the render engine. Inverse correction of textures and colors is performed using the **G. in** setting. **Gamma** should be used accordingly with your monitor gamma. To take full advantage of this feature, a basic monitor calibration is recommended. A display gamma of **2.2** is the standard for the Windows OS. The standard for Macintosh and Linux is **1.8**. Use Gamma=1 to deactivate render gamma correction. More info about this technique: http://www.gijsdezwart.nl/tutorials.php
- **G. In:** Inverse Gamma correction applied to textures and color input. It should be the same value as **Gamma**.
- **Transparent Shadows:** Enable to get **fake shadows** in a **glass** material.
- **Clamp RGB:** Reduces the colors' brightness to a *low dynamic range*, for better *anti aliasing* on areas with fast hight contrast changes. The examples below were made by **sevontheweb.** The upper image has got aliasing issues in areas with strong contrast variations, but colors are crisp. In the lower image **Clamp RGB** has been enabled. There is better *anti aliasing* but colors are duller. Another way of solving this issue is increasing the render resolution and then scaling back to the desired resolution using a good interpolation algorithm.

Comparison between Clamp RBG enabled (top) and disabled (bottom)

- **Threads:** to fork the rendering calculation into two or more simultaneously running tasks, for multiprocessing purposes.
- **Clay Render:** produces a clay render overriding all materials.
- **Output to XML:** The scene is written in a yaf(a)ray XML file. The file is located in the **YFexport** directory as per Blender settings.
- **Draw Render Params:** The render most important parameters are written in a badge in the render image. Use this feature to compare renders and to ask for advice in the YafRay forums.
- **Custom String:** Adds text to the previous render parameters.

7.3 Anti Aliasing settings.

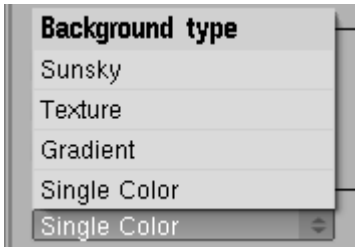


Yaf(a)ray now uses two more *anti aliasing* filters apart from default **Box**, which are **Gauss** and **Mitchell**. The only really good property of **Box** is that it has no danger of amplifying sampling noise. If a slight softness of edges is no problem for your work, try **Gauss**; if you want maximum detail try **Mitchell**.

Pixelwidth (which gets translated to real filter size) has some influence too, the larger the softer the image. Use lower values than default to make your renders sharper. The way **AA** samples are configured has got changed too. **AA samples** defines the number of samples for only the first **AA pass**, while the **AA inc. samples** value is used for subsequent **AA passes**.

7.4 Background Settings.

There are four options for backgrounds, which are:



- **Sunsky:** This kind of background tries to reproduce a realistic sky with its color variations. There is a tutorial about how to set up this kind of background, made by **sandstorm**: http://www.yafray.org/forum/viewtopic.php?t=1349
- **Texture:** A texture is used as a background. You can also use it as a light source (**Use IBL** on/off button). The main purpose of this option is using ***HDRI*** *images* to lit the scene. Besides, there is an option to rotate the background (**Rotation:** button), a multiplier for background color (**Power:** button) and a sampling parameter for the background when used as a light source (**IBL samples:** button)

  The textures need to be loaded using **Blender Texture buttons** (F6). **Yaf(a)ray** uses the **Blender Word** mapping settings as well. In the **texture and input** panel, only the **AngMap**, **Sphere** and **Tube** modes work. In the adjacent **Map to** panel, **Hori** must be enabled for light probes to work.

Settings used in Blender Panels for HDRI backgrounds in Yaf(a)ray

- **Gradient:** A two-parts gradient (Sky and Ground) is used as a background, with the division between them in the camera horizon line. **Power:** button is a multiplier for background color.
- **Single color:** A single color is used as a background. **Power:** button is a multiplier for background color.