.: Feeblemind :. Weblog

« Using texture and material stencils - Your first simple animation, step by step »

By olivS on Saturday, July 10 2010, 19:45 - Blender tutorials

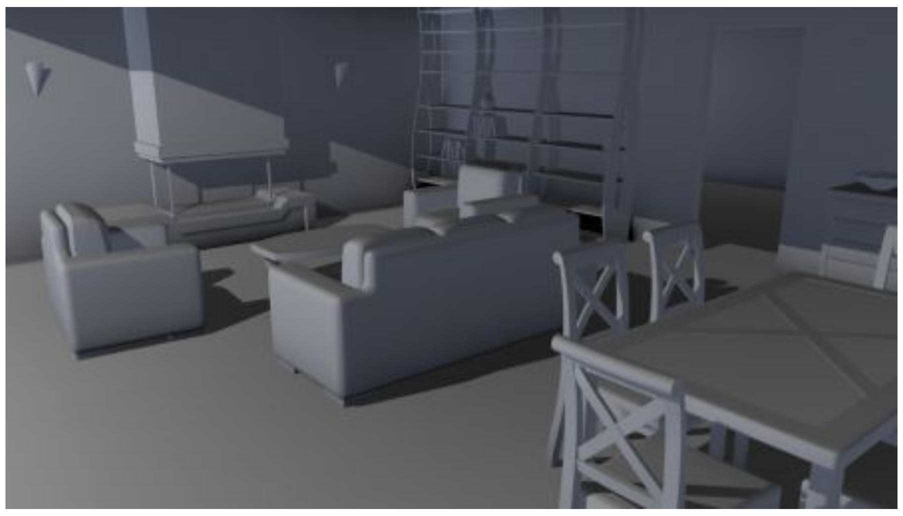
Tags :

- 3D,
- ambient,
- blender,
- indirect,
- light,
- occlusion,
- tutorial

7

Ambient Occlusion and Indirect Lighting

Global illumination is a very popular method for lighting three dimensional scenes. Unfortunately, most of the methods are very slow and requires lots of ressources. Ambient Occlusion is a method for simulating global illumination, while keeping a respectable ratio between results and computing times.



For Blender: 2.50 Alpha 2 and better

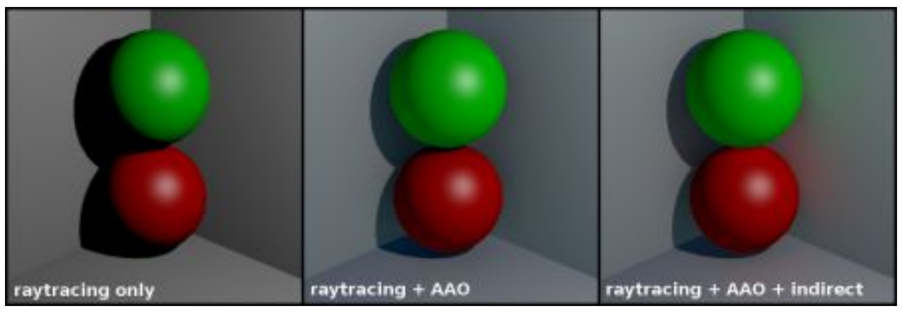
1. Different kinds of illumination

The key of photorealistic renders rely on simulating convincingly the lighting of a given scene. This tutorial should be very helpful to people trying to achieve photorealistic results. An humble attempt to such an usage is showcased there:

Please click to view at full size

- **The case of Raytracing**
  The renderer can only figure out if a pixel is lit or shadowed. If it is shadowed, the pixel will be rendered pitch black ; if it is lit, the color of the object will be displayed according to any self-shadowing and to the shader of the surface.
  In case there are multiple light sources, there are higher chances that a shadowed area will be at least partially lit by a secondary light source, resulting in light shadows instead of pitch black shadows.
  This is the primary way of illumination for Blender ; it does the trick, but it is not strikingly realist. (*left image, below*).
- **And in real life?**
  Careful observation of our environment will show that pitch black shadows are quite rare ; this is because real light rays are not stopped on the surface of the first object met.
  For each ray of light, when it would hit a surface, the renderer would have to calculate the bouncing direction and the new energy available after it has hit the obstacle (lighting the surface costs energy) ; then the ray hits another surface, spends part of its energy to light it, and so on, surface after surface, until depletion of the energy of each single light ray. It may last forever, but this is actually why light seems to "fill" the entire volume of a given room, lighting even the areas that are not directly lit by the original light source. Global illumination is just that: inter-reflections of light from an object to another.
  - **Ambient Occlusion**:
    Instead of letting a ray of light bouncing forever (or until an arbitrary energy threshold is reached) which will drive to never ending or very long renders, Ambient Occlusion is a specific algorithm that tries to simplify and simulate "how much" a given pixel is occluded by a close surface, by "sampling" its environment. The more close are surfaces to the given pixel, the more it will be shadowed.
    Blender uses two kinds of algorithms: the first one actually shoots extra rays from the given pixel in search for nearby obstacles (Ambient Occlusion, or AO), the second one will identify occluded areas in the virtual space based on a radius around each vertex of any objects (Approximate Ambient Occlusion, or AAO). The AO method produces noisy but realistic results, while the second is cleaner but, being more approximative, quicker to render (*middle image, below*).
  - **Indirect Lighting**:
    Each time a raylight hits a surface, it looses part of its energy in order to light it, but its color is also effected by the reflected

surface color. Thus, a white raylight hitting a red surface will turn slightly pink-red after the bounce. If it reaches a white nearby surface, the surface will not show as pure white, but slightly redish instead.
As a consequence, careful observation will show that surfaces are very faintly colored by light interactions from nearby objects ; this subtile color bleeding is caused by Indirect Lighting. The AAO algorithm of Blender can simulate color bleeding through actual Indirect Lighting (*right image, below*).



There are other ways to fake a global illumination, like Radiosity (http://en.wikipedia.org/wiki/Radiosity_%283D_computer_graphics%29) , path tracing (http://en.wikipedia.org/wiki/Path_tracing) , Metropolis light transport (http://en.wikipedia.org/wiki/Metropolis_light_transport) , photon mapping (http://en.wikipedia.org/wiki/Photon_mapping) , and image based lighting (http://en.wikipedia.org/wiki/Image_based_lighting) . The following tutorial is all about Ambient Occlusion, Approximate Ambient Occlusion and Indirect Lighting. How to use them in order to get quite realistic renders for your scenes.

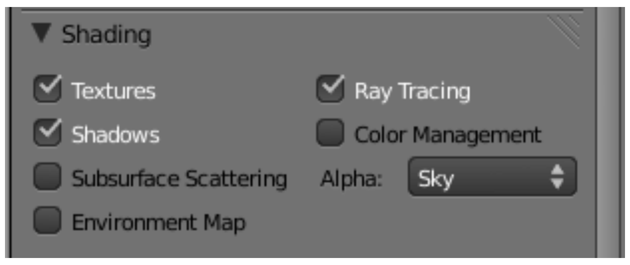
What about other GI options in Blender?

Radiosity was supported until Blender 2.49 but is now deprecated and has been removed from Blender 2.50 Alpha 2. It is replaced by Indirect Lighting which is more flexible and easier to use in production. IBL is supported through the combination of raytraced AO and an HDR image loaded in the World menu as World texture.

2. Preliminaries

2.1 Enabling raytracing

As well as anything related to raytracing with Blender, because of the supplementary calculus time required by your computer, raytracing is only an option that you should feel free to activate and deactivate. This is done by the mean of the ***Render*** *menu*. The **Shading** panel shows two options labeled **Ray Tracing** and **Shadows** you will have to activate in order to use ambient occlusions in your pictures.

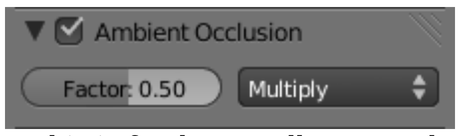


✓ Textures ✓ Ray Tracing
✓ Shadows ☐ Color Management
☐ Subsurface Scattering Alpha: Sky
☐ Environment Map

2.2 Global illumination modes

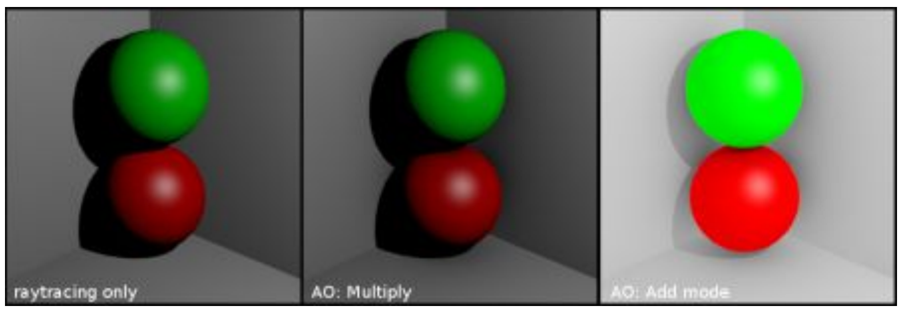
Three panels in the ***World*** *menu* controls the lighting effects. *Ambient Occlusion*, *Environment Lighting* and *Indirect Lighting* could be used independently of each other, or alltogether.

2.2.1 Ambient Occlusion

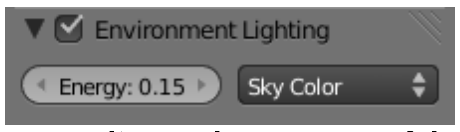

✓ Ambient Occlusion
Factor: 0.50 Multiply

Because it simulates the occlusion from objects to others, this is fundamentally a non-physical effect. The occlusion is stored in a specific pass, and its intensity is controlled by the **Factor** parameter. There are two **Blend Modes**:

- **Multiply**: this is the defaut option, it multiplies the occlusion pass with direct lighting (would produce richer exposed surfaces)
- **Add**: it adds the occlusion pass to direct lighting (would enhance shadowed areas)

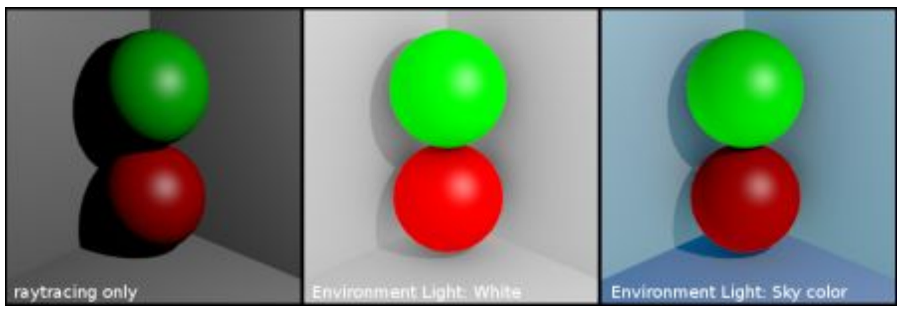


2.2.2 Environment lighting

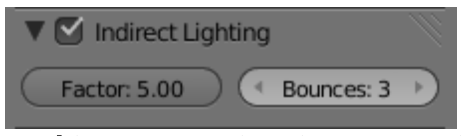

✓ Environment Lighting
Energy: 0.15 Sky Color

Environment lighting tends to light globally your scenes, according to the exposure of the pixel to any virtual bouncing rays of light. By default, the behavior is of "Add" type, but there is no control for it ; instead, you can use negative **Energy** values. By default, the **Environment Color** of light is **White**. But there are two variants available:

- **Sky Color**: takes samples of color from the **World** and taints the scene accordingly
- **Sky Texture**: takes samples of color from the texture assigned to the **World** and taints the scene accordingly

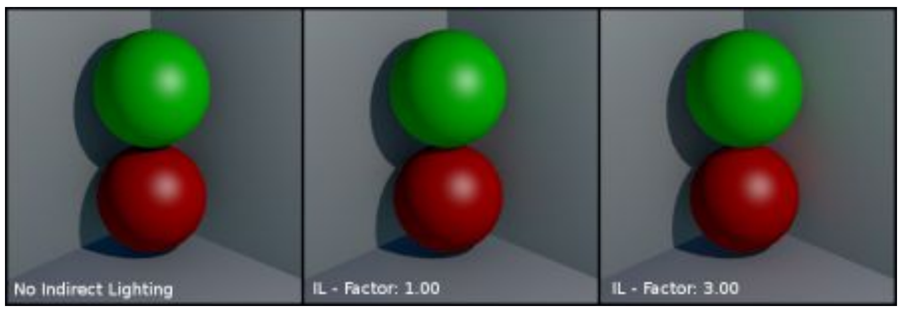


2.2.3 Indirect lighting



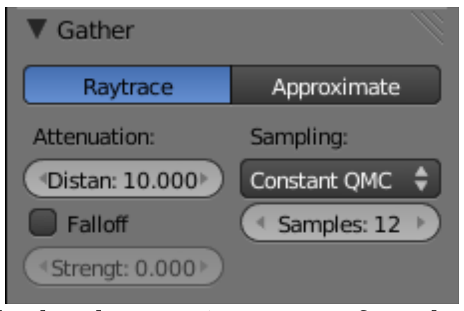
It is only supported by AAO **Gather** type at the moment and is an approximative way to simulate color bleeding. Once again, there is no *Blend Mode* control existing, and it is assumed that *Indirect Lighting* is always "Add".

- **Factor**: specify how much *Indirect Lighting* is added ; a value of 1.0 is physically more correct (*middle image below*), but you can increase or decrease this value for artistic purposes, or for highlighting the color bleeding if it is too subtile (*right image, below*).
- **Bounces**: sets the number of bounces allowed for indirect lighting to occur. Higher values than the default 1 will takes more time to compute, but as color bleeding will occur between many colored objects, complex and rich shading will be produced by higher values, at the price of longer computation times.



2. Gather mode

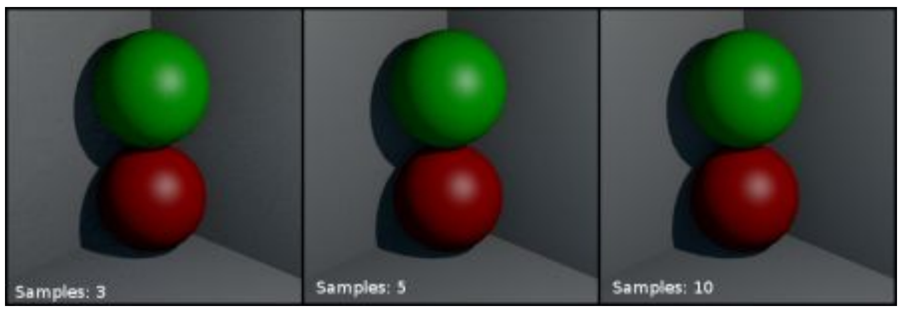
2.1 Raytrace (for AO)



For each pixel, an array of rays is cast is order to sample the close environment. If an obstacle is reached, then the pixel is considered as occluded. If no obstacle is met, then it is considered as lit. The actual light intensity seen by the pixel is the mean of the occluded/lit information returned by all the rays. The more rays return "occluded" information, the more dark the pixel will appear. Unfortunately, if the number of samples cast from the pixel is not high enough, two neighbouring pixels may provide slightly different occlusion information. By nature, the visual result over an entire image is a noisy result, that could only be overcome by increasing the number of sample rays. Of course, the more samples needed, the longer the render takes.

- **Sampling method**: there you can set the method for generating the shadow samples. QMC stands for Quasi Monte-Carlo (http://en.wikipedia.org/wiki/Monte_Carlo_method) , a statistic method to produce random sampling with low variance, e.g., even distribution of the samples. **Constant Jittered** suggest a distribution against a fixed grid. With an equal number of samples, any QMC method will deliver images less noisy than **Constant Jittered**.
  - **Constant QMC**: the samples are very evenly distributed, resulting in clean sample patterns
  - **Adaptive QMC**: the number of samples can be adjusted, with more samples distributed where required - generally speaking the best option
  - **Constant Jittered**: the samples are distributed according to a fix grid

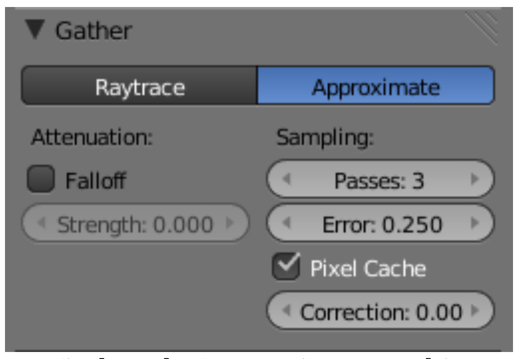
  The **Adaptive QMC** is generally the best option, but some scenes cannot benefit from such a distribution ; the **Constant QMC** then becomes the option of choice. Both QMC methods offer the advantage of being more time efficient compared to the **Constant Jittered**, thanks to their random distribution approach that helps the renderer to produce good quality images with fewer samples. And fewer samples mean shorter rendering time.
- **Samples**: Amount of ray samples. Higher values give smoother results and longer rendering times
- **Attenuation**: the **Distance** parameters sets the length of rays cast, e.g., how far away faces continue to contribute to the occlusion effect of a given pixel



Banding effects:

Because of its very nature, shading using the **Raytraced** AO method tends toward unveiling faces of your objects, even if these faces are theoretically smoothed using the **Smooth** shading mode (see the T key **Object Tools** panel). If you witness this phenomenon, you'd probably want to decrease the value of the Bias parameter.

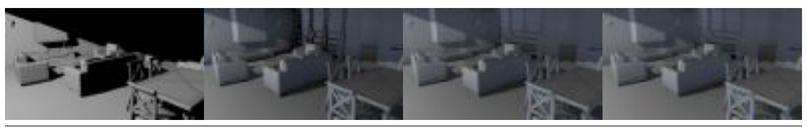
2.2 Approximate (for AAO)



As explained in this article (http://www.bigbuckbunny.org/index.php/approximate-ambient-occlusion/) , this method has been developped specifically to be rendered fast and noise free. As a matter of simplification, each vertex in the scene is considered as a disk. And the more there are disks occluding a point, then the more this point will be considered shadowed. As there may be a lot of disks one behind the other, over-occlusion could unfortunately occur, but running multiple passes would help to get rid of this unwanted result.

> **Important note:**
>
> Because an occluding disk is set at each vertex of the objects, AAO would be effective only with meshes dense enough in respect of the dimensions of the scene. For quick and dirty work, you can set a **Subdivision Surface** modifier, set to **Simple** subdivision algorithm. But this will add unnecessary details on some locations, and impair the rendering performances ; for better results, it is most probably better to smartly use the **Loop Cut** feature (`Ctrl+R`) in order to add details where strictly needed.

- **Passes**: number of preprocessing passes to reduce over-occlusion ; shadow depth is more reliable with at least one preprocessing pass. Higher values may help but will lead to increased render times
- **Error**: low values are slower and higher quality
- **Pixel Cache**: Cache AO values in pixels and interpolate over neighbouring pixels for speedup
- **Correction**: Ad-hoc correction for over-occlusion due to the approximation


Over-occlusion is a very common phenomenon for complexe scenes. From top left to bottom right: raytracing only, AAO with 1 pass, 2 passes and 4 passes

---

Article written on January the 16th, 2005.
Updated on July the 10th, 2010 for Blender 2.50 Alpha 2. Comments re-initialized.

Comments

1 On Sunday, July 18 2010, 17:14 by calambre
Thanks for the update!

2 On Wednesday, July 21 2010, 18:50 by francisco
Thanks for the explanation.
Bye.

3 On Monday, January 31 2011, 16:55 by carrozza
Thanks, I learned a lot from this article.

4 On Monday, September 12 2011, 12:08 by BAka (http://www.lukaszbaczynski.com)
Searched for a good description like that ! keep up the good work.

5 On Wednesday, January 25 2012, 15:47 by oscar
very good explanation, thanks from colombia

6 On Tuesday, May 15 2012, 21:24 by gg
awesome article! thanks a lot.

7 On Thursday, September 13 2012, 09:45 by SZ
Finally I got it!!! Thank you very much. Really comprehensive and easy to understand.

« Freedom » - Powered by Dotclear