Ambient Occlusion

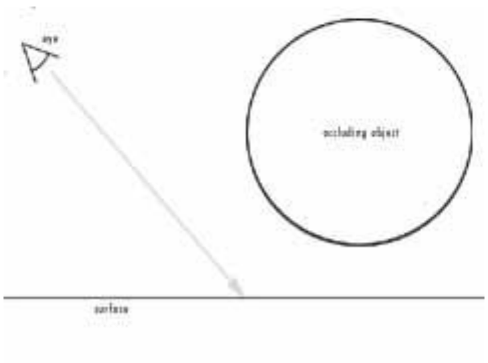
it's better than a kick to the head!

a presentation by Bob Moyer

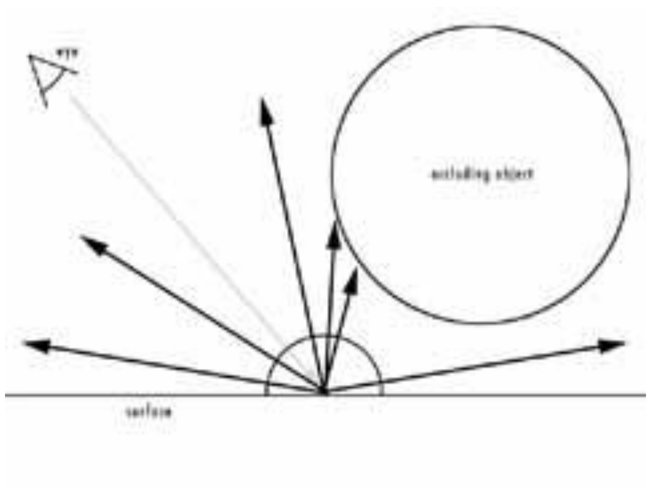
Since standard illumination models do not take into account diffuse reflection (light bouncing off of matte surfaces onto other surfaces), computer lighting tends to be darker than real-world lighting. It becomes necessary to pump up the illumination, and the easiest way to do this is an ambient term in the lighting calculations. The ambient term adds a set value and color uniformly to the surface, ensuring no point on the object is black. The concept of ambient light has been with computer graphics since virtually the beginning, but has largely fallen into disuse. Adding a uniform value tends to make areas look flat, and doesn't take into account cracks and contact shadows. Consequently, studios such as Pixar tend to replace ambient light with a LOT more diffuse lights...which takes up lighting TD hours and render hours.

At SIGGRAPH 2002, technical directors from Industrial Light and Magic presented a technique they call ambient occlusion. Simple, but highly effective, ambient occlusion's been used for years at studios like BlueSky and ILM to achieve grounded realism, without the cost of full global illumination. Fancy name aside, ambient occlusion is simply a ratio of how much ambient light a surface point would be likely to receive. It simulates a huge dome light surrounding the entire scene. If a a surface point is under a foot or table, it will end up much darker than the top of someone's head or the tabletop. This can then be multiplied by various other surface attributes to achieve a subtle, but very powerful, lighting effect.

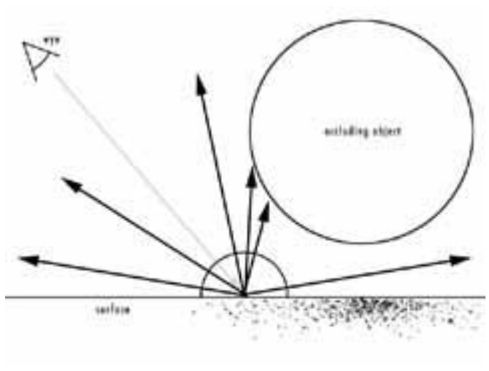
How Does it Work?



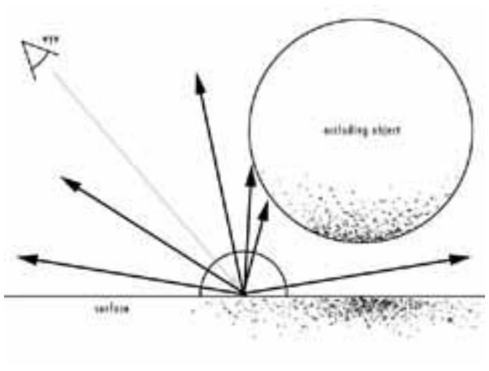
1. The initial eye ray intersects with a surface that has a occlusion shader.



2. The surface point sends out a randomized hemispherical spray of rays, biased towards the normal, out to do hit-tests on the rest of the scene.



3. Each surface point is shaded by a ratio of ray intersections to number of original samples. For example, in this diagram, we see 2 intersections for 6 samples...our ratio is 1/3. Subtracting this ratio from 1 gives us dark areas in the occluded portions of the surface.



4. Lather, rinse, and repeat for every object!

An Example Shader

Let's take a look at a very basic ambient occlusion shader written in Renderman© SL. (This is compiled with PRMan© v11, currently in beta-testing.) I've colored blue any Renderman© reserved variables, and green any Renderman© native functions.

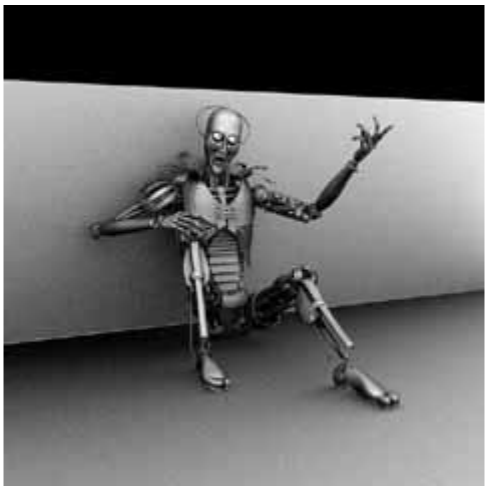
```
surface amb_occ (float samples = 1; color Camb = 1) {

     float sum = 0;
     normal Nn = normalize(N);
     vector R = reflect(normalize(I), Nn);

     gather ("illuminance", P,R, PI/2, samples,
"distribution", "cosine") {
     }
     else { sum = sum + 1; }
     sum = sum / samples;

     Ci = Camb * sum;
     Oi = Os;

}
```

Our shader takes in a number of samples (somewhere between 16 and 128 is a good start), and a color to multiply against (generally white). We calculate the reflection direction and normalize the normal...and then pass to PRMan© 11's gather loop. If you're working with BMRT or Entropy©, you could write this loop by hand...it essentially calculates the spread of rays according to a cosine distribution and returns whether or not there's a hit. The nice thing about the gather loop is that it's optimized, does a lot of the work for us, and also includes the else statement.

The else statement runs if there *isn't* a hit. Which, as we add up the sum, takes care of the subtracting from 1 mentioned in step three. When our gather loop's done, we divide by the number of samples, multiply by the ambient color, and return the final color.

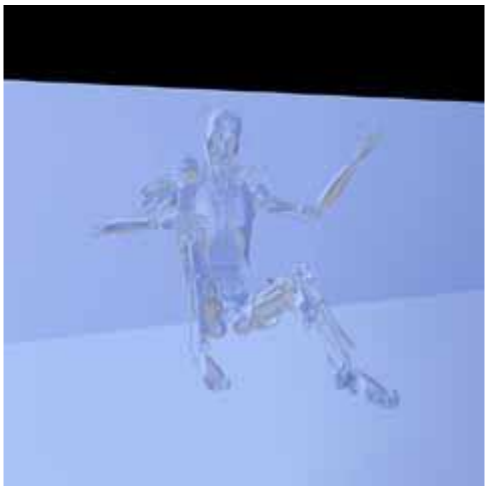
A Practical Case

While this could be used as a portion of a bigger shader, ambient occlusion tends to be very computationally expensive. Consequently, would be smarter to do a single ambient occlusion pass and then either bake it in as a texture or composite it in in post. For example, we might do an ambient occlusion pass, a diffuse pass, and an ambient color pass.

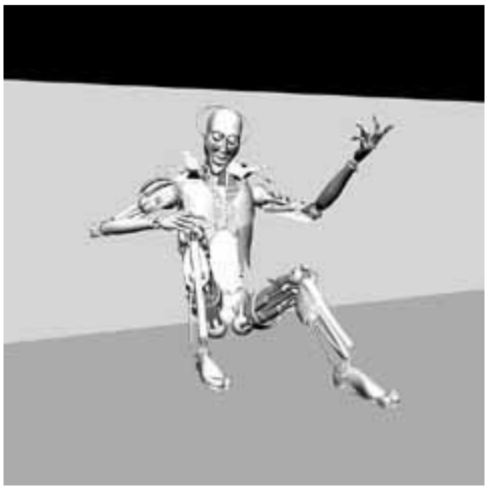


Our **ambient occlusion pass**, done with 64 samples, took roughly 5.5 hours to render. Your mileage may vary from render to render, and from renderer to renderer. Note the noise...many renderers have a function that will blur the noise...but it still comes down to # samples = image quality. This image is darker than I'd like...I still have some tweaking to do in the shader.

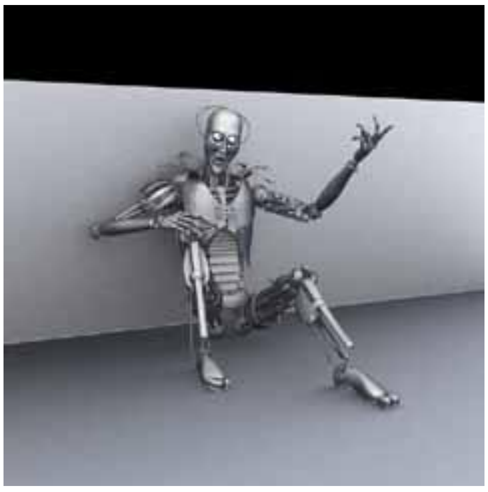
I also did a 16 sample version, which you can see here.



We'd also like our ambient light to have a color. Instead of using the flat object color, we can use an environment map that's been blurred heavily, say 30%. For true accuracy, you can average the rays that don't hit anything, and access the environment map with that instead of the reflection vector. This ensures the color accessed is from the direction of greatest ambient contribution. Our **ambient color pass**, using a map made from 5 images of a sky with one gray image for the floor, took about 20 seconds. Since it's pretty much blue, I didn't bother with the vector averaging.



Our **diffuse pass** uses one light to help give our lighting some direction. You could include shadows with this pass or in a separate pass.



I used Photoshop to **combine** the three layers. The ambient color pass either screens or overlays the diffuse pass, which in turn is multiplied by the occlusion pass. Tweak to taste, and viola. A soft shadow, cloudy day look...with one light!