

Basic 3D Lighting

An Introduction to Simplified Lighting in 3D Computer Graphics Software

Note: This article does not cover artistic aspects of lighting.

For artistic information, please refer elsewhere. Here is some suggested reading:

Digital Lighting & Rendering - Jeremy Birn – Book
Available at most major bookstores. ISBN-13: 978-0321316318

Light - Richard Yot - Web Article
<http://itchy-animation.co.uk/light.htm>

Pixel Cinematography - John Kahrs, Sharon Calahan, Dave Carson, Stephen Poster - Article
https://www.siggraph.org/education/materials/siggraph_courses/s96_course30.pdf

Light Objects in 3D software

3D software does not calculate lighting with perfect accuracy. Generally, real world lighting is far too complex for a computer to calculate, so various levels of simplifications are made. Software packages will often use combinations of different levels of complexity, allowing you to create very complex calculations for some aspects of the lighting but not others.

Here we will focus on the most standardized simplification of lighting in 3D software. Most software simplifies lighting quite heavily, and features light objects, which are object types created for the purpose of illuminating your scene. Light objects in most packages, do not show up by themselves in final renders. If you create a light, and render your scene, it will usually not be visible. Generally, to create the impressions of actual light sources, you have to create geometry to appear as the light source, and apply a bright material to it, usually with incandescence or self illumination set on the material.

The images below contain a white sphere which represents the light's placement. The lights used below have been centered inside a sphere. The example sphere material has been set to have white incandescence. This is done so that it would appear something like a light bulb. Also, the object has been set not to cast or receive shadows, since otherwise, the entire scene would appear black, since the light would be inside of a shadow casting sphere, and the sphere would end up casting a shadow on the entire scene.

The following examples use Lambert shading for the diffuse color of objects, and Blinn shading for the specular highlights. Lambert and Blinn shading are two very popular shading methods named after their inventors.

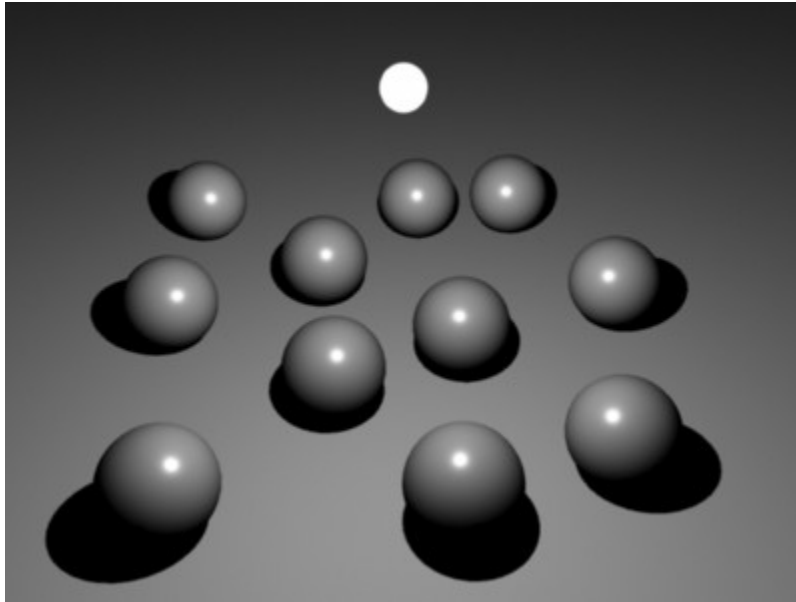
Also, for most basic lighting calculations, 3D software assumes that the surface of the light is simply an infinitely small point, and thus, the lighting is quite “hard” looking. In real life, larger light sources produce softer light. Softer light is quite complex for a computer to calculate, so we first discuss hard lighting, and lighting from tiny light sources.

Light Types

Point/Omni Light

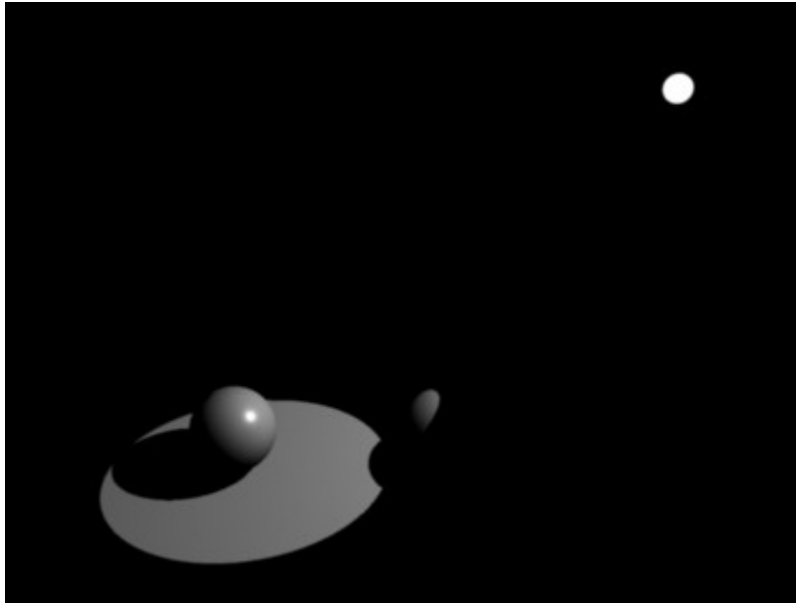
A point light, or “omni light” casts rays in all directions from a single source. Point lights are similar to tiny light bulbs in real life.

Note that in Maya, another type of light called a “volume light” exists, which behaves almost exactly like a point light, but has extra attenuation controls. In Maya, you will likely want to use volume lights instead of point lights.



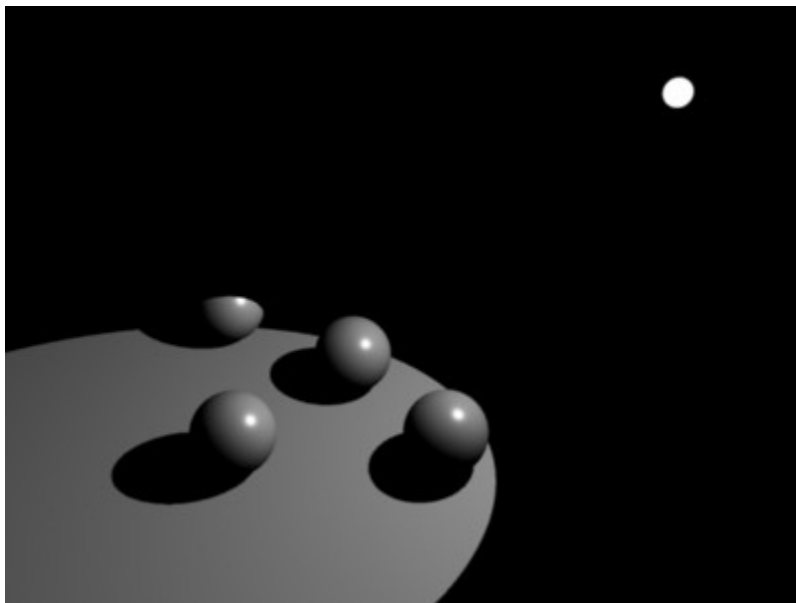
Spot Light

A spotlight is like a point light but casts a beam of light only in one direction, like the light of a flashlight, a spot light in a theater, or a headlight. Spot lights have properties to control the area affected by light.



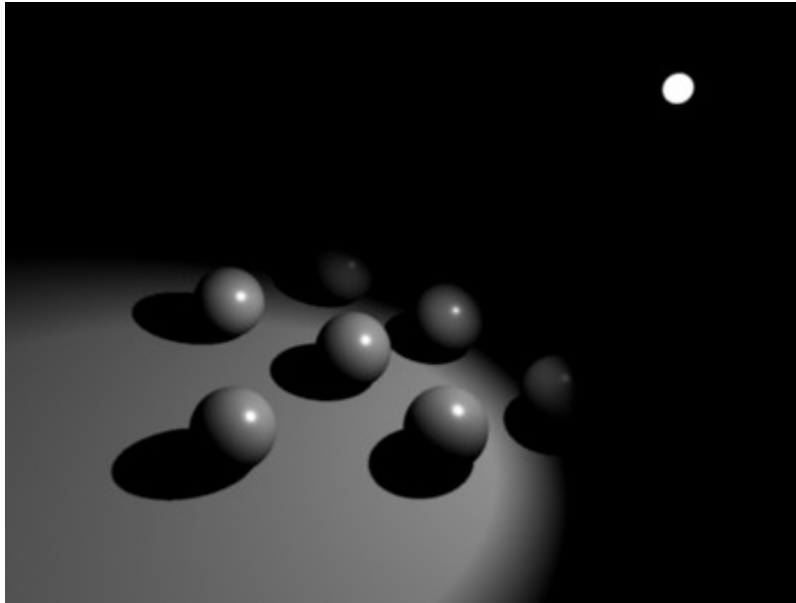
Hotspot / Cone Angle

The cone angle determines the angle in which light will shine, and controls the size of the lit area.



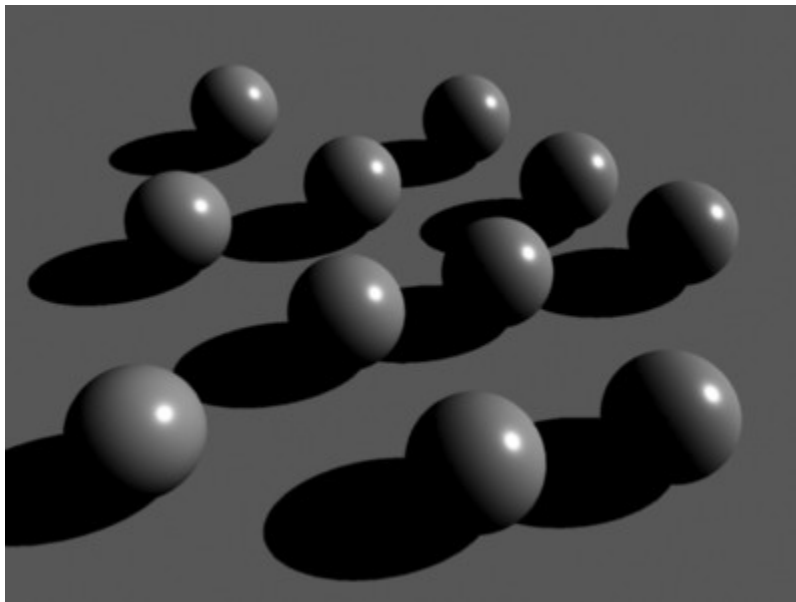
Pnenumbra / Falloff

The falloff angle determines how quickly the light will go from being full brightness to full darkness. It is used to soften the hotspot's edge.



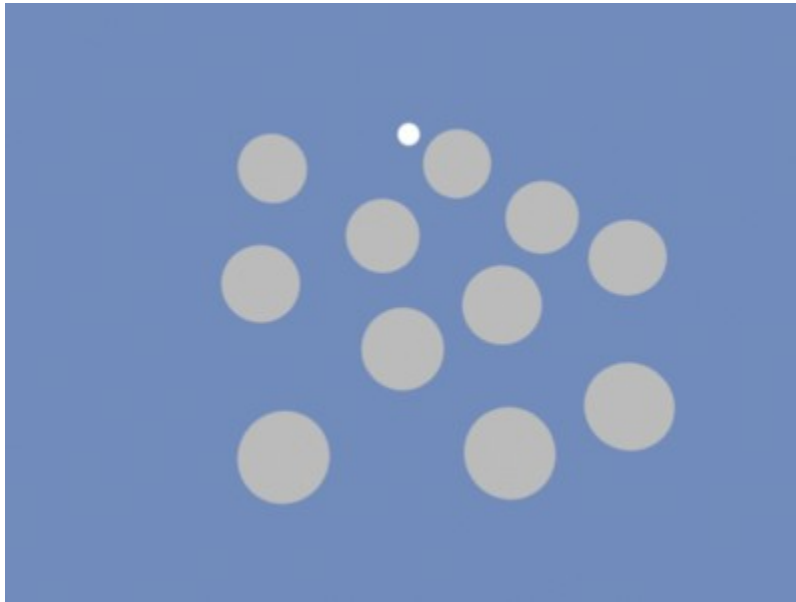
Directional Light (Also called “Infinite” or “Sun” light)

Directional lights cast parallel light rays in a single direction, as the sun does (for all practical purposes) at the surface of the earth. Directional lights are primarily used to simulate sunlight, or any other light coming from a very distant light source. The direction of directional lights is usually controlled by rotating the light. In most programs, the position of the directional light makes no difference, it simply fills the entire scene with light traveling in one direction.



Ambient Light

An ambient light provides perfectly even lighting throughout the entire scene, with no shading. In the image below I changed the ground color so the sphere's silhouettes can be seen.



Ambient lights are frequently used to make special effects and special passes, such as solid black and white “mask layers”.

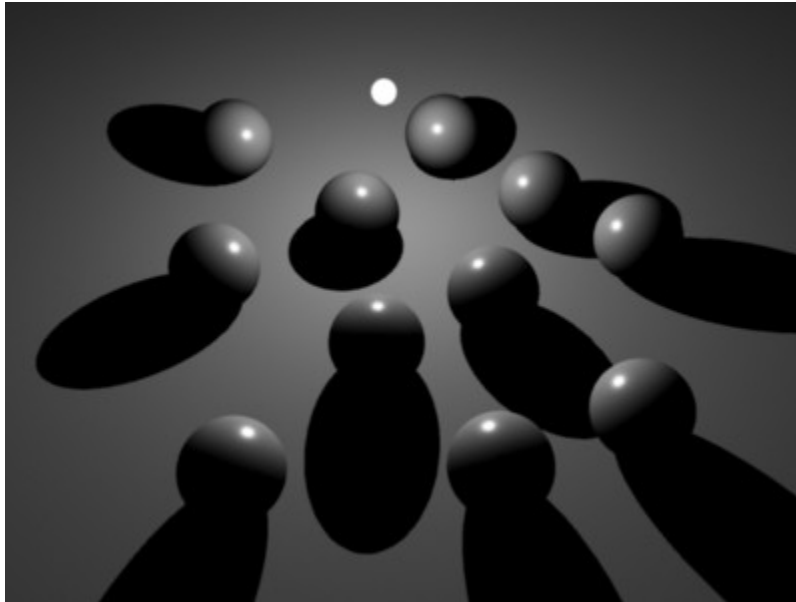
In regular scene lighting, ambient lights should usually be used very sparingly, if at all. They can quickly eliminate contrast in lighting and wash out a scene. In most situations, ambient lights should only be used if their light is combined with other techniques such as ambient occlusion.

Very often, dim “ambient” lighting can be better represented by using several very dim point, spot, or directional lights.

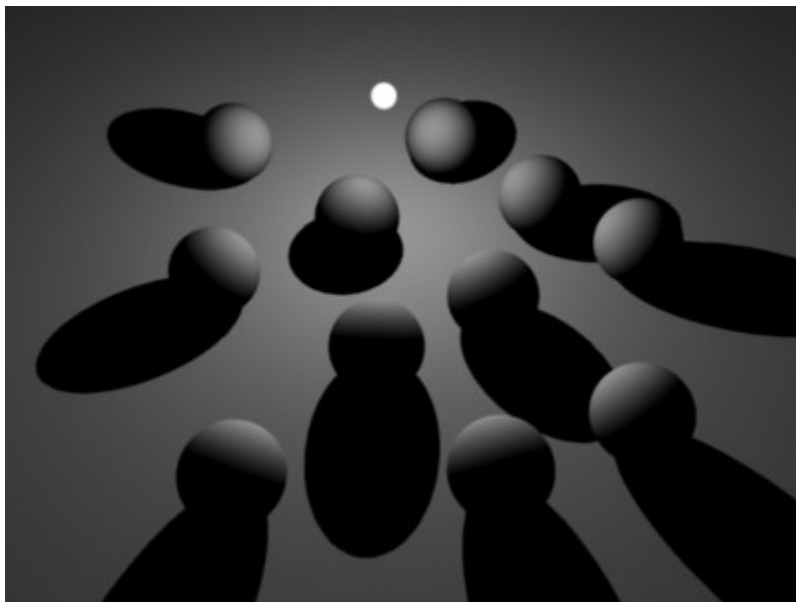
Lighting Attributes

Diffuse and Specular.

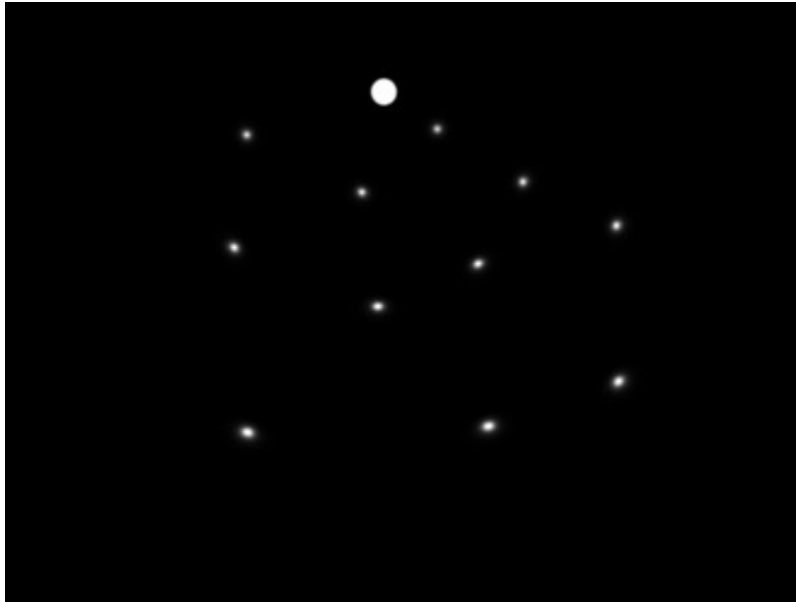
In the image below, the light is affecting both diffuse and specular shading. It is “emitting” both diffuse and specular.



In the image below, the light is emitting diffuse but its specular emission has been turned off. Thus it does not cause specular highlights on objects.

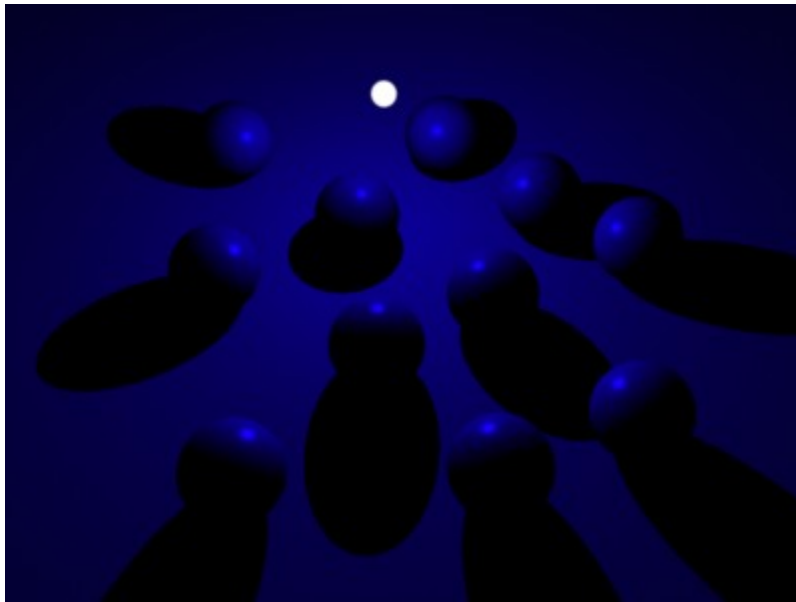


In the image below, the light is emitting specular highlights but its diffuse emission has been turned off. Only specular highlights are created on objects. The light causes no regular, diffuse shading.

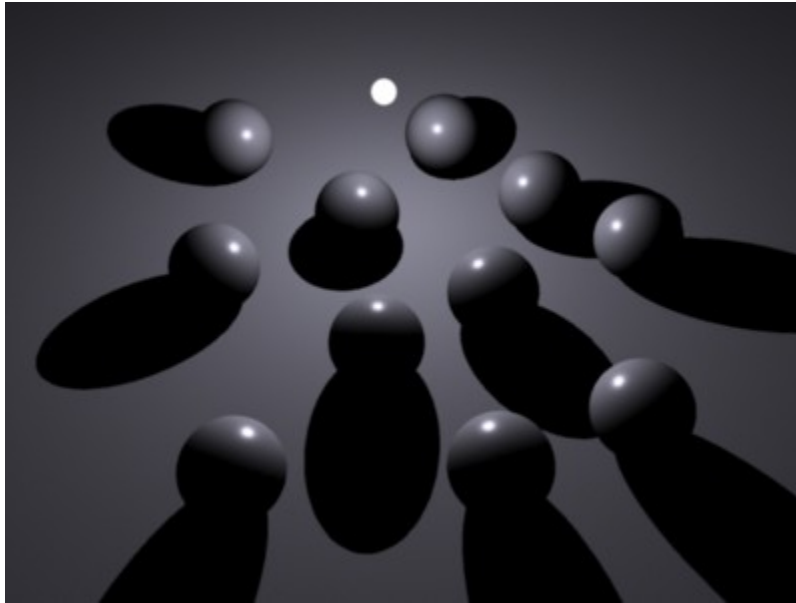


Light Colors

Usually you can control the color of the lights. Lights in real life are usually not perceived as being extremely strong colors, so usually it is best to only tint the color of a light. Light color generally multiplies with the intensity of the light to determine the final brightness and color.

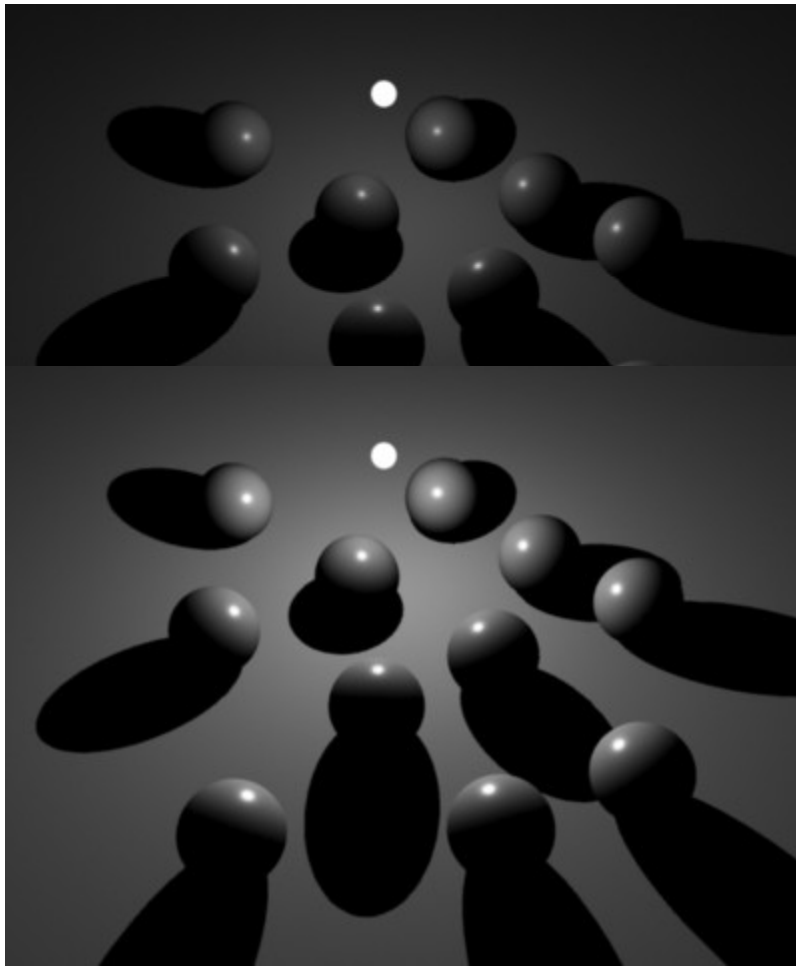


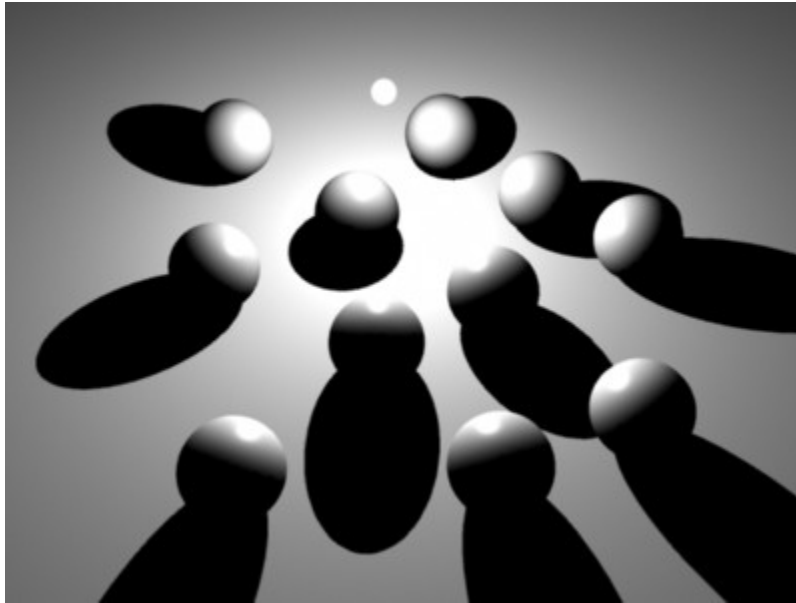
A slight tint usually looks more natural than a strongly colored light source.



Intensity

Intensities, shown at 0.5, 1.0 and 2.0



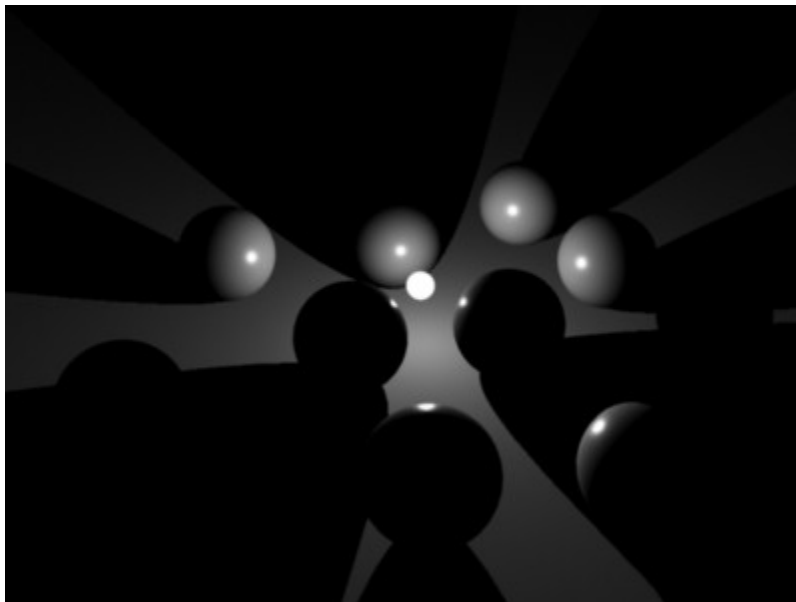


Decay

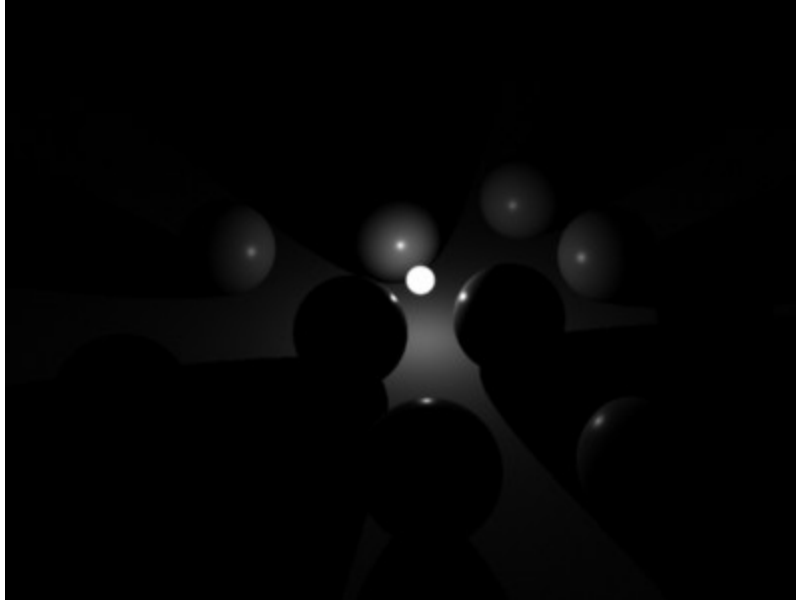
Decay causes light to gradually get darker as it gets farther from the light source. Without decay (or attenuation, described later) it doesn't matter how far away an object is from a light, it will always be affected.

In real life, lights have an inverse square, or quadratic decay. What this means is that twice as far away from a light, the light will be one quarter as bright. In real life, other factors, such as bounce lighting, tend to make the quadratic decay of lights seem less extreme, but in 3D software Quadratic decay is often too much. Thus, some simpler like linear decay or attenuation generally look better. Linear decay for example means that twice as far away from the light, the light is half as bright. Since linear decay only cuts brightness in half, whereas quadratic decay cuts in into quarters, linear decay is less extreme and often looks nicer in renders, allowing the light to spread farther without being too bright.

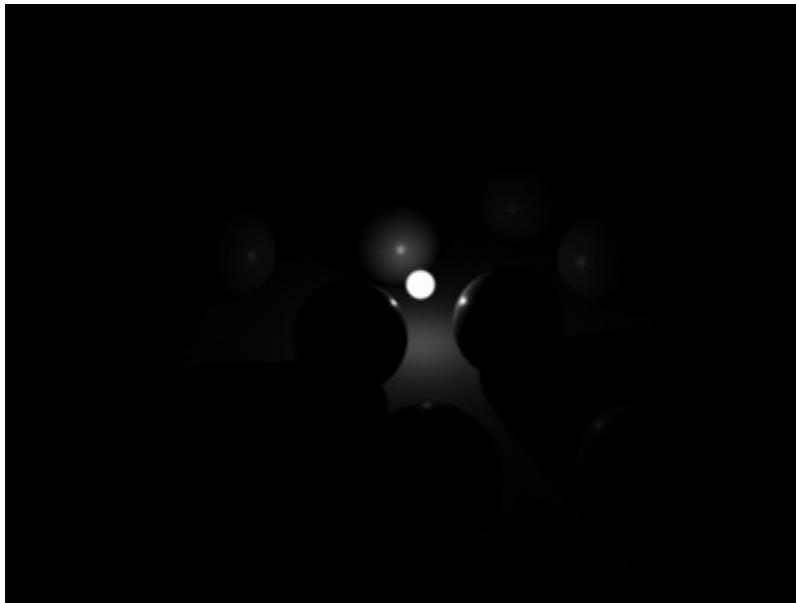
No decay.



Linear decay.



Quadratic decay.



Attenuation

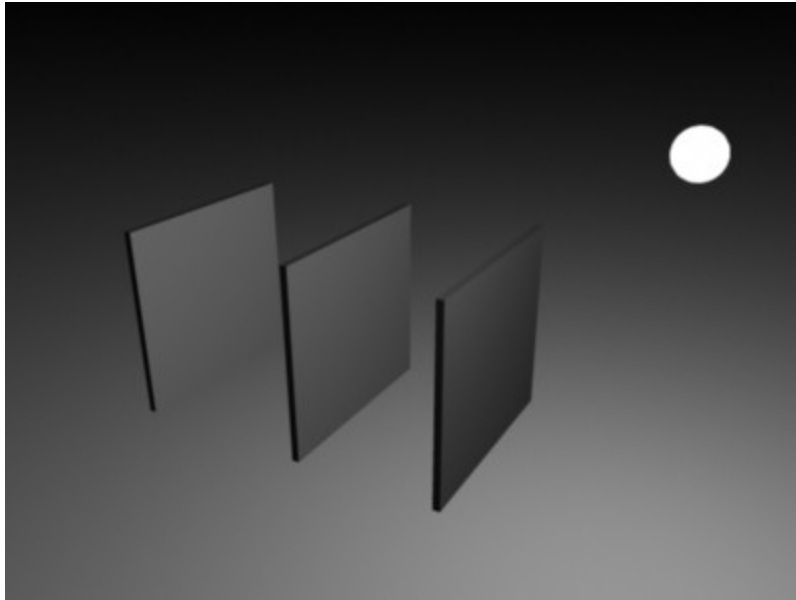
Attenuation is like light decay, but instead of getting darker based on a simple equation, the light gets darker based on user specified ranges. A far range generally specifies the distance at which objects will be in complete darkness. Some software also uses a near range, which indicates the distance at which light will start to attenuate.

Sometimes gradients or graphs are also used to define the exact attenuation of a light. For example, Maya's volume light type provides attenuation with a graph to control brightness at various distances.

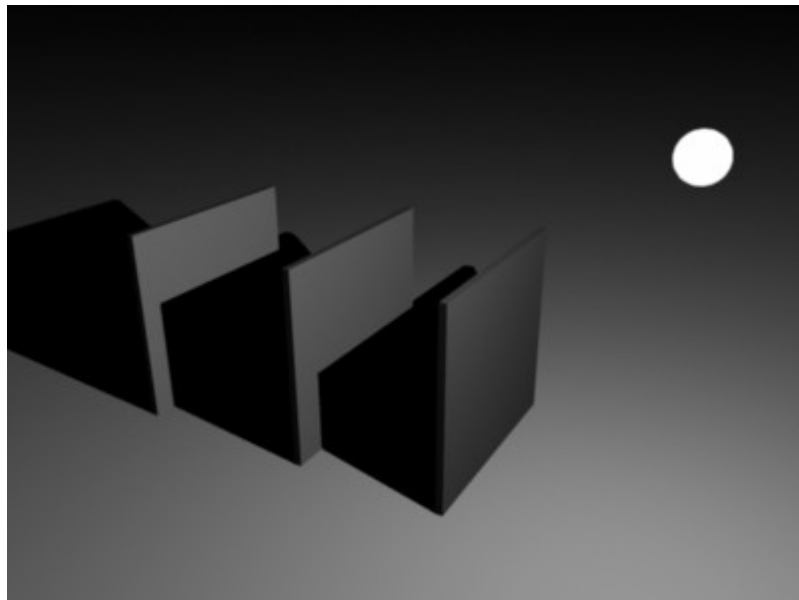
Shadows

Without shadows enabled, light goes right through objects. In most software, shadows are off by default and need to be turned on. The two common shadow types are raytraced and depth mapped shadows.

No shadows



With shadows



Shadow Types

There are several methods that software may use to control and calculate shadows. The two most popular methods are depth mapped shadows, and raytraced shadows. Both are discussed below.

Depth Mapped Shadows

Depth mapped shadows, often simply referred to as “shadow maps” use texture maps to store information about which objects are in the shadows. When rendering, the shadow maps are computed first, and then used in the later phase, rendering the final image.

Shadow maps usually render quite quickly, however they can use a lot of memory at high resolutions.

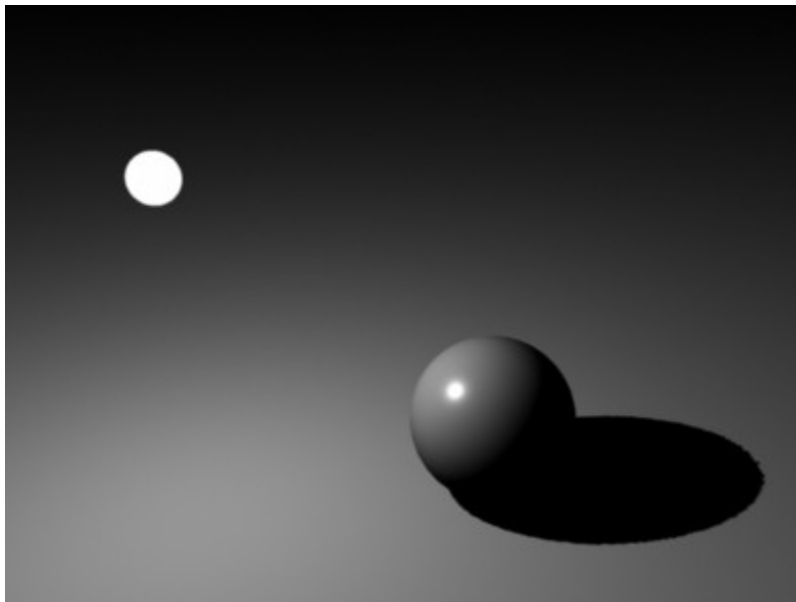
They are good for producing soft shadows, but the soft shadows they produce are not physically correct, and will not match up with the way soft shadows look in real life.

Shadow maps usually don't support transparency, and are especially problematic for casting shadows from objects with texture mapped transparency.

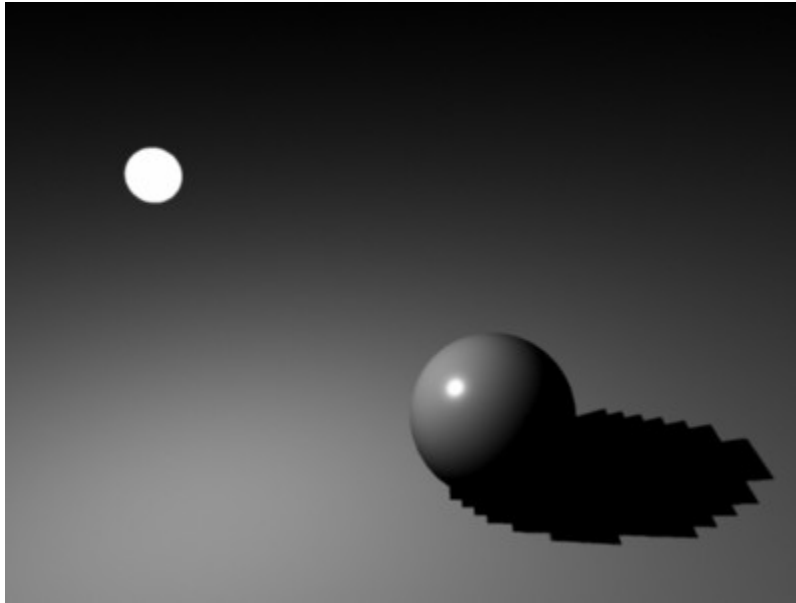
Resolution/ Map Size

The resolution controls how large the generated shadow maps should be. This in turn affects the quality of the generated shadows. It is very similar to image resolution. Lower resolutions can look blocky if viewed close up. Generally, the lowest resolution that looks good should be chosen.

A shadow map size of 512.



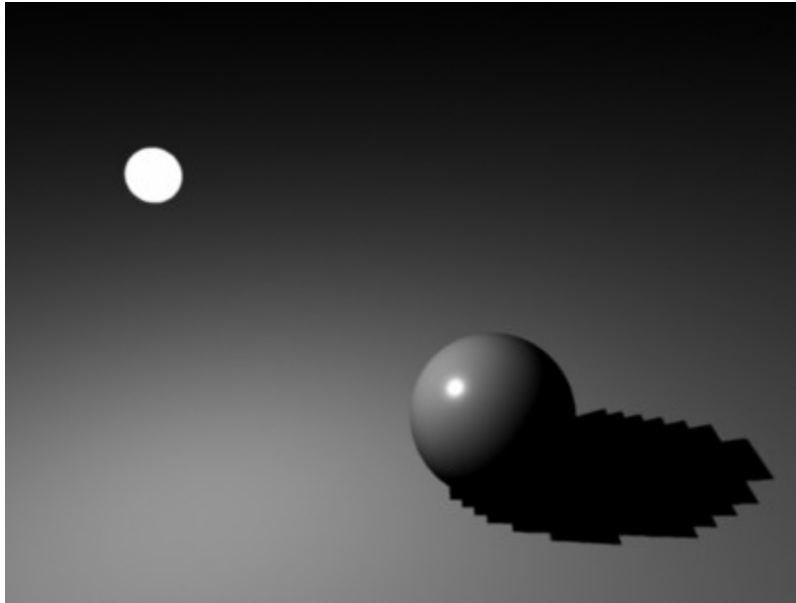
A shadow map size of 64. No filtering. (Filtering turned off to show shadow's jaggy edge.)



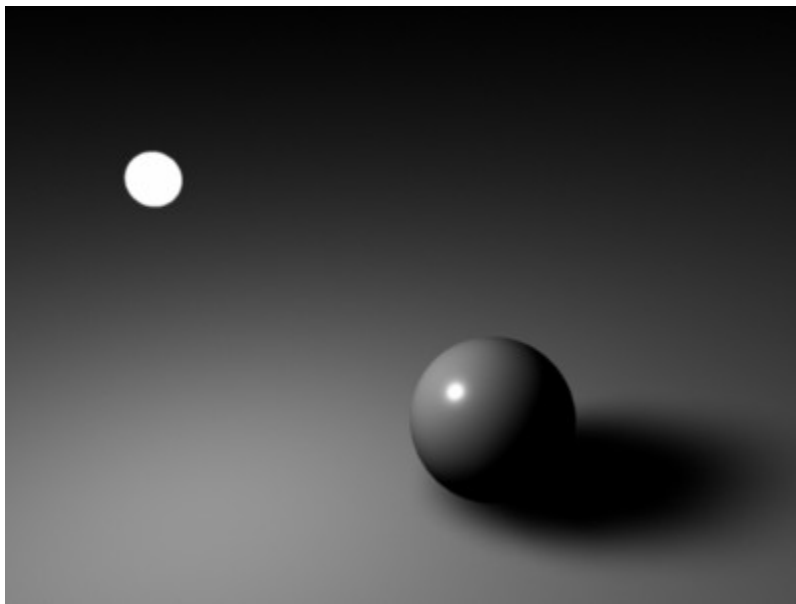
Filter/Focus/Sampling

Filtering blurs the shadows, so that they look softer.

Filter size of zero, no filtering



Filter size of 4



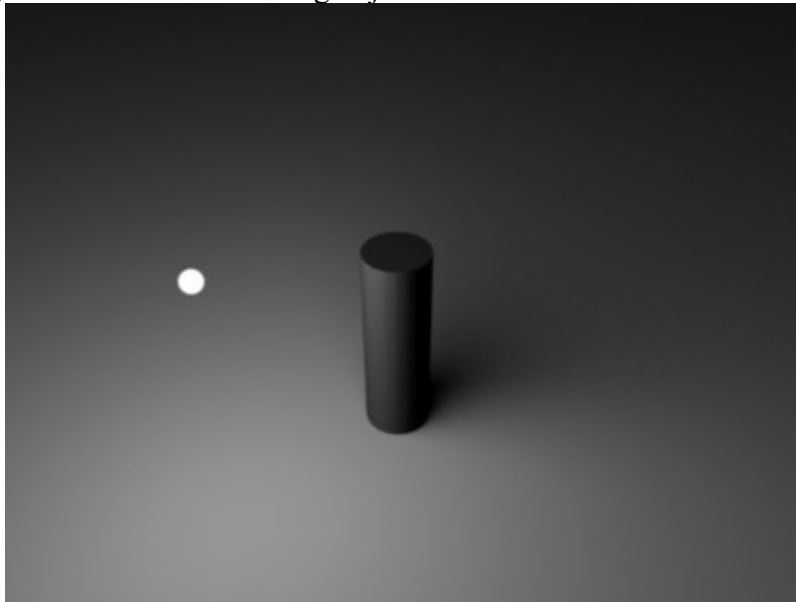
Raytraced Shadows

Raytraced shadows are more accurate than depth mapped shadows, and they support features like transparent shadows, and correct soft shadows. They are usually slower to render than shadow mapped shadows.

Proper Soft Shadows

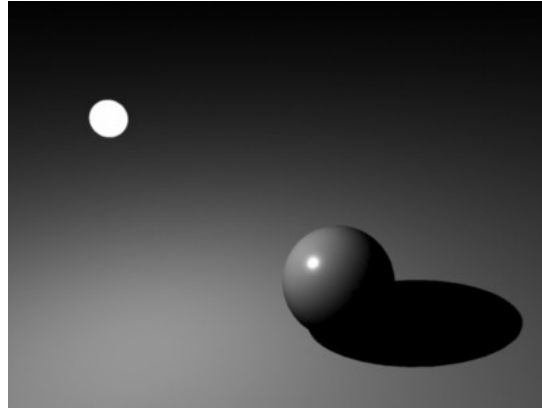
In real life, soft shadows are caused by larger lights. In these examples we are only dealing with tiny, point lights. Software often features controls over soft shadows which can make the shadows appear as if the light is larger. (“Shadow Radius”, “Shadow Spred”, “Area Shadows” are all examples of names that various software uses for these controls.)

Note in the image below how the shadow's edge is sharper closer to the casting object and gets blurry as it get farther away from the shadow casting object.

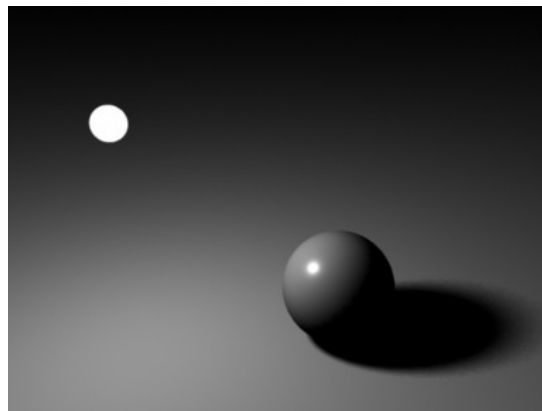


Below are examples of other shadow radius values, and how they affect the look of the cast shadow.

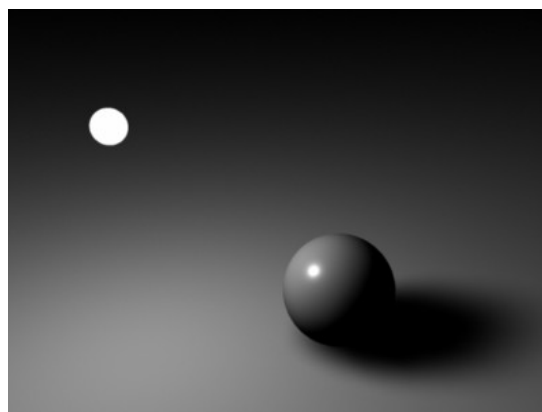
Shadow Radius of 0



Shadow Radius of 0.5



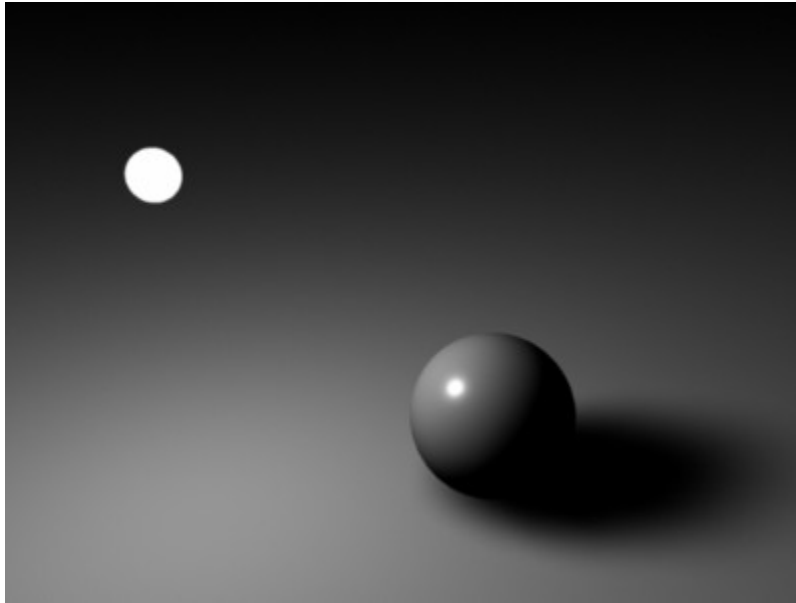
Shadow Radius of 1.0



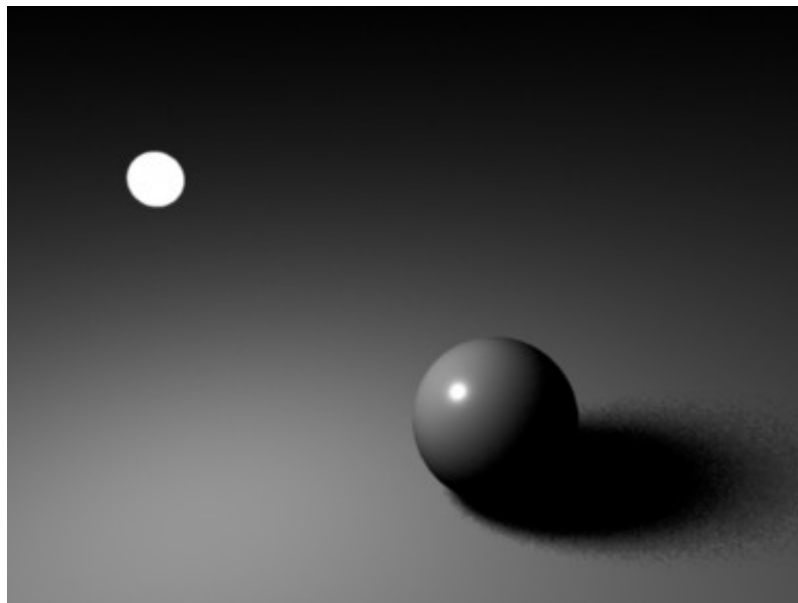
Shadow Rays – Raytraced shadow sampling

Because of the way ray traced soft shadows are sampled, they naturally appear quite grainy. In order to smooth out the shadow, more shadow rays can be used. More shadow rays will cause slower rendering. Thus, in order to optimize rendering, an artist must choose the lowest number of shadow rays necessary for the shadow to look good. Larger lights need more shadow rays to look good.

10 Shadow Rays

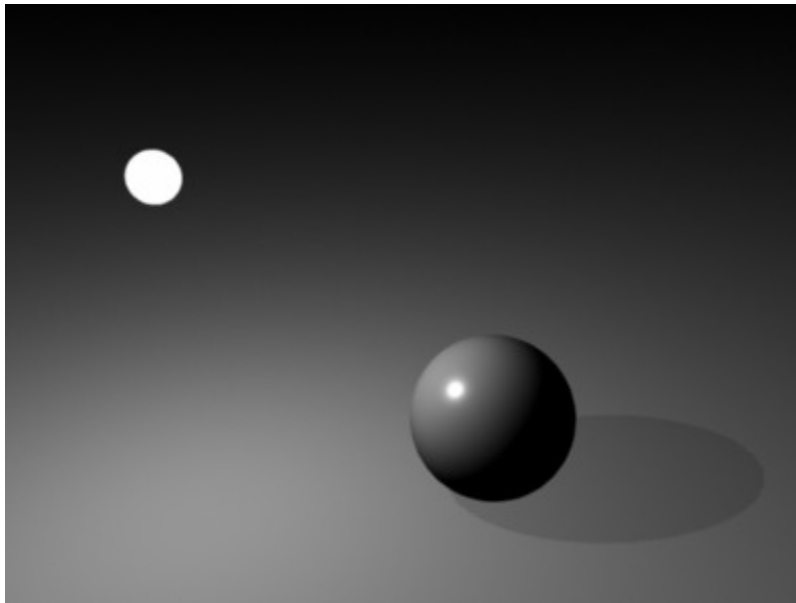
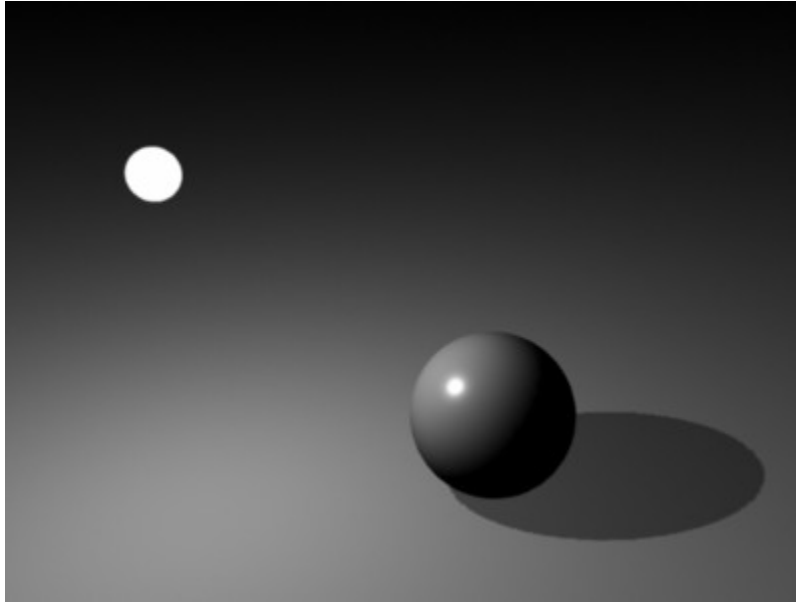


5 Shadow Rays



Shadow Intensity/Color/Density

Shadow color or density controls are used for letting some light into shadow area. Often, shadows appear to dark when rendered, and this can be a quick fix. Note that this isn't very accurate to real life however. In real life, if only one light source exists, and there is no bounce light, and the shadow casting object is not transparent, shadow color is always black. As a quick fix, or “cheat” however, shadow density can be used for simulating light bounces (from objects around shadows) as well as faking shadows being casted from transparent objects.



Transparent Shadows

The biggest difference between shadow maps and raytraced shadows is the fact that shadow maps don't

support transparency. In the images below you can see how the raytraced shadow interprets the thickness of the wine glass as well as how many walls the rays go through. You can adjust the ray depth of a raytraced shadow to give it a longer life to pass through more transparent objects. The top image below uses raytraced shadows (and has a raytraced shadow depth of 10). It shows transparency in the shadows. The second image below is using a shadow map, and does not show transparent shadows. Note that in real life, shadows from transparent objects are generally much more complex. For true transparent shadows, advanced rendering features such as “caustics” need to be used. However, for simple situations with transparent objects, raytraced shadows are usually better than depth mapped shadows.

