Chapter 21

Light (technical)

In this chapter we will describe in more detail how light and reflections are properly measured and represented. These concepts may not be necessary for doing casual computer graphics, but they can become important in order to do high quality rendering. Such high quality rendering is often done using stand alone software and does not use the same rendering pipeline as OpenGL. We will cover some of this material, as it is perhaps the most developed part of advanced computer graphics. This chapter will be covering material at a more advanced level than the rest of this book. For an even more detailed treatment of this material, see Jim Arvo's PhD thesis [3] and Eric Veach's PhD thesis [71].

There are two steps needed to understand high quality light simulation. First of all, one needs to understand the proper units needed to measure light and reflection. This understanding directly leads to equations which model how light behaves in a scene. Secondly, one needs algorithms that compute approximate solutions to these equations. These algorithms make heavy use of the ray tracing infrastructure described in Chapter 20. In this chapter we will focus on the more fundamental aspect of deriving the appropriate equations, and only touch on the subsequent algorithmic issues. For more on such issues, the interested reader should see [71, 30].

Our basic mental model of light is that of "geometric optics". We think of light as a field of photons flying through space. In free space, each photon flies unmolested in a straight line, and each moves at the same speed. When photons hit a surface, they scatter in various directions from that point. We also assume that the field is in equilibrium.

21.1 Units

If we want to carefully simulate realistic images, we first need to understand the units used for measuring light. We will start this discussion with some simple photon mea-

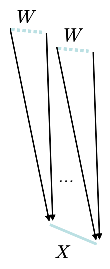


Figure 21.1: A sensor counts the photons that pass through X from incoming directions within the wedge W. This gives us $\Phi(W, X)$, a measurement called radiant flux.

surements. These will lead us to a very useful unit called radiance.

21.1.1 Radiant Flux

We can think of light as a bunch of photons flying through space in various directions. Imagine a "sensor" (W, X) out in space, where X is a smooth imaginary reference surface and W is a *wedge* of directions. In this sensor, we count the number of photons that come in from any direction within the wedge W and pass through the surface X. There may be a physical surface of our scene coincident with X, or the imaginary sensor may be sitting in free space. See Figure 21.1.

This sensor then counts the number of photons it receives each second. Each photon carries energy measured in units of *joules*. By dividing the energy by time (measured in seconds) we get a measurement called radiant flux, measured in *watts*. We use the symbol $\Phi(W, X)$ to represent such a measurement.

Next, we assume (or verify by experiment) that $\Phi(W, X)$ varies continuously as we continuously alter the geometry of the sensor (translate, rotate, or change its size). Given this assumption we are now in the position to define a slew of useful radiometric measurements.

21.1.2 Irradiance

First, we want to define a measurement of light, say over a very small planar sensor X with normal $\vec{n}$, that does not depend on the actual size of the sensor X. We can get this by simply dividing our measured radiant flux by the area of the sensor (measured in square meters). Doing this, we get a measurement

$$E(W, X) := \frac{\Phi(W, X)}{|X|}$$

We can measure $E(W, X)$ for smaller and smaller X around a single point $\tilde{x}$. Under reasonable continuity assumptions about Φ, this ratio converges (almost everywhere)

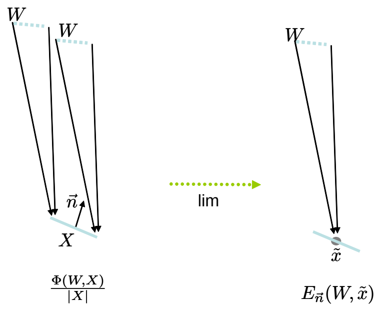


Figure 21.2: We can divide radiant flux by $|X|$. In the limit, this becomes the pointwise incoming irradiance measurement $E_{\vec{n}}(W, \tilde{x})$.

to a value that we call an (incoming) *irradiance* measurement, and write as $E_{\vec{n}}(W, \tilde{x})$. See Figure 21.2. We need to keep around the $\vec{n}$ parameter in order to specify the orientation of the smaller and smaller sensors used in this measurement sequence. If we had considered the same point $\tilde{x}$ in space on a different surface X' with a different normal $\vec{n}'$, we would obtain a different quantity $E_{\vec{n}'}(W, \tilde{x})$. (Moreover, since the wedge W is finite, there is no easy way to relate $E_{\vec{n}}(W, \tilde{x})$ to $E_{\vec{n}'}(W, \tilde{x})$. Later on, when we define a quantity called radiance, we will shrink W down to a single vector. In that case, a simple cosine factor will be able to relate measurements with different normals.)

Often, in the literature, the first argument for E is dropped from the notation, and is inferred somehow from conventions and context. For example, in some contexts, it may be clear that W is the entire upper hemisphere above the point. Similarly, the normal parameter is often dropped from the notation and inferred from context.

Suppose our finite sensor surface X is broken up into a bunch of smaller surfaces, X_i. Then we can compute flux over the entire sensor as the sum

$$\Phi(W, X) = \sum_i \Phi(W, X_i) = \sum_i |X_i| \, E(W, X_i)$$

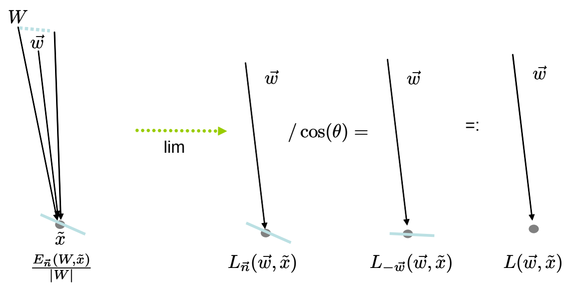
Likewise, under reasonable continuity assumptions about Φ, we can compute flux from pointwise irradiance as

$$\Phi(W, X) = \int_X dA \; E_{\vec{n}(\tilde{x})}(W, \tilde{x})$$

Where $\int_X$ is an integral over the surface X and dA is an area measure over the positions $\tilde{x}$.

21.1.3 Radiance

We next want to define a measurement that does not depend on the size of W, and so we want to divide out by $|W|$, the *solid angle* measure of W. The solid angle of a



W
$\vec{w}$
$\tilde{x}$
$\frac{E_{\vec{n}}(W,\tilde{x})}{|W|}$
lim
$/\cos(\theta) =$
$=:$
$L_{\vec{n}}(\vec{w},\tilde{x})$
$L_{-\vec{w}}(\vec{w},\tilde{x})$
$L(\vec{w},\tilde{x})$

Figure 21.3: We can divide an irradiance measurement by $|W|$ to obtain a measurement $L_{\vec{n}}(W,\tilde{x}) = \frac{E_{\vec{n}}(W,\tilde{x})}{|W|}$. This converges to $L_{\vec{n}}(\vec{w},\tilde{x})$. By dividing out by $\cos(\theta)$ we can convert this to $L_{-\vec{w}}(\vec{w},\tilde{x})$. Dropping the subscript gives us the radiance measurement $L(\vec{w},\tilde{x})$ used for a ray.

wedge of directions from the origin is simply defined as the area covered by the wedge on the unit sphere. These units are called *steradians*, where the wedge of *all* directions covers 4π steradians.

We now define a new radiometric measurement by dividing irradiance by steradians

$$L_{\vec{n}}(W,\tilde{x}) = \frac{E_{\vec{n}}(W,\tilde{x})}{|W|}$$

We can measure this using smaller and smaller wedges around a vector $\vec{w}$ pointing towards $\tilde{x}$. Again, under reasonable continuity assumptions for Φ this converges (almost everywhere) to a measurement which we call $L_{\vec{n}}(\vec{w},\tilde{x})$, where $\vec{w}$ is now a single direction vector, and not a wedge. See Figure 21.3.

We can now drop the dependence on $\vec{n}$ by converting our measurement $L_{\vec{n}}(\vec{w},\tilde{x})$ to that of $L_{-\vec{w}}(\vec{w},\tilde{x})$. In other words, we consider what the measurement would have been had the X plane been perpendicular to the incoming beam. To do this, we have to account for the ratio between the areas of sensor surfaces as they are tilted. This can be calculated as

$$L(\vec{w},\tilde{x}) := L_{-\vec{w}}(\vec{w},\tilde{x}) = \frac{L_{\vec{n}}(\vec{w},\tilde{x})}{\cos(\theta)}$$

where θ is the angle between $\vec{n}$ and $-\vec{w}$. We now drop the normal parameter, writing this simply as $L(\vec{w},\tilde{x})$ instead of $L_{-\vec{w}}(\vec{w},\tilde{x})$. We call this measurement *incoming radiance*. See Figure 21.3. In summary

$$L(\vec{w},\tilde{x}) \quad := \quad \frac{1}{\cos(\theta)} \lim_{W \to \vec{w}} \frac{1}{|W|} \left(\lim_{X \to \tilde{x}} \frac{\Phi(W,X)}{|X|} \right) \tag{21.1}$$

Going the other way, given a spatially and angularly varying $L(\vec{w},\tilde{x})$, we can com-

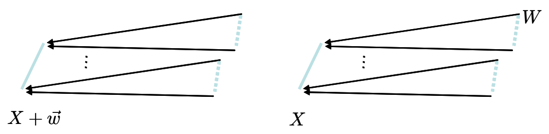


Figure 21.4: We can shift a sensor along a central direction $\vec{w}$. In the limit for small wedges, they will be measuring mostly the same photons. From this we can conclude that radiance is constant along a ray.

pute the radiant flux over a large sensor (W, X) as

$$\Phi(W, X) = \int_X dA \int_W dw \; L(\vec{w}, \tilde{x}) \cos(\theta)$$

where dw is a differential measure of steradians. Radiance measurements allow us to measure light at a point and direction without keeping track of the size and orientation of the measurement device. Moreover, even though our radiance notation includes a 3D point, $\tilde{x}$, as one of its variables, in fact (see below for an argument), radiance remains constant along a ray in free space. That is

$$L(\vec{w}, \tilde{x}) = L(\vec{w}, \tilde{x} + \vec{w})$$

For a point on a surface, it is also useful to have a measurement $L(\tilde{x}, \vec{w})$ of *outgoing radiance* from the point $\tilde{x}$ in the direction $\vec{w}$. (We reverse the order of the arguments to L to distinguish incoming from outgoing radiance). We can define this as

$$L(\tilde{x}, \vec{w}) \quad := \quad \frac{1}{\cos(\theta)} \lim_{W \to \vec{w}} \frac{1}{|W|} \left(\lim_{X \to \tilde{x}} \frac{\Phi(X, W)}{|X|} \right)$$

where $\Phi(X, W)$ is the radiant flux of photons *leaving* a finite surface X and going out along vectors in the wedge W (again, note the order of the arguments).

In free space, it is clear that $L(\vec{w}, \tilde{x}) = L(\tilde{x}, \vec{w})$.

Radiance is the most useful quantity needed for computational light simulation. Indeed, we can go back to Chapters 14 and 20 and interpret those methods using these units. For example, when we calculate the color of some 3D point $\tilde{x}$ in an OpenGL fragment shader, we can interpret this as computing the outgoing radiance $L(\tilde{x}, \vec{v})$, where $\vec{v}$ is the "view vector". This outgoing radiance value is also the incoming radiance value at the corresponding sample location on the image plane, and is thus used to color the pixel. Likewise, in ray tracing, when we trace a ray along the ray $(\tilde{x}, \vec{d})$, we can interpret this as calculating the incoming radiance value, $L(-\vec{d}, \tilde{x})$.

Constancy of Radiance Along a Ray (optional)

Suppose, in free space, that we slide our sensor (W, X) along the vector $\vec{w}$ to obtain the shifted sensor (W, X') where $X' = X + \vec{w}$. (See Figure 21.4). In this case our

Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

measured fluxes will not agree $\Phi(W,X) \neq \Phi(W,X')$. But if we compute radiance, in the limit we will get agreement. In particular

$$\begin{aligned}
L(\vec{w},\tilde{x}) &:= \frac{1}{\cos(\theta)} \lim_{W\to\vec{w}} \frac{1}{|W|}\left(\lim_{X\to\tilde{x}} \frac{\Phi(W,X)}{|X|}\right) \\
&= \frac{1}{\cos(\theta)} \left(\lim_{X\to\tilde{x}} \frac{1}{|X|} \lim_{W\to\vec{w}} \frac{\Phi(W,X)}{|W|}\right) \\
&= \frac{1}{\cos(\theta)} \left(\lim_{X\to\tilde{x}} \frac{1}{|X|} \lim_{W\to\vec{w}} \frac{\Phi(W,X+\vec{w})}{|W|}\right) \\
&= \frac{1}{\cos(\theta)} \left(\lim_{X'\to\tilde{x}+\vec{w}} \frac{1}{|X'|} \lim_{W\to\vec{w}} \frac{\Phi(W,X')}{|W|}\right) \\
&= \frac{1}{\cos(\theta)} \lim_{W\to\vec{w}} \frac{1}{|W|}\left(\lim_{X'\to\tilde{x}+\vec{w}} \frac{\Phi(W,X')}{|X'|}\right) = L(\vec{w},\tilde{x}+\vec{w})
\end{aligned}$$

In the third line we use the fact that, in the limit for very small wedges, our sensor and shifted sensor will be measuring the same set of photons, and thus

$$\lim_{W\to\vec{w}} \frac{\Phi(W,X')}{\Phi(W,X)} = 1$$

As a result of this we say that radiance is constant along a ray.

Fundamentally, we can think of this constancy as arising from the combination of two facts. First, our physical assumptions imply that, in free space, the measured flux, Φ, depends only on the set of directed lines that are measured, not where they are measured. Secondly, for any set of lines, say called S, if we parametrize the lines by direction $\vec{w}$ and by where they hit some plane X, then the *line measurement*: $\int_S \, dw \, dA_X \, \cos(\theta)$ will, in fact, not depend on the choice of plane X. Using this terminology, radiance is simply the *density* of flux over the line measure, and the choice of $\tilde{x}$ is not relevant.

21.2 Reflection

When light comes in from W, a wedge of incoming directions about the vector $\vec{w}$, and hits a point $\tilde{x}$ on a physical surface, then some of that light can get reflected off the surface. We make the simplifying approximation/assumption that all reflection is pointwise, i.e., light hitting a point bounces out from that single point only. Let us measure the light that is reflected out along V, some wedge of outgoing directions around an outgoing vector $\vec{v}$. The particulars of this bouncing are governed by the physical properties of the material. We wish to represent the behavior of this bouncing with some function f describing the ratio of incoming to outgoing light. What kind of units should we use for this ratio? Our governing principle is that we want **a ratio that converges** as we use smaller and smaller incoming and outgoing wedges. This implies our second principle (that is desirable in and of itself): we want a ratio that (for small enough wedges) **does not actually depend on the size of the wedges**.

Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

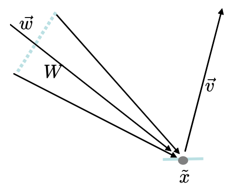


Figure 21.5: A BRDF measures the ratio of incoming irradiance to outgoing radiance.

In this section, we will derive the primary way to describe reflection, called the BRDF. For special materials, like pure mirrors and refractive media, we will use a slightly different representation.

21.2.1 BRDF

We can verify experimentally that most materials (excluding pure mirrors or lenses, see below) have the following diffusing behavior. For any fixed incoming light pattern, the outgoing light measured along any outgoing wedge changes continuously as we rotate this wedge. Thus if we double the size of a small outgoing measurement wedge V, we will see roughly twice the outgoing flux. Therefore, to set the numerator of our ratio f in a way that does not depend on the size of V, we should use units of radiance. Let us call this outgoing measurement: $L^1(\tilde{x}, \vec{v})$. We place a superscript on L^1 to make clear that we are referring here to the measurement of bounced photons.

Similarly, (but possibly surprisingly) we can verify that, for most materials (again mirrors and lenses excluded), if all of the light is coming in from a single small wedge W, and we double the width of this incoming wedge, the amount of flux reflected along a fixed outgoing wedge, and thus the amount of radiance reflected along a fixed outgoing direction, will roughly double as well. Thus, to get a ratio that does not depend on the size of the W, we need the denominator to double in this case. We accomplish this by measuring the incoming light in units of irradiance.

Putting this together, we see that we should measure reflection as

$$\begin{aligned} f_{\tilde{x},\vec{n}}(W, \vec{v}) &= \frac{L^1(\tilde{x}, \vec{v})}{E^e_{\vec{n}}(W, \tilde{x})} \\ &= \frac{L^1(\tilde{x}, \vec{v})}{L^e_{\vec{n}}(W, \tilde{x})|W|} \end{aligned}$$

Where we use the superscript L^e to refer to photons that have been *emitted* by some light sources and have not yet bounced. Once again, by making the incoming wedge smaller and smaller around a fixed $\vec{w}$, this quantity converges to a measurement denoted $f_{\tilde{x},\vec{n}}(\vec{w}, \vec{v})$. This function, f, is called the *bi-directional reflection distribution function*, or *BRDF*. It is a function that can vary with both incoming and outgoing directions. See Figure 21.5.

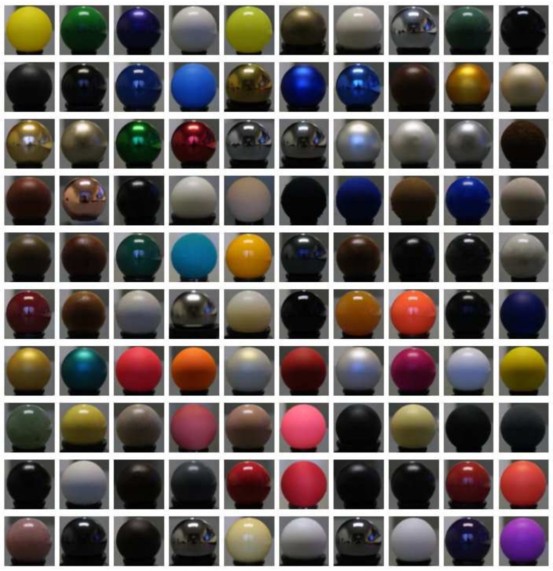


Figure 21.6: An array of BRDFs captured by a measuring device. From [47], ©ACM.

The simplest BRDF is the constant BRDF $f_{\tilde{x},\vec{n}}(\vec{w},\vec{v}) = 1$. This represents the behavior of a diffuse material. In this case, the outgoing radiance at a point does not depend on $\vec{v}$ at all. (Subtlety: this does not mean that incoming photons are scattered equally in all outgoing directions. In fact, on a diffuse surface more of the photons are scattered in the direction of the surface normal, and the amount of scattered photons drops by a cosine factor to zero at grazing angles. In contrast, the outgoing *radiance* does not depend on outgoing angle, since the definition of radiance includes its own cosine factor which cancels this "drop-off" factor. Intuitively speaking, when looking at a diffuse surface from a grazing angle, fewer photons are coming towards you per unit of surface area, but your are also seeing more surface area through your sensor.)

More complicated BRDFs can be derived from a variety of methods.

- We can simply hack up some function that makes our materials look nice in pictures. This is essentially what we did in Section 14.
- We can derive the BRDF function using various physical assumptions and statistical analysis. This involves a deeper understanding of the physics of light and materials.
- We can build devices that actually measure the BRDF of real materials. This can be stored in a big tabular form, or approximated using some functional form. See Figure 21.6.

Suppose we want to compute the outgoing $L^1(\tilde{x},\vec{v})$ at a point on a surface with

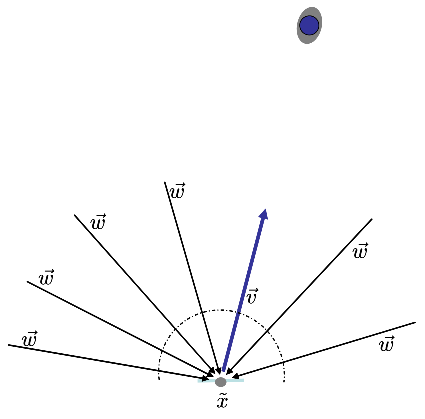


Figure 21.7: To compute the outgoing reflected radiance, $L^1(\tilde{x},\vec{v})$, the reflection equation integrates all of the incoming rays $\vec{w}$ coming towards $\tilde{x}$.

normal $\vec{n}$, due to light coming in from the hemisphere, H above $\tilde{x}$. And suppose H is broken up into a set of finite wedges W_i. Then we can compute the reflected light using the sum

$$L^1(\tilde{x},\vec{v}) \quad = \quad \sum_i |W_i| \; f_{\tilde{x},\vec{n}}(W_i,\vec{v}) \; L^e_{\vec{n}}(W_i,\tilde{x})$$

Likewise, using $f_{\tilde{x},\vec{n}}(\vec{w},\vec{v})$ and $L^e(\vec{w},\tilde{x})$ we can compute the reflected light using the integral:

$$L^1(\tilde{x},\vec{v}) \quad = \quad \int_H dw \; f_{\tilde{x},\vec{n}}(\vec{w},\vec{v}) \; L^e_{\vec{n}}(\vec{w},\tilde{x}) \tag{21.2}$$

$$= \quad \int_H dw \; f_{\tilde{x},\vec{n}}(\vec{w},\vec{v}) \; L^e(\vec{w},\tilde{x}) \; \cos(\theta) \tag{21.3}$$

This is called the *reflection equation*, and it is the basis for most of our models for light simulation. See Figure 21.7.

21.2.2 Mirrors and Refraction

Pure mirror reflection and refraction are not easily modeled using a BRDF representation. In a mirrored surface, $L^1(\tilde{x},\vec{v})$, the bounced radiance along a ray, depends **only** on $L^e(-B(\vec{v}),\tilde{x})$, the incoming radiance along a single ray. Here B is the bounce operator of Equation 14.1.

Doubling the size of an incoming wedge that includes $-B(\vec{v})$ has no effect on $L^1(\tilde{x},\vec{v})$. Thus, for mirror materials, we represent the reflection behavior as

$$k_{\tilde{x},\vec{n}}(\vec{v}) = \frac{L^1(\tilde{x},\vec{v})}{L^e(-B(\vec{v}),\tilde{x})} \tag{21.4}$$

where $k_{\tilde{x},\vec{n}}(\vec{v})$ is some material coefficient. We replace the reflection equation with

$$L^1(\tilde{x},\vec{v}) = k_{\tilde{x},\vec{n}}(\vec{v})\ L^e(-B(\vec{v}),\tilde{x})$$

No integration is required in this case. For this reason, mirror reflection is easy to calculate algorithmically, and easily done in a ray tracing program.

When light passes between mediums with different indices of refraction, such as when light passes into or out of glass, the rays bend using an appropriate geometric rule. Like mirror reflection, at the material interface, the radiance along each outgoing light ray is affected by the radiance of a single incoming light ray. Once again, it is easiest here to use "ratio of radiance" units, as in Equation (21.4).

21.3 Light Simulation

The reflection equation can be used in a nested fashion to describe how light bounces around an environment multiple times. Such descriptions typically result in definite integrals that need to be calculated. In practice, this computation is done by some sort of discrete sampling over the integration domain.

In this section, we will start with the simple light simulation of our shading model from Chapter 14 and then build up to more complicated models.

We use the symbol L with no arguments to represent the entire distribution of radiance measurements in a scene due to a certain set of photons. Such an L includes all incoming and outgoing measurements anywhere in the scene. We use L^e to represent unbounced (emitted) photons, and L^i to represent the radiance of photons that have bounced exactly i times.

21.3.1 Direct Point Lights

In our basic OpenGL rendering model, our light comes not from area lights, but from *point lights*. Such point lights do conform to our continuity assumptions and are not so easily represented with our units. In practice, for point lights we simply replace the reflection equation with

$$L^1(\tilde{x},\vec{v}) = f_{\tilde{x},\vec{n}}(-\vec{l},\vec{v})\ E^e_{\vec{n}}(H,\tilde{x})$$

where E^e is the unbounced irradiance coming in to $\tilde{x}$ due to the point source and $\vec{l}$ is the "light vector" pointing from $\tilde{x}$ to the light. We are free to calculate E^e any way we want. For example, in the real world, the irradiance at a point $\tilde{x}$ due to a very small

Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

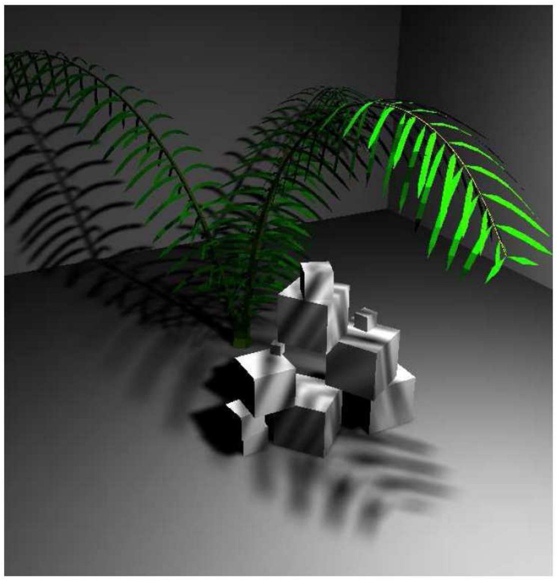


Figure 21.8: Integration over area lights cause soft shadows. From [69], ©ACM.

spherical light source is proportional to $\frac{\cos(\theta)}{d^2}$, where d is the distance between the light and the surface. This is because the solid angle of the small light source drops off with $\frac{1}{d^2}$. (On the other hand, this distance drop off term tends to make pictures too dark and so is often not used). Also note that, in the language of Section 14, we have $\cos(\theta) = \vec{n} \cdot \vec{l}$.

21.3.2 Direct Area Lights

Suppose our light sources have finite area, then we really do need the integral of the reflection equation. In this case, the integrand over H in Equation (21.2) is only non-zero for incoming directions $\vec{w}$ that "see" the light.

If we approximate the integral using some finite number of incoming directions, $\vec{w}_i$, we can use a ray tracing approach to calculate these $L^e(\vec{w}_i, \tilde{x})$ values. When randomness is used to select the directions, this is called distribution ray tracing [13]. As ray intersection is an expensive operation, these integrals can be very expensive to compute accurately.

When the light source is partially occluded by other geometry, this integration will produce shadows that have soft boundaries. This happens because nearby points on a light-receiving surface may see different amounts of the area light source. See Figure 21.8.

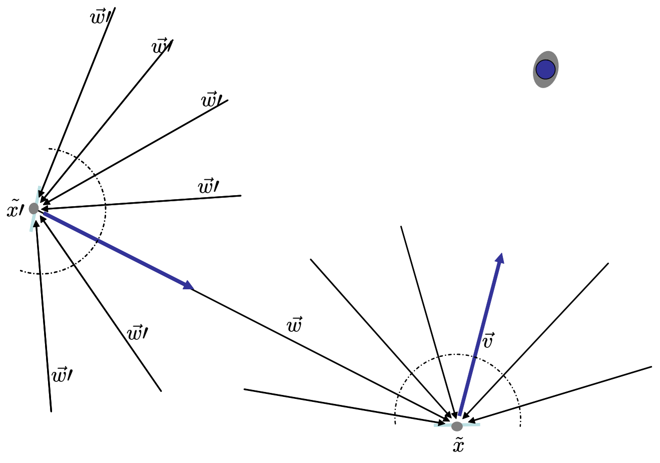


Figure 21.9: To compute L^2, we need two nested integrals. For each direction $\vec{w}$ coming into $\tilde{x}$, there we find the point $\tilde{x}'$ hit by the ray as shown. We then need to integrate the hemisphere above $\tilde{x}'$.

21.3.3 Two Bounces

We can compute L^2, the light distribution of twice bounced photons, by using the reflection equation, but replacing the "input" L^e with L^1.

$$\begin{aligned}
L^2(\tilde{x},\vec{v}) &= \int_H dw\ f_{\tilde{x},\vec{n}}(\vec{w},\vec{v})\cos(\theta)L^1(\vec{w},\tilde{x}) \\
&= \int_H dw\ f_{\tilde{x},\vec{n}}(\vec{w},\vec{v})\cos(\theta)\int_{H'} dw'\ f_{\tilde{x}',\vec{n}'}(\vec{w}',\vec{w})\cos(\theta')L^e(\vec{w}',\tilde{x}') \\
&= \int_H dw\ \int_{H'} dw'\ f_{\tilde{x},\vec{n}}(\vec{w},\vec{v})\cos(\theta)\ f_{\tilde{x}',\vec{n}'}(\vec{w}',\vec{w})\cos(\theta')\ L^e(\vec{w}',\tilde{x}')
\end{aligned}$$

In this expression, $\tilde{x}'$ is the point first hit along the ray $(\tilde{x},-\vec{w})$. At the intersection point, $\vec{n}'$ is the normal, H' is the upper hemisphere and $\vec{w}'$ is an incoming direction, making an angle of θ' with $\vec{n}'$. See Figure 21.9.

Once computed, we can add together $L^e(\tilde{x},\vec{v}) + L^1(\tilde{x},\vec{v}) + L^2(\tilde{x},\vec{v})$ and use this as the point's observed color at the image plane.

As suggested by the second line in the above equation, one way to compute L^2 is by recursively evaluating these nested integrals using distribution ray tracing. That is, an outer loop integrates the hemisphere above $\tilde{x}$. For each sampled direction $\vec{w}$, we hit some point, $\tilde{x}'$. We then use a recursive distribution ray tracing call to integrate the hemisphere above it. See Figure 21.9.

Alternatively, as suggested by the third line in the above equation, there are also

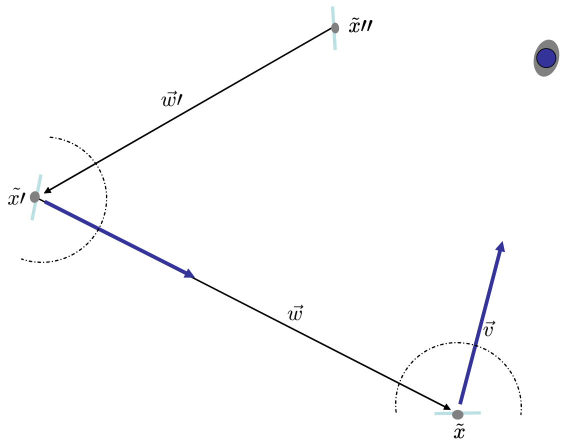


Figure 21.10: The integral that computes L^2 can be thought of as summing over paths of length two.

other ways to organize this integration. Let us denote by $\tilde{x}''$ the point first hit along the ray $(\tilde{x}', -\vec{w}')$. Then, in the integral, each setting of the variables $(\vec{w}, \vec{w'})$ corresponds to a *geometric path of length two*: $(\tilde{x}, \tilde{x}', \tilde{x}'')$. We can think of this integrand as calculating the light emitted at $\tilde{x}''$, reflected at $\tilde{x}'$, and then reflected at $\tilde{x}$ out towards $\vec{v}$. As such, it is often convenient to think of this not as a nested integral over two hemispheres, but as an integral over an appropriate space of paths. This leads to an integration algorithm known as path tracing [71].

This second bounce light L^2 is less important than direct lighting, but it is needed to properly simulate blurry reflections of the surrounding environment. See Figure 21.11. It also produces the less visible effect called color bleeding (see again Figure 21.11). Caustic effects can also be seen due to L^2. In this case, light bounces off mirrored or refracting objects and creates bright spots on diffuse surfaces. This light then bounces off towards the eye. See Figure 21.12.

21.3.4 And So On

In the real world, light bounces around the environment many many times. Thus, the *total* observed light L^t is the sum of light that has come from emitters and bounced **any** number of times.

$$L^t = L^e + L^1 + L^2 + L^3 + ...$$

Some light is absorbed at each bounce, making the higher bounce terms become small, and the sum convergent.

Higher bounces account for the overall distributions of lightness and darkness in

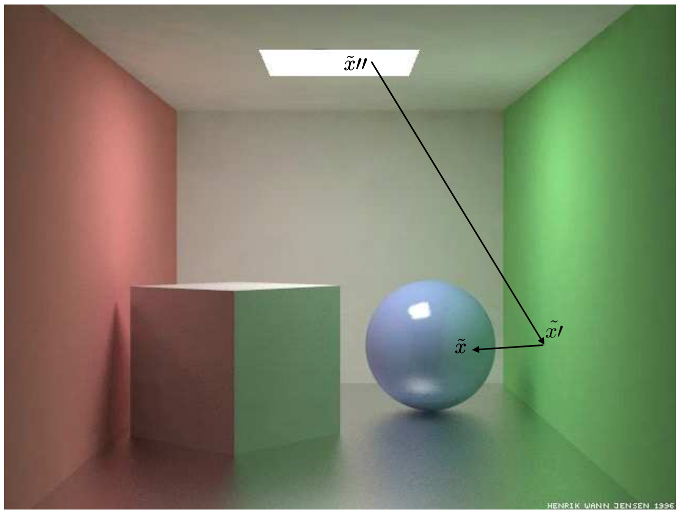


Figure 21.11: Second bounce light L^2 accounts for the blurry reflection in the glossy floor of the sphere as well as the color bleeding from the walls onto the ceiling. One path of length two is shown. From [29], ©Springer.

the environment. See Figure 21.13. In most cases, this can be done with low accuracy and at a low spatial resolution.

In software rendering, such integrals can be computed using sampling and summing. Successful methods here include distribution ray tracing [13], path tracing [71], the "Radiance" algorithm [75] and photon mapping [30].

In OpenGL, most of these effects are simply hacked using a combination of multiple pass rendering and precomputed textures. One popular such technique is called "ambient occlusion" [39].

21.3.5 The Rendering Equation (optional)

Instead of thinking of L^t as an infinite sum of L^i, we can also think of L^t as the solution to the so-called *rendering equation*. This point of view can ultimately lead to other insights and algorithmic approaches to light simulation. We include it here, as it is interesting in its own right.

Let us begin by writing the reflection equation in shorthand as

$$L^1 = \mathcal{B}L^e$$

where $\mathcal{B}$ is the bounce operator mapping light distributions to light distributions. More generally, we can use the reflection equation and bounce operator to write $L^{i+1} = \mathcal{B}L^i$.

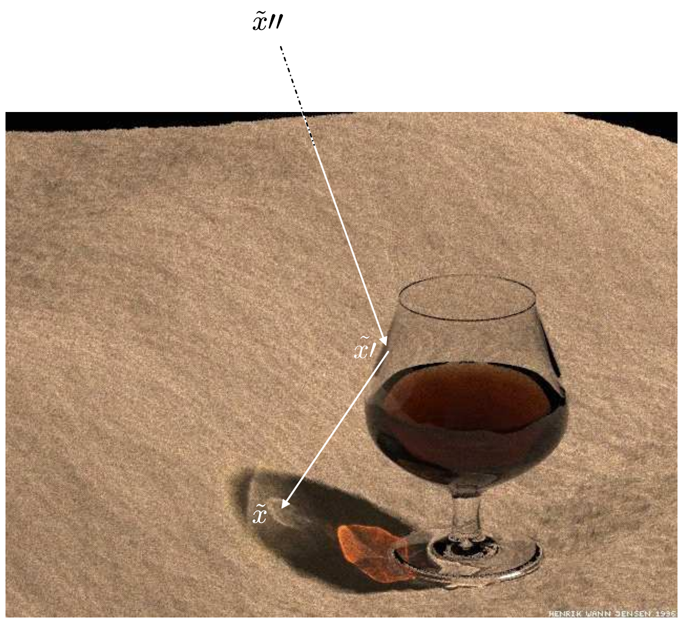


Figure 21.12: Second bounce light can also create the caustic effects seen on the ground. One path of length two is shown. From [29], ©Springer.

Putting this together, we can write

$$\begin{aligned} L^t &= L^e + L^1 + L^2 + L^3 + ... \\ &= L^e + \mathcal{B}(L^e + L^1 + L^2 + L^3 + ...) \\ &= L^e + \mathcal{B}L^t \end{aligned}$$

This expresses an equation that must hold for the total equilibrium distribution L^t.

At a surface point, this can be expanded out as

$$L^t(\tilde{x}, \vec{v}) = L^e(\tilde{x}, \vec{v}) + \int_H dw\ f_{\tilde{x},\vec{n}}(\vec{w}, \vec{v})\ L^t(\vec{w}, \tilde{x}) \cos(\theta)$$

This last form is called the rendering equation. Note that L^t appears on both sides of the equation, so it is not simply a definite integral, but is an *integral equation*.

21.4 Sensors

When we place a (virtual) camera in the scene, an image is captured of the light distribution L^t, the total equilibrium distribution of light in the scene. For a pinhole camera,

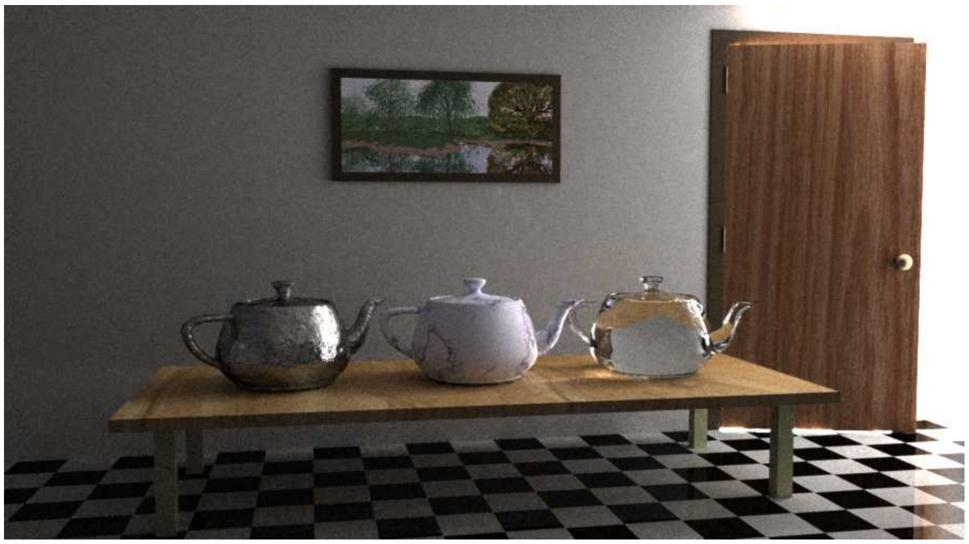


Figure 21.13: By computing multiple bounces, we can compute the correct distribution of light and darkness in an environment. From [72], ©ACM.

we simply capture the incoming radiance at each pixel/sample location along the single ray from the pixel towards the pinhole. In a physical camera (or simulation of one) we need a finite aperture and finite shutter speed in order to capture a finite amount of photons at our sensor plane. See Figure 21.14. Given such a camera, we can model the photon count at pixel (i, j) as

$$\int_T dt \int_{\Omega_{i,j}} dA \int_W dw \; F_{i,j}(\tilde{x}) \; L^t(\vec{w}, \tilde{x}) \cos(\theta) \tag{21.5}$$

where T is the duration of the shutter, $\Omega_{i,j}$ is the spatial support of pixel (i, j)'s sensor, and $F_{i,j}$ is the spatial sensitivity of pixel (i, j) at the film point $\tilde{x}$ and W is the wedge of vectors coming in from the aperture towards the film point.

To organize the light, a lens is placed in front of the aperture. The simplest lens model is called the *thin lens* model. Its geometry is visualized in Figure 21.15. The effect of this lens is to keep objects at a particular depth plane in focus. Objects at other depths appear blurred out, due to the $\int_W$ operation.

We have already seen (Section 16.3) that, integration over the pixel area domain produces anti-aliasing. Integration over the shutter duration produces motion blur. See Figure 21.16. Integration over the aperture produces focus and blur effects, also called *depth of field* effects. See Figure 21.17.

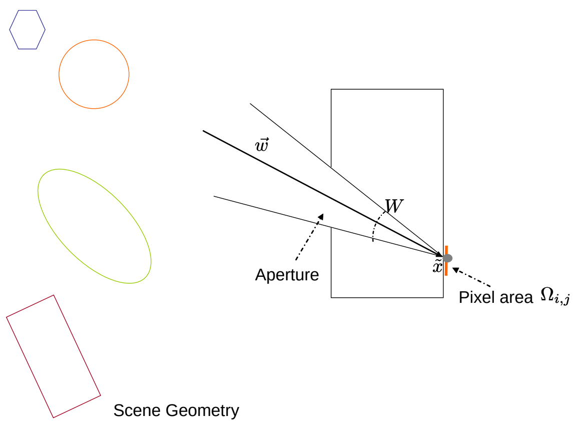


Figure 21.14: In a camera, we must integrate over the aperture and pixel footprint.

21.5 Integration Algorithms

As we have seen, starting from L^e, which is part of the scene definition, the computation of reflected light, L^1, and especially the total equilibrium distribution, L^t, requires the computation of (nested) definite integrals. Moreover, the computation of each pixel value in our sensor requires its own integrals as well. Integral computations are typically approximated by turning them into sums over some set of samples of the integrand.

Much of the work in the photo-realistic rendering field is all about the best ways to approximate these integrals. Key ideas for computing these efficiently include:

- Use randomness to choose the samples [13]. This avoids noticeable patterns in the errors during approximation. Using randomness, we can also use expectation arguments to argue about the correctness of the method.
- Reuse as much computation as possible [75, 29]. If we know the irradiance pattern at a point, perhaps we can share this data with nearby points.
- Do more work where it will have the most effect on the output. For example, it may not be worth it to follow rays of light that don't carry much radiance [71].

Possibly the most important lesson to keep in mind is that there is a certain duality at work here: one the one hand more integrals means more work. But on the other hand, each of the integrals is typically some kind of blurring operation. Thus, more integrals means less accuracy is needed. For efficiency, we should not spend too much effort on computing details that will get blurred out and never impact the final image. For example, we should typically spend more time on calculating direct illumination, and less time on indirect lighting.

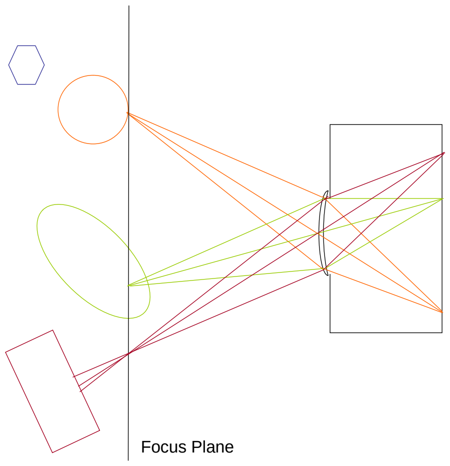


Figure 21.15: Here we show the effect of placing a thin lens in front of the aperture. It has the effect of focusing rays at a preferred depth plane. Objects in front of, or behind this plane are out of focus and blurred.

21.6 More General Effects

There are other optical effects that we have not captured in our simple model of light and reflection. Atmospheric volumetric scattering occurs when light passes through fog. Fluorescence occurs when surfaces absorb light and later re-emit this energy out (often at different wavelengths). Polarization and diffraction effects can occasionally be observed as well.

One interesting effect that turns out to be somewhat important is subsurface scattering. In this case, light enters a material at one point, bounces around inside of the surface, and comes out over a finite region around the point of entrance. This gives an overall softening effect and can be important to properly model surfaces such as skin and marble. See Figure 21.18.

Exercises

Ex. 55 — Given a diffuse wall with constant irradiance from the incoming hemisphere over all points, what is the distribution of outgoing radiance?

Ex. 56 — If we observe the above wall with a finite sensor, and compute $\Phi(W, X)$, how will the flux depend on the distance between the sensor and the wall. What about its dependence on angle?

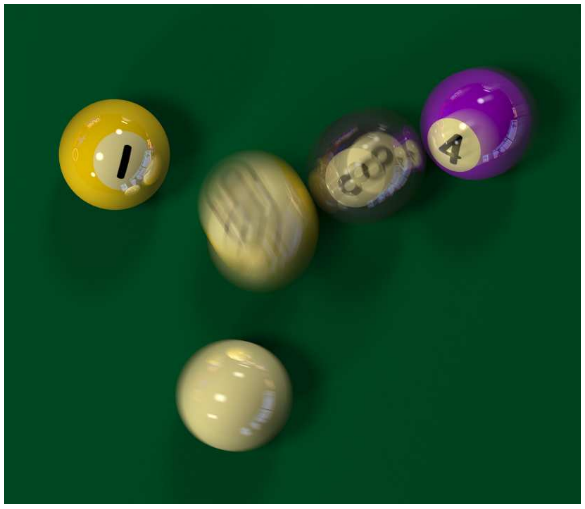


Figure 21.16: One of the first published images rendered using lots of rays per pixel. From [13], ©Pixar.

Ex. 57 — Starting with a ray tracer, use multiple rays to render depth of field effects, or soft shadows from area light sources.

Ex. 58 — Learn about and photon mapping. This is an algorithm which generates photons at the light sources, and sends them out along rays through the scene. The absorbed photons are stored in a *kd-tree* spatial data structure and later used to determine the colors of observed surface points.

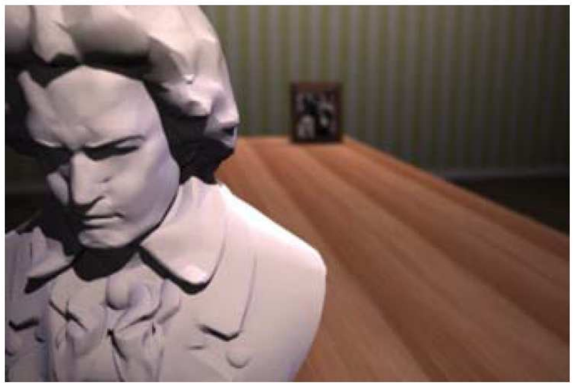


Figure 21.17: Integration over the lens creates focus effects. From [38], ©ACM.

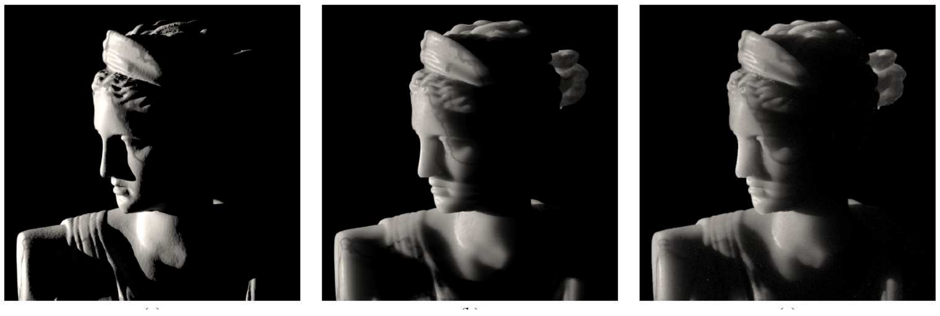


Figure 21.18: From left to right more and more subsurface scattering is used. This can give scale cues, making the rightmost figure look smallest. From [31], ©ACM.

Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012