Rendering Equation

1 Overview

The rendering equation attempts to describe the energy transport that happens by means of visible light. It neglects the wave nature of light (at least in the form discussed here), but is considered a sufficient approximation for computer graphics purposes. The rendering equation is based directly on physics and does not include any non-physics-based 'hacks', like the ambient light term we used in the ray tracing project to make up for multiply reflected light. The rendering equation, in particular, describes multiple light reflections.

2 Setting

The world we want to render will contain several objects. Some of them will be light sources. Light sources will emit some light but otherwise behave like all other objects (in particular, they will reflect light too). The rendering equation will 'simply' say that

$$L(x;\vec{\omega}) = L_e(x;\vec{\omega}) + L_r(x;\vec{\omega}),$$

i.e. that the amount of light leaving a point x in the direction of the vector $\vec{\omega}$ can be obtained by adding the emitted (L_e) light and the reflected light (L_r) leaving in the same direction. The emitted light will generally be zero for points that are not on light sources. Notice that since we are allowed to specify the amount of light leaving in every direction, we can do a lot more than just point light sources emitting the same amount of light in all directions (as we did so far). By sending more light in some directions than others, we can model spot lights, car headlights etc. This is a very flexible model.

3 Radiance: How to measure amount of light?

What is 'amount of light'? We are going to think of it as *radiance*. The idea behind radiance is to measure the 'amount of radiative energy' that travels along a single ray. However, this is not as simple as it seems: a ray is infinitely thin so it alone cannot carry any nonzero energy. Fortunately, there is calculus and derivatives.

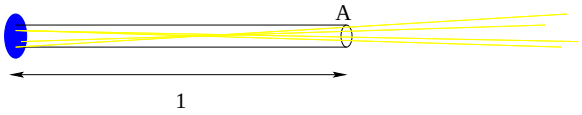

Figure 1: Device for measuring radiance

Intuitively, a 'device' for measuring radiance is a cylinder of unit length built of an ideally black material that absorbs all the light. On one side of the cylinder we have an energy meter that measures the radiative energy that passes through the opening on the other end of the cylinder and hits the meter directly (the yellow lines indicate some paths of photons that will contribute to the measured energy). Direct hit is important - remember that the cylinder is black, so all photons that hit it will immediately die (their energy being turned into heat). The measured amount of energy will decrease quadratically as the area A of the cylinder's base decreases to zero (example: if you decrease A by a factor of two, there are twice fewer rays that hit the meter due to it's smaller area; but also twice fewer rays will fit into the opening on the other side of the cylinder... total decrease of energy is therefore 4-fold). Thus, an interesting physical quantity we are going to call radiance is the second derivative with respect to A:

$$L = \frac{\partial^2 E}{\partial A^2}.$$

One of the most important properties of radiance is that it is *constant along all rays* which don't contain obstacles or scattering particles (e.g. fog or dust). Of course, only vaccum satisfies this property fully, but in practice this works very well for air as well.

For most rasonable cases, radiance is adds physical meaning to what we losely called 'intensity of light comming from some direction' so far. In particular, rendering can be viewed as computing the radiances along rays directed towards the viewpoint and passing through the pixels.

4 Modeling and Representing Surface Reflection

The reflection model we used earlier is too simplistic to simulate complex reflection patterns present in the so-called real world. In general, surfaces like cloth, skin, polished wood etc. are impossible to model too well using just the ambient, specular and diffuse lighting terms (even if textures are used to specify the variation of color).

Scattering of light by a surface is described by the following equation:

$$L_r(x, \vec{\omega}) = \int_\Omega f_r(x; \vec{\omega}', \vec{\omega})(\vec{n} \cdot \vec{\omega}') L_i(x; \vec{\omega}') d\vec{\omega}',$$

in which:

* $\vec{n}$ is the normal vector at x

* $\vec{\omega}$ is an outgoing direction

* $\vec{\omega}'$ (the variable over which we integrate) is an incoming direction

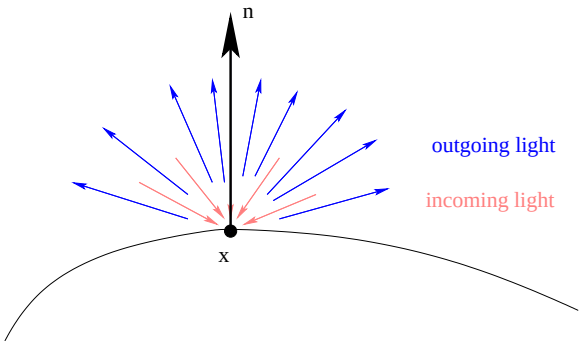

Figure 2: Generally, light hits a point x from all directions and is reflected (scattered) in all directions.

* Ω is the set of all incoming directions; can be thought as a halph-sphere (endpoints of unit vectors that start at x and are directed towards the empty space)

* $L_i(x; \vec{\omega}')$ is the radiance incoming from the direction of $\vec{\omega}'$

* $f_r(x; \vec{\omega}', \vec{\omega})$ is the BRDF (*Bidirectional Reflectance Distribution Function*). Its value is, essentially, the fraction of radiative energy arriving x from the direction of $\vec{\omega}'$ that is scattered in the direction of $\vec{\omega}$.

Why the $(\vec{n} \cdot \vec{\omega}')$ term? All the vectors here are unit, so this is the cosine of the angle of the incoming direction with the normal vector. Theoretically, it can be 'swallowed' by the BRDF but it's convenient to have it separate: basically, it makes up for the increase of area for light hitting the surface at an angle (figure 3). For a beam of (perpendicular) cross-section A hitting a surface from a direction $\vec{\omega}'$, the area of at which the surface intersects the beam is proportional to A and the inverse of the cosine between $\vec{\omega}'$ and the normal vector $\vec{n}$. Therefore, the energy *per unit area of the surface* is proportional to the energy carried by the beam and to $(\vec{n} \cdot \vec{\omega}')$.

4.1 Diffuse Surfaces

For diffuse surfaces, the energy is scattered uniformly in all directions. The BRDF is, in this case, simply a constant. This reduces the scattering equation to

$$L_r(x, \vec{\omega}) = K * \int_{\Omega} (\vec{n} \cdot \vec{\omega}') L_i(x; \vec{\omega}') d\vec{\omega}',$$

which is consistent with our diffuse reflection model (cf project 1): it states that the energy arriving at x from the direction of $\vec{\omega}'$ contributes to the reflected light (i.e. brightness of the point, intensity) an amount proportional to that energy and to the cosine of the angle between the incoming and normal directions.

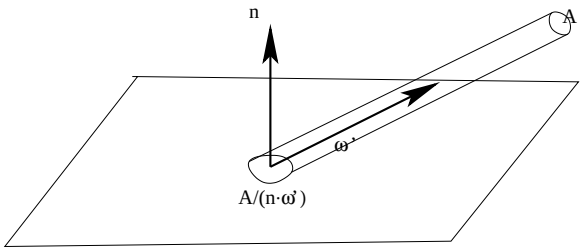

Figure 3: Area of the intersection of a beam with a surface is inversely proportional to the cosine of the angle between the incoming direction and the normal.

5 Global Radiative Transport Equation

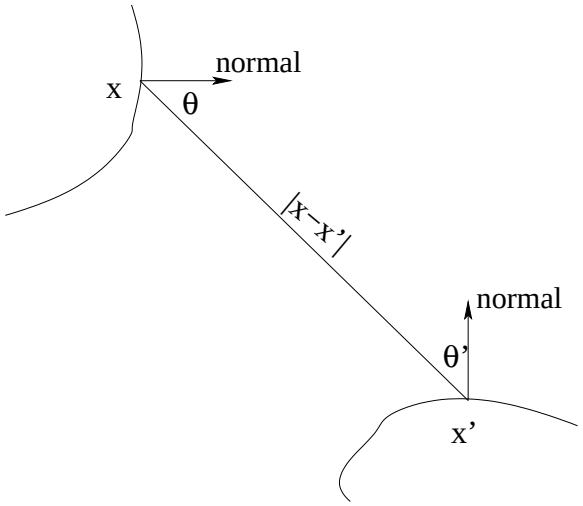

Figure 4: Geometry term

Basically, when you turn on the light, energy starts flowing between the surfaces until the whole system achieves the equilibrium state. The equation that governs this process, describing the energy transport that happens by means of visible light is called the rendering equation and solving it is the ultimate goal of computer graphics, or at least its area called (for obvious reasons) photorealistic rendering. The equation is based entirely on physics and, even though it neglects certain aspects of light (e.g. its wave nature), it is considered sufficiently accurate for computer graphics (at least in vast majority of cases). Here is one

of the forms of the rendering equation

$$L(x' \to x'') = V(x', x'')\Big(L_e(x' \to x'') + \int_S f_r(x', x \to x', x' \to x'')G(x, x')L(x \to x')dx\Big).$$

By $p \to q$ we denote the direction from a point p to another point q. $L(p \to q)$ is the amount of radiance that leaves p and reaches q. S is the union of all surfaces in the world. $V(x', x'')$ is the *visibility term*, one if the points x' and x'' can see each other and zero otherwise. It ensures that if the points don't see each other then the radiance flowing between them is zero. f_r is the BRDF and $G(x, x')$ is the *geometry term*, encapsulating both the attenuation (energy gets disperses proportionally to squared distance) and the adjustment needed for areas as described in Section 4 (Figure 4).

$$G(x, x') = \frac{\cos\theta \cos\theta'}{|x - x'|^2}.$$

The rendering equation basically states that the light leaving a surface point and reaches another (if the two can see each other) equals the emited light plus the scattered light, using the scattering equation of Section 4 to describe the second term. The only difference is a 'change of variables': we don't integrate over all incoming directions, but instead over all points on the surfaces (i.e. points from which the light might potentially be comming).

6 Iterative approach to solving the rendering equation

The 'unknowns' of the rendering equation are the radiances flowing between the points on the surfaces of the world. L can be thought as a function that assigns a radiance to a pair of such point. The right hand side of the rendering equation is an *operator*, or a rule that allows to take one assignment of radiance to pairs of points and compute another one. For equations of this kind, one can start from an 'initial guess' and then apply the rule iteratively hoping that the whole process will converge. This does not always work, but the rendering equation is a case when this is actually quite effective (although this would be a very inefficient solution in terms of accuracy and running time - path tracing, radiosity and photon mapping are the most commonly used approaches in photorealistic rendering). The reason for this is that the iterations cause more and more reflections of light to be included, and the energy tends to decay quite fast with the number of reflections in a typical environment. This makes the iteration converge very well.

6.1 A trivial example

Consider the equation

$$x = .5x + 1.$$

Here, the unknown is x, just a number, and the operator is the function

$$f(x) = .5x + 1.$$

To solve the equation we can start from initial guess, say 0 and then iterate f on it. This gives the following sequence

$$0, f(0) = 1, f(1) = 1.5, f(1.5) = 1.75, f(1.75) = 1.875, \ldots$$

If you keep iterating, you will be getting closer and closer to the true solution, i.e. 2. Of course, the rendering equation is much more complex (in fact it has infinitely many variables, in a way - the variables are radiances for all of the infinitely many directions; it is also impossible to solve it explicitely in any interesting case), but in fact it's not that much different from the simple example described here.

6.2 Iteration 1

Let's say that our first guess for the solution of the rendering equation is zero radiance for each pair of points. If you plug in $L \equiv 0$ to the right hand side, the equation would reduce to

$$L(x' \to x'') = V(x', x'')L_e(x' \to x'').$$

Using this L for rendering would lead to an image showing only the light sources (which have nonzero L_e) with correctly resolved visibility (the V term takes care of it). No light reflection will happen. The only way objects that do not emit light will 'show' is as black spots obscuring light sources.

6.3 Iteration 2

We substitute L from the previous iteration to the right hand side of the rendering equation. This yields:

$$L(x' \to x'') = V(x', x'')\Big(L_e(x' \to x'') + \int_S f_r(x', x \to x', x' \to x'')G(x, x')V(x, x')L_e(x \to x')dx\Big).$$

Using this L means simulating just one scattering. Objects that are not hit directly by light emitted by one of the light sources are completely dark. An image you would obtain would be a lot like the one drawn by the ray tracer you implemented for the first project, except for that it would not contain the 'hack' of ambient light.

6.4 More iterations

Generally, iterating more times means simulating more and more light scattering events. In iteration 3, we would obtain an image with objects lit by both the light comming directly from the light sources and light which is scattered just once. Iteration 4 would include illumination caused by light scattered twice and so on.