

Lecture 17: Recursive Ray Tracing

Where is the way where light dwelleth? Job 38:19

1. Raster Graphics

Typical graphics terminals today are raster displays. A raster display renders a picture scan line by scan line. A *scan line* is a horizontal line consisting of many small, tightly packed dots called *pixels*. To display a scene using a raster display, we need to compute the color and intensity of the light at each pixel. The purpose of this lecture is to explain how to compute the color and intensity of light for each pixel given a 3-dimensional model of the objects in the scene and the location, color, and intensity of the light sources.

2. Recursive Ray Tracing

For realistic rendering, we need the following setup: an eye point E , a viewing screen S , and one or more point light sources with known location, color, and intensity (see Figure 1). We also need to specify the locations and the shapes of all the objects in the scene, together with their reflectivity, transparency, and indices of refraction. Our goal is to display the 2-dimensional projection onto the screen of the 3-dimensional scene as viewed from the eye point, much like an artist painting the scene or a still photographer capturing a vista. We shall accomplish this task by tracing light rays from the scene back to the eye. Below is the basic ray tracing algorithm.

Ray Tracing Algorithm

For each pixel:

- i. Find all the intersections of the ray from the eye to the pixel with every object in the scene.
- ii. Keep the intersection closest to the eye.
- iii. Compute the color and intensity of the light at this intersection point.

Render the scene by displaying the color and intensity at each pixel.

Step i of the ray tracing algorithm requires us to compute the intersections of a straight line with lots of different surfaces. The difficulty of this intersection problem depends on the complexity of the surfaces in the scene. For planar surfaces these intersections are easy to compute: we simply solve one linear equation in one unknown. But for more complicated surfaces, we shall need more sophisticated tools. We will study line-surface intersection algorithms for a variety of different surfaces in Chapters 18 and 19.

Step ii of the intersection algorithm is straightforward; in this chapter we shall concentrate on Step iii of the algorithm.

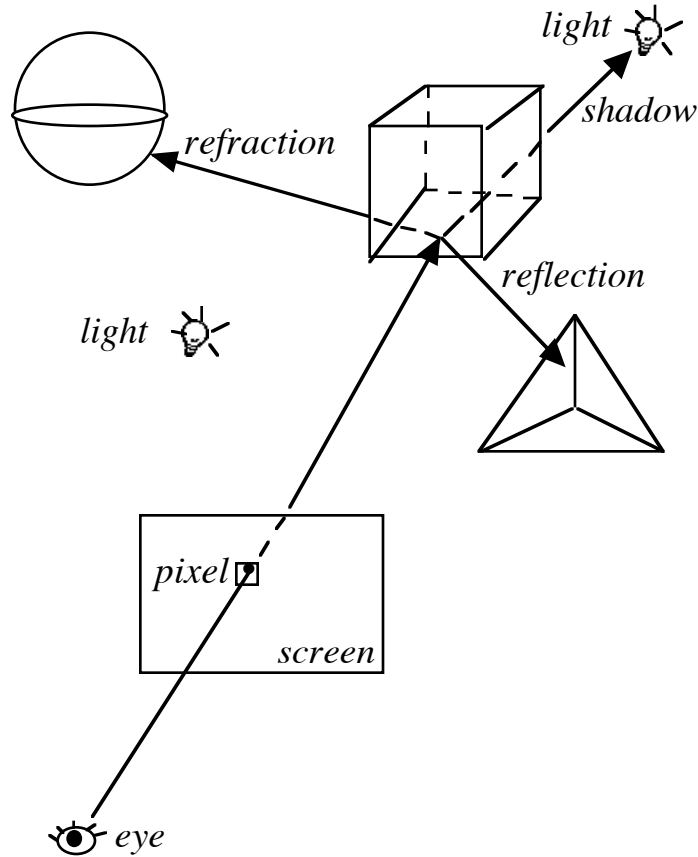


Figure 1: To determine the color and intensity at a pixel, a ray is cast from the eye point through the pixel. The color and intensity of the point where the ray first strikes a surface is computed by summing the contributions from the direct illumination of the unobstructed light sources and the contributions of reflected and refracted rays.

To find the color and intensity of the light at a surface point, we need to consider not only the light shining directly on the surface from the light sources, but also the light reflected and refracted from these sources off of other surfaces. Thus the total intensity I at a surface point is given by

$$I = I_{direct} + k_s I_{reflected} + k_t I_{refracted} \tag{2.1}$$

where, as we observed in Chapter 16,

$$I_{direct} = I_{ambient} + I_{diffuse} + I_{specular} \tag{2.2}$$

and $0 \leq k_s, k_t \leq 1$ are the reflection and refraction (transparency) coefficients. These coefficients depend on the physical characteristics of the surface, which can be determined experimentally or simply set by the programmer. Again the light reflected and refracted from other surfaces depends on the light shining directly on these surfaces as well as on the light reflected and refracted onto these surfaces from other surfaces. These observations lead to the binary light tree depicted in Figure 2.

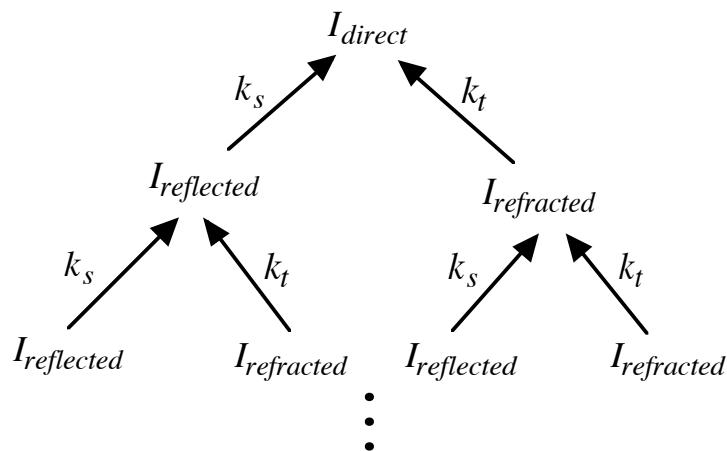


Figure 2: Binary light tree. The intensity of light at a surface point (apex) is the sum (depicted here by arrows) of the intensities of the light from the point light sources shining directly on the surface and the intensity of the light reflected and refracted off of other surfaces. The intensity of the light reflected and refracted from other surfaces also depends on the light shining directly on these surfaces and the light reflected and refracted to these surfaces from other surfaces. This recursive computation is summarized graphically in the binary light tree depicted above. This tree must be truncated at a prespecified depth in order to avoid infinite recursion.

Notice that for each object in the scene, we must decide whether it is reflecting, transparent, semi-transparent, or opaque -- that is, we need to assign values for reflectivity (k_s) and transparency (k_t). Moreover for transparent or semi-transparent objects, we need to assign an index of refraction (c_i). For opaque objects, $c_i = 0$; for air, $c_i = 1$. Note too that for transparent objects $I_{direct} = 0$.

3. Shadows

Just as we may not see every object in a scene because some objects are hidden from view by other objects closer to the eye, so too not every object in the scene is necessarily illuminated by every light source in the scene because some objects may lie in the shadow of other objects. That is, if an opaque object lies between a light source and a surface, then the surface will not be illuminated by the light source.

To determine whether or not a surface point is illuminated by a point light source, we treat the surface point as a virtual eye and we cast a virtual ray from the surface point to the point light source. As in the ray casting algorithm, we compute all the intersections of this virtual ray with every object in the scene. If the virtual ray intersects an opaque object before the virtual ray hits the

light source, then the surface point lies in shadow; otherwise the surface point is illuminated by the light source. Below we summarize this basic shadow casting algorithm.

Shadow Casting Algorithm

For each surface point visible to the eye, cast a *virtual ray* to each light source.

- i. If the virtual ray hits an opaque object before the virtual ray hits a light source, then omit the contribution of this light source. That is, for this light source, $I_{diffuse} = I_{specular} = 0$.
- ii. If the virtual ray hits a transparent or semi-transparent object before the virtual ray hits the light source, then scale the contribution of this light source and continue to look for further intersections.

Note that in step i, an object may be self shadowed -- that is, a point on an object (e.g. a sphere) may lie in the object's own shadow -- so we must compute intersections of the virtual ray even with the object containing the point of interest.

Since we need to compute the intersections of the virtual ray with every object in the scene, shadow computations can be quite time consuming. To speed up shadow computations, we can take advantage of *shadow coherence*. The idea behind shadow coherence is that if a point P lies in the shadow of an object O , then a nearby point Q is also likely to lie in the shadow of the same object O . Therefore to speed up the shadow calculations, we store the current shadowing object and we test first for intersections with this object when we cast the virtual ray from the next point in the scene.

Shadow rays are not refracted; if shadow rays were refracted, these rays would not hit the light source! We could solve this problem by casting rays starting from the light sources instead of from the surface points taking refraction into account, but this approach is very expensive and not generally done. Typically this problem is ignored, since the resulting scenes still appear realistic.

4. Reflection

To find the color and intensity of the light at a surface point, we need to calculate the light reflected off this point from other surfaces in the scene. To compute the color and intensity of this reflected light, we consider the surface point as a virtual eye and we cast a secondary ray from the surface point treating the surface as a mirror (see Figure 3). As in the standard ray casting algorithm, we find all the intersections of this secondary ray with every object in the scene, keep the closest intersection point, and calculate the color and intensity of the light at this intersection point recursively. We then add a scaled value of this color and intensity to the color and intensity at the original point. Below we summarize this reflection algorithm.

Reflection Algorithm

For each visible point on a reflecting surface:

- i. Use the law of mirrors -- angle of incidence = angle of reflection -- to calculate a reflected *secondary ray* (see Figure 3).
- ii. Find all the intersections of this reflected secondary ray with every object in the scene.
- iii. Keep the intersection closest to the surface point.
- iv. Compute the color and intensity of the light at this intersection point recursively and add a scaled value of this contribution, $k_s I_{reflected}$, to the color and intensity at the original point.

To find the reflected secondary ray $R(t)$, let

$$V = \frac{E - P}{|E - P|}$$

denote the unit vector from the point P to the eye point E , and let V_{\parallel} and V_{\perp} denote the components of V parallel and perpendicular to the surface normal N at the point P (see Figure 3). Then

$$V = V_{\parallel} + V_{\perp},$$

so the reflected vector W is given by

$$W = V_{\parallel} - V_{\perp}.$$

But

$$V_{\parallel} = (V \cdot N)N$$

$$V_{\perp} = V - V_{\parallel} = V - (V \cdot N)N.$$

Hence

$$W = 2(V \cdot N)N - V$$

and

$$R(t) = P + tW.$$

Note that the calculation here of the reflected vector W is the same as the calculation of the reflected vector R for specular reflection (see Chapter 16, Section 5).

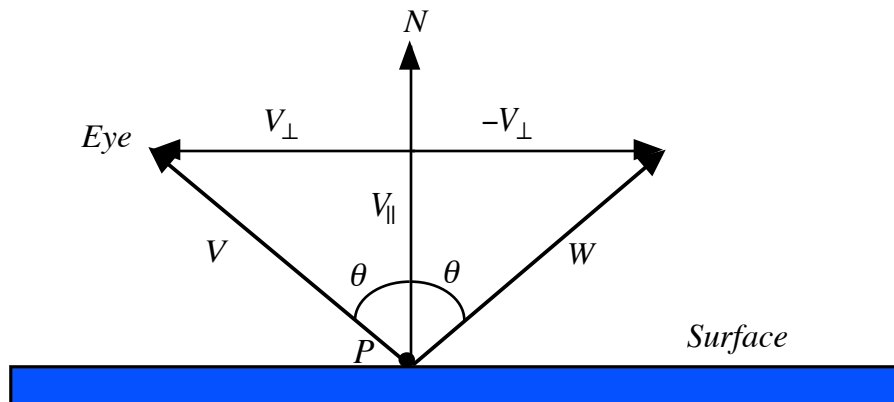


Figure 3: Law of the mirror: angle of incidence = angle of reflection. Thus if V is the unit vector from the point P to the Eye, then the reflected vector W is given by $W = V_{\parallel} - V_{\perp}$.

5. Refraction

To find the color and intensity of the light at a surface point, we need to calculate not only the light reflected off this point from other surfaces in the scene, but also for transparent or semi-transparent surfaces the light refracted through the surface at this point. Refraction is caused by light traveling at different speeds in different mediums: the denser the medium, the slower the speed of light. This change in speed gives the illusion of light bending when light passes between different mediums.

To compute the color and intensity of refracted light, we again consider the surface point as a virtual eye and we cast a secondary ray from the surface point treating the surface as a refracting surface (see Figure 4). As in the standard ray casting algorithm, we again find all the intersections of this secondary refracted ray with every object in the scene, keep the closest intersection point, and calculate the color and intensity of the light at this intersection point recursively. We then add a scaled value of this color and intensity to the color and intensity at the original point. Below we summarize this refraction algorithm.

Refraction Algorithm

For each visible point on a transparent object:

- i. Use Snell's Law to calculate the refracted *secondary ray* (see Figure 4).
- ii. Find all intersections of the refracted ray with every object in the scene.
- iii. Keep the intersection closest to the surface point.
- iv. Compute the color and intensity of the light at this intersection point recursively and add a scaled value of this contribution, $k_t I_{refracted}$, to the color and intensity at the original point.

To find the refracted secondary ray $R(t)$, again let

$$V = \frac{E - P}{|E - P|}$$

denote the unit vector from the point P to the eye point E , and let W denote the direction of the refracted ray (see Figure 4). To compute the refracted ray $R(t)$, we need to calculate the refracted vector W in terms of the known vectors N and V . To perform this computation, we shall apply the following rule from optics:

Snell's Law

Let θ_1 be the angle between the unit vector V and the surface normal N at the point P , and let θ_2 be the angle between the refracted vector W and the unit normal $-N$ at the point P pointing away from the eye. If the indices of refraction on either side of the surface are c_1 and c_2 , then

$$\frac{c_2}{c_1} = \frac{\sin(\theta_2)}{\sin(\theta_1)}. \quad (5.1)$$

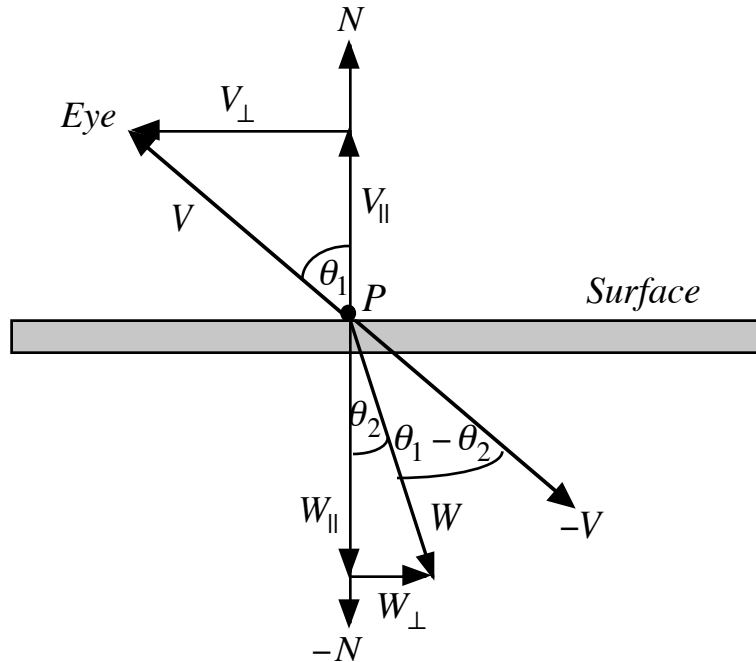


Figure 4: The unit vector V from the point P to the eye, the surface normal N at the point P , and the refracted vector W . Here θ_1 is the angle between the surface normal N and the vector V , and θ_2 is the angle between the refracted vector W and the vector $-N$. By Snell's Law if the indices of refraction on either side of the surface are c_1 and c_2 , then $\frac{c_2}{c_1} = \frac{\sin(\theta_2)}{\sin(\theta_1)}$.

There are several ways to find the refracted vector W in terms of the known vectors N and V . For example, we could rotate $-N$ about the vector $N \times V$ through the angle θ_2 and then apply Snell's Law to eliminate θ_2 (see Exercise 4). Alternatively, we could compute $W = \text{slerp}(-N, -V, \theta_2 / \theta_1)$, and again apply Snell's Law to eliminate θ_2 (see Exercise 3). Below we present a simpler method due to S. Schaefer that avoids both rotation and *slerp*.

Let V_{\parallel} and V_{\perp} denote the parallel and perpendicular components of V relative to N , and let W_{\parallel} and W_{\perp} denote the parallel and perpendicular components of W relative to $-N$ (see Figure 4). Then by construction

$$\begin{aligned} W_{\parallel} &\parallel -V_{\parallel} \\ W_{\perp} &\parallel -V_{\perp}. \end{aligned} \tag{5.2}$$

Moreover, since V and W unit vectors

$$\begin{aligned} |V_{\parallel}| &= \cos(\theta_1) & |V_{\perp}| &= \sin(\theta_1) \\ |W_{\parallel}| &= \cos(\theta_2) & |W_{\perp}| &= \sin(\theta_2). \end{aligned} \tag{5.3}$$

Therefore by Equation (5.2), (5.3) and Snell's Law (Equation (5.1))

$$W_{\parallel} = \frac{\cos(\theta_2)}{\cos(\theta_1)}(-V_{\parallel}) = -\frac{\cos(\theta_2)}{(V \cdot N)}V_{\parallel} \quad (5.4)$$

$$W_{\perp} = \frac{\sin(\theta_2)}{\sin(\theta_1)}(-V_{\perp}) = -\frac{c_2}{c_1}V_{\perp} . \quad (5.5)$$

But recall that

$$V_{\parallel} = (V \cdot N)N$$

$$V_{\perp} = V - (V \cdot N)N .$$

Substituting these formulas for V_{\parallel} and V_{\perp} into Equations (5.4) and (5.5) yields

$$W_{\parallel} = -\frac{\cos(\theta_2)}{(V \cdot N)}(V \cdot N)N = -\cos(\theta_2)N$$

$$W_{\perp} = -\frac{c_2}{c_1}(V - (V \cdot N)N) ,$$

so

$$W = W_{\parallel} + W_{\perp} = \left(\frac{c_2}{c_1}(V \cdot N) - \cos(\theta_2) \right) N - \frac{c_2}{c_1}V . \quad (5.6)$$

It remains only to solve for $\cos(\theta_2)$ in terms of N and V . We proceed by applying the trigonometric identity $\cos^2(\theta) + \sin^2(\theta) = 1$ together with Snell's Law:

$$\cos(\theta_2) = \sqrt{1 - \sin^2(\theta_2)} = \sqrt{1 - \frac{c_2^2}{c_1^2} \sin^2(\theta_1)} = \sqrt{1 - \frac{c_2^2}{c_1^2} (1 - \cos^2 \theta_1)} = \sqrt{1 - \left(\frac{c_2}{c_1} \right)^2 (1 - (N \cdot V)^2)} .$$

Thus we conclude that

$$W = \left(\frac{c_2}{c_1}(N \cdot V) - \cos(\theta_2) \right) N - \frac{c_2}{c_1}V , \quad (5.7)$$

where

$$\cos(\theta_2) = \sqrt{1 - \left(\frac{c_2}{c_1} \right)^2 (1 - (N \cdot V)^2)} . \quad (5.8)$$

The equation of the refracted ray is simply

$$R(t) = P + tW .$$

Notice that for each color, we use the same index of refraction and therefore the same refracted ray. Technically this procedure is incorrect, but this tactic saves tremendously on computation time and the results still look realistic, so most ray tracing programs adopt this approach.

There are some angles θ_1 between the vector V to the eye and the normal vector N to the surface at which refraction cannot occur. This phenomenon is called *total reflection*. Total reflection may occur when $c_2 / c_1 > 1$. In this case for some angles θ_1 ,

$$\frac{c_2}{c_1} \sin(\theta_1) > 1,$$

so by Snell's Law

$$\sin(\theta_2) > 1$$

which is impossible. To check for total reflection, test

$$\frac{c_2^2}{c_1^2} (I - (N \cdot V)^2) > 1 \Leftrightarrow I - (N \cdot V)^2 > \left(\frac{c_1}{c_2}\right)^2 \Leftrightarrow I > (N \cdot V)^2 + \left(\frac{c_1}{c_2}\right)^2.$$

In this case, $\cos(\theta_2)$ in Equation (5.7) is imaginary and the refracted vector W does not exist. Be careful to test for total reflection, otherwise your ray tracing program may crash.

6. Summary

Recursive ray tracing is one of the most powerful tools in Computer Graphics for generating realistic images of virtual scenes. Shadows, reflections, and refractions are all modeled effectively by recursively tracing the paths of light rays through the scene. In recursive ray tracing, the total intensity at each surface point is given by the following recursive computation:

Total Intensity

$$I = I_{direct} + k_s I_{reflected} + k_t I_{refracted} \quad 0 \leq k_s, k_t \leq 1$$

$$I_{direct} = I_{ambient} + I_{diffuse} + I_{specular}$$

To find the reflected and refracted rays at each surface point, let N represent the unit normal vector at the surface point and let V represent the unit vector from the surface point to the eye. In addition, let c_1 and c_2 denote the indices of refraction on opposite sides of the surface. Then we have the following formulas:

Reflected Vector

$$W = 2(V \cdot N)N - V$$

Refracted Vector

$$W = \left(\frac{c_2}{c_1} (N \cdot V) - \cos(\theta_2) \right) N - \frac{c_2}{c_1} V$$

$$\cos(\theta_2) = \sqrt{1 - \left(\frac{c_2}{c_1}\right)^2 (I - (N \cdot V)^2)}$$

Total Reflection

$$(N \cdot V)^2 + \left(\frac{c_1}{c_2}\right)^2 < 1$$

To implement recursive ray tracing, we need to be able to calculate two things: surface normals and line-surface intersections. We shall take up the calculation of these items for a variety of different surfaces in the next two chapters.

Exercises:

1. Show that the reflection vector W in Figure 3 can be computed in each of the following ways:
 - a. By finding the mirror image of the vector V in the plane perpendicular to V_{\perp} .
 - b. By rotating the vector V by 180° around the normal vector N .
 - c. By finding the mirror image of the vector V in the plane perpendicular to the normal vector N and then negating the result.
2. Using the result of Exercise 1, part c, show that light following the law of the mirror -- angle of incidence = angle of reflection -- takes the shortest path from the light source through the mirror to the eye.
3. Consider the refraction vector W in Figure 4.
 - a. Show that $W = \text{slerp}(-N, -V, t)$, for $t = \theta_2 / \theta_1$ (see Chapter 11, Section 6).
 - b. Use the result of part a together with Snell's Law to derive Equation (5.7) for the refraction vector W .
4. Consider the refraction vector W in Figure 4.
 - a. Show that W is the result of rotating the vector $-N$ through the angle θ_2 about the axis vector $\frac{N \times V}{|N \times V|}$.
 - b. Using part a, show that
$$W = \frac{\sin(\theta_2 - \theta_1)N - \sin(\theta_2)V}{\sin(\theta_1)}.$$
 - c. Use the result of part b together with Snell's Law to derive Equation (5.7) for the refraction vector W .