An Introduction to 3D Computer Graphics, Stereoscopic Image, and Animation in OpenGL and C/C++

Fore June

Chapter 16 Ray Tracing

16.1 Local vs. Global Illumination

The Phong lighting model that we discussed in the previous chapters is called a **local illumination** model. In the model, the shaded color or the illumination at a point of a surface depends purely on the local surface configuration such as the surface normal, the viewing direction, and the light direction. A light ray from a source only causes a point of the surface to glow; the reflected light ray has no effect on any other surface. Even if a surface is emissive (e.g. *glMaterialfv(GL_FRONT, GL_EMISSION, ..);*), it only glows itself but does not illuminate any other surface. Also, an object does not block the light source from illuminating another object and thus it won't cast any shadows. Figure 16-1 shows an example of illuminating two sphere by the same distant light source on the right. In reality, the sphere on the left should not be illuminated as the light is blocked by the right sphere. However, using the Phong model, the light source illuminates both spheres identically. Therefore, in situations like this, local illumination is a poor approximation to reality.

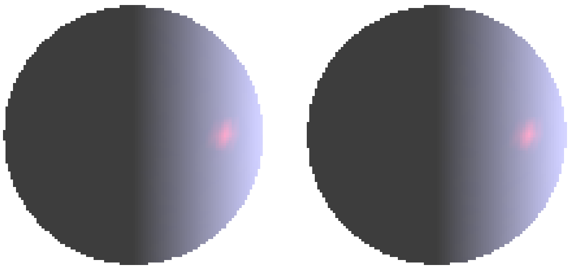


Figure 16-1 Two Spheres Illuminated by Same Source using Phong Model

Global illumination would make a lit scene look a lot more realistic. In a global illumination model, objects can block light to create shadows. A light ray reflected off a surface may illuminate other objects. Therefore, the image of an object may appear on a reflective surface. Also, emissive surfaces and refracted light rays may illuminate other objects. **Ray tracing** is one of the global illumination techniques; it eliminates the use of a depth buffer to determine hidden surfaces and allows for many special effects.

In summary, ray tracing unifies in one framework hidden surface removal, shadow computation, reflection of light, refraction of light, and global specular interaction. With all these advantages, ray tracing is a wonderful tool for creating sophisticated images. However, ray tracing is very computing intensive; a single image generated by ray tracing may take minutes, hours or even days to render. Despite the high computational costs, ray tracing has been widely used to create high quality and photorealistic images especially in animated movies. It is fortunate that each frame of an animated movie can be rendered independently by a different computer using ray tracing; it is common that hundreds of computers are used in parallel to produce a movie.

Another popular global illumination technique is **radiosity**. Ray tracing follows the paths of light rays around a 3-D scene to compute the illumination and shading on various surfaces. It assumes light sources to be point sources. On the other hand, radiosity simulates the diffuse propagation of light starting at the light sources, which may be broad surfaces.

16.2 Ray Tracing Concepts

To follow the paths of light rays, ray tracing makes use of the characteristics of light:

1. Light rays travel in straight lines.
2. Light rays do not interfere with each other even if they cross over.
3. Light paths are reversible. If a light ray can travel from point A to point B, then it is able to travel from B to A.

A straightforward way to do ray tracing is to follow light rays bounced off surfaces that will reach the viewplane as shown in Figure 16-2. However, as one can see from the figure, many of the rays originated from the light source may not hit the viewplane and thus cannot be seen by the viewer. Therefore, a lot of the computing work of following light paths is wasted.

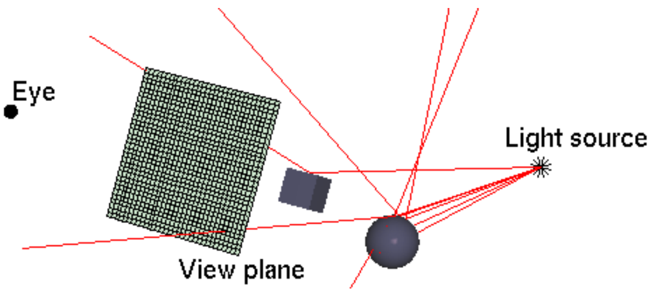


Figure 16-2 Tracing Rays From a Light Source

In practice, we only want to track those rays that can finally reach the eye. Since light paths are reversible, we can start tracing rays from the viewpoint, passing through viewplane pixels to the light source. You may ask: *Won't many of the rays miss the light source and thus the method is just as inefficient as the one before?* You are certainly right! Ray tracing actually does not work by tracing light paths only. It must work with a local lighting model, in which a light 'source' illuminates a surface like the Phong model. The task of ray tracing is to find out whether a surface that is visible to the viewer is illuminated by various light 'sources'. Visibility of surfaces can be found by throwing (or casting) light rays from the viewpoint, passing through pixels of the viewplane, into the scene as shown in Figure 16-3. This simple process is usually referred to as **ray casting**. If a ray hits an object, the eye will see the object at the intersection point, which is illuminated by the light source and the intensity value at the point is calculated according to the local lighting model. Furthermore, this point will be mapped to the pixel of the viewplane that the ray passes through.

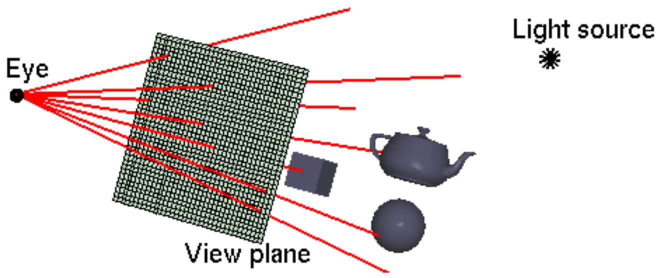


Figure 16-3 Casting Rays From Eye (Viewpoint)

Ray casting, which throws rays from the viewpoint to the scene to determine the visibility of surfaces, can be considered as a special case (the simplest case) of ray tracing. Conversely, we can consider ray tracing as an extension of ray casting. In ray casting, a light source is a point source that gives out light. In ray tracing, a light 'source' can be a point reflecting or transmitting a light ray incident on it. Therefore, ray tracing is a recursive process. It recursively casts from the points of intersections between light rays and object surfaces as shown in Figure 16-4 below.

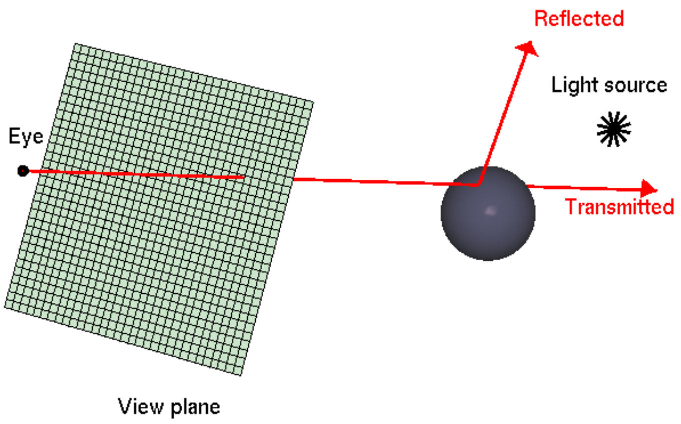


Figure 16-4 Recursive Ray Tracing

16.3 Basic Ray Tracing

16.3.1 Ray Casting (Nonrecursive Ray Tracing)

In this method, we determine the color of a pixel P on a viewplane by sending a ray from the eye through the pixel center to find out the first point where the ray hits an object in the scene. This first point of intersection is colored (shaded) according to a local lighting model such as the Phong model. This point is mapped to the pixel of the viewplane and is what the eye sees at that pixel. The following is the basic algorithm for ray casting:

```
for each pixel
{
  Shoot a ray from the eye through a pixel of viewplane.
  Find the first primitive intersection point by ray.
  Determine color at intersection point using local model.
  Draw color at the intersection point.
}
```

For example, Figure 16-5 below shows a ray that would intersect both the sphere and the lower cube. However, the ray hits the sphere first. So we will only see this point on the pixel. Note that using this simple ray casting model, we have not achieved any new visual effects as compared to the local lighting model. We have simply replaced the method of discarding hidden surfaces using a depth buffer by a ray tracing method to determine visible surfaces. To obtain more interesting visual effects, we must trace the paths recursively to include reflection rays, transmission rays and shadow feelers.

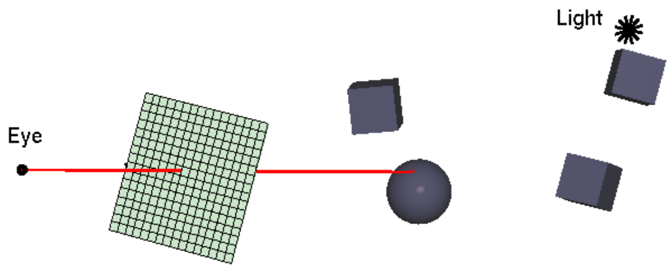


Figure 16-5 Determining First Intersection Point of Ray

16.3.2 Recursive Ray Tracing

In recursive ray tracing, the path tracing for a pixel deos not stop after the ray has hit a surface. We have to reflect the ray off the surface specularly (mirror-like) and continue the tracing along the reflected path. If the surface is that of a translucent media such as water, in addition to the reflected ray, we also have to shoot a ray through the surface along the refraction direction. The reflected and refracted ray are referred to as *secondary rays*. We carry on this process recursively until the secondary rays could 'see' the light source.

In ray tracing, there are several kinds of rays we need to consider, depending on the material properties of the scene.

Shadow Rays

A *shadow ray*, also called *shadow feeler*, is a ray sent from an intersection point to the light source to determine whether the intersection point is in shadow (the point is not visible from the light source) or not. As we discussed above, local light models like Phong model assume that every light source is always visible and no objects block light rays. Therefore, a local light model alone does not create shadows (see Figure 16-1). However, by casting rays into a scene and emitting shadow rays from the intersection points, we can create shadows in a scene from a simple local light model.

Figure 16-6 below shows an example of four shadow feelers shown as four broken lines. Four primary rays are sent from the eye through four pixels of the viewplane to determine four intersection points. From each of the intersection points, a shadow ray (shown as broken line) is sent to the light source. If the shadow ray hits an object before reaching the light source, then the intersection point is occluded and is in a shadow, not illuminated by the light. In the figure, two intersection points labeled “I” are illuminated and the other two intersecion points labeled “S” are shadowed.

Reflection Rays

When a light ray is bounced off a surface, the reflected ray may hit another surface and illuminates it. The reflected ray may be reflected again and illuminates other objects. To account for this effect, we add *reflection rays* in a ray tracing algorithm. When a ray intersects a surface and reflected, we treat the intersection point as the view point (eye) and follow the path of the reflected ray to continue the ray tracing. When this reflected ray hits a surface, we use a shadow ray discussed above to determine whether the hit point is shadowed or illuminated by a light source.

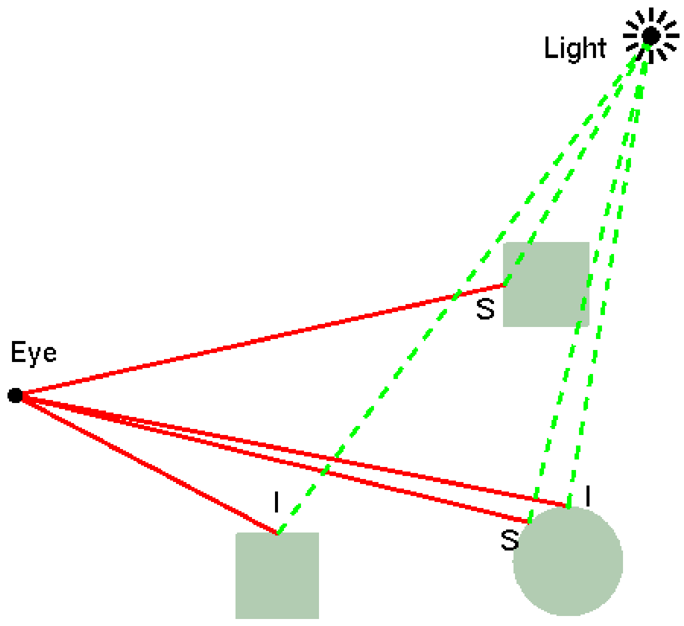


Figure 16-6 Shadow Feelers (S = shadowed, I = illuminated)

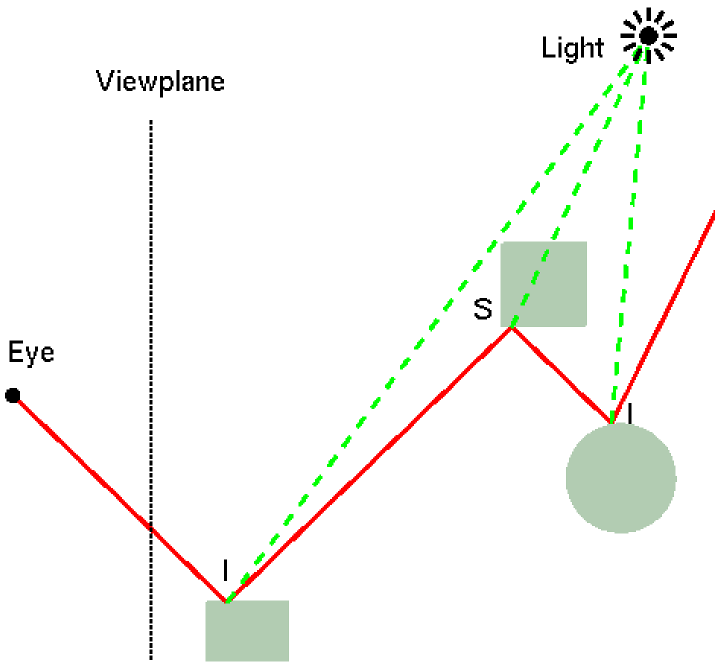


Figure 16-7 Reflection Rays

This process is carried on recursively as shown in Figure 16-7. Therefore, the illumination at a point is the sum of direct local illumination (if the point is not shadowed) from the local light model plus illumination due to reflection. This can be computed by a formula of the form

$$\mathbf{I} = \mathbf{I}_{local} + \rho_{rg}\mathbf{I}_r \tag{16.1}$$

where **I** denotes a color vector consisting of the red, green, and blue color components:

$$\mathbf{I} = \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{16.2}$$

The subscript g denotes "global" and r denotes reflection. The brightness (color vector) $\mathbf{I}_{local}$ is the illumination at the reflection point computed using a local lighting model such as Phong lighting. The brightness I_r is the illumination due to reflection from the reflected direction. It is computed recursively by Equation (16.1); at the very end (the reflection point on the sphere of Figure 16-7), there is no more reflection, and $I = I_{local}$. The reflection coefficient ρ_{rg}, depending on the material property of the surface, is the fraction of incident light from the reflection direction that will be reflected.

Transmission (Refracted) Rays

When a light ray passes through the boundary between two different media (such as air and glass), it is partially reflected and partially refracted. Part of the light is blent and transmitted through the media. By adding **transmission rays** in ray tracing, we can simulate transparency effects in a scene.

When a ray hits a translucent object, besides the reflection ray, a transmission ray is generated as shown in Figure 16-8.

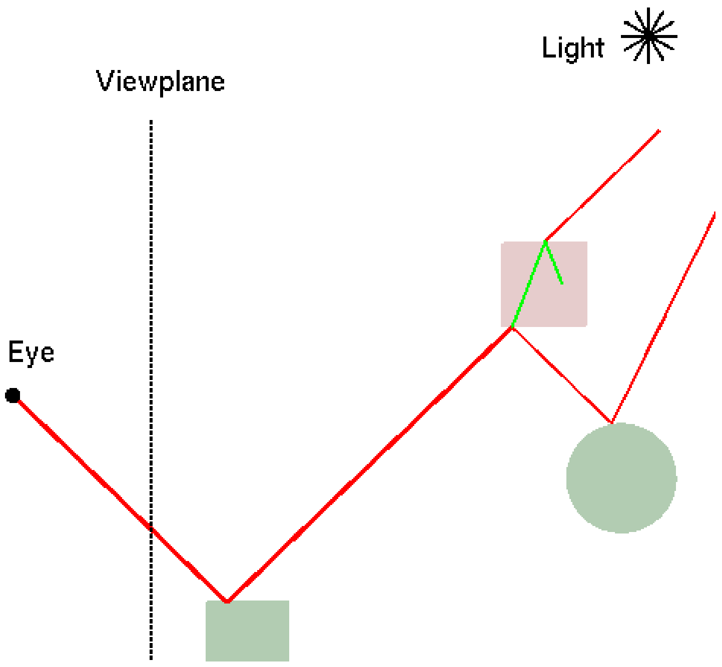


Figure 16-8 Transmission and Reflection Rays

The direction of the transmission ray is different from that of the original ray due to refraction. The degree of refraction is calculated using the the refraction indices of the media as discussed in the next section. When the transmission ray hits the other boundary of the object like that shown in Figure 16-8, it will be refracted and reflected again. Therefore, transmission rays also need to be traced recursively like the reflection rays.

When both transmission rays and reflected rays have been traced, the intensity (color vector) I at a point is calculated according to the formula

$$\mathbf{I} = \mathbf{I}_{local} + \rho_{rg}\mathbf{I}_r + \rho_{tg}\mathbf{I}_t \tag{16.3}$$

where the subscript t denotes transmission. The intensity $\mathbf{I}_t$ is the transmission illumination due to refraction in the transmission direction and is calculated recursively. The transmission coefficient ρ_{tg} is the fraction of light transmitted through the surface.

16.4 Global Intensity Calculation

16.4.1 Basic Ray Tracing Vectors

Figure 16-9 below shows the basic ray tracing vectors at an intersection point Q. In the figure we hava a surface between two media with different refractive indices η_v and η_t. The refractive index of a medium is the ratio of the speed of light in vacuum to the light speed in the medium. The two media could be air ($\eta \approx 1$), water ($\eta \approx 1.33$), glass ($\eta \approx 1.5$) or any other materials. It does not matter which refractive index, η_v or η_t, is greater. All that matters is that η_v is the refractive index of the medium the ray comes from (incident ray), and η_t is that of the medium it transmits (refractive ray).

We trace the path of a ray originated from the viewpoint, which intersects a surface at point Q. In the figure, this direction is $-\mathbf{v}$; we assume that this vector and others in the figure are normalized (i.e. magnitude=1). Besides v, vectors of reflection, transmission, and other related rays are shown in Figure 16-9:

$$\begin{array}{ll}
\mathbf{v} & = \text{ viewing vector from } Q \text{ to the viewpoint} \\
\mathbf{r_v} & = \text{direction of perfect reflection of tracing ray} \\
\mathbf{t} & = \text{perfect transmission direction of tracing ray} \\
\mathbf{L} & = \text{direction from } Q \text{ to the light source at same side as viewpoint} \\
\mathbf{L_t} & = \text{direction from } Q \text{ to the light source at opposite side of viewpoint} \\
\mathbf{r} & = \text{perfect reflection direction of light source vector } \mathbf{L} \\
\mathbf{n} & = \text{normal to the surface at } Q \\
|\mathbf{n}| & = |\mathbf{v}| = |\mathbf{r_v}| = |\mathbf{t}| = |\mathbf{L}| = |\mathbf{r}| = 1
\end{array} \tag{16.4}$$

We can split the direction vector in components orthogonal (perpendicular) and parallel to the surface at Q. We call these the normal component $\mathbf{b}_\perp$ and the tangential component $\mathbf{v}_{||}$ of a vector $\mathbf{v}$.

The normal component of **v** can be found by orthogonal projection of **v** on **n**:

$$\mathbf{v}_\perp = (\mathbf{v} \cdot \mathbf{n})\mathbf{n} = \cos\theta_v \mathbf{n} \tag{16.5}$$

The sum of the normal component $\mathbf{v}_\perp$ and tangential component $\mathbf{v}_{||}$ is equal to $\mathbf{v}$. Therefore, the difference between $\mathbf{v}$ and $\mathbf{v}_\perp$ gives the tangent componet:

$$\mathbf{v}_{||} = \mathbf{v} - \mathbf{v}_\perp = \mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n} \tag{16.6}$$

The dot product of $\mathbf{v}_\perp$ and $\mathbf{v}_{||}$ is zero:

$$\begin{aligned}
\mathbf{v}_\perp \cdot \mathbf{v}_{||} &= \mathbf{v}_\perp \cdot \mathbf{v} - \mathbf{v}_\perp \cdot \mathbf{v}_\perp \\
&= (\mathbf{v} \cdot \mathbf{n})\mathbf{n} \cdot \mathbf{v} - (\mathbf{v} \cdot \mathbf{n})^2 \mathbf{n} \cdot \mathbf{n} \\
&= ((\mathbf{v} \cdot \mathbf{n})^2 - (\mathbf{v} \cdot \mathbf{n})^2 \\
&= 0
\end{aligned} \tag{16.7}$$

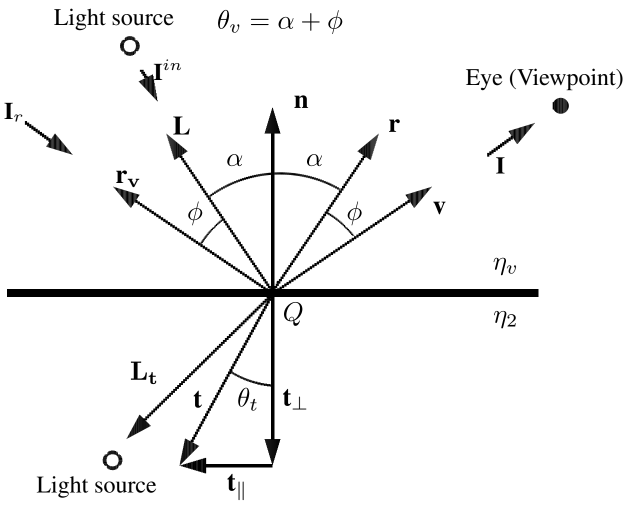


Figure 16-9 Basic Vectors for Ray Tracing

Note that all the normal components and $\mathbf{n}$ are parallel to each other and all the tangent components are parallel as well. Also,

$$\begin{aligned} \cos\theta_v &= |\mathbf{v}_\perp| \\ \sin\theta_v &= |\mathbf{v}_{||}| \end{aligned} \tag{16.8}$$

16.4.2 Reflection

The direction of perfect reflection is $\mathbf{r_v}$ of Figure 16-9. It can be calculated from the normal and viewing vector:

$$\begin{aligned} \mathbf{r_v} &= \mathbf{r_{v\perp}} + \mathbf{r_{v||}} \\ &= \mathbf{v}_\perp - \mathbf{v}_{||} \\ &= 2\mathbf{v}_\perp - \mathbf{v} \\ &= 2(\mathbf{v}\cdot\mathbf{n})\mathbf{n} - \mathbf{v} \end{aligned} \tag{16.9}$$

Note that $|\mathbf{r_v}|^2 = |\mathbf{v}|^2 = 1$.

Suppose we have used a Phong lighting model as our local lighting model. From Figure 16-9, we see that $\mathbf{r_v}\cdot\mathbf{L} = \mathbf{r}\cdot\mathbf{v}$, and from Equation (7.15) of Chapter 7, the direct illumination at Q due to the light source is given by

$$\begin{aligned} \mathbf{I}_{local} &= \mathbf{I}_a + \mathbf{I}_d + \mathbf{I}_s + \mathbf{I}_e \\ &= \mathbf{c}_a \odot \mathbf{I}_a^{in} + \mathbf{c}_d \odot \mathbf{I}_d^{in}(\mathbf{L}\cdot\mathbf{n}) + \mathbf{c}_s \odot \mathbf{I}_s^{in}(\mathbf{r_v}\cdot\mathbf{L})^f + \mathbf{I}_e \end{aligned} \tag{16.10}$$

There could be more than one light source in the scene. Suppose there are N lights and we use subscript i to associate a vector with light i. So $\mathbf{L_i}$ is the unit vector in the direction of light i. We introduce the variable δ_i to denote whether the intersection point Q is illuminated by light i ($\delta_i = 1$) or it is shadowed ($\delta_i = 0$) from that light source. To determine the value of δ_i, we first check whether $\mathbf{L_i}\cdot\mathbf{n} > 0$. If so, the light is above the surface and we need to use a shadow ray to determine its visibility.

The total local lighting due to all the light sources above the surface is the sum of the illumination of all the visible lights:

$$\mathbf{I}_{local} = \mathbf{c}_a \odot \mathbf{I}_a^{in} + \mathbf{c}_d \odot \sum_{i=1}^{N} \delta_i \mathbf{I}_d^{in,i}(\mathbf{L_i}\cdot\mathbf{n}) + \mathbf{c}_s \odot \sum_{i=1}^{N} \delta_i \mathbf{I}_s^{in,i}(\mathbf{r_v}\cdot\mathbf{L_i})^f + \mathbf{I}_e \tag{16.11}$$

As mentioned in Chapter 7, the coefficient vectors $\mathbf{c}_*$ are tuples of coefficients of red, green, and blue intensities. The symbol $\odot$ denotes a component-wise multiplication.

In addition to the local illumination, the intensity$\mathbf{I}$ at Q as seen from the ray direction $\mathbf{v}$ consists of intensity from reflection:

$$\mathbf{I} = \mathbf{I}_{local} + \rho_{\mathbf{rg}} \odot \mathbf{I}_r \tag{16.12}$$

where ρ_{rg} is the reflection coefficient of the surface. This coefficient can be a scalar independent of the light wavelength for simple applications or it can be a 3-tuple consisting of three differents values for the red, green and blue colors if we need more accurate calculation of the reflected light.

16.4.3 Transmission

The transmission ray can be calculated using Snell's law, which states that the ratio of the refractive indices is equal to the inverse of the ratio of the sines of the angles:

$$\frac{\eta_v}{\eta_t} = \frac{\sin\theta_t}{\sin\theta_v} \tag{16.13}$$

The transmission angle θ_t can be calculated by:

$$\sin\theta_t = \frac{\eta_v}{\eta_t}\sin\theta_v \tag{16.14}$$

Equation (16.14) implies a special situation when $\eta_v \sin\theta_v > \eta_t$, leading to $\sin\theta_t > 1$ which is impossible. Physically, this means that total internal reflection has occurred. Therefore, we have to put the constraint, $\eta_v \sin\theta_v \leq \eta_t$ in (16.14).

Our goal is to calculate the transmission vector $\mathbf{t}$ from the normal $\mathbf{n}$ and viewing vector $\mathbf{v}$ in Figure 16-9. Recall that all the vectors in the the figure are normalized (magnitude=1) and a vector can be decomposed into a normal component and a parallel component. So

$$\mathbf{t} = \mathbf{t}_\perp + \mathbf{t}_{||} \tag{16.15}$$

The parallel component $\mathbf{t}_{||}$ is in the opposite direction of $\mathbf{v}_{||}$ which is given by (16.6). Thus

$$|\mathbf{t}_{||}| = \sin\theta_t = \frac{\eta_v}{\eta_t}\sin\theta_v = \frac{\eta_v}{\eta_t}|\mathbf{v}_{||}| \tag{16.16}$$

and

$$\mathbf{t}_{||} = -\frac{\eta_v}{\eta_t}\mathbf{v}_{||} = \frac{\eta_v}{\eta_t}((\mathbf{v}\cdot\mathbf{n})\mathbf{n} - \mathbf{v}) \tag{16.17}$$

Also, $\mathbf{t}_\perp$ is in the negative direction of the normal $\mathbf{n}$ and $|\mathbf{t}_\perp|^2 + |\mathbf{t}_{||}|^2 = 1$. So

$$\mathbf{t}_\perp = -\sqrt{1 - |\mathbf{t}_{||}|^2}\,\mathbf{n} \tag{16.18}$$

The transmission vector is obtained by summing the two components, that is $\mathbf{t} = \mathbf{t}_\perp + \mathbf{t}_{||}$.

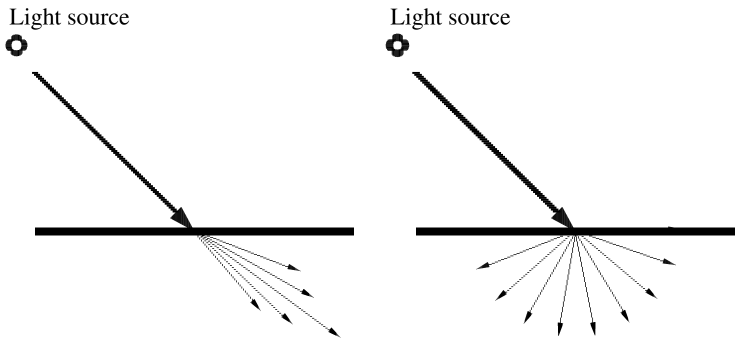
In the situation that $\sin^2\theta_t = |\mathbf{t}_{||}|^2 \geq 1$, total internal reflection occurs and there will be no transmission light. The incoming angle at which total internal reflection occurs is called the *critical angle* θ_c. The calculation of the transmission vector $\mathbf{t}$ can be easily implemented as follows, makng use of *Vector3* class discussed in previous chapters:

```
//calculating transmission director; vector returned in t
bool transmitDir(double eta_v, double eta_t, const Vector3 &v,
                                const Vector3 &n, Vector3 &t)
{
  Vector3 t_parallel = (eta_v / eta_t ) * ((v * n) * n - v );
  double tp_square = t_parallel * t_parallel;
  if ( tp_square >= 1.0 )     //total internal reflection
     return false;
  Vector3 t_perp = -sqrt ( 1 - tp_square ) * n;
  t = t_parallel + t_perp;   //transmission vector

  return true;
}
```

16.4.4 Phong Lighting for Transmission

When we consider the illumination due to reflection, we have to apply a local lighting model to calculate the lighting due to different material surfaces as we did in (16.11) where a Phong lighting model has been used. We can extend Phong lighting to calculate the transmission illumination in a similar way. For transmission illumination, the light may come from the opposite side of the surface from the viewer or incoming ray. Just like reflection illumination, transmission illumination is modeled as consisting of two types of lighting, specular and diffuse lighting as shown in Figure 16-10.



(a) Specular Transmission (b) Diffuse Transmission

Figure 16-10 Phong Lighting for Transmission

Specular transmission transmits primarily along the direction of perfect transmission, the direction $\mathbf{t}$ found using Snell's law. Diffuse transmission transmits equally in all directions. In this model, we define $\mathbf{L_t}$ as the direction from the intersection point Q to a light source (see Figure 16-9). Similar to the illumination of the Phong reflection lighting model, the local transmission illumination due to light source i is given by

$$\mathbf{I}^i_{local} = \mathbf{c}_a \odot \mathbf{I}_a^{in,i} + \delta_i^t(\mathbf{c}_{td} \odot \mathbf{I}_d^{in,i}(\mathbf{L_{ti}} \cdot (-\mathbf{n})) + \mathbf{c}_{ts} \odot \mathbf{I}_s^{in,i}(\mathbf{t} \cdot \mathbf{L_{ti}})^f \tag{16.19}$$

where $\mathbf{c}_{td}$ and $\mathbf{c}_{ts}$ are transmission coefficients for diffuse and specular materials respectively. The variable δ_{ti} is 1 if the light and the viewpoint are on opposite side of the surface (i.e. $\mathbf{L_{ti}} \cdot \mathbf{n} < 0$), otherwise it is equal to 0; this can be determined by a shadow feeler. The total local transmission lighting at the point is the sum of all illumination due to light sources that are on the opposite side of the surface.

The total lighting, including both reflection and transmission illumination, is given by

$$\begin{aligned}\mathbf{I}_{local} = \; & \mathbf{c}_a \odot \mathbf{I}_a^{in} + \mathbf{c}_d \odot \sum_{i=1}^{N} \delta_i \mathbf{I}_d^{in}(\mathbf{L_i} \cdot \mathbf{n}) + \mathbf{c}_s \odot \sum_{i=1}^{N} \delta_i \mathbf{I}_s^{in}(\mathbf{r_v} \cdot \mathbf{L_i})^f + \mathbf{I}_e \\ & + \mathbf{c}_{td} \odot \sum_{i=1}^{N} \delta_{ti} \mathbf{I}_d^{in,i}(\mathbf{L_{ti}} \cdot (-\mathbf{n})) + \mathbf{c}_{ts} \odot \sum_{i=1}^{N} \delta_{ti} \mathbf{I}_s^{in,i}(\mathbf{t} \cdot \mathbf{L_{ti}})^f\end{aligned} \tag{16.20}$$

Note that for each light i, both δ_i and δ_{ti} may be 0, but only one of them can be 1 as the light source cannot be above and below the surface at the same time.