# 11 Basic Ray Tracing

## 11.1 Basics

- So far, we have considered only *local* models of illumination; they only account for incident light coming directly from the light sources.
- *Global* models include incident light that arrives from other surfaces, and lighting effects that account for global scene geometry. Such effects include:
  - Shadows
  - Secondary illumination (such as color bleeding)
  - Reflections of other objects, in mirrors, for example
- Ray Tracing was developed as one approach to modeling the properties of global illumination.
- The basic idea is as follows:

  For each pixel:
  - Cast a ray from the eye of the camera through the pixel, and find the first surface hit by the ray.
  - Determine the surface radiance at the surface intersection with a combination of local and global models.
  - To estimate the global component, cast rays from the surface point to possible incident directions to determine how much light comes from each direction. This leads to a recursive form for tracing paths of light backwards from the surface to the light sources.

> *Aside:*
> Basic Ray Tracing is also sometimes called Whitted Ray Tracing, after its inventor, Turner Whitted.

**Computational Issues**

- Form rays.
- Find ray intersections with objects.
- Find closest object intersections.
- Find surface normals at object intersection.
- Evaluate reflectance models at the intersection.

 

## 11.2 Ray Casting

We want to find the ray from the eye through pixel $(i, j)$.

- Camera Model
  $\bar{e}^W$ is the origin of the camera, in world space.
  $\vec{u}$, $\vec{v}$, and $\vec{w}$ are the world space directions corresponding to the $\vec{x}$, $\vec{y}$, and $\vec{z}$ axes in eye space.
  The image plane is defined by $(\bar{p} - \bar{r}) \cdot \vec{w} = 0$, or $\bar{r} + a\vec{u} + b\vec{v}$, where $\bar{r} = \bar{e}^W + f\vec{w}$.

- Window
  A window in the view-plane is defined by its boundaries in camera coordinates: $w_l$, $w_r$, $w_t$, and $w_b$. (In other words, the left-most edge is the line $(w_l, \lambda, f)$.)

- Viewport
  Let the viewport (i.e., output image) have columns $0...n_c - 1$ and rows $0...n_r - 1$. $(0, 0)$ is the upper left entry.
  The camera coordinates of pixel $(i, j)$ are as follows:

$$\bar{p}^C_{i,j} = (w_l + i\Delta u, w_t + j\Delta v, f)$$

$$\Delta u = \frac{w_r - w_l}{n_c - 1}$$

$$\Delta v = \frac{w_b - w_t}{n_r - 1}$$

  In world coordinates, this is:

$$\bar{p}^W_{i,j} = \begin{pmatrix} | & | & | \\ \vec{u} & \vec{v} & \vec{w} \\ | & | & | \end{pmatrix} \bar{p}^C_{i,j} + \bar{e}^W$$

- Ray: Finally, the ray is then defined in world coordinates as follows:

$$\bar{r}(\lambda) = \bar{p}^W_{i,j} + \lambda\vec{d}_{i,j}$$

  where $\vec{d}_{i,j} = \bar{p}^W_{i,j} - \bar{e}^W$. For $\lambda > 0$, all points on the ray lie in front of the viewplane along a single line of sight.

## 11.3 Intersections

In this section, we denote a ray as $\bar{r}(\lambda) = \bar{a} + \lambda\vec{d}$, $\lambda > 0$.

 

### 11.3.1 Triangles

Define a triangle with three points, $\bar{p}_1$, $\bar{p}_2$, and $\bar{p}_3$. Here are two ways to solve for the ray-triangle intersection.

- Intersect $\bar{r}(\lambda)$ with the plane $(\bar{p} - \bar{p}_1) \cdot \vec{n} = 0$ for $\vec{n} = (\bar{p}_2 - \bar{p}_1) \times (\bar{p}_3 - \bar{p}_1)$ by substituting $\bar{r}(\lambda)$ for $\bar{p}$ and solving for $\lambda$. Then test the half-planes for constraints. For example:

$$(\bar{a} + \lambda\vec{d} - \bar{p}_1) \cdot \vec{n} = 0$$

$$\lambda^* = \frac{(\bar{p}_1 - \bar{a}) \cdot \vec{n}}{\vec{d} \cdot \vec{n}}$$

  What does it mean when $\vec{d} \cdot \vec{n} = 0$? What does it mean when $\vec{d} \cdot \vec{n} = 0$ and $(\bar{p}_1 - \bar{a}) \cdot \vec{n} = 0$?

- Solve for $\alpha$ and $\beta$ where $\bar{p}(\alpha, \beta)\bar{p}_1 + \alpha(\bar{p}_2 - \bar{p}_1) + \beta(\bar{p}_3 - \bar{p}_1)$, i.e. $\bar{r}(\lambda) = \bar{a} + \lambda\vec{d} = \bar{p}_1 + \alpha(\bar{p}_2 - \bar{p}_1) + \beta(\bar{p}_3 - \bar{p}_1)$. This leads to the 3x3 system

$$\begin{pmatrix} | & | & | \\ -(\bar{p}_2 - \bar{p}_1) & -(\bar{p}_3 - \bar{p}_1) & \vec{d} \\ | & | & | \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \lambda \end{pmatrix} = (\bar{p}_1 - \bar{a})$$

  Invert the matrix and solve for $\alpha$, $\beta$, and $\lambda$. The intersection is in the triangle when the following conditions are all true:

$$\alpha \geq 0$$
$$\beta \geq 0$$
$$\alpha + \beta \leq 1$$

### 11.3.2 General Planar Polygons

For general planar polygons, solve for the intersection with the plane. Then form a ray $\mathbf{s}(t)$ in the plane, starting at the intersection $\bar{p}(\lambda^*)$. Measure the number of intersections with the polygon sides for $t > 0$. If there is an even number of intersections, the intersection is inside. If the number of intersection is odd, it is outside.

> *Aside:*
> This is a consequence of the Jordan Curve Theorem. As related to this problem, it states that two points are both inside or both outside when the number of intersections on a line between them is even.

 

### 11.3.3 Spheres

Define the unit sphere centered at $\bar{c}$ by $||\bar{p} - \bar{c}||^2 = 1$.
Substitute a point on the ray $\bar{r}(\lambda)$ into this equation:

$$(\bar{a} + \lambda\vec{d} - \bar{c}) \cdot (\bar{a} + \lambda\vec{d} - \bar{c}) - 1 = 0$$

Expand this equation and write it in terms of the quadratic form:

$$A\lambda^2 + 2B\lambda + C = 0$$
$$A = \vec{d} \cdot \vec{d}$$
$$B = (\bar{a} - \bar{c}) \cdot \vec{d}$$
$$C = (\bar{a} - \bar{c}) \cdot (\bar{a} - \bar{c}) - 1$$

The solution is then:

$$\lambda = \frac{-2B \pm \sqrt{4B^2 - 4AC}}{2A} = -\frac{B}{A} \pm \frac{\sqrt{D}}{A}, D = B^2 - AC$$

If $D < 0$, there are no intersections. If $D = 0$, there is one intersection; the ray grazes the sphere. If $D > 0$, there are two intersections with two values for $\lambda$, $\lambda_1$ and $\lambda_2$.

When $D > 0$, three cases of interest exist:

- $\lambda_1 < 0$ and $\lambda_2 < 0$. Both intersections are behind the view-plane, and are not visible.
- $\lambda_1 > 0$ and $\lambda_2 < 0$. The $\bar{p}(\lambda_1)$ is a visible intersection, but $\bar{p}(\lambda_1)$ is not.
- $\lambda_1 > \lambda_2$ and $\lambda_2 > 0$. Both intersections are in front of the view-plane. $\bar{p}(\lambda_2)$ is the closest intersection.

### 11.3.4 Affinely Deformed Objects

**Proposition**: Given an intersection method for an object, it is easy to intersect rays with affinely deformed versions of the object. We assume here that the affine transformation is invertible.

- Let $F(\bar{y}) = 0$ be the deformed version of $f(\bar{x}) = 0$, where $\bar{y} = \mathbf{A}\bar{x} + \vec{t}$.
  i.e. $F(\bar{y}) = f(\mathbf{A}^{-1}(\bar{y} - \vec{t})) = 0$, so $F(\bar{y}) = 0$ iff $f(\bar{x}) = 0$.
- Given an intersection method for $f(\bar{x}) = 0$, find the intersection of $\bar{r}(\lambda) = \bar{a} + \lambda\vec{d}$ and $F(\bar{y}) = 0$, where $\lambda > 0$.
- **Solution:** Substitute $\bar{r}(\lambda)$ into the implicit equation $f = F(\bar{y})$:

$$\begin{aligned} F\left(\bar{r}(\lambda)\right) &= f\left(\mathbf{A}^{-1}\left(\bar{r}\left(\lambda\right) - \vec{t}\right)\right) \\ &= f(\mathbf{A}^{-1}(\bar{a} + \lambda\vec{d} - \vec{t})) \\ &= f(\bar{a}' + \lambda\vec{d}') \\ &= f(\bar{r}'(\lambda)) \end{aligned}$$

 

where

$$\bar{a}' = \mathbf{A}^{-1}(\bar{a} - \vec{t})$$
$$\vec{d}' = \mathbf{A}^{-1}\vec{d}$$

i.e. intersecting $F(\bar{y})$ with $\bar{r}(\lambda)$ is like intersecting $f(\mathbf{x})$ with $\bar{r}'(\lambda) = \bar{a}' + \lambda\vec{d}'$ where $\lambda > 0$. The value of $\lambda$ found is the same in both cases.

- **Exercise:** Verify that, at the solution $\lambda^*$, with an affine deformation $\bar{y} = \mathbf{A}\bar{x} + \vec{t}$, that $\bar{r}(\lambda^*) = \mathbf{A}\bar{r}'(\lambda^*) + \vec{t}$.

### 11.3.5 Cylinders and Cones

A right-circular cylinder may be defined by $x^2 + y^2 = 1$ for $|z| \leq 1$. A cone may be defined by $x^2 + y^2 - \frac{1}{4}(1 - z^2) = 0$ for $0 \leq z \leq 1$.

- Find intersection with "quadratic wall," ignoring constraints on $z$, e.g. using $x^2 + y^2 = 1$ or $x^2 + y^2 - \frac{1}{4}(1 - z^2) = 0$. Then test the $z$ component of $\bar{p}(\lambda^*)$ against the constraint on $z$, e.g. $z \leq 1$ or $z < 1$.

- Intersect the ray with the planes containing the base or cap (e.g. $z = 1$ for the cylinder). Then test the $x$ and $y$ components of $\bar{p}(\lambda^*)$ to see if they satisfy interior constraints (e.g. $x^2 + y^2 < 1$ for the cylinder).

- If there are multiple intersections, then take the intersection with the smallest positive $\lambda$ (i.e., closest to the start of the ray).

 

## 11.4 The Scene Signature

The scene signature is a simple way to test geometry intersection methods.

- Create an image in which pixel $(i, j)$ has intensity $k$ if object $k$ is first object hit from ray through $(i, j)$.
- Each object gets one unique color.

*Note:*
**Pseudo-Code: Scene Signature**

$<$ Construct scene model = { obj, $(\mathbf{A}, \vec{t})$, objID } $>$
sig: array[nc, nr] of objID
**for** j = 0 to nr-1 (loop over rows)
  **for** i = 0 to nc-1 (loop over columns)
    $<$ Construct ray $\vec{\mathbf{r}}_{ij}(\lambda) = \bar{p}_{ij} + \lambda(\bar{p}_{ij} - \bar{e})$ through pixel $\bar{p}_{ij}$ $>$
    $\lambda_{i,j} \leftarrow \infty$
    **loop** over all objects in scene, with object identifiers objID$_k$
      $<$ find $\lambda^*$ for the closest intersection of the ray $\vec{\mathbf{r}}_{ij}(\lambda)$ and the object $>$
      **if** $\lambda^* > 0$ and $\lambda^* < \lambda_{i,j}$ then
        $\lambda_{i,j} \leftarrow \lambda^*$
        sig[i,j].objID $\leftarrow$ objID$_k$
      end **if**
    end **loop**
  end **for**
end **for**

## 11.5 Efficiency

Intersection tests are expensive when there are large numbers of objects, and when the objects are quite complex! Fortunately, data structures can be used to avoid testing intersections with objects that are not likely to be significant.

**Example:** We could bound a 3D mesh or object with a simple bounding volume (e.g. sphere or cube). Then we would only test intersections with objects if there exists a positive intersection with the bounding volume.

**Example:** We could project the extent onto the image plane so you don't need to cast rays to determine potential for intersections.

 

## 11.6 Surface Normals at Intersection Points

Once we find intersections of rays and scene surfaces, and we select the first surface hit by the ray, we want to compute the shading of the surface as seen from the ray. That is, we cast a ray out from a pixel and find the first surface hit, and then we want to know how much light leave the surface along the same ray but in the reverse direction, back to the camera.
Toward this end, one critical property of the surface geometry that we need to compute is the surface normal at the hit point.

- For mesh surfaces, we might interpolate smoothly from face normals (like we did to get normals at a vertex). This assumes the underlying surface is smooth.
- Otherwise we can just use the face normal.
- For smooth surfaces (e.g. with implicit forms $f(\bar{p}) = 0$ or parametric forms $\mathbf{s}(\alpha, \beta)$), either take

$$\vec{n} = \frac{\nabla f(\bar{p})}{||\nabla f(\bar{p})||}$$

or

$$\vec{n} = \frac{\frac{\partial \mathbf{s}}{\partial \alpha} \times \frac{\partial \mathbf{s}}{\partial \beta}}{||\frac{\partial \mathbf{s}}{\partial \alpha} \times \frac{\partial \mathbf{s}}{\partial \beta}||}.$$

### 11.6.1 Affinely-deformed surfaces.

Let $f(\bar{p}) = 0$ be an implicit surface, and let $Q(\bar{p}) = \mathbf{A}\bar{p} + \vec{t}$ be an affine transformation, where $\mathbf{A}$ is invertible. The affinely-deformed surface is

$$F(\bar{q}) = f(Q^{-1}(\bar{p})) = f(\mathbf{A}^{-1}(\bar{p} - \vec{t})) = 0 \tag{1}$$

A normal of $F$ at a point $\bar{q}$ is given by

$$\frac{\mathbf{A}^{-T}\vec{n}}{||\mathbf{A}^{-T}\vec{n}||} \tag{2}$$

where $\mathbf{A}^{-T} = (\mathbf{A}^{-1})^T$ and $\vec{n}$ is the normal of $f$ at $\bar{p} = Q^{-1}(\bar{q})$.

*Derivation:*
Let $\bar{s} = \bar{r}(\lambda^*)$ be the intersection point, and let $(\bar{p} - \bar{s}) \cdot \vec{n} = 0$ be the tangent plane at the intersection point. We can also write this as:

$$(\bar{p} - \bar{s})^T \vec{n} = 0 \tag{3}$$

Substituting in $\bar{q} = \mathbf{A}\bar{p} + \vec{t}$ and solving gives:

$$(\bar{p} - \bar{s})^T \vec{n} \;=\; (\mathbf{A}^{-1}(\bar{q} - \vec{t}) - \bar{s})^T \vec{n} \tag{4}$$

 

$$= \quad (\bar{q} - (\mathbf{A}\bar{s} + \vec{t}))^T \mathbf{A}^{-T} \vec{n} \tag{5}$$

In other words, the tangent plane at the transformed point has normal $\mathbf{A}^{-T}\vec{n}$ and passes through point $(\mathbf{A}\bar{s} + \vec{t})$.
preserved so the tangent plane on the deformed surface is given by $(\mathbf{A}^{-1}(\bar{q} - \vec{t}))^T \vec{n} = D$.

This is the equation of a plane with *unit* normal $\frac{\mathbf{A}^{-T}\vec{n}}{||\mathbf{A}^{-T}\vec{n}||}$.

## 11.7 Shading

Once we have cast a ray through pixel $\bar{p}_{i,j}$ in the direction $\vec{d}_{i,j}$, and we've found the closest hit point $\bar{p}$ with surface normal $\vec{n}$, we wish to determine how much light leaves the surface at $\bar{p}$ into the direction $-\vec{d}_{i,j}$ (i.e., back towards the camera pixel). Further we want reflect both the light from light sources that directly illuminate the surface as well as secondary illumination, where light from other surfaces shines on the surface at $\bar{p}$. This is a complex task since it involves all of the ways in which light could illuminate the surface from all different directions, and the myriad ways such light interacts with the surface and it then emitted or reflected by the surface. Here we will deal first with the simplest case, known widely as Whitted Ray Tracing.

*Aside:*
First, note that if we were to ignore all secondary reflection, then we could just compute the Phong reflectance model at $\bar{p}$ and then color the pixel with that value. Such scenes would look similar to those that we have rendered using shading techniques seen earlier in the course. The main differences from earlier rendering techniques are the way in which hidden surfaces are handled and the lack of interpolation.

### 11.7.1 Basic (Whitted) Ray Tracing

In basic ray tracing we assume that that the light reflected from the surface is a combination of the reflection computed by the Phong model, along with one component due to specular secondary reflection. That is, the only reflection we consider is that due to perfect mirror reflection. We only consider perfect specular reflection for computational efficiency; i.e., rather than consider secondary illumination at $\bar{p}$ from all different directions, with perfect specular reflection we know that the only incoming light at $\bar{p}$ that will be reflected in the direction $-\vec{d}_{i,j}$ will be that coming from the corresponding mirror direction (i.e., $\vec{m}_s = -2(\vec{d}_{i,j} \cdot \vec{n}))\vec{n} + \vec{d}_{i,j}$). We can find out how much light is incoming from direction $\vec{m}_s$ be casting another ray into that direction from $\bar{p}$ and calculating the light reflected from the first surface hit. Note that we have just described a recursive ray tracer; i.e., in order to calculate the reflectance at a hit point we need to cast more rays and compute the reflectance at the new hit points so we can calculate the incoming light at the original hit point.

In summary, for basic (Whitted) ray tracing, the reflectance model calculation comprises:

 

- A local model (e.g., Phong) to account for diffuse and off-axis specular reflection (highlights) due to light sources.
- An ambient term to approximate the global diffuse components.
- Cast rays from $\bar{p}$ into direction $\vec{m}_s = -2(\vec{d}_{i,j} \cdot \vec{n}))\vec{n} + \vec{d}_{i,j}$ to estimate ideal mirror reflections due to light coming from other ojects (i.e., secondary reflection).

For a ray $r(\lambda) = \bar{a} + \lambda\vec{d}$ which hits a surface point point $\bar{p}$ with normal $\vec{n}$, the reflectance is given by

$$E = r_a I_a + r_d I_d \max(0, \vec{n} \cdot \vec{s}) + r_s I_s \max(0, \vec{c} \cdot \vec{m})^\alpha + r_g I_{spec}$$

where $r_a$, $r_d$, and $r_s$ are the reflection coefficients of the Phong model, $I_a$, $I_d$, and $I_s$ are the light source intensities for the ambient, diffuse and specular terms of the Phong model, $\vec{s}$ is the light source direction from $\bar{p}$, the emittant direction of interest is $\vec{c} = -\vec{d}_{i,j}$, and $\vec{m} = 2(\vec{s} \cdot \vec{n}))\vec{n} - \vec{s}$ is the perfect mirror direction for the local specular reflection. Finally, $I_{spec}$ is the light obtained from the recursive ray cast into the direction $\vec{m}_s$ to find secondary illumination, and $r_g$ is the reflection coefficient that determines the fraction of secondary illumination that is reflected by the surface at $\bar{p}$

### 11.7.2 Texture

- Texture can be used to modulate diffuse and a mbient reflection coefficients, as with Gouraud shading.
- We simply need a way to map each point on the surface to a point in texture space, as above, e.g. given an intersection point $\bar{p}(\lambda^*)$, convert into parametric form $\mathbf{s}(\alpha, \beta)$ and use $(\alpha, \beta)$ to find texture coordinates $(\mu, \nu)$.
- Unlike Gouraud shading, we don't need to interpolate $(\mu, \nu)$ over polygons. We get a new $(\mu, \nu)$ for each intersection point.
- Anti-aliasing and super-sampling are covered in the Distribution Ray Tracing notes.

### 11.7.3 Transmission/Refraction

- Light that penetrates a (partially or wholly) transparent surface/material is refracted (bent), owing to a change in the speed of light in different media.
- Snell's Law governs refraction:

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{c_1}{c_2}$$

 

- The index of refraction is the ratio of light speeds $c_1/c_2$. For example, the index of refraction for passing from air to water is $\frac{c_{air}}{c_{water}} = 1.33$, and for passing from air to glass, it is $\frac{c_{air}}{c_{glass}} = 1.8$.
  Note: There is also a wavelength dependence. We ignore this here.

- Example:
  - If $c_2 < c_1$, light bends towards the normal (e.g. air to water). If $c_2 < c_1$, light bends away from the normal (e.g. water to air).
  - The critical angle $\theta_c$, when $c_2 > c_1$, is when $\theta_1 \rightarrow \theta_c$ and $\theta_2 \rightarrow 90$. Beyond $\theta_c$, $\theta_1 > \theta_c$, and total internal reflection occurs. No light enters the material.

- Remarks:
  - The outgoing direction is in the plane of the incoming direction and $\vec{n}$. This is similar to the perfect specular direction.
  - When $\theta_1 = 0$, then $\theta_2 = 0$, i.e. there is no bending.

- For ray tracing:
  - Treat global transmission like global specular, i.e. cast one ray.
  - Need to keep track of the speed of light in the current medium.

### 11.7.4 Shadows

- A simple way to include some global effects with minimal work is to turn off local reflection when the surface point $\bar{p}$ cannot see light sources, i.e. when $\bar{p}$ is in shadow.

- When computing $E$ at $\bar{p}$, cast a ray toward the light source, i.e. in the direction $\mathbf{s} = (\mathbf{l} - \bar{p})$.

$$\bar{p}^W(\lambda) = \bar{p}^W + \lambda(\mathbf{l}^W - \bar{p}^W)$$

- Find the first intersection with a surface in the scene. If $\lambda^*$ at the first intersection point is $0 \leq \lambda \leq 1$, then there exists a surface that occludes the light source from $\bar{p}$.
  - We should omit diffuse and specular terms from the local Phong model.
  - The surface radiance at $\bar{p}$ becomes

$$E = r_a I_a + r_g I_{spec}$$

 

*Note:*

**Pseudo-Code: Recursive Ray Tracer**

**for** each pixel (i,j)
  $<$ compute ray $\vec{\mathbf{r}}_{ij}(\lambda) = \bar{\mathbf{p}}_{ij} + \lambda\vec{\mathbf{d}}_{ij}$ where $\vec{\mathbf{d}}_{ij} = \bar{\mathbf{p}}_{ij} - \vec{\mathbf{e}}$ $>$
  $I$ = rayTrace( $\bar{\mathbf{p}}_{ij}$, $\vec{d}_{ij}$, 1);
  setpixel(i, j, $I$)
end **for**

**rayTrace**( $\bar{\mathbf{a}}$, $\vec{\mathbf{b}}$, depth)
  findFirstHit( $\bar{\mathbf{a}}$, $\vec{\mathbf{b}}$, *output var* obj, $\lambda$, $\bar{\mathbf{p}}$, $\vec{\mathbf{n}}$)
  **if** $\lambda > 0$ then
    $I$ = rtShade(obj, $\bar{\mathbf{p}}$, $\vec{\mathbf{n}}$, $-\vec{\mathbf{b}}$, depth)
  **else**
    $I$ = background;
  end **if**
  return($I$)

**findFirstHit**( $\bar{\mathbf{a}}$, $\vec{\mathbf{b}}$, *output var* OBJ, $\lambda_h$, $\bar{\mathbf{p}}_h$, $\vec{\mathbf{n}}_h$)
  $\lambda_h = -1$;
  **loop** over all objects in scene, with object identifiers $\text{objID}_k$
    $<$ find $\lambda*$ for the closest legitimate intersection of ray $\vec{\mathbf{r}}_{ij}(\lambda)$ and object $>$
    **if** ( $\lambda_h < 0$ or $\lambda^* < \lambda_h$) and $\lambda^* > 0$ then
      $\lambda_h = \lambda^*$
      $\bar{\mathbf{p}}_h = \bar{\mathbf{a}} + \lambda^*\vec{\mathbf{b}}$;
      $<$ determine normal at hit point $\vec{\mathbf{n}}_h$ $>$
      OBJ = $\text{objID}_k$
    end **if**
  end **loop**

**rtShade**(OBJ, $\bar{\mathbf{p}}$, $\vec{\mathbf{n}}$, $\vec{\mathbf{d}}_e$, depth)
  /* Local Component */
  findFirstHit( $\bar{\mathbf{p}}$, $\vec{\mathbf{l}}^w - \bar{\mathbf{p}}$, *output var* temp, $\lambda_h$);
  **if** $0 < \lambda_h < 1$ then
    $I_l$ = ambientTerm;
  **else**
    $I_l$ = phongModel($\bar{\mathbf{p}}$, $\vec{\mathbf{n}}$, $\vec{\mathbf{d}}_e$, OBJ.localparams)
  end **if**
  /* Global Component */
  **if** depth $<$ maxDepth then

```
    if OBJ has specular reflection then
        < calculate mirror direction m_s = -d_e + 2n · d_e n >
        I_spec = rayTrace( p, m_s, depth+1)
        < scale I_spec by OBJ.specularReflCoef >
    end if
    if OBJ is refractive then
        < calculate refractive direction t >
        if not total internal reflection then
            I_refr = rayTrace( p, t, depth+1)
            < scale I_refr by OBJ.refractiveReflCoef >
        end if
    end if
    I_g = I_spec + I_refr
else
    I_g = 0
end if
return(I_l + I_g)
```