Photorealistic Rendering
Lesson XI

Joel Isaacson Bar-Ilan
Computer Science

16th January 2005

1 Ray Tracing

Ray tracing extends the techniques that we previously discussed to handle reflective and transparent objects. This technique has the ability to handle interactions between objects. The techniques that we have considered until now are called *local illumination models*. In this model each reflection is defined by the local parameters at the point of reflection and the direct illumination. The only inter-object interaction discussed is hidden surfaces and shadows. The more general illumination model is called the *global illumination model*. Ray tracing generates an approximate solution to the global illumination model. In ray tracing we calculate higher order contributions to the illumination from light reflected off other objects.

The basic idea of ray tracing is simple: light travels along straight lines between objects (rays). We can invert the direction of the ray to trace it back to the point of illumination. While this approach is powerful enough to render scenes using raytracing alone we generally use a combination of local and global methods.

Light might reach the surface of an object indirectly via reflection from other surfaces, transmission through partially transparent objects or a combination of these. Ray tracing has a number of disadvantages. The chief disadvantage is that it can take quite a long time to render. Its advantage is that it can give a good approximation to the global illumination problem.

1.1 The basic idea

Ray tracing works in object space. For each point on the view surface a ray is projected from the view point through the view surface and backtracked onto the first object encountered. The local contribution to the color of the pixel is calculated from Phong reflection model and in addition potentially two rays (one reflective and another transmitted) are backtracked to add contributions due to global illumination.

Fig. 1 shows the backtracking of multiple reflections and transmission of light as performed in raytracing. Note that culling of non-visible surface can not be performed in environments that have reflective surfaces.

1.2 Ray tracing programming

Because of the tree structure of the ray tracing algorithm a recursive implementation is advantageous. The following pseudo-C(C++) code illustrates the naive ray tracing algorithm.

```
struct Intersection // This describes the intersection of a ray and object
{
  Vector location;
  Vector reflection_direction;
  Vector transmission_direction;
  Object object;
}

// A simple ray tracer. Just sit back and wait.
```

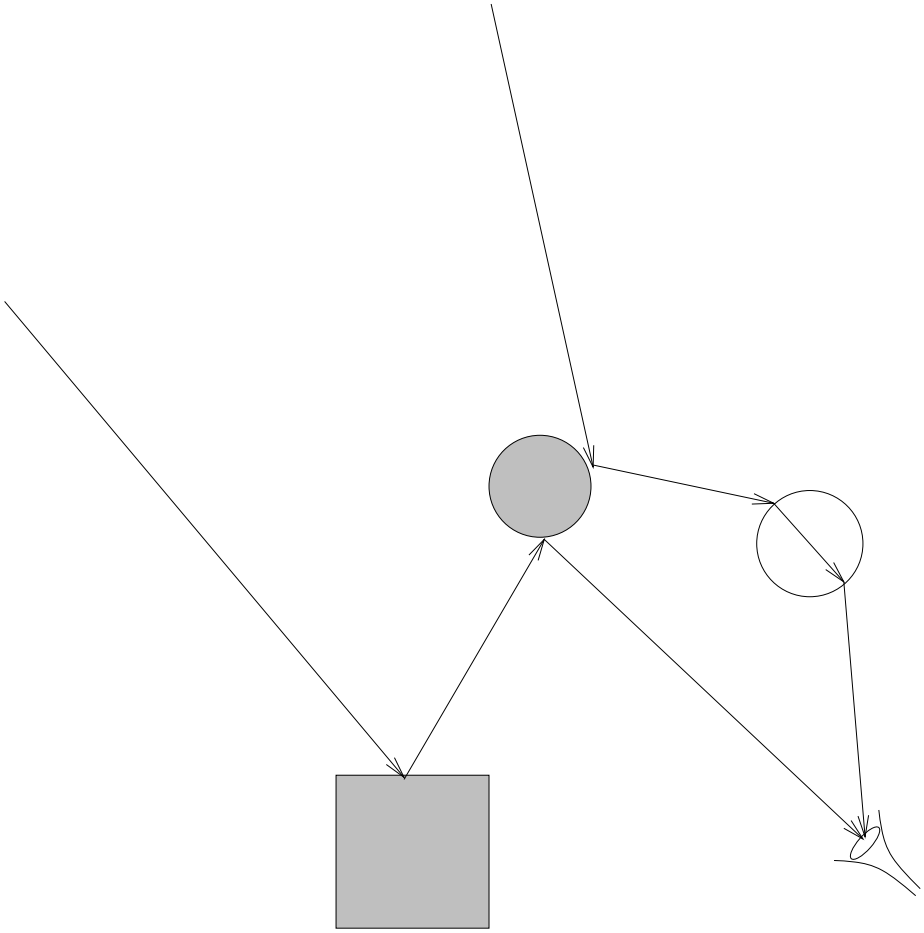

Figure 1: Raytracing of two reflective objects and one transparent object

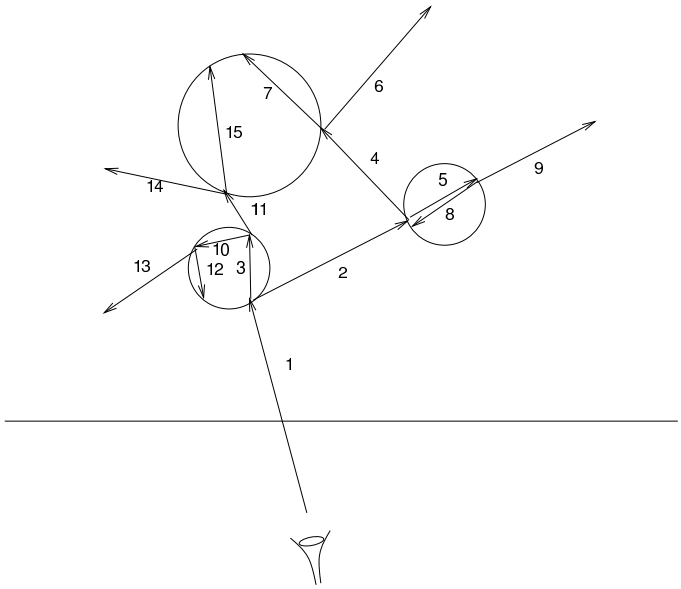

Figure 2: Recursive raytracing until level four

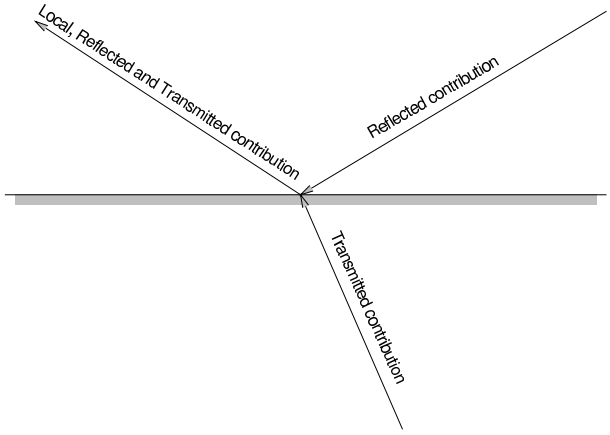

Figure 3: The three contributions to the reflectance of a raytraced point on the view surface

```
Color raytrace(Vector start, Vector direction, int depth)
{
  Vector transmission-direction, reflection-direction;
  Color local_color, reflective_color, transmission_color;
  Intersection intersection;

  if(depth > MAXDEPTH)
  {
return(BLACK_COLOR);
  }
  intersection= find_first_intersection(start, direction);

  local_color= color_at_intersection(intersection); // Phong formula

  // Calculate the reflective contribution
  if(reflective_coefficient(intersection) != 0)
  {
reflective_color= reflective_coefficient(intersection) *
  raytrace(intersection.location, intersection.reflection_direction, depth+1);
  }
  else
reflective_color= BLACK_COLOR;


  // Calculate the transmission contribution
  if(transmission_coefficient(intersection) != 0)
  {
transmission_color= transmission_coefficient(intersection) *
  raytrace(intersection.location, intersection.transmission_direction, depth+1);
  }
  else
transmission_color= BLACK_COLOR;

  return(combine_colors(intersection, local_color, reflective_color,
transmission_color));
}
```

Ray tracing is a very compute intensive application. In order to make it computationally feasible on scenes that are even moderately complex optimization techniques must be seriously considered.

1.3 Ray tracing geometry

Most of the time spend in ray tracing is in calculating the intersections of rays with objects. Because of the simplicity of calculating the intersection of a ray with a sphere many examples that use ray tracing have spheres prominently displayed. Frequently spheres are used as bounding volumes around objects since the test for intersection with a sphere is simple and a negative result can obviate a much more complex calculation.

1.3.1 Intersection of a ray with a sphere

A ray, passing through the points (x_1, y_1, z_1) and (x_2, y_2, z_2), can be written parametrically as:

$$x = x_1 + (x_2 - x_1)t = x_1 + it \tag{1}$$

$$y = y_1 + (y_2 - y_1)t = y_1 + jt \tag{2}$$

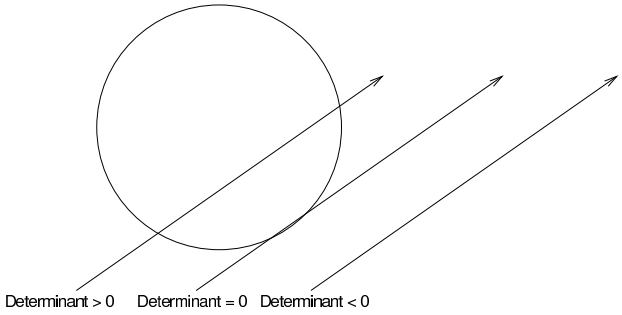

Figure 4: The intersection of a ray with a sphere

$$z = z_1 + (z_2 - z_1)t = z_1 + kt \tag{3}$$

Where t is $[0...1]$. A sphere with centers at (l, m, n) with radius r is given by:

$$(x - l)^2 + (y - m)^2 + (z - n)^2 = r^2 \tag{4}$$

Substituting for x, y and z gives a quadratic equation in t of the form:

$$at^2 + bt + c = 0 \tag{5}$$

where:

$$a = i^2 + j^2 + k^2 \tag{6}$$

$$b = 2i(x_1 - l) + 2j(y_1 - m) + 2k(z_1 - n) \tag{7}$$

$$c = l^2 + m^2 + n^2 + {x_1}^2 + {y_1}^2 + {z_1}^2 + 2(-lx_1 - my_1 - nz_1) - r^2 \tag{8}$$

This is a quadratic equation that has the well know solution:

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{9}$$

The determinant $(b^2 - 4ac)$ of the solution can be used to determine if the ray intersects the sphere. If the determinant is positive then the ray intersects the sphere at two points. If the determinant is zero then the ray tangent to the spheres surface. If the determinant is negative the ray doesn't intersect the sphere.

When we solve for t a negative value of t will indicate that the intersection is not in the direction of the ray, but rather in back of the starting point. If there are two positive solutions then smaller value is the closest to the initial point. The intersection point (x_i, y_i, z_i) can be calculated by substituting t in the parametric equations for $x(t), y(t), z(t)$. The surface normals at this point is just:

$$\mathbf{N} = \left(\frac{x_i - l}{r}, \frac{y_i - m}{r}, \frac{z_i - n}{r} \right) \tag{10}$$

1.3.2 Intersection of a ray with a polygon

The plane of a polygon can be represented by a linear equation:

$$ax + by + cz + d = 0 \tag{11}$$

If we substitute the parametric equation for the ray into this equation we can solve form t:

$$t = -\frac{ax_1 + by_1 + cz_1 + d}{ai + bj + ck + d} \tag{12}$$

Substituting t in the parametric equation for the ray gives the intersection point (x_i, y_i, z_i). If we are dealing with a convex polygon then the sum of angles between the lines from the intersection point and the vertices is $360°$. If the intersection point is outside the polygon then the sum of angles is less than $360°$ (in fact less than $180°$).

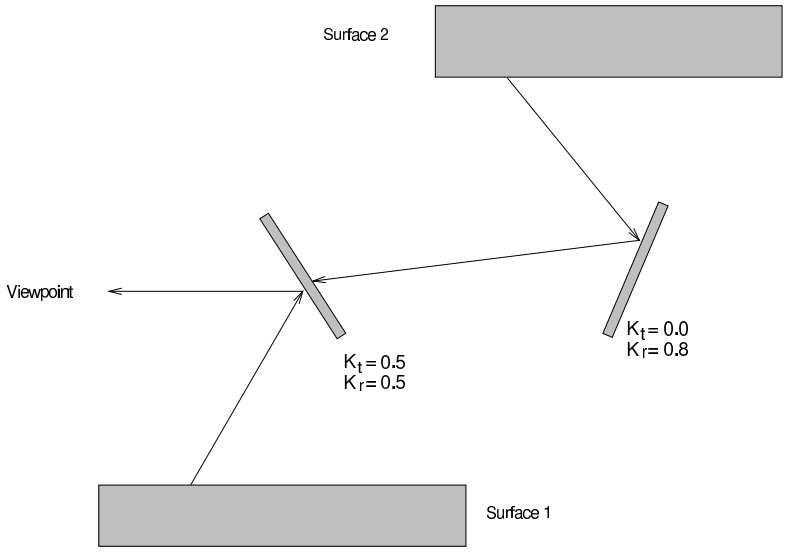

Figure 5: A simple 2d raytraced scene

1.4 Reflection and Refraction

See the third lecture.

1.5 Reflection-illumination model

The reflection model has to be extended to include ray traced reflection and transmission. The Phong model is taken as the local reflectance contribution. It has to be extended because of the possibility of specular transmission by translucent media. The Phong reflectance formula is (as defined in lecture III) is:

$$I_{local}(\mathbf{V}) = I_a K_a + I_i(k_d(\mathbf{L}\cdot\mathbf{N}) + k_s(\mathbf{R}\cdot\mathbf{V})^n) \tag{13}$$

If we include a specular transmission term to this formula we get:

$$I_{local}(\mathbf{V}) = I_a K_a + I_i(k_d(\mathbf{L}\cdot\mathbf{N}) + k_s(\mathbf{R}\cdot\mathbf{V})^{n_s} + k_t(\mathbf{T}\cdot\mathbf{V})^{n_t}) \tag{14}$$

Note that these term are driven only by the illumination (I_a and I_i), possibly modulated by shadows caused by other objects. It doesn't take into account secondary (or higher order) scattering of light off other objects or transmission through objects. In order to account for these effect we add two terms:

$$I(\mathbf{V}) = I_{local}(\mathbf{V}) + k_{tg} I(\mathbf{T}) + k_{rg} I(\mathbf{R}) \tag{15}$$

We can example a simple ray traced scene as show in Fig. 5. The value of the color for this ray is

$$I = 0.5 I_1 + 0.4 I_2 \tag{16}$$

1.6 Shadows

Shadows can be simply be added to the ray tracing model by performing an additional ray trace in the direction of the light source when performing the local illuminance calculation. If there is no obscuring objects then the calculation is performed otherwise that point is judged to be shadowed. This additional ray is called a **ray feeler**. This scheme can't properly handle light that is refracted through transparent objects. This type of phenomena is called caustics and can't be treated within a simple ray tracing model.

2 Radiosity

Ray tracing does a good job of modeling specular reflection and refractive transparency. We still have to make use of directionless ambient light in order to model global lighting. In order of model correctly lighting we have to solve the *radiosity* equations. These equations attempt to handle all light emitted, reflected or absorbed by all surfaces. The radiosity solution is then used to render a scene.

2.1 The Radiosity Equation

In the shading algorithms previously studied, light sources have always been treated separately from the surfaces that they illuminate. In the radiosity methods all surfaces to light other surfaces. Light sources are treated as having area.

We break up the surfaces into a finite number of discrete patches, each of which is assumed to be of finite size, emiting and reflecting light uniformly over their surfaces. The radiosity equation for a patch, i is:

$$B_i A_i = E_i A_i + R_i \sum_{j=1}^{n} B_j F_{ji} A_j \tag{17}$$

B_i is the radiosities of patch i measured in units of power/area (watts/ m^2). E_i is the rate in which light is emitted from the surface i. R_i is the reflectivity of the surface i. F_{ji} is called the form factor, which describes the amount of energy leaving surface j that arrives at all of the surface i.

There is a reciprocity relationship that hold between form factors:

$$A_i F_{ij} = A_j F_{ji} \tag{18}$$

Thus the radiosity equation becomes:

$$B_i = E_I + R_i \sum_{j=1}^{n} B_j F_{ij} \tag{19}$$

or

$$B_I - R_i \sum_{j=1}^{n} B_j F_{ij} = E_i \tag{20}$$

This equation can be written as the following set of simultaneous equations:

$$\begin{bmatrix} 1 - R_1 F_{11} & -R_1 F_{12} & ... & -R_1 F_{1n} \\ -R_2 F_{21} & 1 - R_2 F_{22} & ... & -R_1 F_{2n} \\ . & . & ... & . \\ . & . & ... & . \\ . & . & ... & . \\ -R_n F_{n1} & -R_n F_{n2} & ... * 1 - R_n F_{nn} & \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ . \\ . \\ . \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ . \\ E_n \end{bmatrix} \tag{21}$$