Introduction to Ray-tracing

Objectives

• Define ray-tracing as a means of rendering
• Ray-tracing for spheres
• Combining with shading model
• An algorithm framework
Light vs. Rendering

(Local) Ray-tracing

Compute intersections with objects
(Local) Ray-tracing

1. Find intersection of ray with nearest object in scene (Eye ray)

2. Determine if that object is in shadow relative to light source(s) (Shadow ray)

3. Determine illumination of object from light source(s)

A LARGE part of the work is in calculating intersections. Performance can be improved by making intersection calculations more efficient or by avoiding unnecessary intersection calculations.
Defining E-ray

Define ray by parametric line with parameter t,

\[ R(t) = R_o + R_d \times t \] for all values \( t > 0 \)

\( R_o \) = Origin of the ray (at eye position)
\( R_d \) = Direction of ray, unit vector pointing from eye to pixel on image plane (different for each pixel)

Intersecting Objects

1. Find intersection of ray nearest object in scene (Eye ray)

-Objects in scene could be:
  Polygons
  Curved Surfaces
  Spheres - easiest to compute...

-Ray-sphere intersection:
  Define sphere as:
  \[ (x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - S_r^2 = 0 \]
  with \( S_r \) as radius and \( S_c \) as center
Ray- Sphere Intersection

What are the possible scenarios?

1. Find value of t for ray-sphere intersection

Substitute the ray equation:
- $x = xo + xd * t$
- $y = yo + yd * t$
- $z = zo + zd * t$

Into sphere equation as:

$$(xo + xd*t - x_c)^2 + (yo + yd*t - y_c)^2 + (zo + zd*t - z_c)^2 - r^2 = 0$$

Note, this is a quadratic equation in terms of t
Ray- Sphere Intersection

1. Find value of t for ray-sphere intersection

\[(x_o + x_d*t - x_c)^2 + (y_o + y_d*t - y_c)^2 + (z_o + z_d*t - z_c)^2 - r^2 = 0\]

becomes \(A t^2 + B t + C = 0\) where:

\[A = x_d^2 + y_d^2 + z_d^2 = 1\]

\[B = 2[x_d(x_o - x_c) + y_d(y_o - y_c) + z_d(z_o - z_c)]\]

\[C = (x_o - x_c)^2 + (y_o - y_c)^2 + (z_o - z_c)^2 - r^2\]

and so \(t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}\) discriminant

Ray- Sphere Intersection

How does t correspond to intersection?

\[t_0, t_1\] discriminant = 0

\[t_0, t_1\] discriminant < 0
Ray- Sphere Intersection

Actual intersection point:
\[ xi = xo + xd*ti \]
\[ yi = yo + yd*ti \]
\[ zi = zo + zd*ti \]

Compute normal (unit length) at intersection:
\[ Rn = \frac{[ (xi - xc)/Sr, (yi - yc)/Sr, (zi - zc)/Sr]}{Sr} \]

Note, if ray is inside the sphere, negate to point at ray:

Summary:
Calculate \( A, B, \) and \( C \) and discriminant for quadratic
If discriminant < 0 then no intersection
Calculate to
If to < 0, find \( t1 \)
If \( t1 < 0 \), no intersection
Compute actual intersection point
Compute normal at intersection

To speed up precompute \( Sr^2 \) and \( 1/Sr \)

Be careful of round-off errors which may appear like intersections for very small non-zero values of \( t \)
Shadow Rays

Once an intersection is computed, we send a shadow ray to each light source to see if it is in shadow or not.

We can treat shadow rays the same as eye rays except:
- They begin at the point of intersection \( R_0 = \{x_i, y_i, z_i\}^T \)
- They are directed toward the light source
  (normalize vector to find \( D_0 \))

If no intersection is found, the point is not in shadow from that light source, compute illumination.
If it is in shadow, only add the ambient term for the light.
Note, no need to compute exact intersection value or normal.

Combining with shading

- Normal is determined by Ray-sphere
- Angle of incidence = angle of reflection

\[
\theta_i \quad \theta_r
\]

\[
\text{light ray} \\
-(s-ray)
\]

\[
\text{reflected ray}
\]

\[
R_n
\]

Sphere
Combining with shading

• Angle $\phi$ is between eye-ray and reflected light ray

For each light source and each color component, the Phong model adds to the illumination as:

$$I' = k_d L_d \cos \theta + k_s L_s \cos^e \phi + k_a L_a$$

Again, shadow ray lets us determine if the point on the sphere is in shadowed by the particular light

If shadowed, do not compute diffuse and specular (only ambient)
Algorithm framework

For each pixel $j$

Determine closest intersection with objects in the scene

If no hits, color pixel $j$ background color

Else

Illumination $j$ for $i = 0$;

For each light $k$

Compute shadow ray intersection

If hit, illumination $j +=$ ambient $k$

Else find illum $k$ and add to illum $j$

end light

end hit

end pixel

Notes:

Illumination is computed for R, G, B values individually

If the intensity goes above 1 for these values, will wash out the color - especially when we have a lot of lights in the scene, may want to re-normalize... but better is to keep contribution of each light small (for example ambient total to be $< .4$ or so)
Intersecting Other Objects

Other objects in the scene follow similar procedures

We will look at polygons next time