Chapter 13

Ray-Tracing

Global Illumination Models

- Simple shading methods simulate local illumination models
- No object interaction
- To simulate global illumination models need more sophisticated & more computation-intensive algorithms
- Ray-tracing deals with
  - Reflectivity
  - Transparency
  - Shadows

Ray-Tracing Algorithm

Reflection and Refraction

\[ c_2 \sin \theta_1 = c_1 \sin \theta_2 \]

Snell’s Law

Sub-Routines

- ReflectRay(r, obj) - computes reflected ray (use obj normal at intersection)
- RefractRay(r, obj) - computes refracted ray
  - Note: ray is inside obj
- Shade(reflect_color, refract_color, obj) - compute illumination given three components

Basic Ray-Tracing Algorithm

```
RayTrace(r, scene)
obj := FirstIntersection(r, scene)
if (no obj) return BackgroundColor;
else begin
  if ( Reflect(obj) ) then
    reflect_color := RayTrace(ReflectRay(r, obj));
  else
    reflect_color := Black;
  if ( Transparent(obj) ) then
    refract_color := RayTrace(RerefactRay(r, obj));
  else
    refract_color := Black;
  return Shade(reflect_color, refract_color, obj);
end;
```
Ray-Object Intersections

- Kernel of ray-tracing ⇒ must be extremely efficient
- Usually involves solving a set of equations

**Example:** Ray-Sphere intersection

- Ray: \( r(t) = p_v + v_t t \)
- Unit sphere: \( x^2 + y^2 + z^2 = 1 \)
- Quadratic equation in \( t \):
  \[
  0 = (p_v + v_t t)^2 + (p_x + v_x t)^2 + (p_y + v_y t)^2 - 1 \\
  = t^2 (v_x^2 + v_y^2 + v_z^2) + 2(p_x v_x + p_y v_y + p_z v_z) t + (p_v^2 + p_x^2 + p_y^2) - 1
  \]

More About Ray-Tracing

- Algorithm above has a BUG….
- Does not terminate
- Termination Criteria
  - No intersection
  - Contribution of secondary ray attenuated below threshold - each reflection/refraction attenuates ray
  - Maximal depth is reached

Optimized Ray-Tracing

- Basic algorithm simple but VERY expensive
- Optimize...
  - Reduce number of rays traced
  - Reduce number of ray-object intersection calculations
- Methods
  - Bounding Boxes
  - Spatial Subdivision
  - Visibility & Intersection
  - Tree Pruning

Simulating Shadows

- Trace ray from each ray-object intersection point to light sources
- If the ray intersects an object in between ⇒ point is shadowed from the light source

```
shadow = RayTrace(LightRay(obj,r,light));
return Shade(shadow,reflect_color,refract_color,obj);
```

Ray-Tracing With Shadows

- Can (not always efficiently) simulate
  - Soft Shadows
  - Fog
  - Frequency Dependent Light (diamonds & prisms)
  - Barely handle S*DS^*
    - S - Specular
    - D - diffuse

Advanced Phenomena