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
Reflection and Refraction

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

Snell’s Law
Basic Ray-Tracing Algorithm

\[ \text{RayTrace}(r, \text{scene}) \]
\[ \text{obj} := \text{FirstIntersection}(r, \text{scene}) \]
\[ \text{if (no obj)} \quad \text{return BackgroundColor;} \]
\[ \text{else begin} \]
\[ \quad \text{if ( Reflect(obj) ) then} \]
\[ \quad \quad \text{reflect\_color} := \text{RayTrace}(\text{ReflectRay}(r, \text{obj})); \]
\[ \quad \text{else} \]
\[ \quad \quad \text{reflect\_color} := \text{Black}; \]
\[ \quad \text{if ( Transparent(obj) ) then} \]
\[ \quad \quad \text{refract\_color} := \text{RayTrace}(\text{RefractRay}(r, \text{obj})); \]
\[ \quad \text{else} \]
\[ \quad \quad \text{refract\_color} := \text{Black}; \]
\[ \quad \text{return Shade(reflect\_color, refract\_color, \text{obj});} \]
\[ \text{end;} \]
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\((\text{reflect\_color, refract\_color, obj})\) – compute illumination given three components
Ray-Object Intersections

- Kernel of ray-tracing \( \Rightarrow \) must be extremely efficient

- Usually involves solving a set of equations

**Example:** Ray-Sphere intersection

ray: \( x(t) = p_x + v_x t, \quad y(t) = p_y + v_y t, \quad z(t) = p_z + v_z t \)

(unit) sphere: \( x^2 + y^2 + z^2 = 1 \)

quadratic equation in \( t \):

\[
0 = (p_x + v_x t)^2 + (p_y + v_y t)^2 + (p_z + v_z t)^2 - 1
\]

\[
= t^2 (v_x^2 + v_y^2 + v_z^2) + 2t(p_x v_x + p_y v_y + p_z v_z)
+ (p_x^2 + p_y^2 + p_z^2) - 1
\]
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

```plaintext
shadow = RayTrace(LightRay(obj,r,light));
return Shade(shadow,reflect_color,refract_color,obj);
```
Ray-Tracing With Shadows

- Eye
- Image Plane
- Light Source
- Reflected Ray
- Refracted Ray
- rayshow
Advanced Phenomena

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