CSE452 Computer Graphics

Lecture 12: Ray Tracing
Review

- **Local Illumination Model (1-hop reflection only)**
  - Non-physical model: “looks good”
  - Ambient, diffuse and specular components

\[
I = I_{amb} + I_{diff} + I_{spec} \\
= I_A k_a + I_L f_{att} (k_d (N \cdot L) + k_s (R \cdot V)^n)
\]
Review

- Drawing **polygons** using local illumination
  - Visibility culling (z-buffer)
  - Shading (flat, Gouraud, and Phong)
  - Texturing
What are we missing?

Local illumination

Global illumination

By Michael Moran, 2000
Ray Tracing

• A global illumination method
  – Shadows
  – Reflection
  – Refraction

Global illumination
By Michael Moran, 2000
What Is Ray Tracing

- **Goal:** Capture multiple hops of light rays
- **Forward ray casting**
  - Trace the path of each ray coming out of the light source
What Is Ray Tracing

• Goal: Capture multiple hops of light rays

• Forward ray casting
  – Trace the path of each ray coming out of the light source
    • Wasteful. Many rays don’t come to the camera

• Backward ray casting
  – Trace backwards in each view direction
    • Initiate one ray per pixel
    • When the ray hits a surface, calculate color using local illumination (if not in shadow), and spawn new rays along reflective and refractive directions
    • Accumulate color for all rays
Backward Ray Casting

When to stop casting?
1. Ray hits no object
2. Maximum recursion depth
3. Contribution to total illumination too small

\[
I(R_0) = I_{\text{local}}(q_1) + k_r^B I(R_1) + k_t^B I(T_1)
\]

\[
I(R_1) = k_r^A I(R_2)
\]

\[
I(T_1) = k_r^B I(R_3) + k_t^B I(T_3)
\]

$k_r$: reflective coefficient
$k_t$: refractive coefficient
Recursive Algorithm

- Main loop

  For each pixel on the screen
  - Form a ray \( L \) from the eye to the pixel
  - \( \text{pixel color} = \text{RayTrace}(L) \)

- Recursive ray-tracer

  \[
  \text{RayTrace}(L) = \begin{cases} 
  \text{Find nearest intersection of } L \text{ with all surfaces} \\
  \text{If no intersection found} \\
  \quad \text{Return 0} \\
  \text{Else} \\
  \quad \text{Compute local illum. } I \text{ at intersection} \\
  \quad \text{Cast reflection ray } R, \text{ refraction ray } T \\
  \quad \text{Return } I + k_r \text{ RayTrace}(R) + k_t \text{ RayTrace}(T) 
  \end{cases}
  \]

- That’s all!
Forming A Ray

- Locating a pixel (i,j) in world coordinates
  - Viewport: w pixels wide, h pixels high
  - 3D pixel location (on the far plane) after WTC transform:
    \[ q_s = \left\{ (i + 0.5) \frac{2}{w} - 1, 1 - (j + 0.5) \frac{2}{h}, -1 \right\} \]
  - 3D pixel location in world coordinates:
    \[ q_w = (S_{xyz} S_{xy} R T)^{-1} q_s = T^{-1} R^{-1} S_{xy}^{-1} S_{xyz}^{-1} q_s \]

- Representing the ray (parametric equation)
  - Eye point: P
    \[ P + t (q_w - P) \]
Ray-Object Intersection

• General approach
  – Represent ray in *parametric* form
    \[ q = P + t d \]
  – Represent surface in *implicit* form
    \[ f[q] = 0 \]
  – Substitute ray into surface, and solve for \( t \) (\( P, d \) are known)
    \[ f[P + t d] = 0 \]
  – Substitute \( t \) back into ray equation, find intersection point \( q \)
    • Use the *smallest positive* \( t \) (to find nearest intersection point)
## Implicit Functions

<table>
<thead>
<tr>
<th>Plane</th>
<th>Sphere</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Plane Diagram" /></td>
<td><img src="image" alt="Sphere Diagram" /></td>
</tr>
<tr>
<td>$f[q] = (q - g) \cdot n$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cylinder (no cap)</th>
<th>Cone (no cap)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Cylinder Diagram" /></td>
<td><img src="image" alt="Cone Diagram" /></td>
</tr>
</tbody>
</table>
Example: Ray-Plane Intersects

- Plane definition
  \[ f[q] = (q - g) \cdot n = 0 \]

- Substituting ray into equation and solve
  \[ f[P + td] = (P + td - g) \cdot n = 0 \]
  \[ t = \frac{(g - p) \cdot n}{d \cdot n} \]

- Substitute t back and find intersection
  \[ q = P + td = P + \frac{(g - p) \cdot n}{d \cdot n} \cdot d \]
Example: Ray-Plane Intersects

- Plane definition
  
  \[ f[q] = (q - g) \cdot n = 0 \]

- Substituting ray into equation and solve
  
  \[ f[P + td] = (P + td - g) \cdot n = 0 \]
  
  \[ t = \frac{(g - p) \cdot n}{d \cdot n} \]

- Substitute \( t \) back and find intersection
  
  \[ q = P + td = P + \frac{(g - p) \cdot n}{d \cdot n} \cdot d \]

- No intersection if
  
  - \( d \cdot n = 0 \) (Ray parallel to plane) or
  
  - \( t < 0 \) (behind the ray origin).
# Implicit Functions

<table>
<thead>
<tr>
<th></th>
<th>Plane</th>
<th>Sphere</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f[q]$</td>
<td>$(q - g) \cdot n$</td>
<td>$(q - g)^2 - r^2$</td>
</tr>
<tr>
<td><strong>a.</strong></td>
<td><img src="image" alt="Plane" /></td>
<td><img src="image" alt="Sphere" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Cylinder (no cap)</th>
<th>Cone (no cap)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g$</td>
<td>![Cylinder (no cap)]</td>
<td>![Cone (no cap)]</td>
</tr>
<tr>
<td>$n$</td>
<td>![Cylinder (no cap)]</td>
<td>![Cone (no cap)]</td>
</tr>
<tr>
<td>$r$</td>
<td>![Cylinder (no cap)]</td>
<td>![Cone (no cap)]</td>
</tr>
</tbody>
</table>
Example: Ray-Sphere Intersects

- Sphere definition

\[ f[q] = (q - g)^2 - r^2 = 0 \]

- Substituting ray into equation and solve

\[ f[P + td] = (P + td - g)^2 - r^2 = At^2 + Bt + C = 0 \]

where \( A = d^2 \), \( B = 2d \cdot (P - g) \), \( C = (P - g)^2 - r^2 \)

\[ t_1 = \frac{-B - \sqrt{B^2 - 4AC}}{2A}, \quad t_2 = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \quad (t_1 \leq t_2) \]
Example: Ray-Sphere Intersects

- Sphere definition

\[ f[q] = (q - g)^2 - r^2 = 0 \]

- Substituting ray into equation and solve

\[
\begin{align*}
  f[P + td] &= (P + td - g)^2 - r^2 = At^2 + Bt + C = 0 \\
  \text{where } A &= d^2, \ B = 2d \cdot (P - g), \ C = (P - g)^2 - r^2 \\
  t_1 &= \frac{-B - \sqrt{B^2 - 4AC}}{2A}, \ t_2 = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \quad (t_1 \leq t_2)
\end{align*}
\]

- Find Intersection

No intersection if \( B^2 < 4AC \) or \( t_2 < 0 \)

If \( t_1 < 0 \) and \( t_2 > 0 \) : \( q = P + t_2 d \)

If \( t_1 > 0 \) : \( q = P + t_1 d \)
### Implicit Functions

<table>
<thead>
<tr>
<th>Plane</th>
<th>Sphere</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f[q] = (q - g) \cdot n )</td>
<td>( f[q] = (q - g)^2 - r^2 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cylinder (no cap)</th>
<th>Cone (no cap)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v = q - g )</td>
<td>( v = q - g )</td>
</tr>
<tr>
<td>( f[q] = (v - (v \cdot n) n)^2 - r^2 )</td>
<td>( f[q] = (v - (v \cdot n) n)^2 - r^2 (v \cdot n)^2 )</td>
</tr>
</tbody>
</table>
Constructive Solid Geometry (CSG)
Constructive Solid Geometry (CSG)
Constructive Solid Geometry (CSG)

• How to do ray to CSG intersection?
Constructive Solid Geometry (CSG)

- Call A: cube, B: sphere
- This is A – B
- Find intersections with A
  - Throw away if inside B
- Find intersections with B
  - Throw away if outside A
Constructive Solid Geometry (CSG)

- Boolean results of basic shapes
  - Example: capped cylinder = cylinder ∩ 2 half-spaces
  - Defined by multiple implicit functions

- Side: \( f(q) = ((q - g) - ((q - g) \cdot n) n)^2 - r^2 \)
- Top: \( f_1(q) = (q - g - h n) \cdot n \)
- Bottom: \( f_2(q) = (q - g + h n) \cdot n \)

- Point \( q \) lies inside or on the capped cylinder if
  \[ f(q) \leq 0 \land f_1(q) \leq 0 \land f_2(q) \geq 0 \]
Ray Intersection with CSG

- Example: a capped cylinder
  - A point \( q \) lies on the top plane if
    \[ f_1[q] = 0, \ f[q] \leq 0 \]
  - A point \( q \) lies on the bottom plane if
    \[ f_2[q] = 0, \ f[q] \leq 0 \]
  - A point \( q \) lies on the side if
    \[ f[q] = 0, \ f_1[q] \leq 0 \land f_2[q] \geq 0 \]
Constructive Solid Geometry (CSG)

- Example: A capped cone
  - How many implicit functions define this shape?
    - Infinite cone: \( f[q] = (v - (v \cdot n) n)^2 - r^2 (v \cdot n)^2 \), where \( v = q - g \)
    - Top plane: \( f_1[q] = (q - g) \cdot n \)
    - Bottom plane: \( f_2[q] = (q - g - h n) \cdot n \)
  - Point \( q \) lies inside or on the capped cone if
    \[ f[p] \leq 0 \land f_1[p] \leq 0 \land f_2[p] \geq 0 \]
  - Point \( q \) lies
    - On the side if: \( f[q] = 0, f_1[q] \leq 0 \land f_2[q] \geq 0 \)
    - On the base if: \( f_2[q] = 0, f[q] \leq 0 \)
Triangles

• First test if ray intersects the plane
  – Triangle vertices $g_1, g_2, g_3$
  – Plane defined by point $g_1$ and normal $n = (g_2 - g_1) \times (g_3 - g_2)$

• Next test if the intersection lies in the triangle
  – Let intersection be $q$
  – If $q$ lies inside, the triangles $qg_i g_{i+1}$ have same orientation as the triangle $g_1 g_2 g_3$

$$n \cdot ((g_{i+1} - g_i) \times (q - g_{i+1})) > 0$$
$$i = 1, 2, 3$$
When Ray Hits A Surface...

• Compute local illumination at the intersection
  — If not occluded, compute diffuse and specular light
  — Add ambient light

• Cast more rays and keep tracing
  — Reflected ray (if the reflection coefficient is not zero)
  — Refracted ray (if the refraction coefficient is not zero)

• Sum up all illumination along traced rays
Computing Illumination

- Local illumination at intersection
  - Ambient reflection: \( I_{amb} = I_A k_a \)

  - Cast a *shadow ray* to each light source
    - A light source is *visible* if the ray is unblocked

- For each visible light source \( i \):
  - Diffuse reflection: \( I_{i,\text{diff}} = I_i f_{\text{att}} k_d (N \cdot L_i) \)
  - Specular reflection: \( I_{i,\text{spec}} = I_i f_{\text{att}} k_s (R_i \cdot V)^n \)

- Together:
  \[
  I_{local} = I_{amb} + \sum_{\text{visible source } i} (I_{i,\text{diff}} + I_{i,\text{spec}})
  \]
Reflection Ray

- Mirrored by the surface normal

\[ \mathbf{v} = (\mathbf{L} \cdot \mathbf{n}) \mathbf{n} \]
\[ \mathbf{h} = \mathbf{v} - \mathbf{L} \]
\[ \mathbf{R} = \mathbf{L} + 2 \mathbf{h} = 2 (\mathbf{L} \cdot \mathbf{n}) \mathbf{n} - \mathbf{L} \]
Refraction Ray

- **Snell’s Law**
  
  \[
  \frac{\sin(\alpha)}{\sin(\beta)} = \frac{\eta_B}{\eta_A}
  \]

  - \(\eta_A, \eta_B\): refraction index (speed of light in vacuum / speed of light in that material)
  - Compute refracted ray \(T\):
    
    \[
    T = \frac{\tan(\beta)}{\tan(\alpha)} h - v
    \]
Recursive Algorithm

- Main loop

  For each pixel on the screen
  
  Form a ray $L$ from the eye to the pixel
  
  pixel color = RayTrace($L$)

- Recursive ray-tracer

  RayTrace($L$)
  
  Find nearest intersection of $L$ with all surfaces
  
  If no intersection found
    
    Return 0
  
  Else
    
    Compute local illum. $I$ at intersection
    
    Cast reflection ray $R$, refraction ray $T$
    
    Return $I + k_r \text{ RayTrace}(R) + k_t \text{ RayTrace}(T)$
Examples

- Internet Ray Tracing Competition (irtc.org)

First Place, January-February 2006
Examples

- Internet Ray Tracing Competition (irtc.org)

Third Place, January-February 2006
Speed Up Ray Intersection

- Bounding boxes
  - Using coarse bounding objects for intersection first
    - If no intersection, then ignore the entire object
    - If yes, then intersect with the actual object
  - Types
    - Sphere (ellipsoid)
    - Axes-aligned bounding boxes (AABB)
    - Oriented bounding boxes (OBB)
  - Often hierarchical

An OBB tree
Speed Up Ray Intersection

• Spatial partitioning
  – Divide space up into small cells
    • Record objects in each cell
    • Trace cells along the ray, intersect only with objects in the cells
  – Types
    • Uniform 3D lattice
    • Adaptive lattice (octree, k-d tree)
    • Binary space partitioning (BSP)

An octree