Ray Tracing

Outline

• Ray
  - Forward
  - Backward
• Intersection
• Shadow
• Ray tracing basics
Illumination models

• Interaction between light sources and objects in scene that results in perception of intensity and color at eye

• Local vs. global models
  - Local illumination: Perception of a particular primitive only depends on light sources directly affecting that one primitive
    • Geometry
    • Material properties
  - Global illumination: Also take into account indirect effects on light of other objects in the scene
    • Shadows cast
    • Light reflected/refracted

Forward Ray “Following”

• Proper global illumination means simulation of physics of light
  - Rays are emitted from light source, bounce off objects in the scene, and some eventually hit our eye, forming an image

• Problem: Not many rays make it to the image
  - Waste of computation for those that don’t
Backward Ray “Following”

- Idea: Only consider those rays that **do** create the image—where did they come from?

Backward Ray “Following”: Types

- **Ray casting**: Compute illumination at first intersected surface point only
- **Ray tracing**: Recursively spawn rays at hit points to simulate reflection, refraction
Ray Casting

• Simulation of irradiance (incoming light ray) at each pixel
• Send a ray from the focal point through each pixel and out into the scene until it intersects an object
  - “Background” color if nothing hit
• Local shading model is applied to first point hit
  - Easy to apply exact rather than faceted shading model to objects for which we have an analytic description (spheres, cones, cylinders, etc.)

Ray Casting: Details

• Must compute 3-D ray into scene for each 2-D image pixel (Chap. 14.2 of Hill)
• Compute 3-D position of ray’s intersection with nearest object and normal at that point
• Apply shading model such as Phong to get color at that point and fill in pixel with it
Phong Illumination Model

• Let $c = (r, g, b, \alpha)$ be material color, $s(l)$ be color of light $l$

• Sum over all lights $l$ for each color channel (clamp overflow to $[0, 1]$):

$$c_{total} = \sum_l c_{amb}(l) + c_{diff}(l) + c_{spec}(l)$$

$$c_{amb}(l) = m_{amb} \otimes s_{amb}(l)$$

$$c_{diff}(l) = \max(0, n \cdot l(l)) m_{diff} \otimes s_{diff}(l)$$

$$c_{spec}(l) = \max(0, v \cdot r(l))^{\text{shine}} m_{spec} \otimes s_{spec}(l)$$

Does Ray Intersect any Scene Primitives?

• Test each primitive in scene for intersection individually

• Different methods for different kinds of primitives
  - Polygon
  - Sphere
  - Etc.

• Make sure intersection point is nearest one and in front of eye
Ray-Polygon Intersection

- First, does ray intersect plane $\pi$ of polygon?
- If ray does intersect $\pi$ at 3-D point $p$, is the 2-D location of $p$ on $\pi$ inside the 2-D polygon?
  - Every point on plane $\pi$ is expressible as linear combination of 3-D axis vectors: $p = xu + yv$, where normal $n = u \times v$
  - $(x, y)$ are 2-D coordinates of point on plane
  - Point-in-polygon test: Like Cohen-Sutherland clipping—AND together "correct side of line" tests for each 2-D polygon edge

Ray-Plane Intersection (from BSP lecture)

- Parametrize plane from polygon
  - Cross product of edges gets plane normal $n = (a, b, c)^T$
  - Solve for $d$ from single point on plane $(x, y, z)^T$ via plane equation $ax + by + cz + d = 0$
- Ray-plane intersection
  - Express point $p$ on a ray as some distance $t$ along direction $d$ from origin $o$: $p = o + td$ for positive $t$
  - Use plane equation $n \cdot x + d = 0$, substitute $o + td$ for $x$, and solve for $t$
  - Then plug $t$ back into $p = o + td$ to get $p$
Polygon normals

• Let polygon vertices \(v_0, v_1, v_2, \ldots, v_{n-1}\) be in counterclockwise order and co-planar
• Calculate normal with cross product:
  \[ \mathbf{n} = (v_1 - v_0) \times (v_{n-1} - v_0) \]
• Normalize to unit vector with \(\mathbf{n}/\|\mathbf{n}\|\)

Ray-Sphere Intersection I

• Combine implicit definition of sphere
  \[ |\mathbf{p} - \mathbf{p}_c|^2 - r^2 = 0 \]
  with ray equation
  \[ \mathbf{p} = \mathbf{o} + t\mathbf{d} \]
  (where \(\mathbf{d}\) is a unit vector) to get:
  \[ |\mathbf{o} + t\mathbf{d} - \mathbf{p}_c|^2 - r^2 = 0 \]
Ray-Sphere Intersection II

- Substitute $\Delta \mathbf{p} = \mathbf{p}_c - \mathbf{o}$ and use identity $|\mathbf{a} + \mathbf{b}|^2 = |\mathbf{a}|^2 + 2\mathbf{a} \cdot \mathbf{b} + |\mathbf{b}|^2$ to solve for $t$, resulting in a quadratic equation with roots given by:

$$t = d \cdot \Delta \mathbf{p} \pm \sqrt{(d \cdot \Delta \mathbf{p})^2 - (|\Delta \mathbf{p}|^2 - r^2)}$$

- Notes
  - Real solutions mean there actually are 1 or 2 intersections
  - Negative solutions are behind eye

Shadow Rays

- For point being locally shaded, must spawn new ray in each light direction and check for intersection to make sure light is “visible”
Ray-Cast Scene with and without Shadows

Ray Tracing

- **Recursively** cast rays from hit points to compute:
  - Reflections on specular objects
    - Spawn ray along reflection direction
  - Transmission through transparent object
    - Spawn ray along refraction direction
Ray Tracing: Recursion

a)

b)

Ray Tracing: Example