CMSC 427: Chapter 13
Ray Tracing

Reading: Chapt 4 in Shirley.
Overview:
- Ray Tracing: Basic approach, recursive ray tracing.
- Reflection and refraction
- Ray-Object intersections
- Distributed ray tracing
- Accelerating ray tracing

Overview

- Basic Elements
- Reflection, Transmission, Lighting
- Computing Intersections
- Fine points: Antialiasing, acceleration
Rendering: Adding Realism

So far we have studied rendering with the Phong lighting:

Local illumination model: supports:
  - ambient
  - diffuse
  - specular
  - no inter-object light transmission

Shadows are not supported: but can be added through:
  - real-time shadow rendering
  - light maps
  - shadow volumes

Reflections not supported: but we can use environment mapping:
  - only an approximation to true reflection.
  - expensive if there are many reflective objects.
  - cannot capture complex inter-object reflections.

Chapter 13, Slide 3
Copyright © D. M. Mount and A. Varshney

Ray Tracing

Ray Tracing:
- Ray tracing is a conceptually simple methods for synthesizing highly realistic images.
- It is based on tracing light rays backwards from the eye to the objects of the scene and then to the light sources.

Unlike real-time systems such as OpenGL:
- It naturally incorporates elements of a global lighting model, including accurate rendering of:
  - shadows
  - reflective surfaces
  - transparent surfaces
- It is slow and is typically used off-line (not interactively), but it is useful for generating realistic texture maps and environment maps.
- Combined with GPU acceleration, simple ray-tracers can achieve nearly frame-rate speeds.

Chapter 13, Slide 4
Copyright © D. M. Mount and A. Varshney
Ray Tracing: Basic Idea

Motivation:
- Light travels from the light sources, reflects off objects, and eventually enters our eye.
- But forward tracing of light rays would result in many wasted rays that never reach the eye.
- Instead, we trace rays backwards from eye towards the objects of the scene.
Ray Tracing: Basic Idea

Basic Idea:
- Cast a ray through each of the pixels of the image plane.
- Find the first intersection with objects of scene.
- Cast rays to each light source to determine illumination. If we hit an object before the light source, we are in a shadow.

Reflection and Transmission

Reflection and Transmission:
- In contrast to environment maps, ray tracing can accurately simulate complex reflection and transmission (with refraction).
- When the ray hits a reflective or transparent object, we compute the reflection and/or refraction rays, trace them recursively, and blend the associated colors.
Recursive Ray Tracing

Ray Tree: Recursion tree for ray shoots.
- Reflection rays (R)
- Transmission rays (T)
- Light rays (also called shadow rays) (s)

In practice the recursion depth is limited, say, to a maximum of 6 levels.

Recursive Ray Tracing: Pseudo-code

rayTrace ( ): Given camera parameters, scene, and image size:
  a) generate eye ray Rij from eye through the center of each pixel [i, j].
  b) color[i, j] ← trace ( R ).

trace ( Ray R ):
  a) shoot ray R into the scene; let X ← first object hit; let p ← contact point.
  b) if X is reflective:
     • compute the reflection ray Rr at p. Let Cr ← trace ( Rr ).
  c) if X is transparent:
     • compute the transmission ray (refraction) Rt at p. Let Ct ← trace ( Rt ).
  d) for each light source L:
     • shoot light ray Rl from p to L.
     • if Rl does not hit any object until reaching L
       - apply the (Phong) lighting model to determine the color Cl at this point.
  e) blend colors Cr, Ct, and Cl (for all lights) to determine the final color C.
  f) return C.
The Power of Ray Tracing

Ray tracing is a very powerful general rendering model. The method simultaneously captures the notions of:

- **Scan-conversion**: by shooting rays through each pixel of the image plane.
- **Hidden surface removal**: since we only keep the closest object hit by the ray.
- **Global illumination**: we can easily model shadows, reflection, and transmission since we have access to the complete 3-d scene.
- **Anti-aliasing**: can easily be incorporated by shooting multiple rays per pixel (super-sampling) and averaging the results.

Other realistic elements can be incorporated:

- **Depth of Field**: Can generate regions that are out of focus.
- **Motion blur**: Blurring to create the illusion of motion.

Image source Hiroshima University

Ray Tracing: Elements

Key Elements to Ray Tracing Design:

1. Ray generation.
2. Computing ray-object intersections.
3. Computing reflection rays.
5. Computing shading.

We will discuss items 2-4 in detail.

- Ray generation is straightforward given the image dimensions, and the camera set-up. (Makes for a good exercise.) The first step is computing the camera view-frame.
- Shading can be computed using the Phong illumination model.
- (Both were discussed earlier this semester.)
Basic Representations

Ray Representation: A ray is represented by a pair \( R = (p, u) \), where
- \( p \) is the origin point of the ray.
- \( u \) is the directional vector (unit length) of the ray.

Point on ray: is represented as a scalar parameter \( t \geq 0 \), where
- \( R(t) = p + t u \).

Trimming: When a ray hits an object at point \( R(t) \), we store this as a trimmed ray by storing the pair \( (R, t) \).

In-Class Exercise

Generating a ray through pixel \([i,j]\):
- Suppose that you have already computed a view frame centered at point eye, with axes \( v_x \) (viewer right), \( v_y \) (viewer up), and \( v_z \) (negation of view direction).
- The image plane is 1-unit in front of the viewer.
- The \( y \)-field of view is \( \text{fovy} \) degrees.
- The image size is \( \text{width} \times \text{height} \). Origin in upper-left corner.
- What is the ray through pixel \([i,j]\)?
Overview

- Basic Elements
- Reflection, Transmission, Lighting
- Computing Intersections
- Fine points: Anti-aliasing, acceleration

Reflection Ray

**Reflection Ray:** Earlier this semester we showed how to compute a reflection vector. The same principle applies here.
- Suppose that the ray $R = (p, u)$ hits the surface at $R(t) = q$.
- Let $u$ be the (normalized) **ray direction**, and let $v = -u$.
- Let $n$ be the (normalized) **surface normal**.
- Recall that the **reflection vector** is:
  $$ r = 2(v \cdot n)n - v. $$
- The **reflection ray** is $R_r = (q, r)$. 
Transmission Ray (Refraction)

Refraction Ray: Recall that light refraction occurs when a light ray makes the transition between two media in which the speed of light differs. (E.g. from air into water.)

Index of Refraction: From physics this is defined to be the ratio of the speed of light of in a vacuum to the speed in the medium.

- Air (vacuum): $\eta = 1.0$
- Water: $\eta = 1.333$
- Glass: $\eta = 1.5$
- Diamond: $\eta = 2.47$

Geometric Optics: We ignore wavelength effects (rainbows).

Transmission Ray (Refraction)

Snell's Law: The ratio of the sines of the incidence angles is inversely proportional to the ratios of the indices of refraction.

$$\frac{\sin \theta_i}{\sin \theta_t} = \frac{\eta_t}{\eta_i}$$

Total Internal Reflection: When moving from high to low index of refraction, it is possible that refraction ray is reflected back.
**Transmission Ray (Refraction)**

**Refraction Ray:**
- Suppose that the ray \( R = (p, u) \), hits the surface at \( R(t) = q \).
- Let \( u \) be the (normalized) ray direction, and let \( v = -u \).
- Let \( n \) be the (normalized) surface normal.
- Our goal is to compute the transmission vector, \( t \).

**Intermediates:**
- Let \( m_i \) be the projection of \( v \) onto \( n \):
  \[ m_i = (v \cdot n) n = (\cos \theta_i) n \]
  and let \( w_i = m_i - v \).
- Since \( v \) and \( t \) are unit length we have
  \[ \frac{n_i}{n} = \frac{\sin \theta_i}{\sin \theta_t} = \frac{|w_i|/|v|}{|w_t|/|t|} = \frac{|w_i|}{|w_t|} \]

**Intermediates: (continued)**
- Thus:
  \[ \frac{n_i}{n} = \frac{\sin \theta_i}{\sin \theta_t} = \frac{|w_i|/|v|}{|w_t|/|t|} = \frac{|w_i|}{|w_t|} \]
- Since \( w_i \) and \( w_t \) are parallel we have
  \[ w_t = \frac{n}{n_i} w_i = \frac{n}{n_i} (m_i - v) \]
- The projection of \( t \) onto \( -n \) is \( m_t = -(\cos \theta_t) n \).
Transmission Ray (Refraction)

Putting it All Together:
- **Given:** \( m \) and \( w \), we can now express the desired vector \( t \) as

\[
\begin{align*}
t &= w_t + m_t = \frac{n}{n_t} (m_t - v) - (\cos \theta_i) n = \frac{n}{n_t} ((\cos \theta_i) n - v) - (\cos \theta_i) n \\
&= \left( \frac{n}{n_t} \cos \theta_i - \cos \theta_i \right) n - \frac{n}{n_t} v.
\end{align*}
\]

- We know that \( \cos \theta_i = (v \cdot n) \), so the only remaining unknown is:

\[
\cos \theta_t = \sqrt{1 - \sin^2 \theta_t} = \sqrt{1 - \left( \frac{n}{n_t} \right)^2 \sin^2 \theta_i}
\]

\[
= \sqrt{1 - \left( \frac{n}{n_t} \right)^2 (1 - \cos^2 \theta_i)} = \sqrt{1 - \left( \frac{n}{n_t} \right)^2 (1 - (v \cdot n)^2)}
\]

- Plugging \( \cos \theta_i \) and \( \cos \theta_t \) into the above formula gives \( t \).

Transmission Ray (Refraction)

Final Considerations:
- What if the quantity in the square root is negative?
- This is a case of **total internal reflection**. In this case just compute the reflection ray, as given above, and shoot this rather than the transmission ray.
- Watch out for objects that are both reflective and transparent, since we don't want to combine the effects of shooting this same ray twice.
Lighting and Shading

**Enhanced Phong Model:**
Light sources: \( L_1, L_2, \ldots, L_n \) are RGB intensities of light sources. Let \( L_a \) denote the ambient intensity.
Visibility: Let \( \text{Vis}(i, p) \) be a function, which returns 1, if point \( p \) is visible to light source \( i \), and 0 otherwise.
Material color: Let \( C \) be the RGB value of the surface color.
Reflection: Let \( \rho_a, \rho_d, \) and \( \rho_s \) be the coefficients of ambient, diffuse, and specular reflection. Let \( \alpha \) be the specular strength.
Light properties: Let \( n, l, \) and \( h \) be surface normal, light vector, and halfway vector. Let \( d \) be the distance to \( L_i \). (Recall Lecture 5.)
Attenuation: Let \( a, b, c \) be attenuation parameters. (Recall Lecture 5.)
Reflection/Transmission: Let \( \rho_r \) and \( \rho_t \) be the coefficients of reflection and transmission. Let \( r \) and \( t \) be reflection/trans vectors.

**Phong Lighting Equation:**
\[
I = \rho_a L_a C + \sum_{i} \frac{\text{Vis}(P,i) L_i}{a + bd + cd^2} \left[ \rho_d \cdot C \cdot \max(0, n \cdot l) + \rho_s \cdot \max(0, n \cdot h)^\alpha \right] + \\
\rho_s \cdot \text{trace}(P,r) + \rho_t \cdot \text{trace}(P,t).
\]

Overview

- Basic Elements
- Reflection, Transmission, Lighting
  - Computing Intersections
- Fine points: Antialiasing, acceleration
Ray-Sphere Intersection

Ray-Sphere Intersection:
- Perhaps the easiest of all ray intersection computations.
- The basic approach applies to other implicit surface representations.

Problem Setup:
- Let $R: (p, u)$ be the ray. Recall that $R(t) = p + tu$, and $|u| = 1$.
- Let $c = (c_x, c_y, c_z)$ be the center of the sphere and $r$ be its radius.

Objective:
- Compute $t \geq 0$ such that $R(t)$ is the first intersection point of the ray with the sphere.
- Compute the surface normal $n$ at this intersection point (for lighting).

Intersection Derivation:
- Let $|w|$ denote the length of a vector $w$.
- The point $R(t)$ hits the sphere iff
  \[ r = |R(t) - c| = |(p + tu) - c| \]
- First, define the vector $v = p - c$. We want:
  \[ r^2 = |v + tu|^2 = (v + tu) \cdot (v + tu) \]
- Expanding:
  \[ r^2 = (v \cdot v) + 2t(v \cdot u) + t^2(u \cdot u) \]
  \[ 0 = (u \cdot u)t^2 + 2(v \cdot u)t + ((v \cdot v) - r^2) = at^2 + bt + c \]
  where $a = (u \cdot u)$, $b = 2(v \cdot u)$, and $c = (v \cdot v) - r^2$. Note that $a = 1$, because $u$ is a unit-length vector.
- This is a quadratic equation, whose real roots are:
  \[ t^- = \frac{-b - \sqrt{b^2 - 4c}}{2} \] and \[ t^+ = \frac{-b + \sqrt{b^2 - 4c}}{2} \]
Ray-Sphere Intersection

Why two roots?
- Ray may hit the sphere in 0, 1, or 2 points.
- How do we determine whether there is an intersection, and which root to take?

Case Analysis:
\[ t = \frac{-b - \sqrt{b^2 - 4c}}{2} \quad \text{and} \quad t' = \frac{-b + \sqrt{b^2 - 4c}}{2} \]

- If the discriminant \( b^2 - 4c \) is negative, there are no real roots. The ray misses the sphere.
- If both roots are negative, the sphere is behind the ray.
- If \( t^- < 0 \) and \( t^+ > 0 \), the ray origin lies inside the sphere. Take \( t^+ \).
- Otherwise, both roots positive, take \( t^- \).

Surface Normals: Implicit Surfaces

Surface Normal at Contact Point:
- Computing the normal vector for a point \( q \) on a sphere is easy; it is the vector normalize(\( q - c \)), from the center \( c \) of the sphere to \( q \).
- How do we compute normal vectors for general surfaces?

Normal Vectors for Implicit Representation Surfaces:
- Consider a surface given by an implicit representation \( f(x, y, z) = 0 \).
- The surface normal can be computed as the gradient of \( f \). In particular,
  \[ \n = \text{normalize}(\nabla), \quad \text{where} \quad \nabla = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{pmatrix} \]
- Evaluate this at the point of contact.
- Need to adjust sign depending on whether ray hits from inside or outside.
Surface Normals: Implicit Surfaces

Example: Surface normal for sphere:
- \( f(x, y, z) = (x-c_x)^2 + (y-c_y)^2 + (z-c_z)^2 - r^2 = 0. \)
- \( \nabla (x, y, z) = (2(x-c_x), 2(y-c_y), 2(z-c_z)). \)
- The normal vector at \( q \) is normalize\((q_x - c_x, q_y - c_y, q_z - c_z)\).

Example: Surface normal of a paraboloid:
- \( f(x, y, z) = 3(x-1)^2 + 2y^2 + 5z + 7 = 0. \)
- Thus: \( f(x, y, z) = (3x^2-6x+3) + 2y^2 + 5z + 7 = 0. \)
- \( \nabla (x, y, z) = (6x-6, 4y, 5). \)
- The normal vector at \( q \) is normalize\(((6q_x - 6, 4q_y, 5)). \)

Surface Normals: Parametric Surfaces

Normal Vectors for Parametric Surfaces:
- Consider a surface given by a parametric representation:
  \[ \phi(u,v) = \begin{pmatrix} \phi_x(u,v) \\ \phi_y(u,v) \\ \phi_z(u,v) \end{pmatrix}. \]
- To compute the surface normal we first compute the partial derivatives with respect to the parameter variables:
  \[ \frac{\partial \phi}{\partial u} = \begin{pmatrix} \frac{\partial \phi_x}{\partial u} \\ \frac{\partial \phi_y}{\partial u} \\ \frac{\partial \phi_z}{\partial u} \end{pmatrix} \quad \text{and} \quad \frac{\partial \phi}{\partial v} = \begin{pmatrix} \frac{\partial \phi_x}{\partial v} \\ \frac{\partial \phi_y}{\partial v} \\ \frac{\partial \phi_z}{\partial v} \end{pmatrix}. \]
- The final normal is the cross product of these partials (evaluated at the \((u, v)\) values of the contact point):
  \[ n = \text{normalize} \left( \frac{\partial \phi}{\partial u} \times \frac{\partial \phi}{\partial v} \right). \]
Surface Normals: Parametric Surfaces

Example: Surface Normal of a Torus:
- Consider a torus at the origin with major radius $R$ (about the z-axis) and minor radius $r$. It has the parametric representation:
  \[ \phi_s(u,v) = (R + r \cdot \cos v) \cos u \]
  \[ \phi_t(u,v) = (R + r \cdot \cos v) \sin u \]
  \[ \phi_n(u,v) = r \cdot \sin v. \]
- The partial derivatives are:
  \[ \frac{\partial \phi}{\partial u} = \begin{pmatrix} -(R + r \cdot \cos v)(\sin u) \\ (R + r \cdot \cos v)(\cos u) \\ 0 \end{pmatrix} \quad \text{and} \quad \frac{\partial \phi}{\partial v} = \begin{pmatrix} -r \cdot (\sin v)(\cos u) \\ -r \cdot (\sin v)(\sin u) \\ r \cdot \cos v \end{pmatrix}. \]
- The final surface normal is their cross product:
  \[ n = \text{normalize} \left( \frac{\partial \phi}{\partial u} \times \frac{\partial \phi}{\partial v} \right). \]

Intersection with Transformed Objects

Transformed Objects:
- We know how to do ray-sphere intersection. How about an ellipse?
- Consider an ellipse $E$ and a ray $R(t) = p + t \cdot u$.
- We want to determine the smallest positive $t$ such that $p + t \cdot u \in E$. 

\[ E \]
\[ R(t) \]
\[ p \]
\[ u \]
Intersection with Transformed Objects

Observation:
- An ellipse $E$ is the image of a sphere $S$ under some affine transformation (scaled, rotated, translated).
- In homogeneous coordinates, we can express this as $E = T \cdot S$, where $T$ is an appropriate 4x4 matrix.

Cute Trick: Transform the ray, not the object!
- Let $T^{-1}$ denote the inverse of $T$, thus $S = T^{-1}E$.
- We want to find the smallest $t$ such that:
  \[ p + tu \in E \iff T^{-1}(p + tu) \in T^{-1}E = S \]
  \[ \iff (T^{-1}p) + t(T^{-1}u) \in S. \]
- Let $p' = T^{-1}p$ and let $u' = T^{-1}u$. Rather than writing a new ray-ellipse intersection function, we simply shoot ray $R': (p', u')$ at sphere $S$.
- Note: Care is needed to transform the normal back properly.
Do You Understand This?

Try the following exercises to check that you understand this:

- An infinite cylinder of radius $r$ centered about the z-axis:
  - What is the implicit function of this shape?
  - Express the intersection $t$ values as a quadratic equation.
  - Which root do you take as the intersection point?
  - What is the surface normal at the contact point?
- Repeat the above, but for a finite cylinder running from $z=0$ to $z=h$. (Hint: Derive an intersection test for a flat circular disk.)
- Repeat the above with an infinite cone whose central axis is the z-axis, and whose central angle with respect to the axis is $\theta$.
- Repeat the above cube of side length $s$ centered at the origin.

Overview

- Basic Elements
- Reflection, Transmission, Lighting
- Computing Intersections
  - Fine points: Antialiasing, acceleration
Anti-aliasing and Ray Tracing

Single-sample per pixel:
- May cause aliasing artifacts.
- Reflections/refractions can result high frequency variations even if objects are smooth (of low frequency).

Super Sampling:
Shoot multiple rays per pixel:
- regular grid
- random jittering
- Poisson disk sampling (random but filtering out pairs that are too close)

Average:
- unweighted
- weighted (more heavily near pixel center)

Adaptive Sampling:
- If high variation amongst super-sampled rays per pixel, shoot more.

Distributed Ray Tracing

Distributed Ray Tracing (Cook 1986):
- Ray-traced images tend to appear too sharp, too crisp.
- A simple idea, which can be applied to generate a number of different blurring effects.

Variations: We randomly perturb (or jitter) rays and average the results. The manner of perturbation results in different effects:
- Blurred reflections: jitter reflection rays.
- Convincing translucency: jitter transmission rays.
- Soft shadows (for area light sources): jitter shadow rays.
- Stochastic anti-aliasing: jitter primary rays over each pixel.
- Motion blur: jitter primary rays over time.
- Depth of field effects: jitter primary rays across a virtual lens.

Chapter 13, Slide 39
Copyright © D. M. Mount and A. Varshney

Chapter 13, Slide 40
Copyright © D. M. Mount and A. Varshney
Accelerating Ray-Tracing

Where the time goes:
- Most of the time in a simple ray-tracer is spent in ray-object intersection tests.
- We are only interested in the first object hit by a ray, is there a better way than testing against every object in the scene?

Spatial Hierarchies:
- Octrees Trees: Regular decomposition into axis-aligned cubes.
- k-d Trees: Decomposition into axis-aligned hyper-rectangles.
- BSP Trees: Decomposition using arbitrary splitting planes. (Binary space partition.)

Bounding Volume Hierarchies:
- AABBs: Axis-aligned bounding boxes.
- OBBs: Oriented bounding boxes.
- Bounding spheres/ellipsoids.

Octrees

Octrees: First used in ray tracing by Glassner.
Construction:
- Root is associated with a bounding cube for entire scene.
- Split about midpoint into 8 subcubes, each is a child.
- Construction terminates when the number of objects in cell is small, or a maximum depth is reached.

Answering a Ray-Shooting Query:
- Start at root (bounding box for entire scene).
- Test for intersections with child boxes.
- Recursively visit all child nodes whose box is hit (in increasing order of distance along the ray).
- When arriving at a leaf, test for intersection against all objects stored in this node.
- Return t value of closest hit.
Octrees

Advantages:
- Easy to implement.
- Low storage overhead.
- Efficiently prunes the vast majority of objects from consideration, allowing the algorithm to focus on the most promising objects.
- Ray-traversal can be highly optimized [Revelles 2000]

Disadvantages:
- Potentially very unbalanced. (Imagine a small soccer ball in a large stadium.)
- Long thin objects may overlap many cells, thus increasing storage. Cubes cannot adapt their shape to handle thin objects.

Octrees: Issues

Determining the closest intersection:
- When R enters leaf A it will be tested against both \( w_0 \) and \( w_2 \)
- The ray intersects \( w_0 \), but this is not the correct (first) intersection. (It hits \( w_1 \) first.)

Solution:
- Only accept an intersection if it occurs within the current cell.
- Cache intersections (e.g., by hashing them with the object identifier) so the intersection calculations do not need to be repeated.
Bounding-Volume Hierarchies

Bounding-Volume (BV) Hierarchies:
- Organize bounding volumes as a multi-way tree.
- Each ray starts at the root of the tree and traverses down through the hierarchy.
- For each enclosing shape that hit, recursively search its children.
- Visit children in increasing order of distance along the ray. Once an intersection is found we can prune later nodes.

Figure courtesy of University of North Carolina

Bounding-Volume Hierarchies

Bounding ellipsoid hierarchy:
Summary

Summary:
- Ray Tracing: Basic approach, recursive ray tracing.
- Reflection and refraction
- Ray-Object intersections
- Distributed ray tracing
- Accelerating ray tracing