Raycasting

Adam Mally
CIS 277 Spring 2015
University of Pennsylvania
What is raycasting?

- Creating a line that passes through the viewing frustum and travels from the eye to some endpoint on a slice of the frustum (e.g. the far clip plane)
- The line’s endpoint is determined by the pixel on our screen from which we want to raycast
Normalized device coordinates

- Recall that your GL window ranges from -1 to 1 on both the X and Y axes
- We can convert to NDC from any given pixel

\[
\begin{align*}
sx &= (2 \times \frac{px}{scr\_width}) - 1 \\
sy &= 1 - (2 \times \frac{py}{scr\_height}) \\
px \text{ and } py \text{ are the given pixel's x and y coordinates}
\end{align*}
\]
Review: Camera axes

\[ F = \text{normalize}(\text{ref} - \text{eye}) \]

\[ R = \text{normalize}(F \times \text{world\_up}) \]

\[ U = \text{normalize}(R \times F) \]
Screen point to world point

\[ p = \text{ref} + \text{sx} \times H + \text{sy} \times V \]

- \( \text{sx}, \text{sy} \) are in NDC
- \( \text{len} = |\text{ref} - \text{eye}| \)
- \( V = U \times \text{len} \times \tan(\alpha) \)
- \( H = R \times \text{len} \times \text{aspect} \times \tan(\alpha) \)
- \( \alpha = \text{FOVY}/2 \)
Getting a ray from the world point

- ray_origin = eye
- ray_direction = normalize(p - eye)
- Arbitrary point on ray = eye + t*ray_direction
Rays: what are they good for?

- Finding an intersection in world space
- Click on the screen to select an object
- Cast rays from your camera to simulate light bounces for rendering
  - aka raytracing, which is covered in CIS 560
- One of the first well-known usages of raycasting was the game Wolfenstein 3D

Spooky raycast demo!
Ray-polygon intersection

- Most common intersection test is ray-triangle
- We’ll also cover ray-sphere, ray-cylinder, and ray-cube
  - All four are commonly used in basic raytracer testing
Ray-sphere intersection

- Sphere defined as \((x_s - x_c)^2 + (y_s - y_c)^2 + (z_s - z_c)^2 = r_s^2\)
  - Sphere center = \(<x_c y_c z_c>\)
  - All points on the sphere surface = \(<x_s y_s z_s>\)
  - \(r_s\) is the sphere’s radius

- Ray defined as: \(R_0 + t * R_d\)
  - \(R_0 = <x_0 y_0 z_0>\)
  - \(R_d = <x_d y_d z_d>\)
  - \(t\) is a parameterization of \(R_d\) (i.e. a float)
Ray-sphere intersection

- Substitute $<x_s y_s z_s>$ for the ray equation:
  \[
  (x_\theta + t*x_d - x_c)^2 + (y_\theta + t*y_d - y_c)^2 + (z_\theta + t*z_d - z_c)^2 = r_s^2
  \]

- Can also be written as:
  - $A t^2 + B t + C = 0$
    - $A = x_d^2 + y_d^2 + z_d^2$
    - $B = 2(x_d(x_\theta - x_c) + y_d(y_\theta - y_c) + z_d(z_\theta - z_c))$
    - $C = (x_\theta - x_c)^2 + (y_\theta - y_c)^2 + (z_\theta - z_c)^2 - r_s^2$

- Note that we now have a quadratic equation
  - We can solve for $t$ using the quadratic formula!
Ray-sphere intersection

- \( t_0, t_1 = (-B \pm \sqrt{B^2 - 4AC})/(2A) \)
  - \( t_0 \) is for the - case and \( t_1 \) is for the + case
- Remember: if the discriminant is negative, then there is no real root and therefore no intersection
  - Discriminant = \( B^2 - 4AC \)
- If \( t_0 \) is positive, then we’re done. If not, then compute \( t_1 \).
Ray-sphere intersection

- Once we have $t$, we can plug it into our ray equation to find the closest point of intersection on our sphere.
  - If all we care about is whether or not we hit the sphere, we can just check:
    - $\text{near\_clip} < t < \text{far\_clip}$
- $P = R_0 + t \times R_d$
Converting from local to world space

- It is important to remember that the t value we have found is in object space (i.e. only valid for an untransformed sphere).
- To find a t value in world space, we can use our object-space t to find the point of intersection Po on our unit sphere.
- We can then transform Po by our sphere's transformation matrix to find the point of intersection in world space Pw.
- We can use Pw to compute our world-space t value as follows:
  - \( \text{world}_t = \text{length}(Pw - \text{camera.\text{eye}}) \)
  - Remember that camera.\text{eye} is the origin of your untransformed ray in world space.
Ray-cube intersection

- Begin by storing $t_{\text{near}} = -\infty$ and $t_{\text{far}} = \infty$
- For each pair of planes associated with the X, Y, and Z axes (the example uses the X “slab”):
  - If $x_d$ is 0, then the ray is parallel to the X slab, so
    - If $x_0 < x_{\text{min}}$ or $x_0 > x_{\text{max}}$ then we miss
    - $t_0 = (x_{\text{min}} - x_0)/x_d$
    - $t_1 = (x_{\text{max}} - x_0)/x_d$
    - If $t_0 > t_1$ then swap them
    - If $t_0 > t_{\text{near}}$ then $t_{\text{near}} = t_0$
    - If $t_1 < t_{\text{far}}$ then $t_{\text{far}} = t_1$
    - Repeat for Y and Z
  - If $t_{\text{near}} > t_{\text{far}}$ then we miss the box

We can’t see the z_slab because it’s coming directly out of the screen
Ray-cylinder intersection (Barrel)

- Point P on infinite cylinder if:
  - \( ((P - C_S) \times C)^2 = r_c^2 C^2 \)
- Insert the ray equation \( R_0 + t \times R_d \)
  - \( ((R_0 - C_S) \times C + t \times (R_d \times C))^2 = r_c^2 C^2 \)
- Convert to semi-quadratic form:
  - \( (|J|^2 t^2) + (2 \times \text{dot}(H, J) \times t) + (|H|^2 - K) = 0 \)
    - \( H = (R_0 - C_S) \times C \)
    - \( J = R_d \times C \)
    - \( K = r_c^2 |C|^2 \)
Ray-cylinder intersection (Barrel)

- Re-formulate to fully quadratic form:
  - \( At^2 + Bt + C = 0 \)
    - \( A = |J|^2 \)
    - \( B = 2 \times \text{dot}(H, J) \)
    - \( C = |H|^2 - K \)
- \( t_0, t_1 = (-B \pm \sqrt{B^2 - 4AC})/(2A) \)
- Remember: if the discriminant is negative, then there is no real root and therefore no intersection
  - Discriminant = \( B^2 - 4AC \)
Ray-cylinder intersection (Barrel)

- Now that we have a $t$ value in object space, we can compute the intersection with the cylinder in object space.
- Assuming a test against a cylinder aligned with the Y axis, if the Y value of the local intersection is less than $C_E.y$ and greater than $C_S.y$, then the ray intersects the barrel of the cylinder.
- HOWEVER, we still need to test the cylinder’s end caps.
Ray-cylinder intersection (Endcaps)

- We can use ray-plane intersection (detailed in subsequent slides) to find the points of intersection with the cylinder’s endcap planes.
- If either of the points found by this test are within a distance of $r$ from the corresponding endcap’s center, then the ray intersects the endcap at that point with the $t$ value found with the ray-plane equation.
  - $r$ is the cylinder’s radius.
Ray-plane intersection

- Plane defined as: \( \text{dot}(N, (P - S)) = 0 \)
  - \( N \) is the plane’s normal
  - \( S \) is some point on the plane
  - \( P \) is the point of intersection
- Ray defined as: \( R_0 + t * R_d \)
- Substitute \( P \) for ray:
  - \( \text{dot}(N, (R_0 + t * R_d - S)) = 0 \)
- Solve for \( t \):
  - \( t = \frac{\text{dot}(N, (S - R_0))}{\text{dot}(N, R_d)} \)
Point-in-triangle

- Use **barycentric coordinates** to test if $P$ is within the bounds of a triangle
- $S = \text{area}(P_1, P_2, P_3)$
- $S_1 = \frac{\text{area}(P, P_2, P_3)}{S}$
- $S_2 = \frac{\text{area}(P, P_3, P_1)}{S}$
- $S_3 = \frac{\text{area}(P, P_1, P_2)}{S}$
- Therefore, $P = S_1P_1 + S_2P_2 + S_3P_3$
- So, $P$ is within the triangle if:
  - $0 \leq S_1 \leq 1$
  - $0 \leq S_2 \leq 1$
  - $0 \leq S_3 \leq 1$
  - $S_1 + S_2 + S_3 = 1$