Overview

Ray Intersection
- Triangle
- Sphere
- Cylinder
- Cone

Textures
- Checker Board
- Special

Shadow

Animation
Ray Intersection – Triangle

Plane:

\[ N \cdot x = Dist \]

Triangle:

• Algebraic
• Geometric

For the normal on triangle at intersection: no requirement for pointing inside or outside
Ray Intersection – *Triangle*

**Plane:**

\[ \mathbf{N} \cdot \mathbf{x} = \text{Dist} \]

**Triangle:**

- Algebraic
- Geometric

For the normal on triangle at intersection:
no requirement for pointing inside or outside

mesh.json
Ray Intersection – *Triangle*

**Geometric**

\[
N = \text{normalize}( (T_2 - T_1) \times (T_3 - T_1) )
\]

P...

\[
\alpha = \frac{\text{Area}(T_1 T_2 P)}{\text{Area}(T_1 T_2 T_3)}
\]

\[
\beta = \frac{\text{Area}(T_1 PT_3)}{\text{Area}(T_1 T_2 T_3)}
\]

\[
\text{Area}(T_1 T_2 T_3) = \frac{1}{2} \left\| (T_2 - T_1) \times (T_3 - T_1) \right\|
\]

\[
= \frac{1}{2} < (T_2 - T_1) \times (T_3 - T_1) , N >
\]

\[
\text{Area}(T_1 T_2 P) = \frac{1}{2} < (T_2 - T_1) \times (P - T_1) , N >
\]

\[
\text{Area}(T_1 PT_3) = \frac{1}{2} < (P - T_1) \times (T_3 - T_1) , N >
\]
Ray Intersection – *Sphere*

Look out reflecting/refracting ray!

\[ P = P_0 + tV \]

\[ t_1 = t_{ca} - t_{hc} \quad \checkmark \]

or

\[ t_2 = t_{ca} + t_{hc} \quad \times \]
Ray Intersection – *Sphere*

If we always choose the “nearest” one:

\[ P = P_0 + tV \]

\[ t_1 = t_{ca} - t_{hc} \] ✓

or

\[ t_2 = t_{ca} + t_{hc} \] ✗

When a reflective/refractive ray bounds at an intersection, \( t_1 = 0 \)
Ray Intersection – *Sphere*

Check the “nearest valid(positive)” intersection

\[ t_1 = t_{ca} - t_{hc} \]
\[ t_2 = t_{ca} + t_{hc} \]

If \((t_1 > 0)\) return \(t_1\);
elseif \((t_2 > 0)\) return \(t_2\);
return \(\text{INFINITY}\);
Ray Intersection – *Sphere*

Check the “nearest valid(positive)” intersection

\[ t_1 = t_{ca} - t_{hc} \]
\[ t_2 = t_{ca} + t_{hc} \]

If \((t_1 > 0)\) return \(t_1\);
elseif \((t_2 > 0)\) return \(t_2\);
return \(\text{INFINITY}\);
Ray Intersection – Cylinder

1. Intersect with open cylinder & Check if the intersection is between the planes
2. Intersect with two caps
3. Out of all intersections, choose the one with minimal dist
Ray Intersection – *Infinite Cylinder*

Infinite cylinder along $y$ of radius $r$ axis has equation $x^2 + z^2 - r^2 = 0$.

The equation for a more general cylinder of radius $r$ oriented along a line $p_a + v_a t$:

$$(q - p_a - (v_a, q - p_a)v_a)^2 - r^2 = 0$$

where $q = (x, y, z)$ is a point on the cylinder.

1999, Denis Zorin
Ray Intersection – *Infinite Cylinder*

Infinite cylinder along $y$ of radius $r$ axis has equation

$$x^2 + z^2 - r^2 = 0.$$

The equation for a more general cylinder of radius $r$ oriented along a line $p_a + v_a t$:

$$(q - p_a - (v_a, q - p_a)v_a)^2 - r^2 = 0$$

where $q = (x,y,z)$ is a point on the cylinder.

1999, Denis Zorin
Ray Intersection – *Infinite Cylinder*

To find intersection points with a ray $p + vt$, substitute $q = p + vt$ and solve:

$$(p - p_a + vt - (v_a, p - p_a + vt)v_a)^2 - r^2 = 0$$

reduces to

$$At^2 + Bt + C = 0$$

with

$$A = (v - (v, v_a)v_a)^2$$

$$B = 2(v - (v, v_a)v_a, \Delta p - (\Delta p, v_a)v_a)$$

$$C = (\Delta p - (\Delta p, v_a)v_a)^2 - r^2$$

where $\Delta p = p - p_a$
Ray Intersection – Cylinder

POV ray like cylinder with caps: cap centers at \( p_1 \) and \( p_2 \), radius \( r \).

Infinite cylinder equation: \( p_a = p_1, v_a = (p_2 - p_1)/|p_2 - p_1| \)

The finite cylinder (without caps) is described by equations:

\[
(q - p_a - (v_a, q - p_a)v_a)^2 - r^2 = 0 \text{ and } (v_a, q - p_1) > 0 \text{ and } (v_a, q - p_2) < 0
\]

The equations for caps are:

\[(v_a, q - p_1) = 0, (q - p_1)^2 < r^2 \quad \text{bottom cap} \]
\[(v_a, q - p_2) = 0, (q - p_2)^2 < r^2 \quad \text{top cap} \]
Ray Intersection – *Cone*

Similar to cylinder:
1. Intersect with open cone & Check if the intersection is between the planes
2. Intersect with the cap
3. Out of all intersections, choose the one with minimal dist
Ray Intersection – *Infinite Cone*

Infinite cone along $y$ with apex half-angle $\alpha$ has equation

$$x^2 + z^2 - y^2 = 0.$$ 

The equation for a more general cone oriented along a line $p_a + v_a t$, with apex at $p_a$:

$$\cos^2 \alpha (q - p_a - (v_a, q - p_a)v_a)^2 - \sin^2 \alpha (v_a, q - p_a)^2 = 0$$

where $q = (x, y, z)$ is a point on the cone, and $v_a$ is assumed to be of unit length.
Ray Intersection – *Infinite Cone*

Similar to the case of the cylinder: substitute \( q = p + vt \) into the equation, find the coefficients \( A, B, C \) of the quadratic equation, solve for \( t \). Denote \( \Delta p = p - p_a \).

\[
\cos^2 \alpha (vt + \Delta p - (v_a, vt + \Delta p)v_a)^2 - \sin^2 \alpha (v_a, vt + \Delta p)^2 = 0
\]

\[
A = \cos^2 \alpha \left( v - \left( v, v_a \right)v_a \right)^2 - \sin^2 \alpha \left( v, v_a \right)^2
\]

\[
B = 2 \cos^2 \alpha \left( v - \left( v, v_a \right)v_a, \Delta p - (\Delta p, v_a)v_a \right) - 2 \sin^2 \alpha (v, v_a)(\Delta p, v_a)
\]

\[
C = \cos^2 \alpha (\Delta p - (\Delta p, v_a)v_a)^2 - \sin^2 \alpha (\Delta p, v_a)^2
\]
Ray Intersection – *Infinite Cone*

In the assignment,

\[ N_{\text{axis}} = \text{normalize}(\text{axis}); \]

To get the normal in the infinite cone:

\[ E = P - P_{\text{apex}} \]
\[ \text{Normal} = \text{normalize}(E - ||E||/\text{COS } \alpha \ast N_{\text{axis}}); \]
Refraction

From a medium with a higher refractive index to a lower one
Refraction

When $> \text{critical angle}$, you could let refraction return black; Or you could return internal reflection;

In this scene, the "refraction ratio" of the sphere = 1.1
Texture

• Checkerboard
  – 2D: floor(x)+floor(y) is odd/even
  – 3D: view normal as the z-axis for the new coordinate, then find x, y
Texture

• Special: Perlin noise

Using noise as an offset to create handwritten lines.

By applying a simple gradient, a procedural fire texture can be created.

Perhaps the quintessential use of Perlin noise today, terrain can be created with caves and caverns using a modified Perlin Noise implementation.
Perlin Noise

Random

Perlin
Idea

1. Generate random values at grid points.

2. Interpolate smoothly between these values.
Perlin Noise

Step 1 Cut to grids

Step 2 Pseudorandom gradient vector.

[NEW] Randomly pick from (1,1,0),(-1,1,0),(1,-1,0),(-1,-1,0), (1,0,1),(-1,0,1),(1,0,-1),(-1,0,-1), (0,1,1),(0,-1,1),(0,1,-1),(0,-1,-1)
Perlin Noise

Step 3 Distance vectors

Step 4 Dot product on each grid point \(<V_{\text{grad}}, V_{\text{dist}}>\)

Step 5 Interpolation

fade function: \(6t^5 - 15t^4 + 10t^3\)
Hard Shadow

lightVec is the vector from that position to that light

Step 1 generate a ray from position to light

Step 2 find intersection length(distance)

Step 3 if distance is positive and before hitting the light (smaller than the length of lightVec) return true;
Hard Shadow
Soft Shadow
Soft Shadow

Shoot rays around the point light - grid

- Uniformly sample
- Randomly sample
Soft Shadow

Shoot rays around the point light - sphere

- Sample by $\theta [0, 2\pi)$ and $\phi [0, \pi)$
- Sample by solid angle
float pointShadowRatio ( vec3 pos, vec3 lightVec ) {
    float count = 0.0;
    for i = 1...k
        for j = 1...k
            Randomly Sample a new light ray around the original light
            if not pointInShadow(pos, newLightVec)
                count += 1.0;
        return count/float(k*k);
}

In function getLightContribution
    Comment if pointInshadow return black;
    Modify return to contribution * pointShadowRatio (or diffuseColor * pointShadowRatio)
Soft Shadow

Randomly Sample:

\[(X, y, z) = (\frac{2x_1}{\sqrt{1-x_1^2-x_2^2}}, \frac{2x_2}{\sqrt{1-x_1^2-x_2^2}}, \frac{1-2(x_1^2+x_2^2)}{2})\]

) random \(x_1, x_2\) in \([-1, 1]\)

or (\(\begin{align*}
x &= \sqrt{1-u^2} \cos \theta \\
y &= \sqrt{1-u^2} \sin \theta \\
z &= u,
\end{align*}\)

) random \(u\) in \([-1, 1]\), \(\theta\) in \([0, 2\pi]\)
Animation

In intersection with sphere,
Center.Y += K1*abs(round(frame/K2) – frame/K2))