CS 655 – Advanced Computer Graphics

Ray-Object Intersections

Ray-Object Intersections

- Many objects can be represented as implicit surfaces:
  - Sphere (with center at $\boldsymbol{c}$ and radius R): $f_{sphere}(\boldsymbol{p}) = \|\boldsymbol{p} - \boldsymbol{c}\|^2 - R^2 = 0$
  - Plane (with normal $\boldsymbol{n}$ and distance to origin d): $f_{plane}(\boldsymbol{p}) = \boldsymbol{p} \cdot \boldsymbol{n} + D = 0$
- To determine where a ray intersects an object:
  - Need to find the intersection point $\boldsymbol{p}$ of the ray and the object
  - The ray is represented explicitly in parametric form:

    $$\boldsymbol{R}(t) = \boldsymbol{R}_o + \boldsymbol{R}_d t$$
  - Plug the ray equation into the surface equation and solve for t:

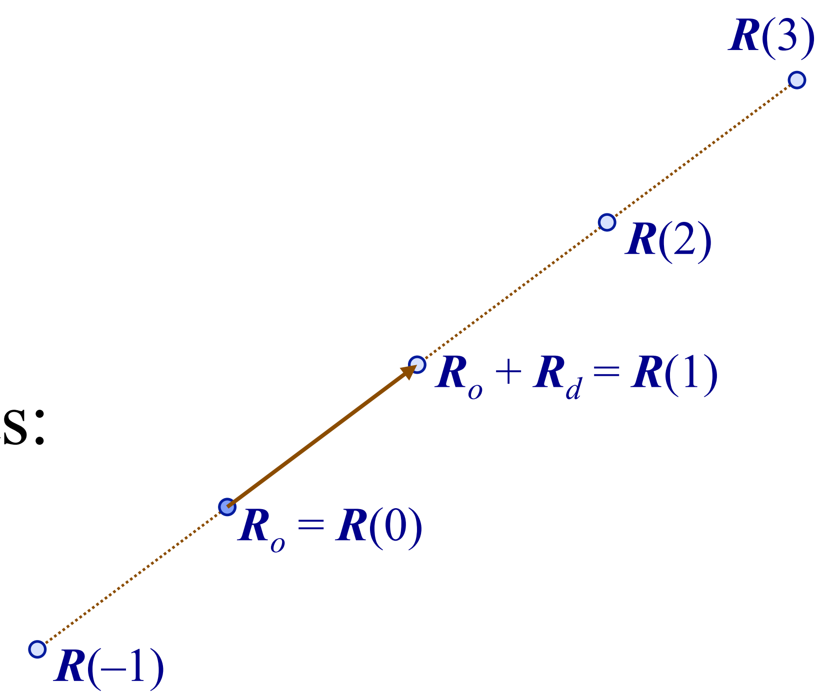
    $$f(\boldsymbol{R}(t)) = 0$$
  - Substitute t back into ray equation to find intersection point $\boldsymbol{p}$:

    $$\boldsymbol{p} = \boldsymbol{R}(t) = \boldsymbol{R}_o + \boldsymbol{R}_d t$$

Recall: Ray Representation

- A ray can be represented explicitly (in parametric form) as an origin (point) and a direction (vector):
  - Origin: $R_o = \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix}$
  - Direction: $R_d = \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix}$
- The ray consists of all points:

  $\boldsymbol{R}(t) = \boldsymbol{R}_o + \boldsymbol{R}_d t$



Ray-Sphere Intersections

To find the intersection points of a ray with a sphere:

- Sphere is represented as:
  - center (point): $S_c = (x_c, y_c, z_c)$
  - and a radius (float): r
- The surface of the sphere is the set of all points (x, y, z) such that:
  - $(x-x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = r^2$
- In this form (implicit form), it is difficult to directly generate surface points
- However, given a point, it is easy to see if the point lies on the surface
- To solve the ray-surface intersection, substitute the ray equation into the sphere equation.

Ray-Sphere Intersections

- First, split the ray into its component equations:

$$x = x_o + x_d t$$

$$y = y_o + y_d t$$

$$z = z_o + z_d t$$

- Next substitute the ray equation into the sphere equation:

$$(x-x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = r^2$$

- Giving:

$$(x_o + x_d t - x_c)^2 + (y_o + y_d t - y_c)^2 + (z_o + z_d t - z_c)^2 = r^2$$

Ray-Sphere Intersections

- Next, simplify the equation:

$$(x_0 + x_d t - x_c)^2 + (y_0 + y_d t - y_c)^2 + (z_0 + z_d t - z_c)^2 = r^2$$

$$\Rightarrow (x_d^2 + y_d^2 + z_d^2)t^2 +$$

$$2(x_d x_o - x_d x_c + y_d y_o - y_d y_c + z_d z_o - z_d z_c)t +$$

$$(x_o^2 - 2x_o x_c + x_c^2 + y_o^2 - 2y_o y_c + y_c^2 + z_o^2 - 2z_o z_c + z_c^2)$$

$$= r^2$$

Ray-Sphere Intersections

- Let

$A = x_d^2 + y_d^2 + z_d^2 = 1$

$B = 2(x_d x_o - x_d x_c + y_d y_o - y_d y_c + z_d z_o - z_d z_c)$

$C = x_o^2 - 2x_o x_c + x_c^2 + y_o^2 - 2y_o y_c + y_c^2 + z_o^2 - 2z_o z_c + z_c^2 - r^2$

- Then solve using the quadratic equation:

$$t = \frac{-B \pm \sqrt{B^2 - 4C}}{2}$$

Ray-Sphere Intersections

- If the discriminant is negative, the ray misses the sphere
- The smaller positive root (if one exists) is the closer intersection point
- We can save some computation time by computing the smaller root first, then only computing the second root if necessary

$$t_0 = \frac{-B - \sqrt{B^2 - 4C}}{2} \qquad t_1 = \frac{-B + \sqrt{B^2 - 4C}}{2}$$

Ray-Sphere Intersections: Algorithm

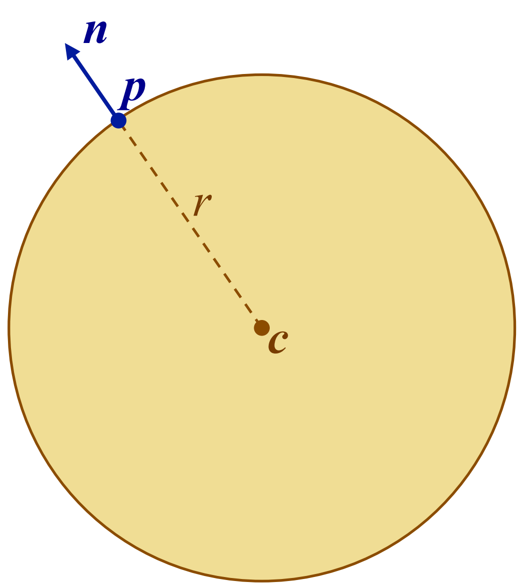
Algorithm for ray-sphere intersection:

1. Calculate B and C of the quadratic
2. Calculate the discriminant: $D = B^2 - 4C$
3. If $D \le 0$ return false (no intersection point)
4. Calculate smaller intersection parameter t_0: $t_0 = \dfrac{-B - \sqrt{D}}{2}$
5. If $t_0 \le 0$ then calculate larger t-value t_1: $t_1 = \dfrac{-B + \sqrt{D}}{2}$
6. If $t_1 \le 0$ return false (intersection point behind ray)
7. else set $t = t_1$
8. else set $t = t_0$
9. Return intersection point: $\boldsymbol{p} = \boldsymbol{r}(t) = \boldsymbol{r}_o + \boldsymbol{r}_d t$

Ray-Sphere Intersections: Normal

The normal ***n*** at an intersection point ***p*** on a sphere is:

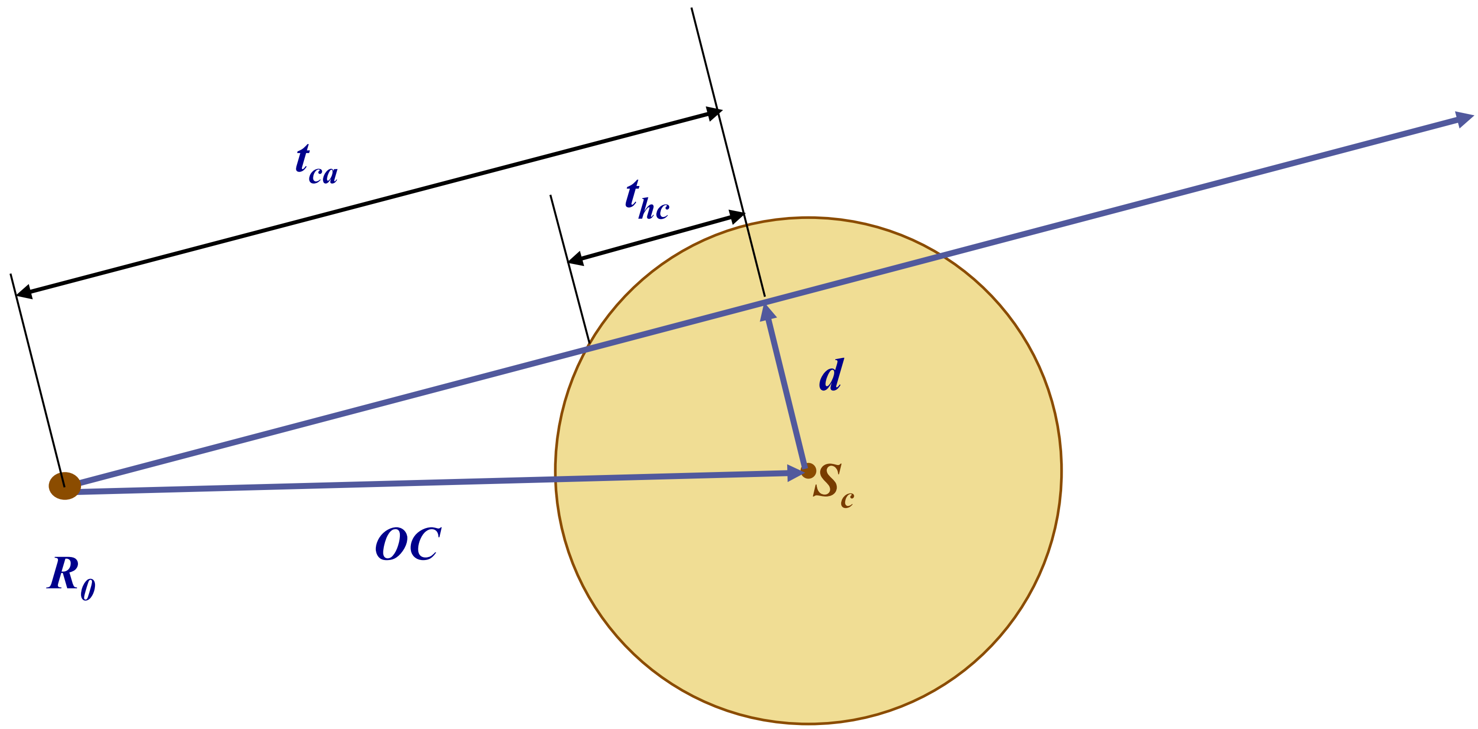
$$\mathrm{n} = \frac{\mathrm{p} - S_{\mathrm{c}}}{r} = \left(\frac{x_i - x_c}{r} \quad \frac{y_i - y_c}{r} \quad \frac{z_i - z_c}{r} \right)$$



Ray-Sphere Intersections

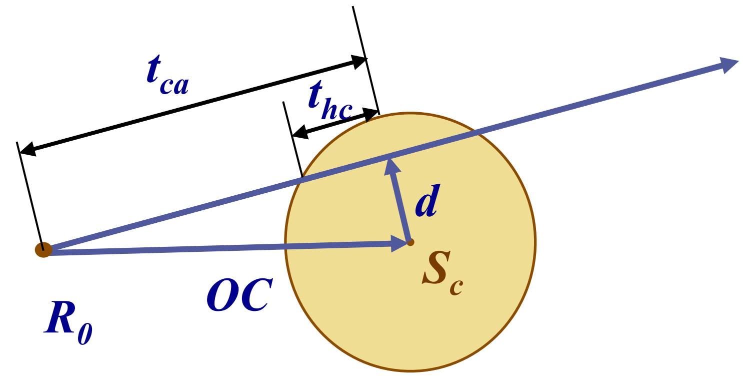
- Computation time:
  - 17 adds/subtracts
  - 17 multiplies
  - 1 square root
- for each ray/sphere test
- Can we reduce the number of intersection calculations?
  - use a geometric approach

Ray-Sphere Intersections - Geometric



Ray-Sphere Intersections - Geometric

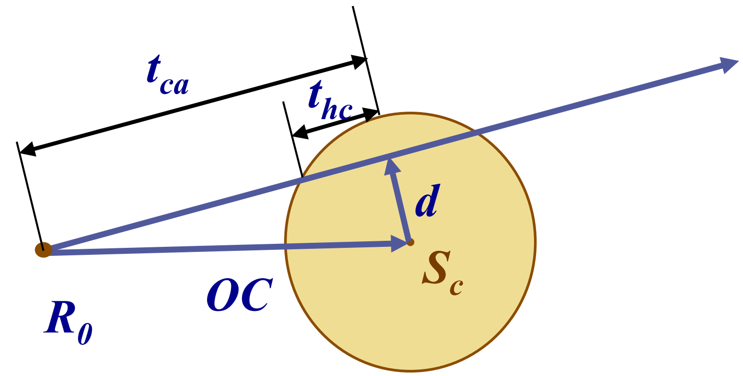
1. Determine whether the ray's origin is outside the sphere
   - if R_0 - S_c < r, then the point is inside the sphere
   - Call the vector between R_0 and S_c OC



2. Find the closest approach of the ray to the sphere's center.
   - let d = distance from S_c to the ray
   - let t_{ca} = distance from R_0 to closest approach of ray to S_c
   - $t_{ca} = R_d \bullet OC$

Ray-Sphere Intersections - Geometric

3. If $t_{ca} < 0$ and R_0 lies outside the sphere, the ray does not intersect the sphere

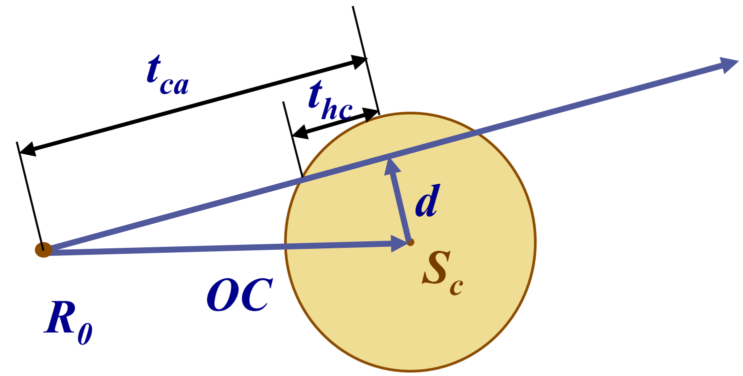


4. Otherwise, compute t_{hc}, the distance from the closest approach to the sphere's surface
   - $t_{hc}^2 = r^2 - d^2$
   - $d^2 = |OC|^2 - t_{ca}^2$, so
   - $t_{hc}^2 = r^2 - |OC|^2 + t_{ca}^2$

Ray-Sphere Intersections - Geometric

5. If $t_{hc}^2 < 0$ the ray does not intersect the sphere

6. Otherwise, calculate the intersection distance



- If R_0 is outside the sphere:

  $t = t_{ca} - t_{hc}$

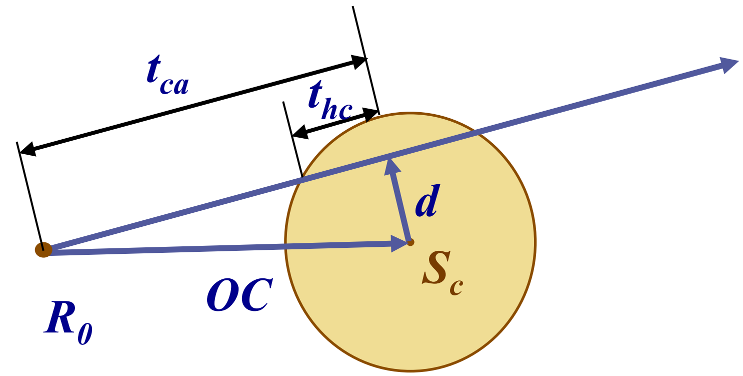
- If R_0 is inside the sphere:

  $t = t_{ca} + t_{hc}$

Ray-Sphere Intersections - Geometric

7. Calculate the intersection point:

$$(x_i, y_i, z_i) = (x_o + x_d t, y_o + y_d t, z_o + z_d t)$$



8. Calculate the normal at the intersection point:

$$n = \left(\frac{x_i - x_c}{r} \quad \frac{y_i - y_c}{r} \quad \frac{z_i - z_c}{r} \right)$$

- This reduces the computation (worst case) by 4 multiplies and 1 add
- Even fewer computations if the ray misses the sphere

Example

- Find the intersection point of the ray

$$R_o = (1,-2,-1), \qquad R_d = (1,2,4)$$

- and the sphere

$$S_c = (3,0,5), \qquad r = 3$$

Example

$$R_o = (1,-2,-1), \qquad R_d = (1,2,4)$$
$$S_c = (3,0,5), \qquad r = 3$$

First: Normalize ray

$$R_d = \left(\frac{1}{\sqrt{21}}, \frac{2}{\sqrt{21}}, \frac{4}{\sqrt{21}}\right)$$

1. Compute OC

$$OC = (3,0,5) - (1,-2,-1) = (2,2,6)$$
$$|OC| = (2,2,6) \bullet (2,2,6) = \sqrt{44}$$

Since $|OC| > r$, R_o is outside the sphere

1. Check ray origin inside/outside sphere
   Compute OC
2. Find closest approach to sphere's center.
3. Check for non intersection
4. Compute t_{hc}
5. Check $t_{hc}^2 < 0$
6. Calculate the intersection distance
7. Calculate intersection point
8. Calculate normal

Example

$$R_o = (1,-2,-1), \qquad R_d = (1,2,4)$$

$$S_c = (3,0,5), \qquad r = 3$$

1. Check ray origin inside/outside sphere
   Compute OC
2. Find closest approach to sphere's center.
3. Check for non intersection
4. Compute t_{hc}
5. Check $t_{hc}^2 < 0$
6. Calculate the intersection distance
7. Calculate intersection point
8. Calculate normal

2. Find the closest approach of the ray to the sphere's center.

$$t_{ca} = R_d \bullet OC = \left(\frac{1}{\sqrt{21}}, \frac{2}{\sqrt{21}}, \frac{4}{\sqrt{21}}\right) \bullet (2,2,6) = 6.546$$

Example

$$R_o = (1,-2,-1), \qquad R_d = (1,2,4)$$

$$S_c = (3,0,5), \qquad r = 3$$

1. Check ray origin inside/outside sphere
   Compute OC
2. Find closest approach to sphere's center.
3. Check for non intersection
4. Compute t_{hc}
5. Check $t_{hc}^2 < 0$
6. Calculate the intersection distance
7. Calculate intersection point
8. Calculate normal

3. If $t_{ca} < 0$ and R_0 lies outside the sphere, the ray does not intersect the sphere

R_0 does lie outside the sphere,

but $t_{ca} > 0$, so continue

Example

$$R_o = (1,-2,-1), \qquad R_d = (1,2,4)$$

$$S_c = (3,0,5), \qquad r = 3$$

4. Compute t_{hc}, the distance from the closest approach to the sphere's surface

$$t_{hc}^2 = r^2 - |OC|^2 - t_{ca}^2$$
$$= 3^2 - 44 + (6.546)^2$$
$$= 7.850$$

1. Check ray origin inside/outside sphere
   Compute OC
2. Find closest approach to sphere's center.
3. Check for non intersection
4. Compute t_{hc}
5. Check $t_{hc}^2 < 0$
6. Calculate the intersection distance
7. Calculate intersection point
8. Calculate normal

Example

$$R_o = (1,-2,-1), \qquad R_d = (1,2,4)$$

$$S_c = (3,0,5), \qquad r = 3$$

5. See if $t_{hc}^2 < 0$

$t_{hc}^2 > 0$, so the ray must hit the sphere

1. Check ray origin inside/outside sphere
   Compute OC
2. Find closest approach to sphere's center.
3. Check for non intersection
4. Compute t_{hc}
5. Check $t_{hc}^2 < 0$
6. Calculate the intersection distance
7. Calculate intersection point
8. Calculate normal

Example

$$R_o = (1,-2,-1), \qquad R_d = (1,2,4)$$

$$S_c = (3,0,5), \qquad r = 3$$

1. Check ray origin inside/outside sphere
   Compute OC
2. Find closest approach to sphere's center.
3. Check for non intersection
4. Compute t_{hc}
5. Check $t_{hc}^2 < 0$
6. Calculate the intersection distance
7. Calculate intersection point
8. Calculate normal

6. Calculate the intersection distance

- Since R_0 is outside the sphere:

$$t = t_{ca} - t_{hc} = 6.546 - \sqrt{7.850} = 3.744$$

so the ray intersects the sphere at t = 3.744

Example

$$R_o = (1,-2,-1), \qquad R_d = (1,2,4)$$

$$S_c = (3,0,5), \qquad r = 3$$

1. Check ray origin inside/outside sphere
   Compute OC
2. Find closest approach to sphere's center.
3. Check for non intersection
4. Compute t_{hc}
5. Check $t_{hc}^2 < 0$
6. Calculate the intersection distance
7. Calculate intersection point
8. Calculate normal

7. Calculate the intersection point

$$\begin{aligned}(x_i, y_i, z_i) &= (x_o + x_d t,\ \ y_o + y_d t,\ \ z_o + z_d t) \\ &= \left(1 + \frac{1}{\sqrt{21}} * 3.744,\ \ -2 + \frac{2}{\sqrt{21}} * 3.744,\ \ -1 + \frac{4}{\sqrt{21}} * 3.744\right) \\ &= (1.817,\ \ -.366,\ \ 2.268)\end{aligned}$$

Example

$$R_o = (1,-2,-1), \qquad R_d = (1,2,4)$$

$$S_c = (3,0,5), \qquad r = 3$$

8. Calculate the normal at the intersection point

$$n = \left(\frac{x_i - x_c}{r}, \quad \frac{y_i - y_c}{r}, \quad \frac{z_i - z_c}{r} \right)$$

$$= \left(\frac{1.817 - 3}{3}, \quad \frac{-.366 - 0}{3}, \quad \frac{2.268 - 5}{3} \right)$$

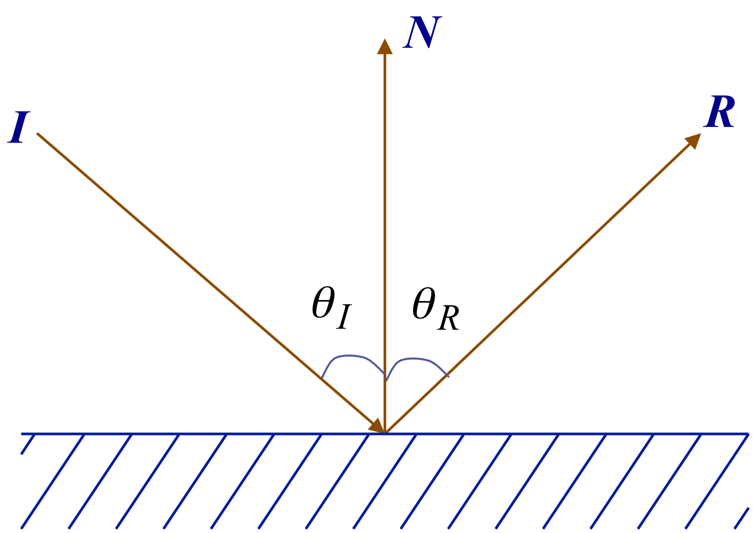
$$= (-0.394, \quad -0.122, \quad -0.911)$$

Done!

1. Check ray origin inside/outside sphere
   Compute OC
2. Find closest approach to sphere's center.
3. Check for non intersection
4. Compute t_{hc}
5. Check $t_{hc}^2 < 0$
6. Calculate the intersection distance
7. Calculate intersection point
8. Calculate normal

Computing the Reflection Ray

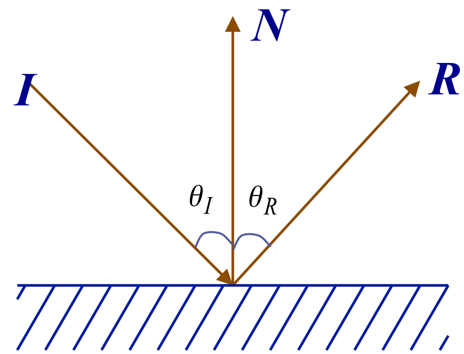
- Angle of incidence = angle of reflection



I = incoming ray
N = Surface normal
R = Reflected ray
(generally want to keep I and N normalized)

Computing the Reflection Ray

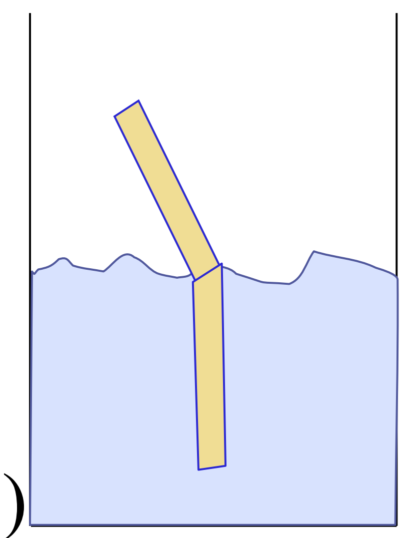
- Knowing I and N, how do we compute R?



1. $\theta_I = \theta_R$
2. I, N, and R are coplanar
3. $R = \alpha I + \beta N$
4. $-I \bullet N = \cos(\theta_I)$
5. $N \bullet R = \cos(\theta_R)$
6. Using 1, 4, and 5, we get:
7. $-I \bullet N = N \bullet R$, so
8. $-I \bullet N = N \bullet (\alpha I + \beta N)$
9. $-I \bullet N = (N \bullet \alpha I) + (N \bullet \beta N)$
10. $-I \bullet N = \alpha(N \bullet I) + \beta (N \bullet N)$
11. $-I \bullet N = \alpha(N \bullet I) + \beta$, since N is normalized
12. Without loss of generality, let $\alpha = 1$, then
13. $\beta = -2 (N \bullet I)$, so
14. $R = I - 2(N \bullet I)N$

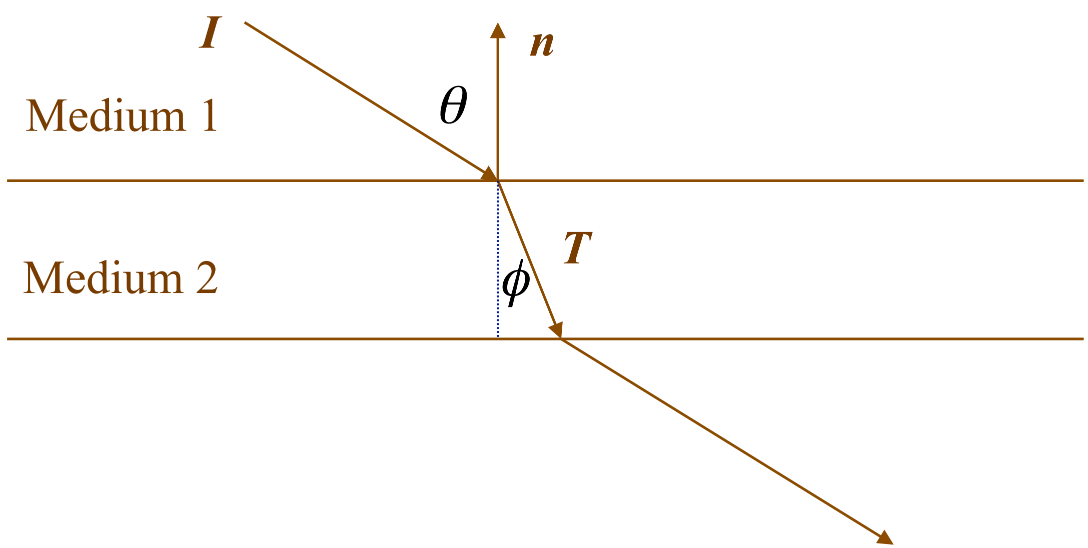
Refraction (transparency)

- When an object is transparent, it transmits light
- Light travels through different materials at different speeds.
- Thus, we get light "bending".
- E.g., pole in water:
- Keep this in mind if you ever get stuck after a plane crash and have to use a bow and arrow to fish for your food.
  (See Hatchet by G. Paulson p. 125)



Refraction (transparency)

- The manner in which light travels through a material is accounted for by the object's "index of refraction"
  - the ratio of the speed of light through the material to the speed of light in a vacuum
- Light bend computed by Snell's law: $$\frac{\sin\theta}{\sin\phi} = \frac{n_2}{n1}$$



Refraction Indices

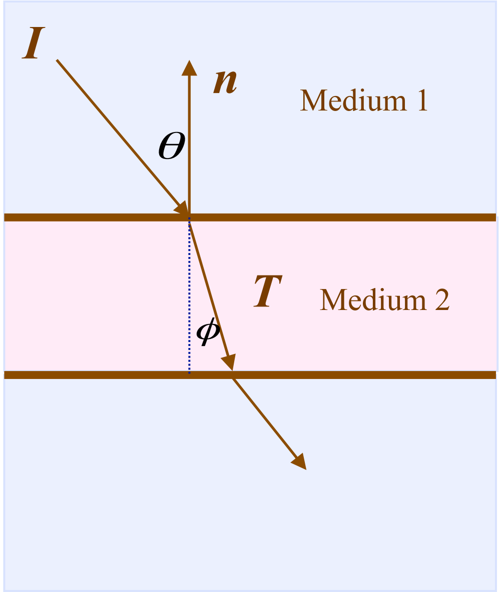
Index of refraction for various materials:

Material	Index
Vacuum	1.0
Air	1.0003
Water	1.33
Alcohol	1.36
Fused quartz	1.46
Crown glass	1.52
Flint glass	1.65
Sapphire	1.77
Heavy flint glass	1.89
Diamond	2.42

Refraction

- How do we compute T?

$$\frac{\sin\theta}{\sin\phi} = \frac{n_2}{n1}$$

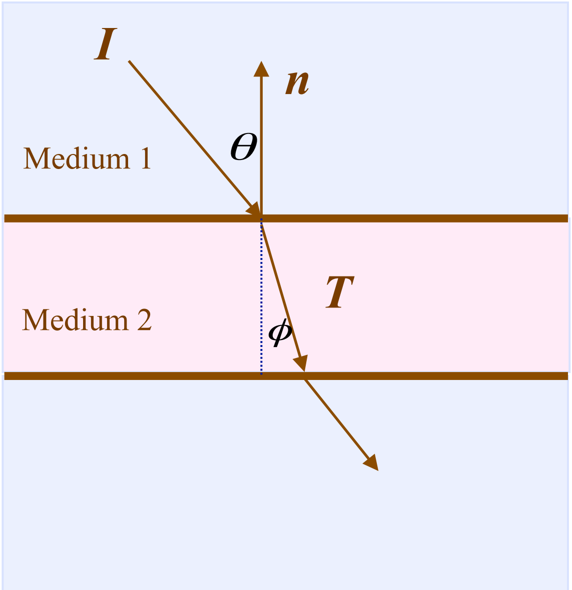


1. Let $n_{IT} = n_1/n_2 = \sin(\phi)/\sin(\theta)$
2. then $n_{IT}^2 = \sin^2(\phi)/\sin^2(\theta)$
3. $\sin^2(\theta)\, n_{IT}^2 = \sin^2(\phi)$
4. since $\sin^2(\theta)\ ^+ \cos^2(\theta) = 1$, we get
5. $(1 - \cos^2(\theta))\, n_{IT}^2 = 1 - \cos^2(\phi)$, so
6. $(1 - \cos^2(\theta))\, n_{IT}^2 - 1 = - \cos^2(\phi)$
7. $(1 - \cos^2(\theta))\, n_{IT}^2 - 1 = - (-N \bullet T)^2$
8. $(1 - \cos^2(\theta))\, n_{IT}^2 - 1 = - (-N \bullet (\alpha I + \beta N))^2$
9. $(1 - \cos^2(\theta))\, n_{IT}^2 - 1 = - (-\alpha(N \bullet I) + \beta(-N \bullet N))^2$
10. $(1 - \cos^2(\theta))\, n_{IT}^2 - 1 = -(\alpha(\cos(\phi)) - \beta)^2$
11. We want T normalized, so $T \bullet T = 1$, so
12. $1 = (\alpha I + \beta N) \bullet (\alpha I + \beta N)$
13. $1 = \alpha^2(I \bullet I) + 2\alpha\beta(I \bullet N) \bullet \beta^2(N \bullet N)$
14. $1 = \alpha^2 + 2\alpha\beta\cos(\theta) \bullet \beta^2$

Refraction

- How do we compute T?

$$\frac{\sin\theta}{\sin\phi} = \frac{n_2}{n1}$$



15. Solve equations 10 and 14 simultaneously:

16. $(1 - \cos^2(\theta))\, n_{IT}^2 - 1 = -(\alpha(\cos(\phi)) - \beta)^2$

$$1 = \alpha^2 + 2\alpha\beta\cos(\theta) \bullet \beta^2$$

17. After solving, we get:

$$\alpha = n_{IT}$$

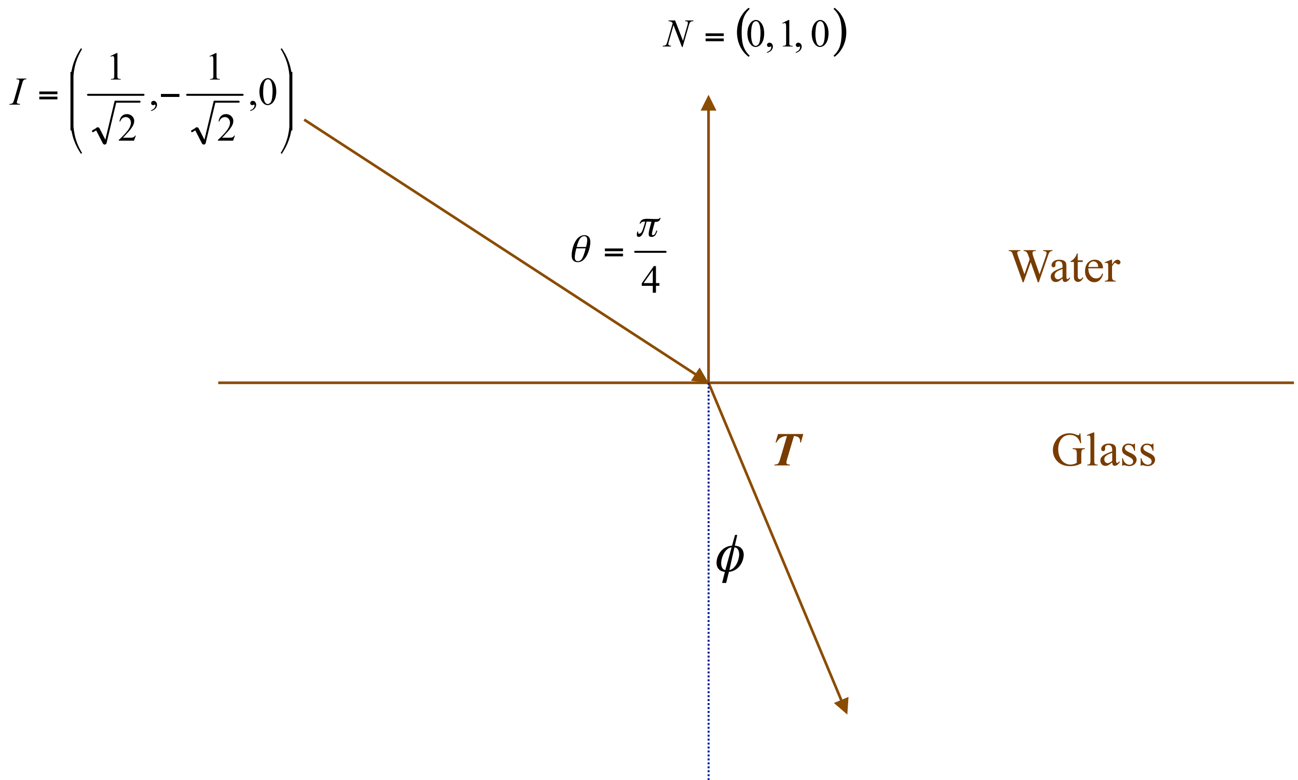
$$\beta = n_{IT}\cos(\theta) - \sqrt{1 + n_{IT}^{\;2}(\cos^2(\theta) - 1)}$$

18. So,

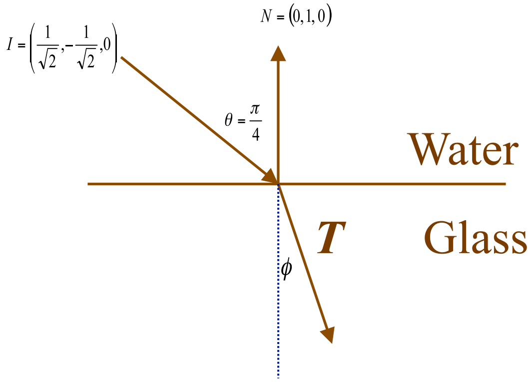
$$T = n_{IT}I + \left(n_{IT}\cos(\theta) - \sqrt{1 + n_{IT}^{\;2}(\cos^2(\theta) - 1)}\right)N$$

Refraction Example

- Compute T for:



Refraction Example



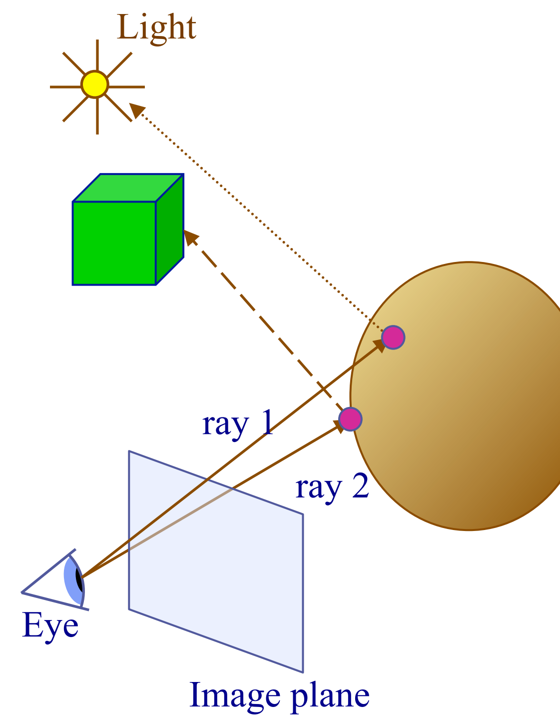
$$T = n_{IT} I + \left(n_{IT} \cos(\theta) - \sqrt{1 + n_{IT}^{2} (\cos^{2}(\theta) - 1} \right) N$$

$$n_{IT} = \frac{n_I}{n_T} = \frac{n_{water}}{n_{glass}} = \frac{1.33}{1.52} = 0.875$$

$$T = 0.875 \left(\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0 \right) + \left(0.875 \cos\left(\frac{\pi}{4}\right) - \sqrt{1 + (0.875)^2 (\cos^2\left(\frac{\pi}{4}\right) - 1} \right) (0, 1, 0)$$

Shadows

- The intensity of what we see at at pixel will depend on whether or not a light actually hits the object
  - The point of intersection between ray 1 and the object is in direct illumination
  - The point of intersection between ray 2 and the object is blocked from the light source (in shadow)



- The final pixel intensity must be computed taking this into account

Shadows

- Approach for computing shadows (hard shadows)
    1. For each ray generated, find the first object intersected
    2. From that intersection point, send a ray directly to each light source
    3. If the ray intersects an object en route to the light source, the object is occluded from that light source

- For shadow rays, we don't care which object the ray intersects, just the fact that it does intersect

Ray-Plane Intersections

To find the intersection points of a ray with an infinite plane:

- Ray equation:
  - Origin: $r_o = (x_o, y_o, z_o)$
  - Direction: $r_d = (x_d, y_d, z_d)$
- Plane equation:

  $ax + by + cz + d = 0$

  with $a^2 + b^2 + c^2 = 1$

  normal vector $p_n = (a, b, c)$

  distance from (0, 0, 0) to plane is d

- Substitute the ray equation into the plane equation:

$$a(x_0 + x_d t) + b(y_0 + y_d t) + c(z_0 + z_d t) + d = 0$$

$$\Rightarrow \quad ax_0 + ax_d t + by_0 + by_d t + cz_0 + cz_d t + d = 0$$

- Solving for t we get:

$$t = -(ax_0 + by_0 + cz_0 + d) / (ax_d + by_d + cz_d)$$

Ray-Plane Intersections

- In vector form we have: $t = \frac{-(p_n \bullet r_o + d)}{p_n \bullet r_d}$

- If $p_n \bullet r_d = 0$ the ray is parallel to the plane and does not intersect

- If $p_n \bullet r_d > 0$ the normal of the plane is pointing away from the ray, and thus the plane is culled

- If $t < 0$ then intersection point is behind the ray, so no real intersection occurs

- Otherwise, compute intersection point: $\boldsymbol{p} = \boldsymbol{r}_o + \boldsymbol{r}_d t$

Ray/Plane Intersection

- Basic Algorithm:

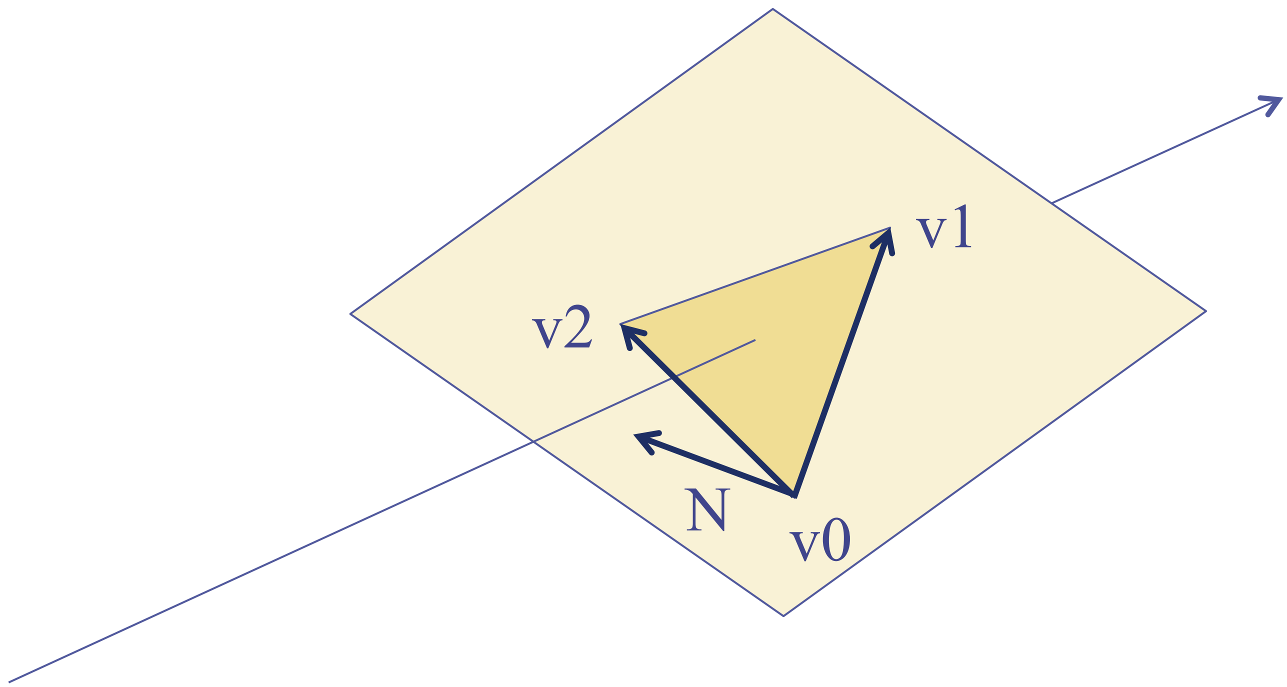
1. Compute $v_d = p_n \cdot r_d$
2. If $v_d > 0$ and assuming 1 sided planes then return
3. If $v_d = 0$ then return (ray parallel to plane)
4. Compute $v_o = -(p_n \cdot r_d + d)$
5. Compute $t = v_o / v_d$
6. If $t < 0$ return
7. If $v_d > 0$ reverse the plane's normal
8. Return $r = (x_o + x_d t,\ y_o + y_d t,\ z_o + z_d t)$

Ray/Triangle Intersection

- Simpler than Ray/Polygon intersection
- Every planar polygon can be reduced to a finite set of triangles.

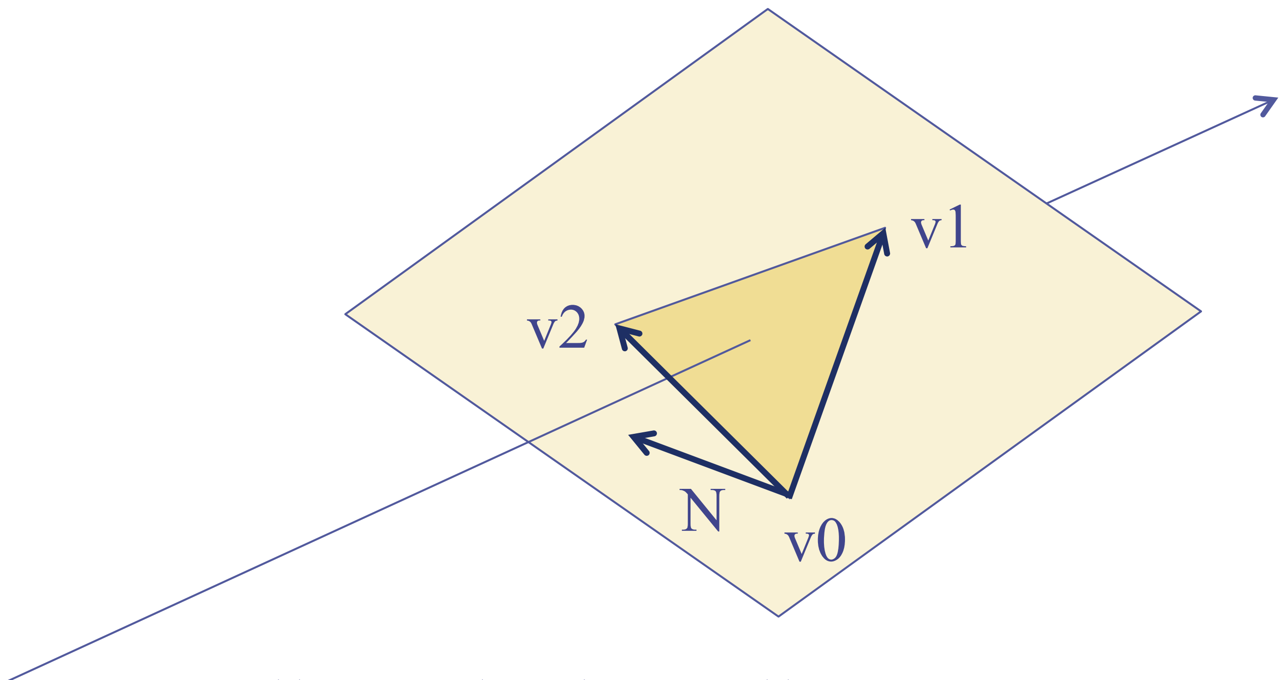
1. Determine if the ray intersects the plan containing the triangle.
2. Project the triangle onto a plane.
3. Find the Barycentric coordinates of the intersection point in 2D on the plane.
4. Reproject back into world coordinates.

Ray/Triangle: The plane



N = norm((v1 - v0) x (v2 - v0))

Ray/Triangle: The plane

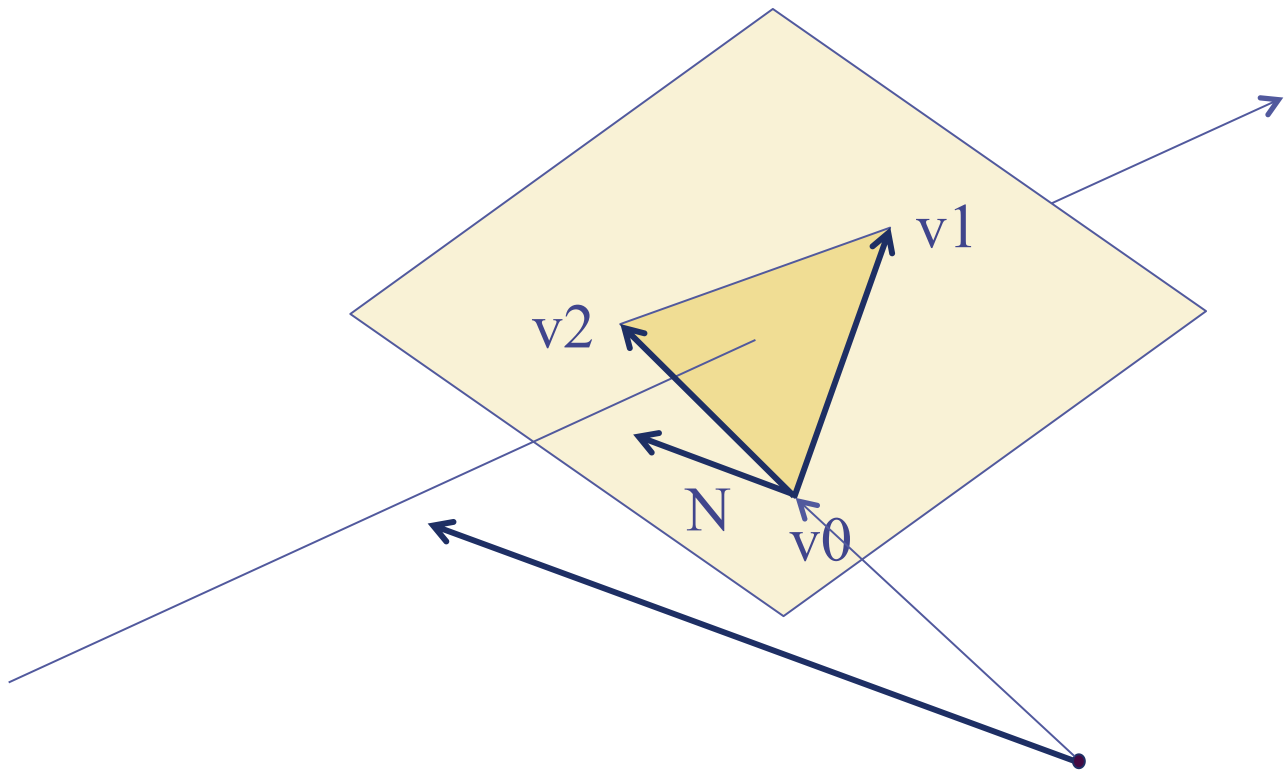


N = norm((v1 - v0) x (v2 - v0))

Plane (x,y,z) = ax + by + cz + d = Nx(x) + Ny(y) + Nz(z) + d = 0

What's d? d = distance from origin to closest point on the plane.

Ray/Triangle: The plane



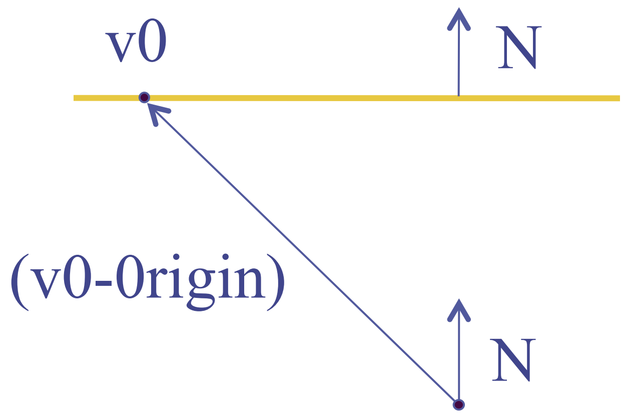
N = norm((v1 - v0) x (v2 - v0))

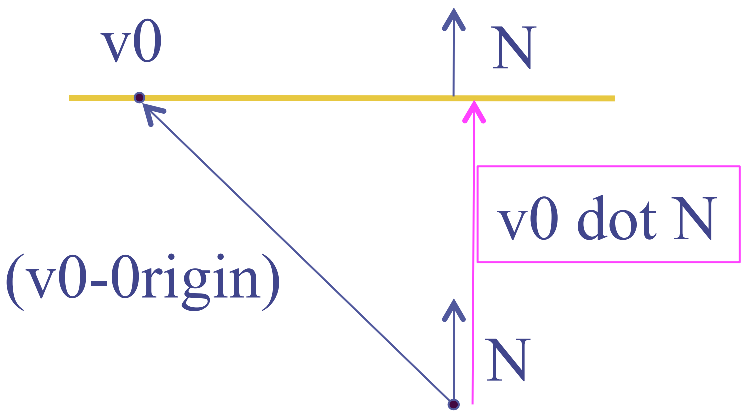
Plane (x,y,z) = ax + by + cz + d = Nx(x) + Ny(y) + Nz(z) + d = 0

What's d? d = distance from origin to closest point on the plane.

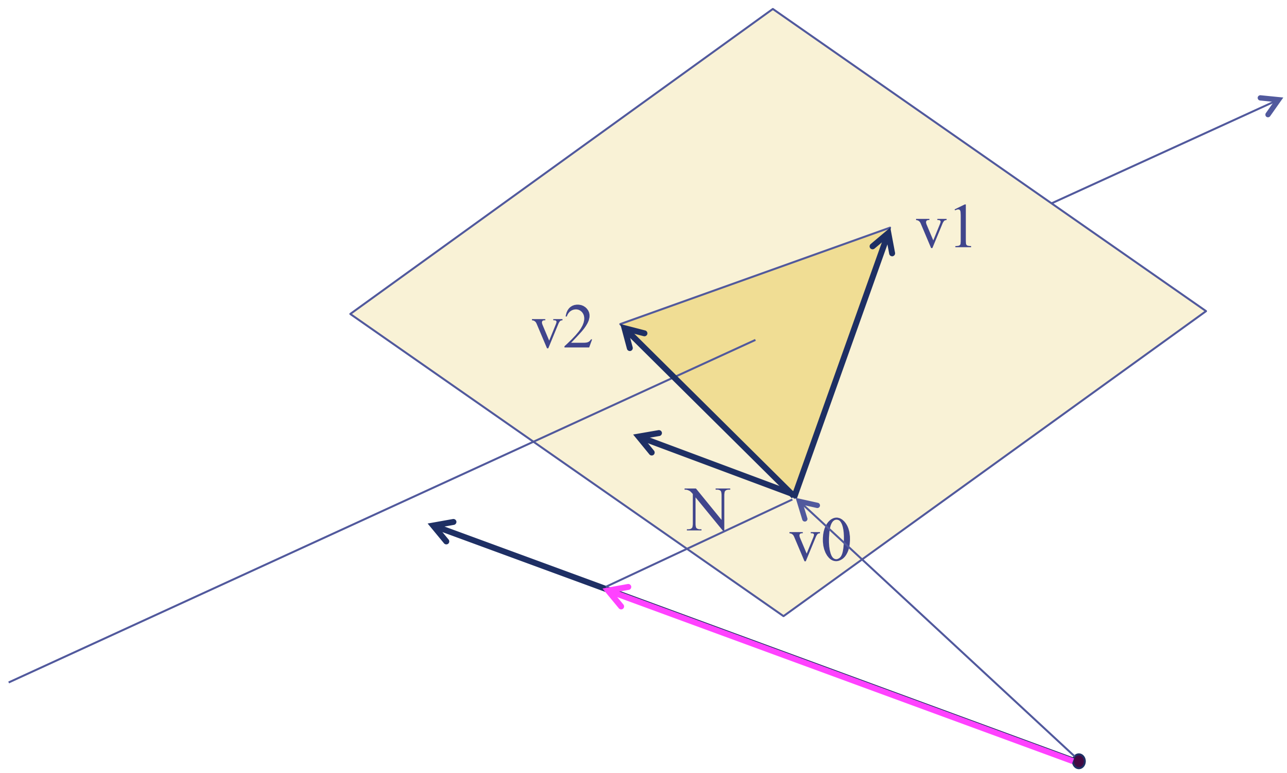
= projection of v0 (as a vector) onto N.

= v0 dot N


v0
N
(v0-0rigin)
N


v0
N
v0 dot N
(v0-0rigin)
N

Ray/Triangle: The plane



N = norm((v1 - v0) x (v2 - v0))

Plane (x,y,z) = ax + by + cz + d = Nx(x) + Ny(y) + Nz(z) + d = 0

What's d? d = distance from origin to closest point on the plane.

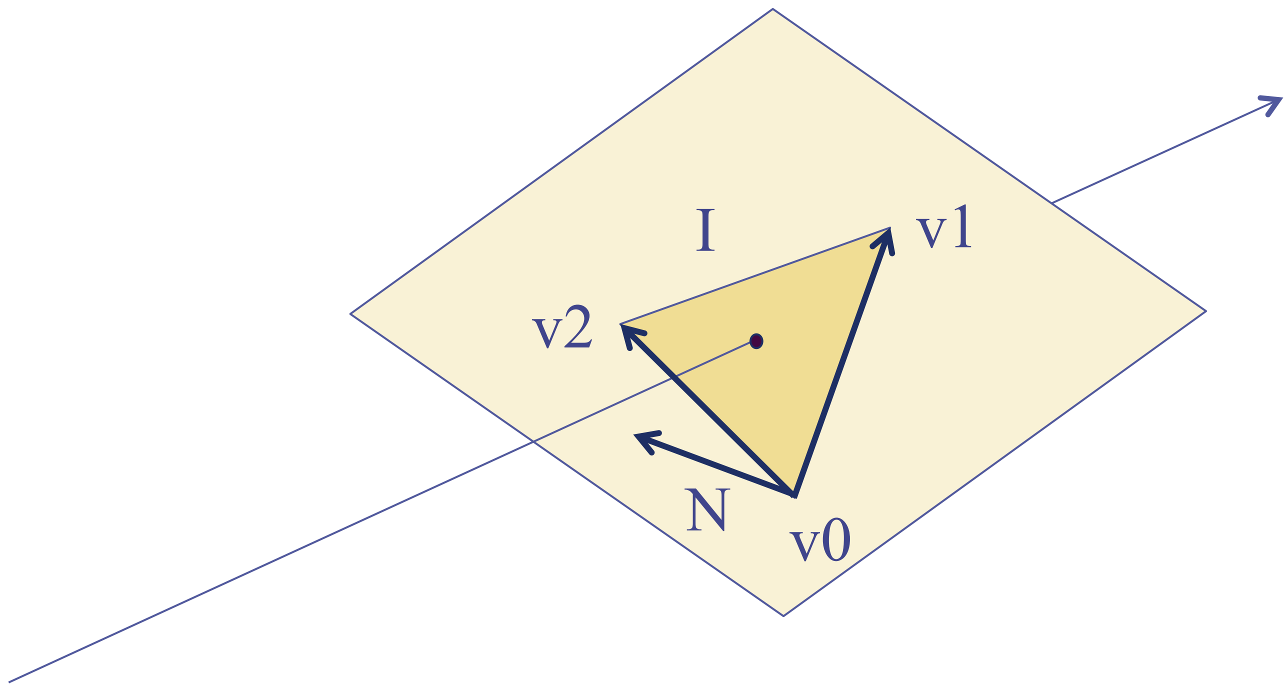
= projection of v0 (as a vector) onto N.

= v0 dot N

Ray/Triangle: Ray/Plane

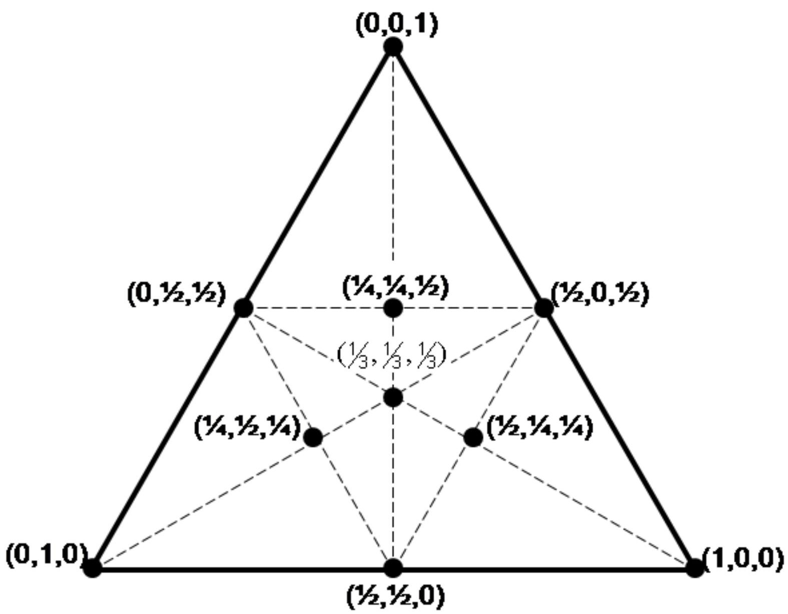
- Now know the plane equation.
- Know the ray equation.
- Compute t then intersection point I

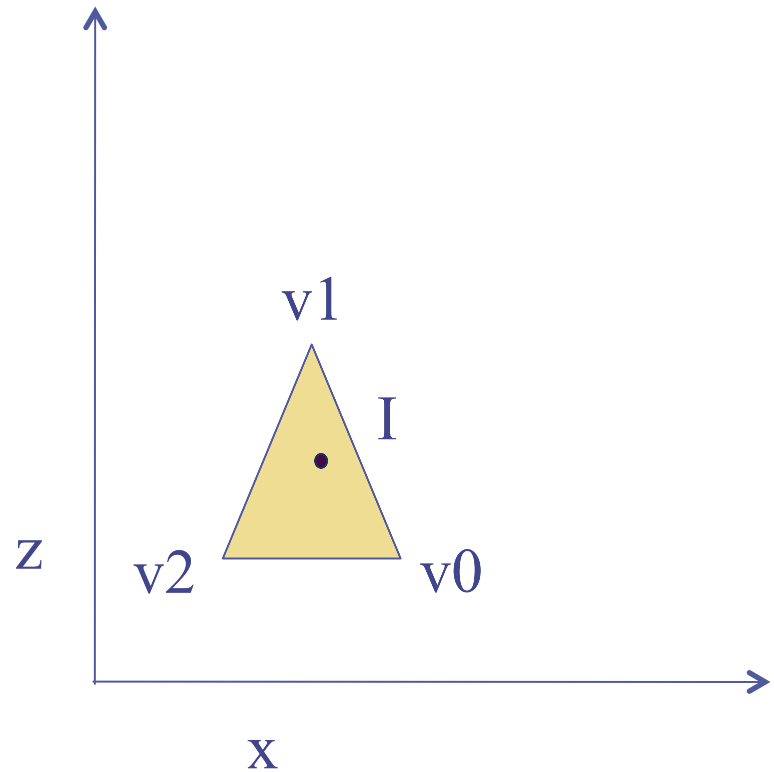
Ray/Triangle: Projection



Project the ray, plane and triangle s.t. drop a coordinate.
Now we only have to work in 2D.
If N = (2,3,1) then y (middle coordinate) is dominant.
Drop all y coordinates in v0, v1, v2 and I.

Ray/Triangle: Is I in triangle?





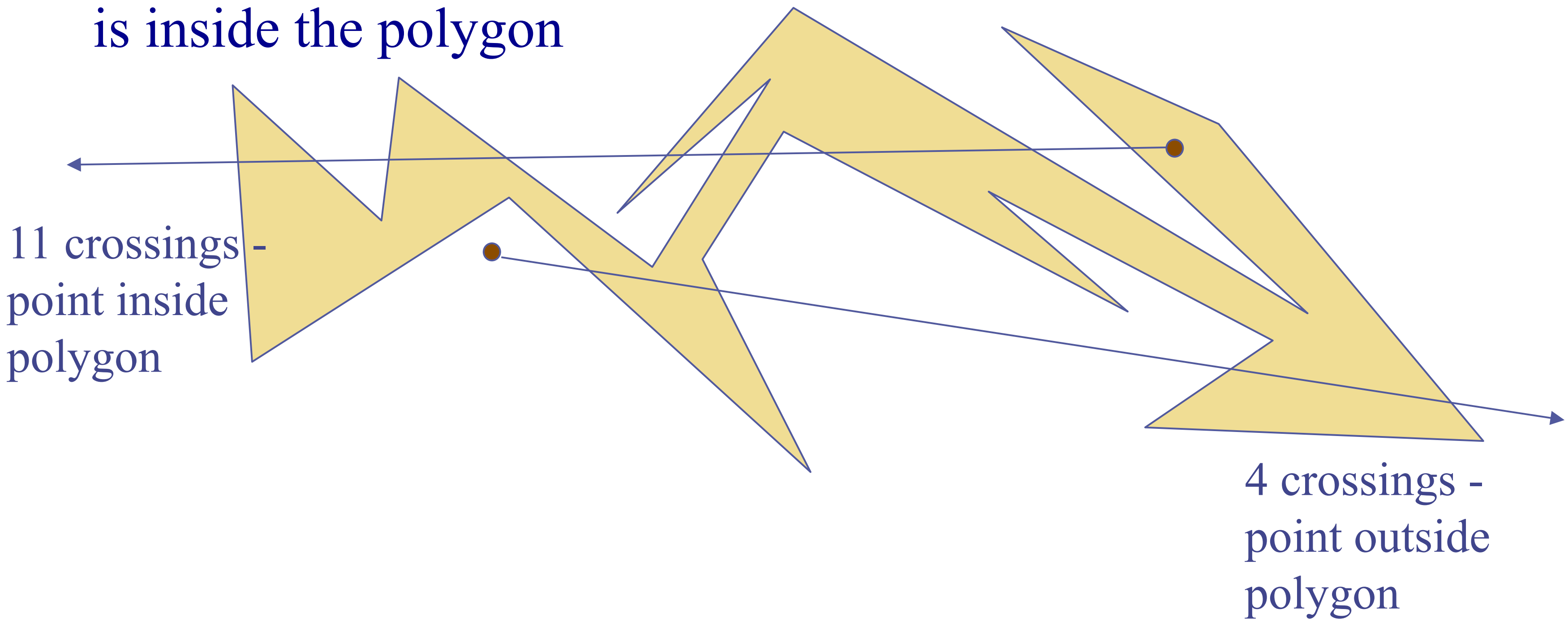
Find the Barycentric coordinates for I.
If all are less than 1, then I is in the triangle.

Ray/Polygon Intersection

- Assume planar polygons
- First perform the ray/plane intersection with the plane in which the polygon lies
- Next, determine whether the intersection point lies within the polygon

Ray/Polygon Intersection

- Idea: shoot a ray from the intersection point in an arbitrary direction
  - if the ray crosses an even number of polygon edges, the point is outside the polygon
  - if the ray crosses an odd number of polygon edges, the point is inside the polygon



Ray/Polygon Intersection

- Polygon:
  - a set of N points $G_n = (x_n, y_n, z_n)$, $n = 0, 1, \ldots N\text{-}1$
- Plane:
  - $Ax + By + Cz + D = 0$, with normal $P_n = (A, B, C)$
- Intersection point:
  - $R_i = (x_i, y_i, z_i)$, R_i on the plane

Ray/Polygon Intersection Algorithm

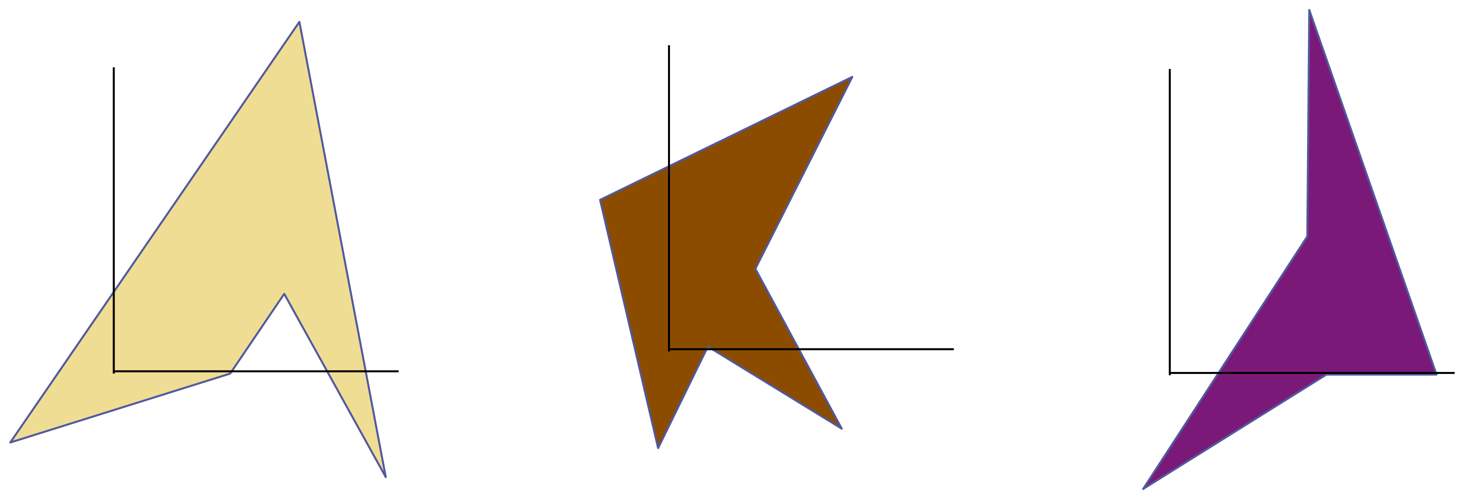
- Step 1:
  - Project the polygon onto a 2D plane
    - one of the coordinate planes
    - simply discard one of the coordinates
    - this will project the polygon onto the plane defined by the other two coordinates
    - topology preserving but not area preserving
    - throw away the coordinate whose magnitude in the plane normal is greatest
    - e.g., if $P_n = (3, -1, -5)$ we would throw away all z coordinates

Ray/Polygon Intersection Algorithm

- Step 2:
  - Translate the polygon so that the intersection point is at the origin
    - apply this translation to all points in the polygon
- Step 3:
  - Send a ray down one of the coordinate axes and count the number of intersections of that ray with the polygon
    - This is done in practice by looking at each polygon edge, one at a time, to see if it crosses the coordinate axis

Ray/Polygon Intersection Algorithm

- Problems:
  - Vertices that lie exactly on the ray
  - Edges that lie exactly on the ray

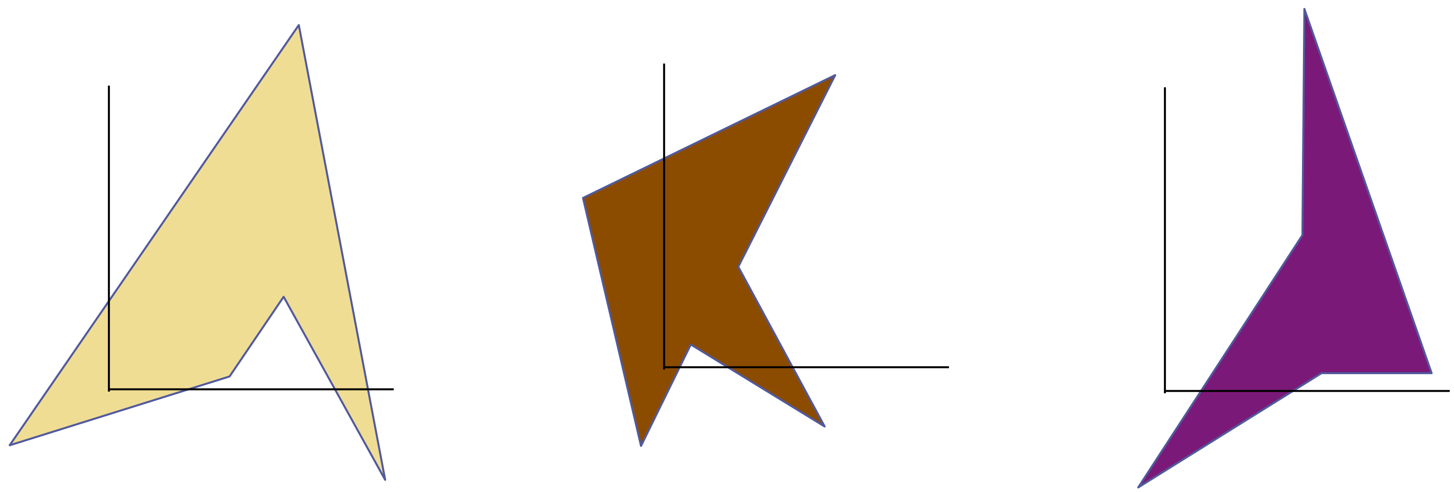


Ray/Polygon Intersection Algorithm

- How do we handle these cases?
  - In essence, shift the vertex or edge up by ε so that it will lie in the plane just above the axis
  - This is done by checking for a y value > 0
- What about if the intersection point is exactly a vertex?
  - Perform a similar ε shift
- What if the intersection point lies exactly on an edge?
  - Specify it as either in or out - as long as we do the specification consistently, it will work okay
  - If the edge is shared by two polygons, the point will be “in” one polygon and “out” of the other.

Ray/Polygon Intersection Algorithm

- The effect of doing this ε shift is:



- These cases all work correctly now.

The Actual Algorithm - part 1

1. Determine the dominant coordinate as the largest magnitude component of P_n
2. For each vertex (x_n, y_n, z_n), $n = 0, 1, \ldots, N-1$ of the polygon, project the vertex onto the dominant coordinate axis, giving (u_n, v_n) vertices.
3. Project the intersection point $(x_{int}, y_{int}, z_{int})$ onto the same coordinate plane as the vertices.
4. Translate all polygon vertices by $(-u_{int}, -v_{int})$, giving (u_n', v_n') vertices.
5. Set numCrossings = 0

The Actual Algorithm - part 2

6. If $v_0' < 0$, set `signHolder = -1`, otherwise set `signHolder = 1`
7. For i = 0 to N-1 (note - when i = N-1, i+1 should be 0)
   a. if $v_{i+1}' < 0$ set `nextSignHolder = -1` else set `nextSignHolder = 1`
   b. if (`signHolder <> nextSignHolder`)
      i. if ($u_i' > 0$ and $u_{i+1}' > 0$) this edge crosses +u' so increment `numCrossings`
      ii. else if ($u_i' > 0$ or $u_{i+1}' > 0$) the edge might cross +u', so compute the intersection with the u' axis

         $u_{cross} = u_i' - v_i' * (u_{i+1}' - u_i') / (v_{i+1}' - v_i')$
      iii. if $u_{cross} > 0$ the edge crosses +u so increment `numCrossings`
   c. set `signHolder = nextSignHolder`
8. If `numCrossings` is odd, the point is inside the polygon

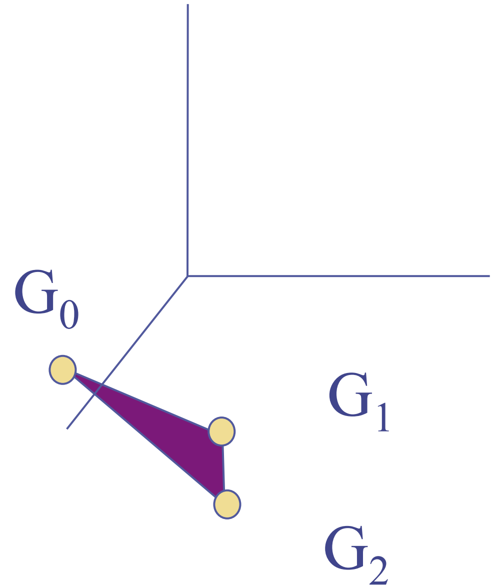
Example

Given a polygon:

$G_0 = (-3, -3, 7)$

$G_1 = (3, -4, 3)$

$G_2 = (4, -5, 4)$



and intersection point

$R_i = (-2, -2, 4)$

Does the intersection point lie within the polygon?

Example

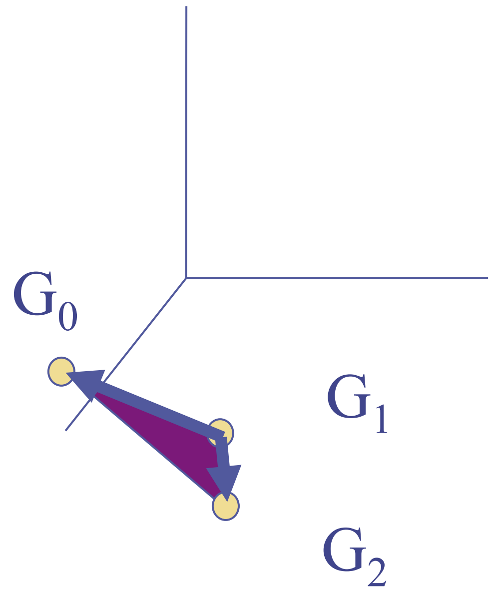
- Step 1: Get the plane normal, determine dominant coordinate

- P_n can be computed from the cross product of two vectors in the plane

- The vertices of the polygon can be used to compute vectors in the plane

$$v_1 = G_0 - G_1 = (-3, -3, 7) - (3, -4, 3)$$
$$= (-6, 1, 4)$$

$$v_2 = G_2 - G_1 = (4, -5, 4) - (3, -4, 3)$$
$$= (1, -1, 1)$$

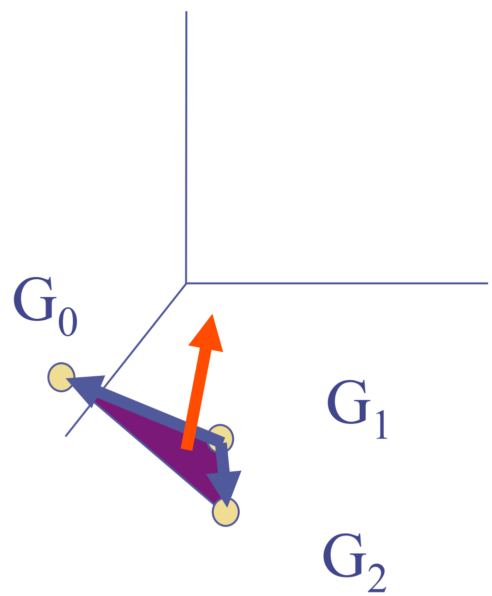


Example

- The plane normal is then v_1 x v_2 (assuming clockwise vertex specification)

$$P_n = (-6, 1, 4) \times (1, -1, 1)$$
$$= (5, 10, 5)$$

So the dominant coordinate is y



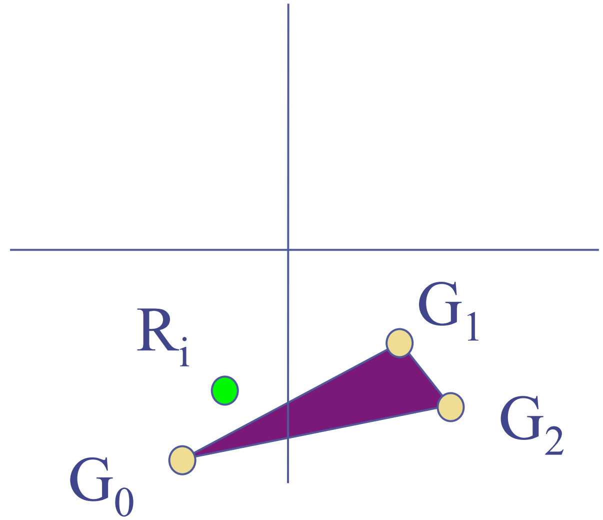
Example

- Step 2: Project the vertices

G_0 = proj of (-3, -3, 7) $\Rightarrow$ (-3, 7)

G_1 = proj of (3, -4, 3) $\Rightarrow$ (3, 3)

G_2 = proj of (4, -5, 4) $\Rightarrow$ (4, 4)



- Step 3: Project the intersection point

R_i = proj of (-2, -2, 4) $\Rightarrow$ (-2, 4)

Example

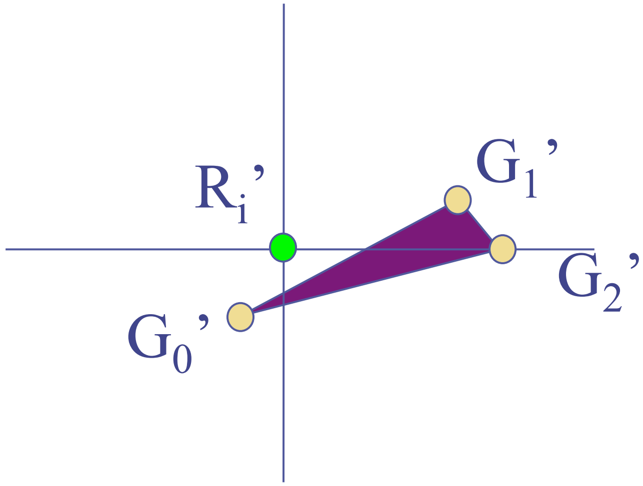
- Step 4: Translate the vertices

$$G_0' = (-3, 7) - (-2, 4) \Rightarrow (-1, 3)$$

$$G_1' = (3, 3) - (-2, 4) \Rightarrow (5, -1)$$

$$G_2' = (4, 4) - (-2, 4) \Rightarrow (6, 0)$$

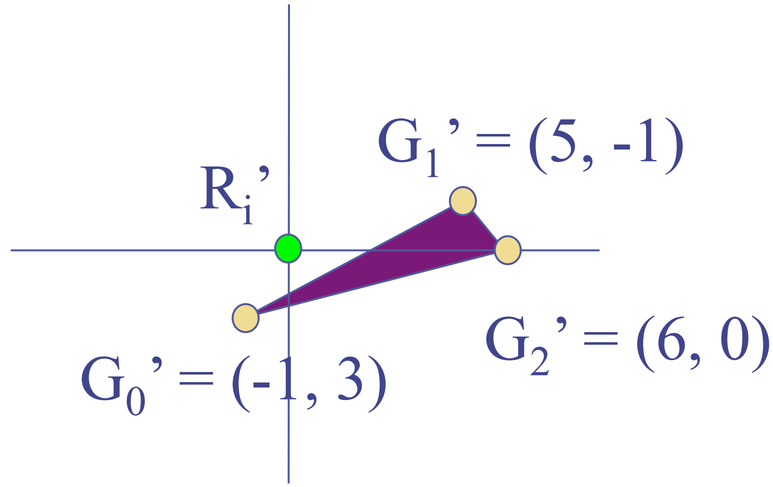
$$R_i' = (-2, 4) - (-2, 4) \Rightarrow (0, 0)$$



Example

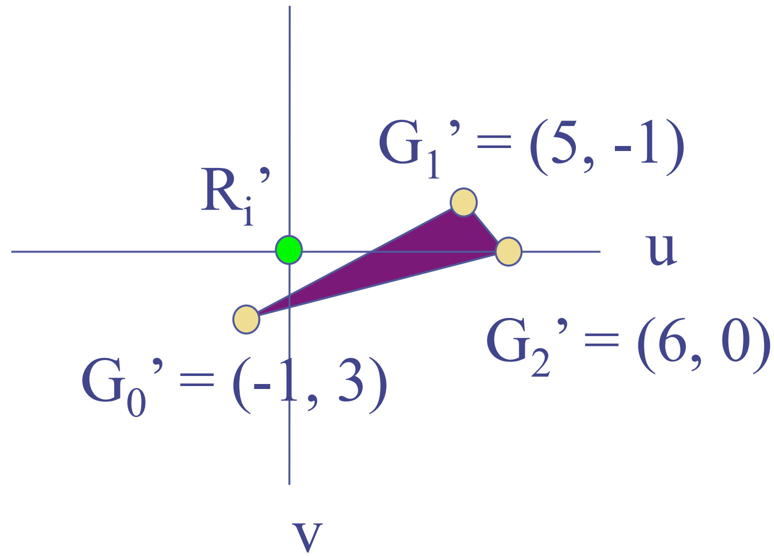
- Step 5: Set numCrossings = 0
- Step 6: $v_0' = 3$, so

  `signHolder = 1`



Example

- Step 7:



i	signHolder	nextSignHolder	numCrossings	intersection point
	+1		0	
0	-1	-1	1	-1-3*(5-(-1))/(-1-3) = 3.5
1	+1	+1	2	
2		+1		

Since numCrossings is even, the point is outside the polygon

Ray/Box Intersection

- i.e., intersecting with bounding boxes
- We will deal with the case of boxes with parallel sides with normals parallel to the coordinate axes.
- Box:
  - Minimum extent $B_l = (x_l, y_l, z_l)$
  - Maximum extent $B_h = (x_h, y_h, z_h)$
- Ray
  - $R(t) = R_o + R_d t$
  - $R_o = (x_o, y_o, z_o)$
  - $R_d = (x_d, y_d, z_d)$

Ray/Box Intersection (2)

- Algorithm:

1. Set $t_{near} = -\infty$, $t_{far} = \infty$
2. For the pair of X planes:
   a. If $x_d = 0$, the ray is parallel to the planes

      If $x_o < x_l$ or $x_o > x_h$ then return FALSE (origin not between planes)

   b. Else the ray is not parallel to the planes, so calculate intersection distances of planes

      $t_1 = (x_l - x_o) / x_d$ (time at which the ray intersects the minimum x plane)

      $t_2 = (x_h - x_o) / x_d$ (time at which the ray intersects the maximum x plane)

      If $t_1 > t_2$ swap t_1 and t_2

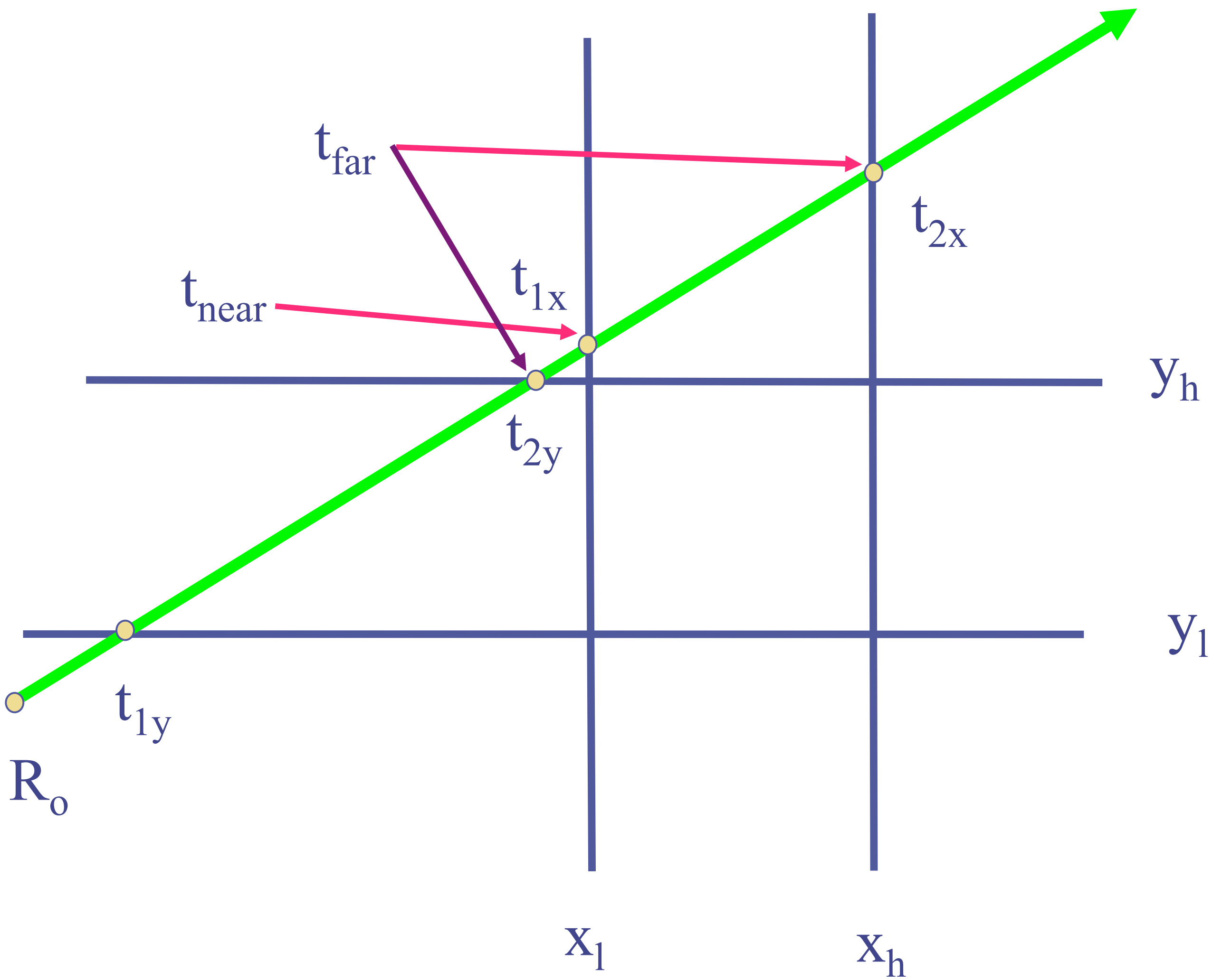
      If $t_1 > t_{near}$, set $t_{near} = t_1$

      If $t_2 < t_{far}$, set $t_{far} = t_2$

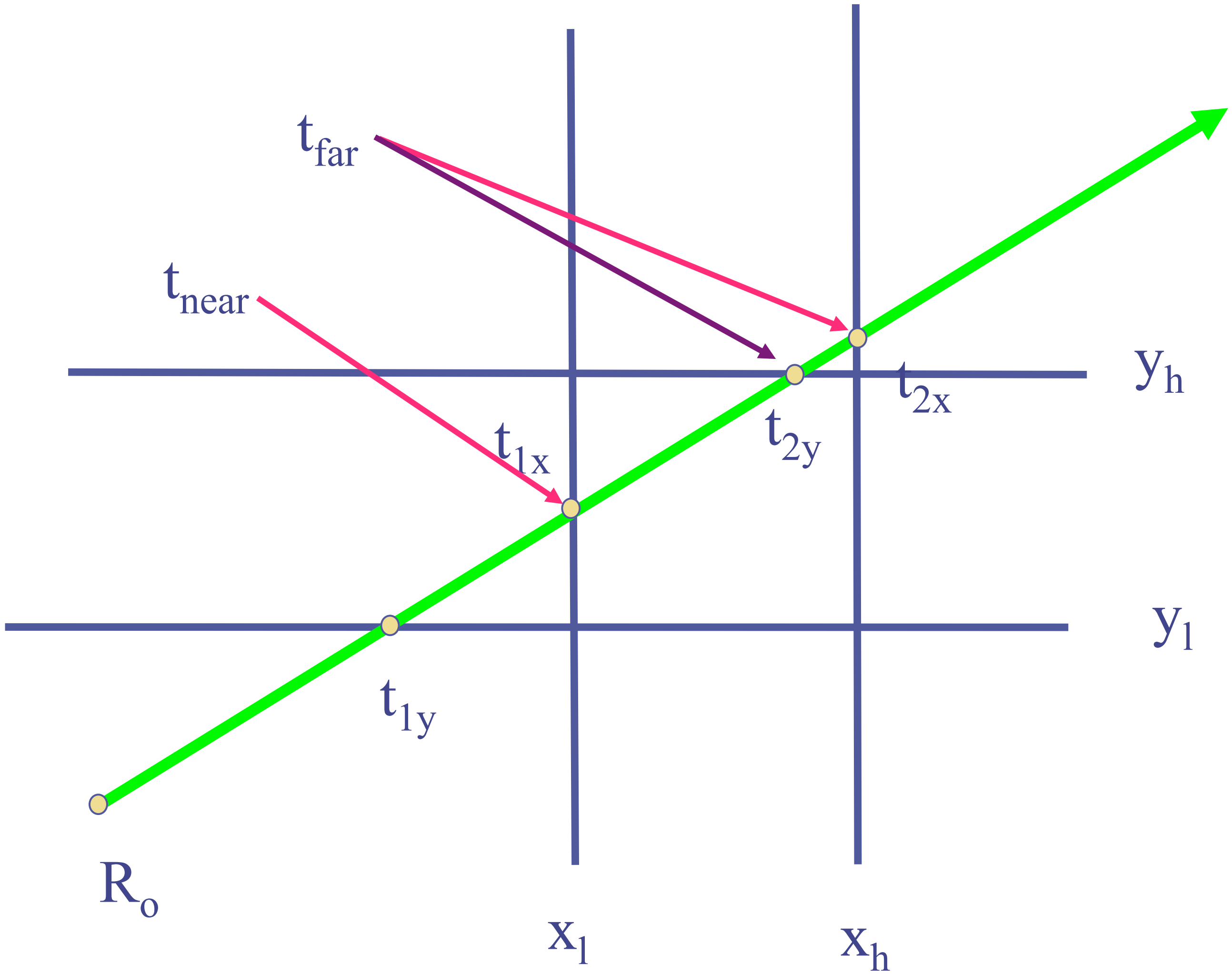
      If $t_{near} > t_{far}$ box is missed so return FALSE

      If $t_{far} < 0$ box is behind ray so return FALSE
3. Repeat step 2 for Y then Z
4. All tests survived, so return TRUE

Ray/Box Intersection example 1



Ray/Box Intersection example 2



Ray/Box Intersection Example 3

- Given the ray
  - $R_o = (0, 4, 2)$
  - $R_d = (0.213, -0.436, 0.873)$
- And the box
  - $B_l = (-1, 2, 1)$
  - $B_h = (3, 3, 3)$
- Does the ray hit the box?

Ray/Box Intersection Example 3

- $t_{1x} = (-1 - 0) / 0.218 = -4.59$
- $t_{2x} = (3 - 0) / 0.218 = 13.8$
- So $t_{near} = -4.59$, $t_{far} = 13.8$
- $t_{1y} = (2 - 4) / (-.436) = 4.59$
- $t_{2y} = (3 - 4) / (-.436) = 2.29$
- $t_{1y} > t_{2y}$ so swap
- Then $t_{near} = 2.29$, $t_{far} = 4.59$
- $t_{1z} = (1 - 2) / (0.873) = -1.15$
- $t_{2z} = (3 - 2) / (0.873) = 1.15$
- t_{near} = stays the same
- $t_{far} = 1.15$
- So, $t_{near} > t_{far}$ so the ray misses the box