clipping

Line Intersect Plane

- 3 cases:
  1. no intersection – plane and line are parallel
  2. intersection at a point
  3. intersection of the whole line

- plane can be represented as the set of points p:

$$(p - p_0) \cdot n = 0 \qquad p_0 \text{ is point on plane}$$

makes sense b/c $(p - p_0)$ must be vector in that plane so if n is normal of plane, dot product of both must be 0 by def of normal to plane

- vector equation for line:

$$p = l_0 + l d \quad d \in \mathbb{R} \qquad l_0 \text{ is point on line}, \; l \text{ is vector in dir of line}$$

- we have eq of line made up of points so want to see if any of these points on line intersect plane

- so substitute line eq (all its points) for p in plane equation

$$((l_0 + ld) - p_0) \cdot n = 0$$

$$(ld + (l_0 - p_0)) \cdot n = 0 \quad \text{associativity}$$

$$(ld) \cdot n + (l_0 - p_0) \cdot n = 0 \quad \text{distribute}$$

$$(l \cdot n) d + (l_0 - p_0) \cdot n = 0 \quad \text{property of scalar mult}$$

solve for d

$$(l \cdot n) d = -(l_0 - p_0) \cdot n$$

$$\boxed{d = \frac{(p_0 - l_0) \cdot n}{l \cdot n}}$$

- examine equation for d to get intersection
  - if $l \cdot n = 0$ that means line and plane are parallel and there are 2 cases here
  - first case: intersection is the full line

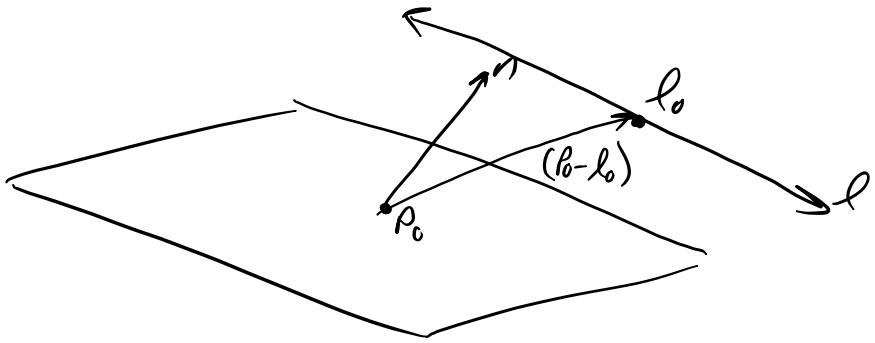
happens when $(p_0 - l_0) \cdot n = 0$

↗ can only happen when vector equals

by $(P_0 - l_0)$ is on plane so if line starts on plane and parallel to plane then line completely on plane

- Second case; no intersection

happens when $(P_0 - l_0) \cdot n \neq 0$



means line stays parallel to plane and never intersects

- if $l \cdot n \neq 0$ then there is a single point of intersection b/c if a line isn't parallel to plane, it's bound to intersect it once

  - in this case get the 1 point of intersection by substitute d into line eq:

$$p = l_0 + ld$$

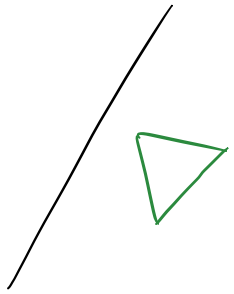
Why?

- You don't want to draw triangles that can't be viewed
- as you get closer, some triangles can become infinitely big

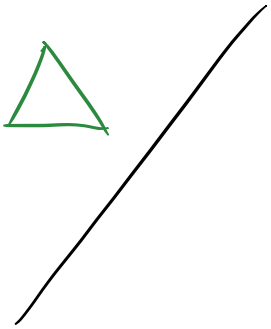
Clip Against Plane

- 4 cases

1) Triangle completely on "inside" of plane so no clipping and full original triangle drawn

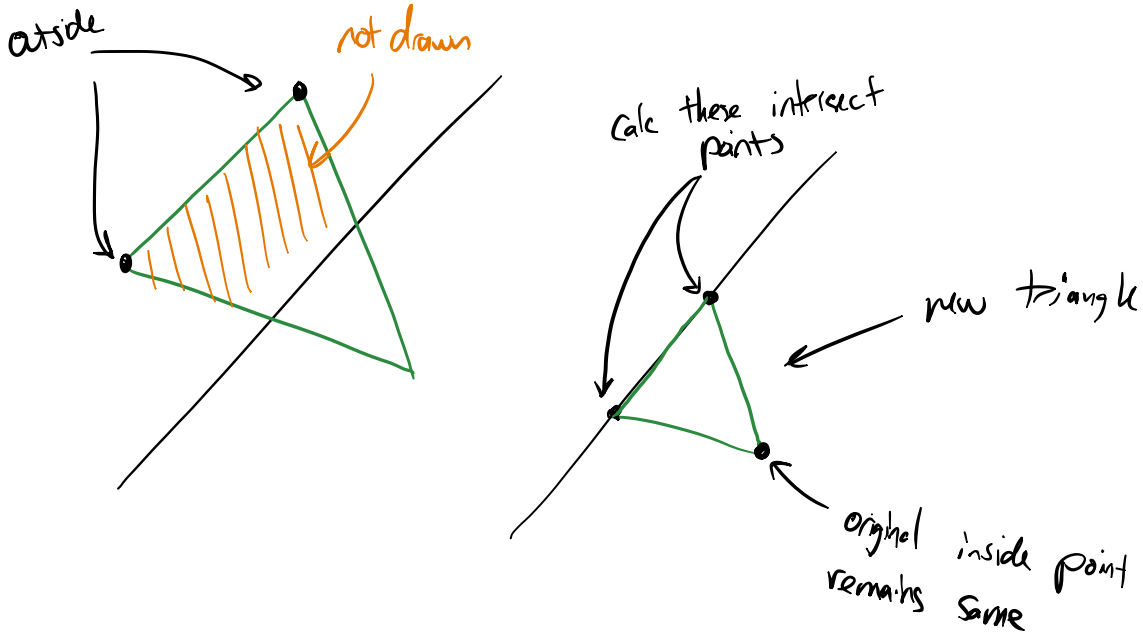


2) Triangle completely on outside of plane so nothing drawn

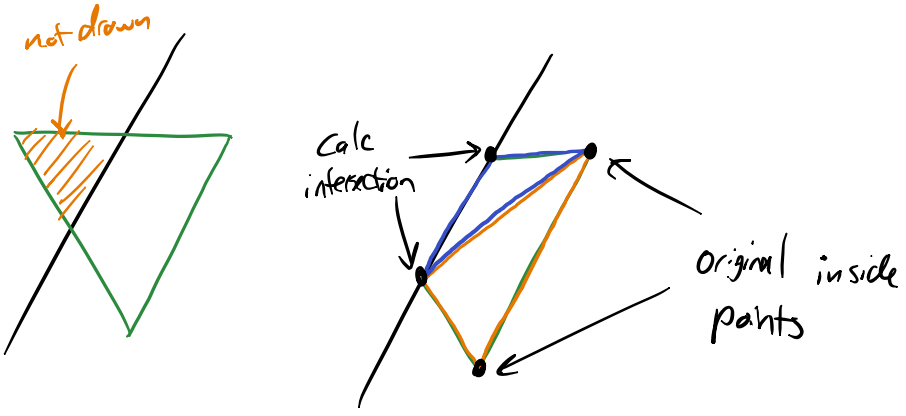


3) Triangle has 1 inside and 2

3) Triangle has 1 inside point and 2 outside points so just becomes a smaller triangle drawn

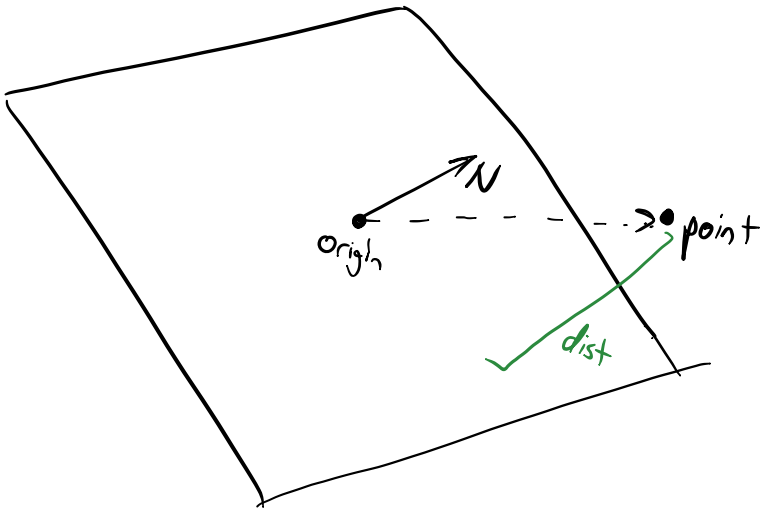


4) Triangle has 2 inside points and 1 outside point so this forms a quad which can then be divided into 2 new triangles



Getting Inside / Outside points

- determine inside/outside by getting distance of point to plane and sign of dist is in/out
- assume that plane and point are at the origin
  - normal vector doesn't change b/c assume that it's at origin
  - to translate plane w/ point of interest to origin assume that plane is at origin but translate point by negative of same point on plane
    - negative of point on plane translates space so plane is at origin and point is same relative dist away from plane



point on plane is $P_0 = (x, y, z)$

normal of plane is $N = \langle a, b, c \rangle$ ← unit vector

point moved to origin is point = $\langle x_0 - x, y_0 - y, z_0 - z \rangle$

Get dist of point to plane in plane norm vector units by getting projection of point onto plane normal

* however, realize that N is unit vector so scalar its multiplied by is the actual distance *

$$\text{proj}_N(\text{point}) = \underbrace{(\text{point} \cdot N)}_{\text{actual dist}} N$$

$$= \left(a(x_0 - x) + b(y_0 - y) + c(z_0 - z)\right) N$$

$$= \left(N \cdot (x_0, y_0, z_0) - N \cdot p_0\right) N$$

(x_0, y_0, z_0): original loc of point of interest

distance: $\overbrace{N \cdot (x_0, y_0, z_0) - N \cdot p_0}^{\text{what is used in the code}} = N \cdot \text{point}$

- if this coefficient is positive, point lies "inside"
- if this coefficient is negative, point lies "outside"

Clipping against sides of screen algorithm

- we have clipped against the plane right in front of the camera and have projected the triangles, now we need to cut off the parts that go beyond the 4 screen bounds
- each triangle to clip can generate at most 2 other triangles for each plane it clips against
- loop through clipping all 4 bounds of the screen

- start with a queue that has triangles to clip against a certain screen plane and only clip these triangles against that screen plane
  - the generated triangles from clipping against this screen plane shouldn't be tested again because they were generated so that no clipping on this plane is necessary
- once all the triangles originally in the queue have been clipped against the screen plane, there will be the generated triangles from clipping, and now clip them against the next screen plane in the same process

pseudo code:

```
for tri in tris_projected
        queue = [tri]
        for side in screen_bounds(4)
                while tris_not_yet_clipped in queue
                        tri_to_clip = queue.shift() # get first elem in
queue
                        clipped_tris = clip(tri, side)
                        queue.push(clipped_tris)
                # all tris clipped after while loop for this side, clip
against all the
                # new triangles generated by clipping against this side for
the next side
```