14 - Geometric Intersections for Raytracing

Dr. Rafael Kuffner dos Anjos

Agenda

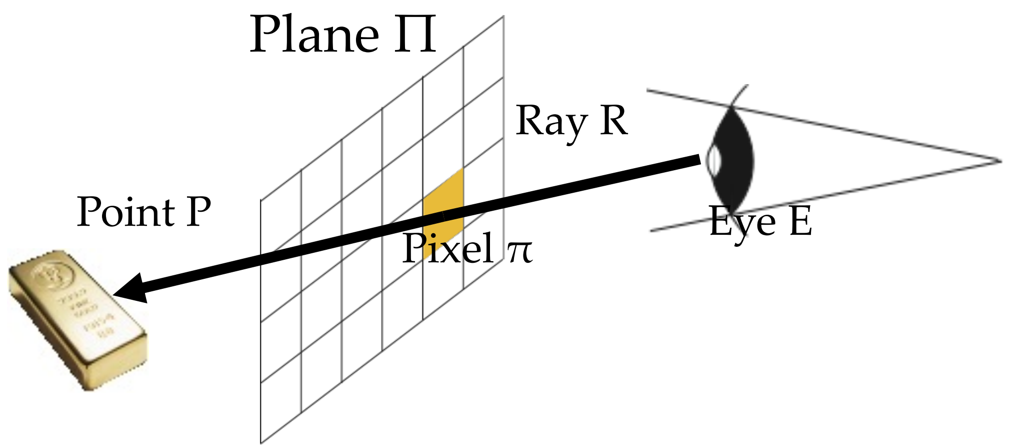
- Raycasting
- Shadows
- 1D intersection
- 2D intersection
- 2.5D intersection
- 3D intersection

Assumptions

- Assume we have:
  - A triangle mesh M, with:
    - Vertex positions
    - Vertex normals
    - Texture coordinates (& a texture)
  - An eye E
  - An image plane Π made up of pixels

High-Level Design

- For each pixel π:
  - Start at the eye (E)
  - Cast a ray R through π
  - Keep going until you find P
    - Where the light came from



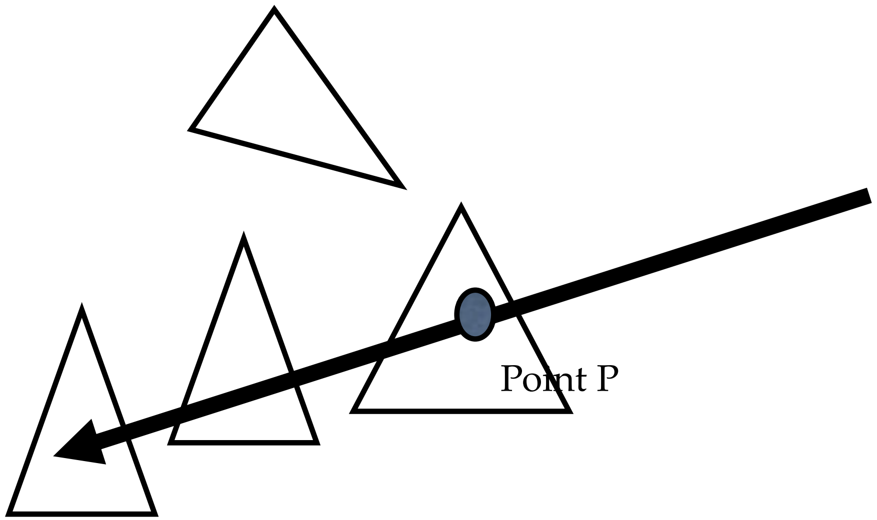
Ray Casting

```
for each pixel p_ij = (i,j)
     let R_ij be the ray from E through p_ij
     cast ray R into scene
     find point P where R first intersects an object
     compute lighting at point P
     store in image[i,j]
```

- How do we find P?

Finding P

- P is a point on a triangle
- But which triangle?
- Simple strategy:
  - Test all triangles T
  - Take the one closest to the eye
- We could sort, but we actually select
  - Because we only care about the closest



Selecting an Item

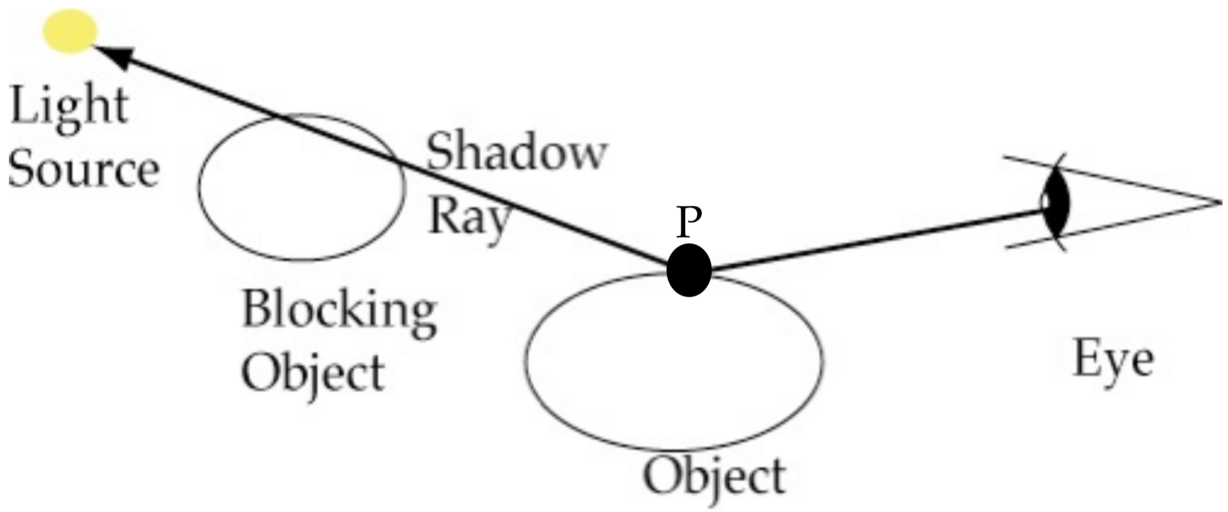
- Set distance = infinity
- Then for each triangle T
  - Compute the point P where R intersects T
  - Compute the distance d
  - If d < distance
    - Compute the lighting and store it
- At the end, we have the correct value

Basic Raycaster

```
for each pixel p_ij = (i,j)
     let R_ij be the ray from E through p_ij
     closest = infinity
     for each triangle T
          find P at intersection of R_ij and T
          find d = distance(P,E)
          if (d > closest)
               continue
          compute lighting at point P
          store in image[i,j]
```

Adding Shadows

- We assumed light cannot be blocked
- Other objects, however, might be in the way
- So we cast a *shadow ray* from P to the light
- And see if P is visible from the light



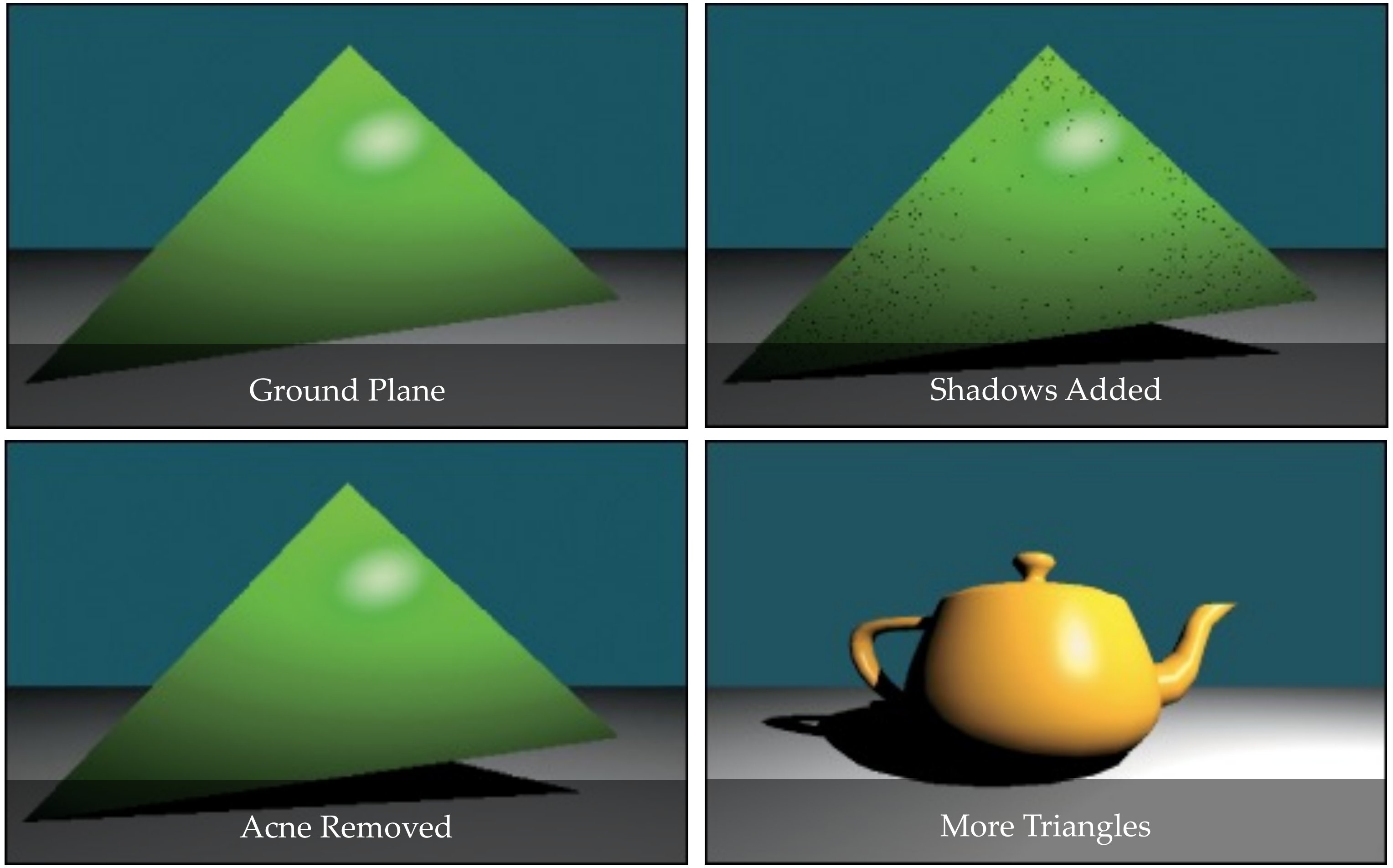
UNIVERSITY OF LEEDS

Shadow Ray Casting

- This uses the same method that found P
- But along the ray from L to P
- If P is the closest to L, it’s lit
- We allow a small tolerance ε>0
  - To avoid *shadow acne*

UNIVERSITY OF LEEDS

Shadow Examples



Rasterisation vs Raytracing

```
for each pixel p_ij = (i,j)
      depth[i,j] = infinity

for each triangle T
      for each pixel p_ij = (i,j)
            let R_ij be the ray from E through p_ij
            find P at intersection of R_ij and T
            find d = distance(P,E)
            if (d > depth[i,j])
                  continue
            compute lighting at point P
            store in image[i,j]
```

- What are we doing here?

More similar than you think

- Just invert the two loops:
  - Triangle -> pixel : Rasterization
  - Pixel->triangle: Raytracing

```
for each triangle T
        for each pixel p_ij = (i,j)

for each pixel p_ij = (i,j)
        for each triangle T
```

- On rasterization we try not to iterate all pixels
  - We can determine which pixels will be generated
- On raytracing, we try not to test all triangles
  - Acceleration structures

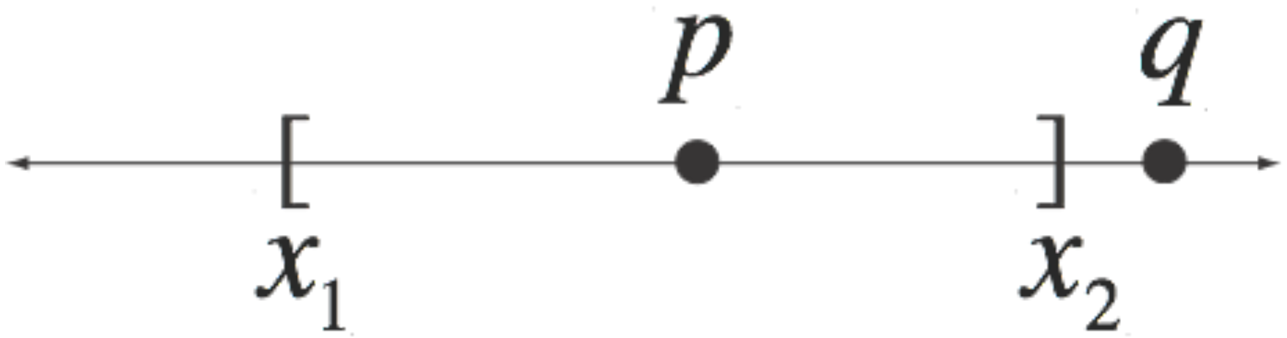
Geometric Intersection

- Recurrent problem in graphics
  - Interpolation
  - Raytracing of all forms
  - Mesh manifold tests
  - Collision Detection
  - Mesh processing

Intersection Tests

- 1D: point in segment, 2 segments
- 2D: points, lines, segments, circles, boxes, triangles, polygons
- 2.5D: rays & triangles in 3D
- 3D: points, lines, segments, planes, spheres, boxes, tetrahedra, polyhedra, cylinders, capsules

1D: The Easy One

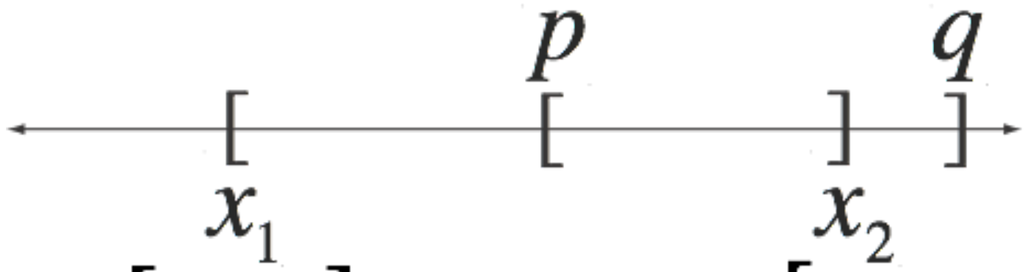


Is p in $[x_1, x_2]$?

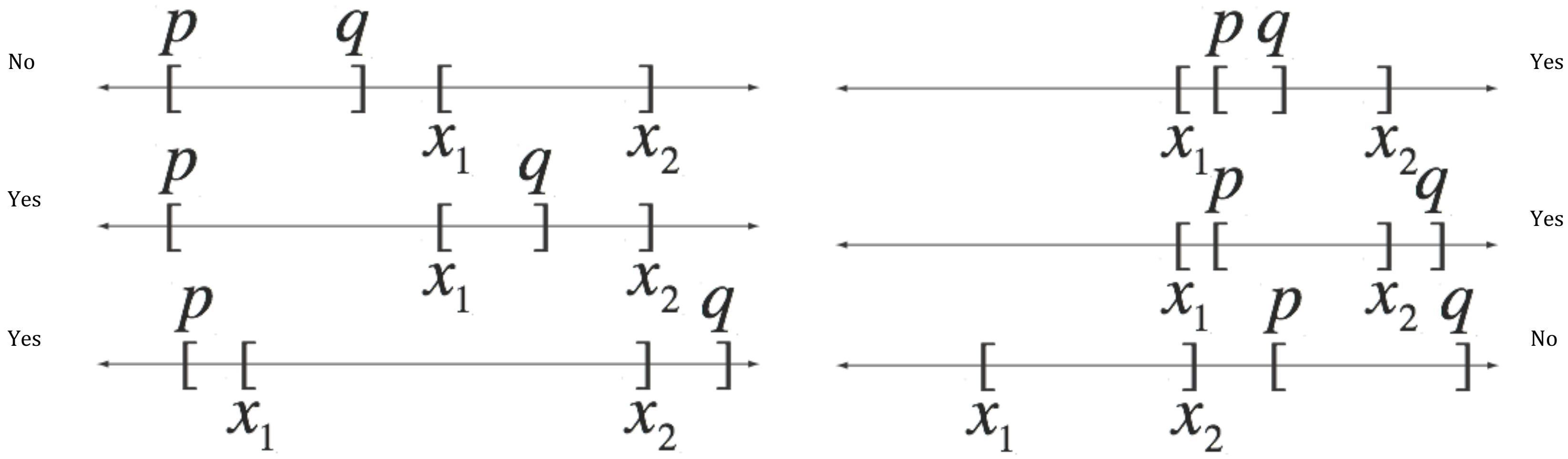
Yes, if $x_1 \le p \le x_2$

Is q?

1D: Two Segments



Does $[p, q]$ intersect $[x_1, x_2]$?



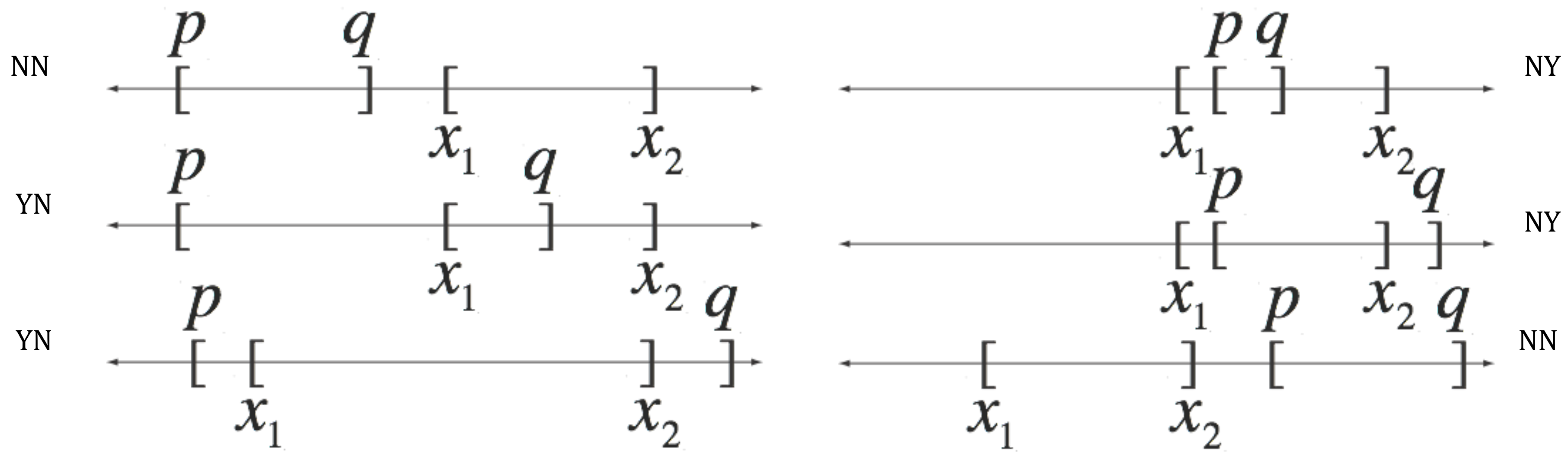
Test

$$\text{Yes, if } x_1 \in [p,q], x_2 \in [p,q],$$
$$p \in [x_1, x_2], \text{ or } q \in [x_1, x_2]$$

- But is there an easier way?

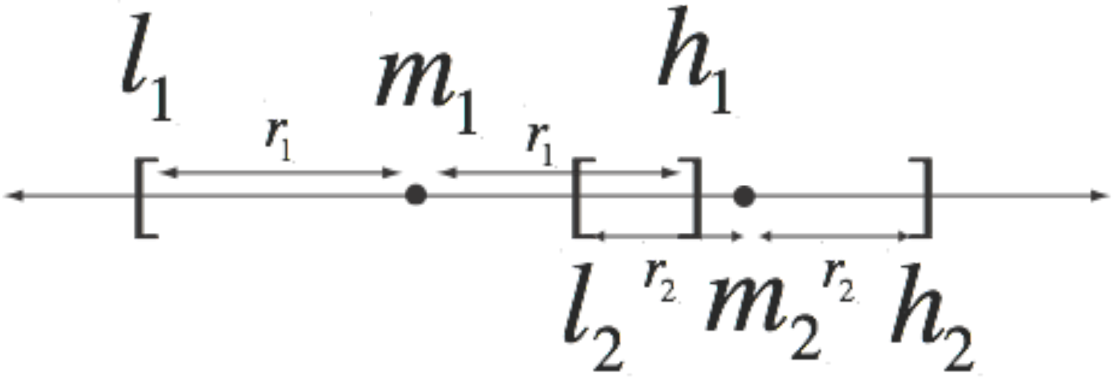
Optimized

$$\text{Yes, if } x_1 \in [p,q], \text{ or } p \in [x_1,x_2]$$



UNIVERSITY OF LEEDS

Better Yet



Does $[l_1, h_1]$ intersect $[l_2, h_2]$?

Let $m_1 = \dfrac{l_1 + h_1}{2}$ $\quad r_1 = \dfrac{h_1 - l_1}{2}$

Is $|m_1 - m_2| \le r_1 + r_2$?

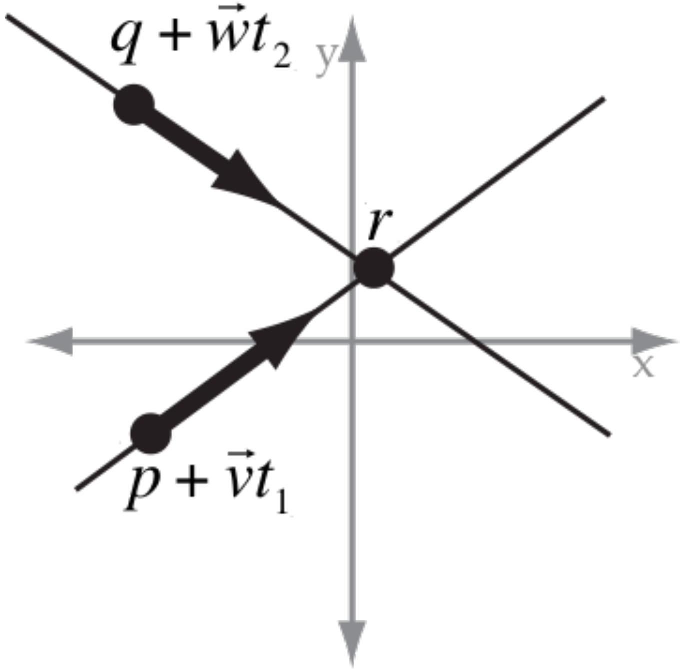
UNIVERSITY OF LEEDS

2D Intersections

- Line - Line
- Box - Box (AABB)
- Circle - Circle
- Triangle - Triangle
- Polygon - Polygon

Two Parametric Lines

- Each has a separate parameter
- We want to find *r*
  - by finding t_1 at *r*
  - or t_2 at *r*



UNIVERSITY OF LEEDS

$$p + \vec{v}t_1 = q + \vec{w}t_2$$

$$or:$$

$$p_x + v_x t_1 = q_x + w_x t_2$$
$$p_y + v_y t_1 = q_y + w_y t_2$$

$$or:$$

$$v_x t_1 - w_x t_2 = q_x - p_x$$
$$\mathrm{v}_y t_1 - w_y t_2 = q_y - p_y$$

System of Equations

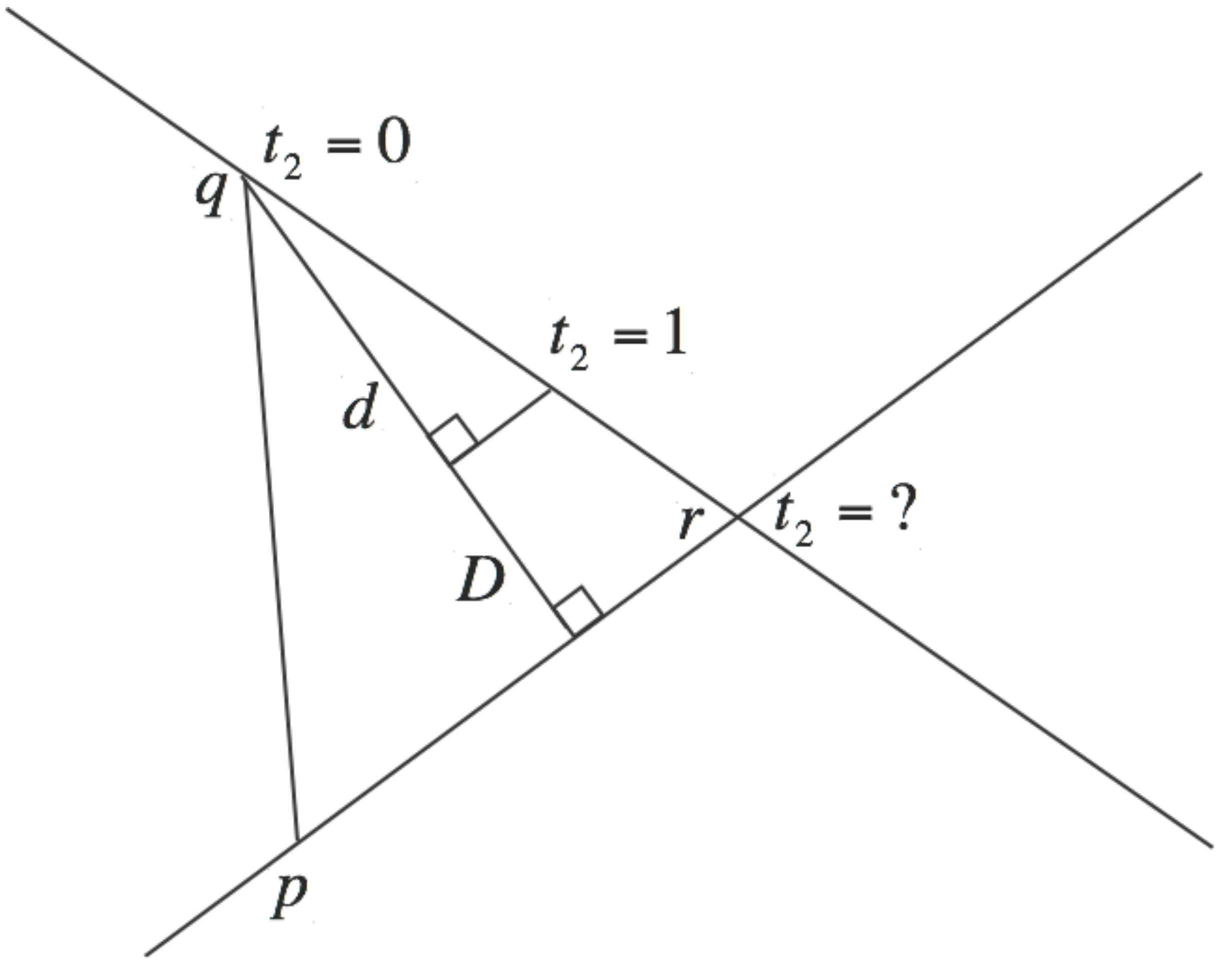
- Two equations in two unknowns,
- which means we can solve it . . .
- or we can find a faster way

Similar Triangles

- Find t_2 at r
- Similar triangles, so

$$\frac{t_2}{1} = \frac{D}{d}$$

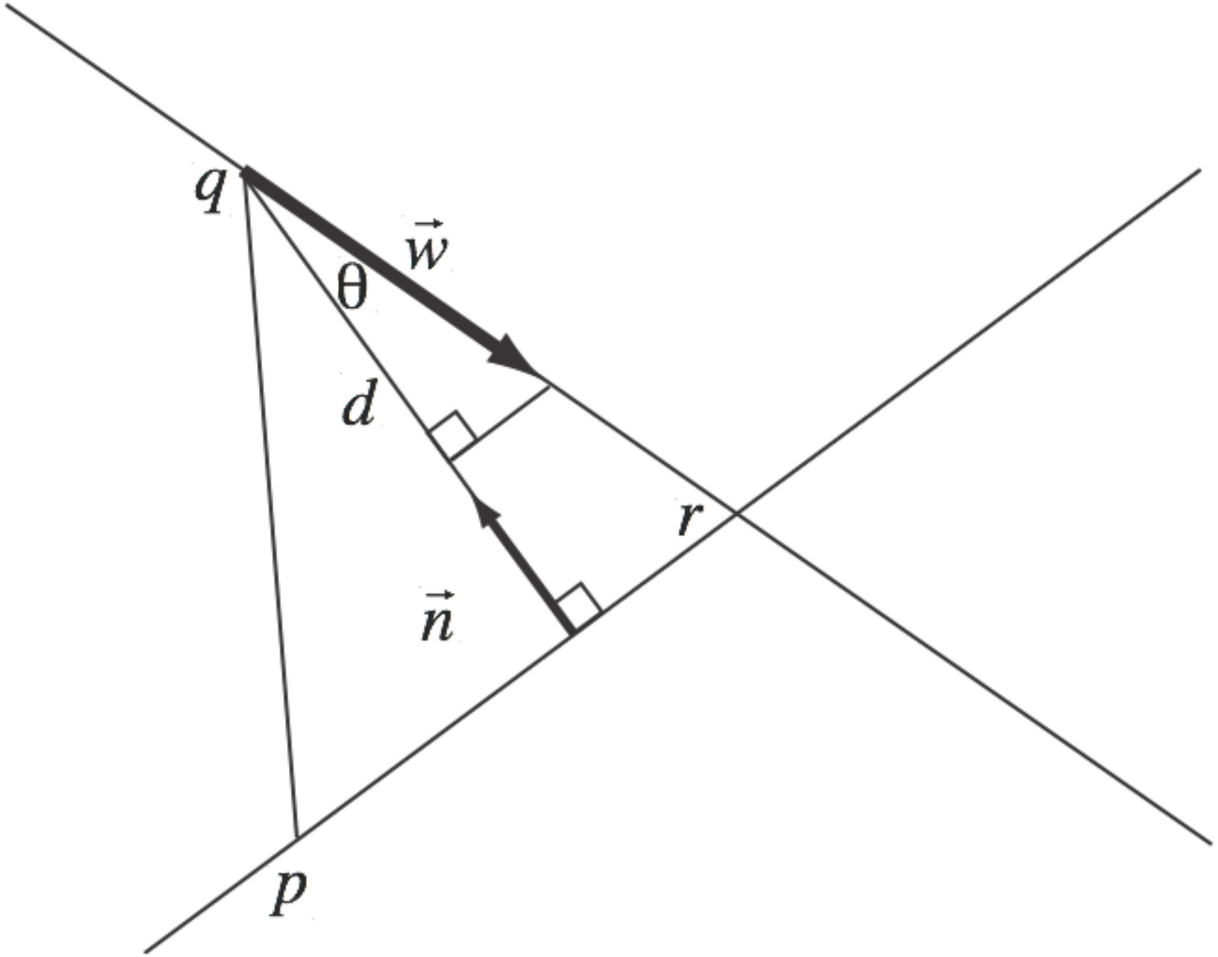
- find d and D



Finding d

- Use the dot product:

$$
\begin{aligned}
d &= \left\|\Pi_{\vec{n}}\left(\vec{w}\right)\right\| \\
&= \left\|\vec{w}\right\| \cos \boldsymbol{q} \\
&= \left\|\vec{w}\right\| \frac{\vec{n} \cdot \vec{w}}{\left\|\vec{w}\right\|\left\|\vec{n}\right\|} \\
&= \frac{\vec{n} \cdot \vec{w}}{\left\|\vec{n}\right\|}
\end{aligned}
$$

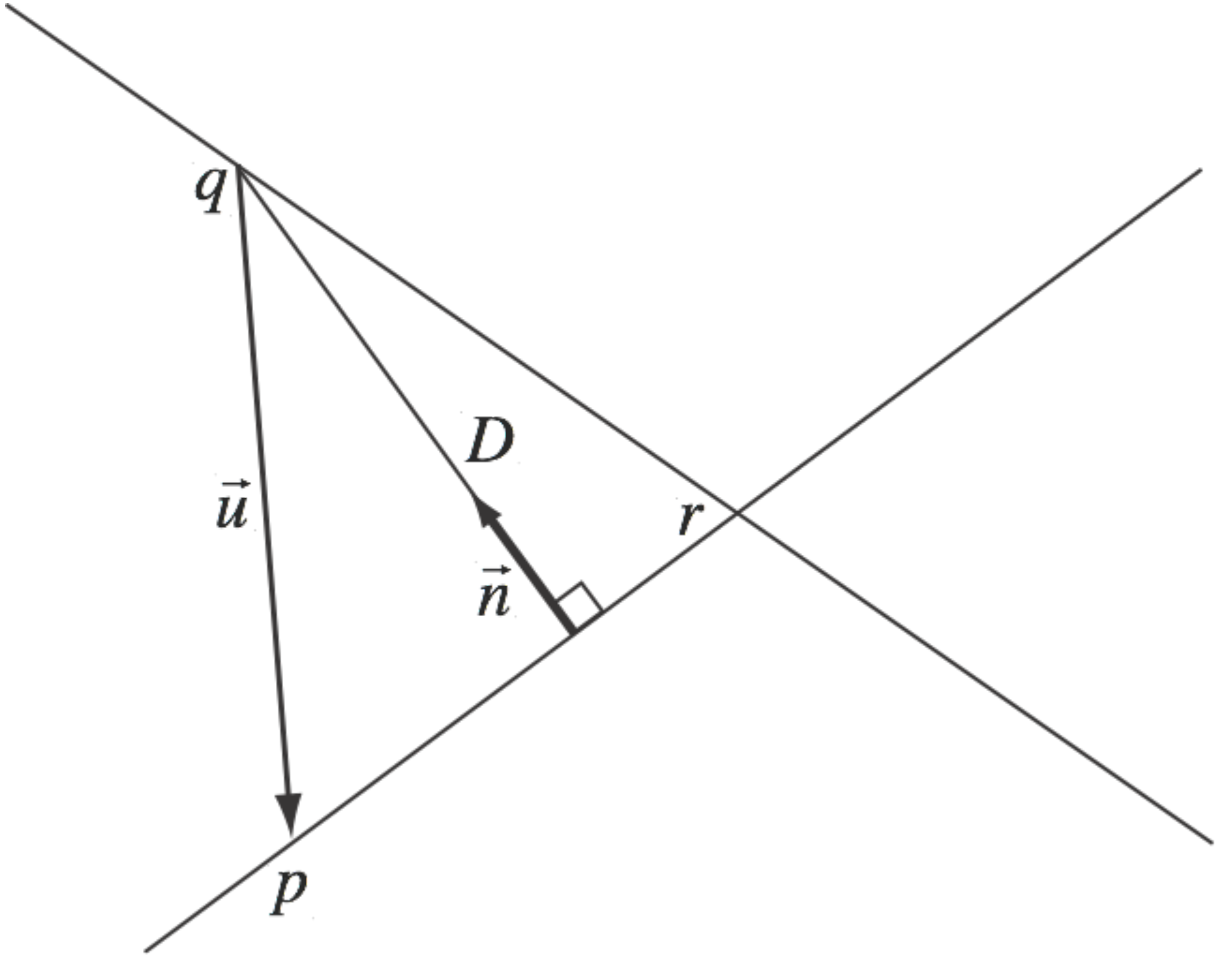


UNIVERSITY OF LEEDS

Finding D

- Similarly,

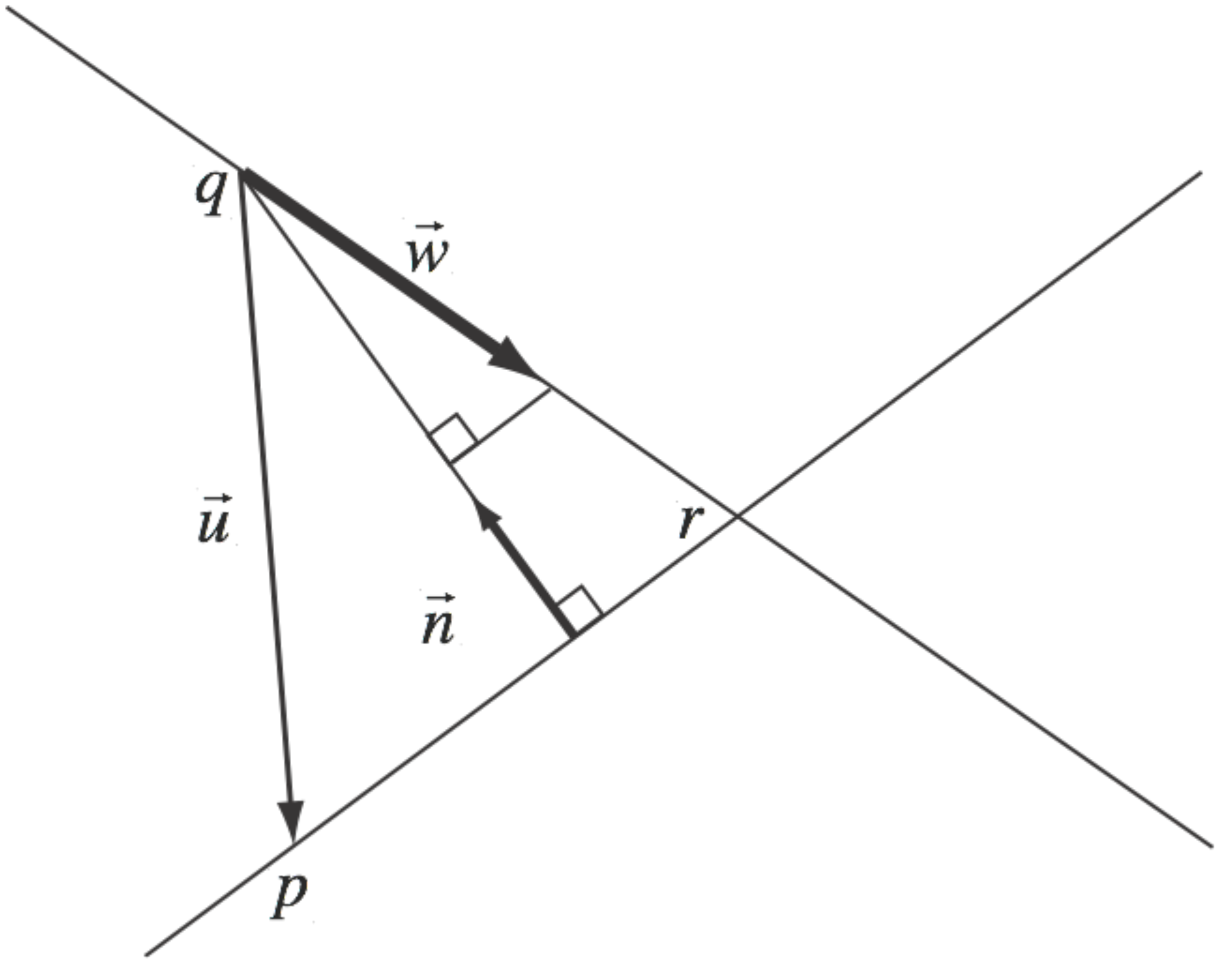
$$
\begin{aligned}
D &= \left\|\Pi_{\vec{n}}\left(\vec{u}\right)\right\| \\
&= \left\|\vec{u}\right\|\cos q \\
&= \left\|\vec{u}\right\|\frac{\vec{n}\cdot\vec{u}}{\left\|\vec{u}\right\|\left\|\vec{n}\right\|} \\
&= \frac{\vec{n}\cdot\vec{u}}{\left\|\vec{n}\right\|}
\end{aligned}
$$



UNIVERSITY OF LEEDS

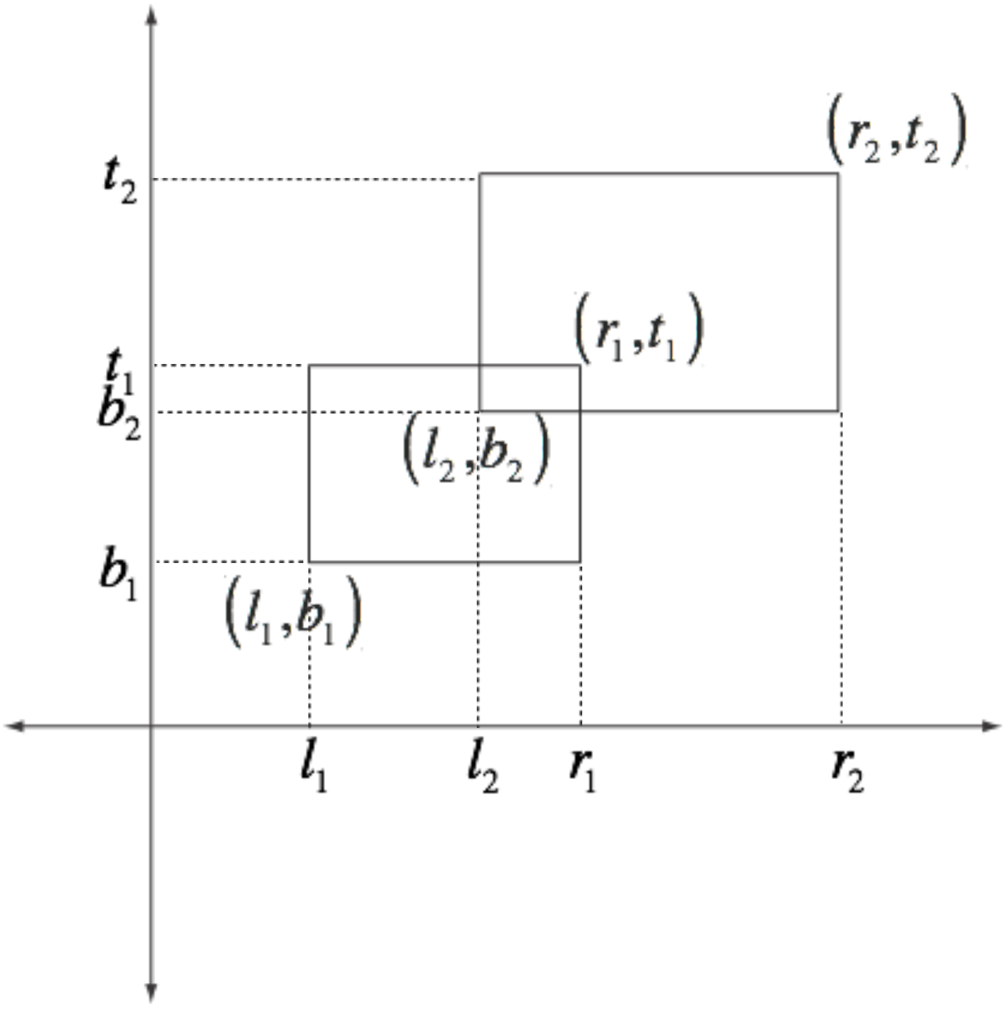
Solution

$$
\begin{aligned}
r &= q + \vec{w}t_2 \\
&= q + \vec{w}\left(\frac{D}{d}\right) \\
&= q + \vec{w}\left(\frac{\frac{\vec{n}\cdot\vec{u}}{\|\vec{n}\|}}{\frac{\vec{n}\cdot\vec{w}}{\|\vec{n}\|}}\right) \\
&= q + \frac{\vec{n}\cdot\vec{u}}{\vec{n}\cdot\vec{w}}\vec{w}
\end{aligned}
$$



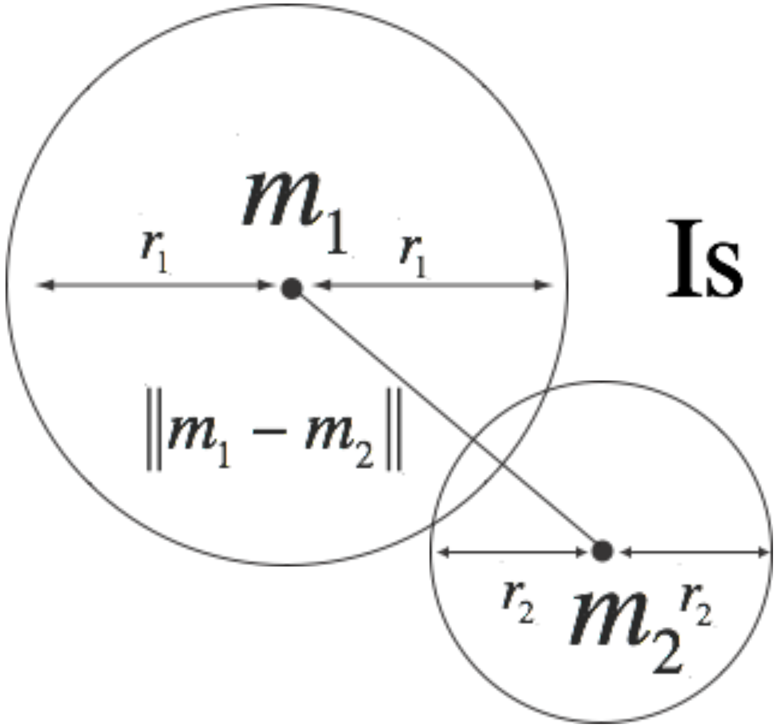
Axis-Aligned Boxes (AAB)

- Line segment test in both of x, y axes



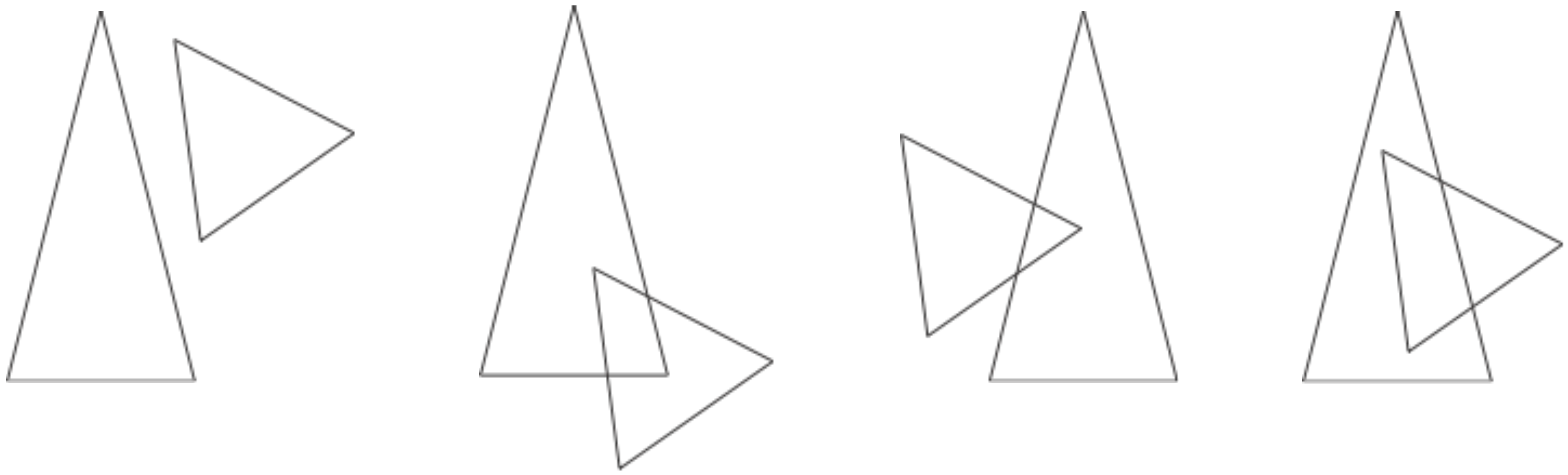
UNIVERSITY OF LEEDS

Circle - Circle



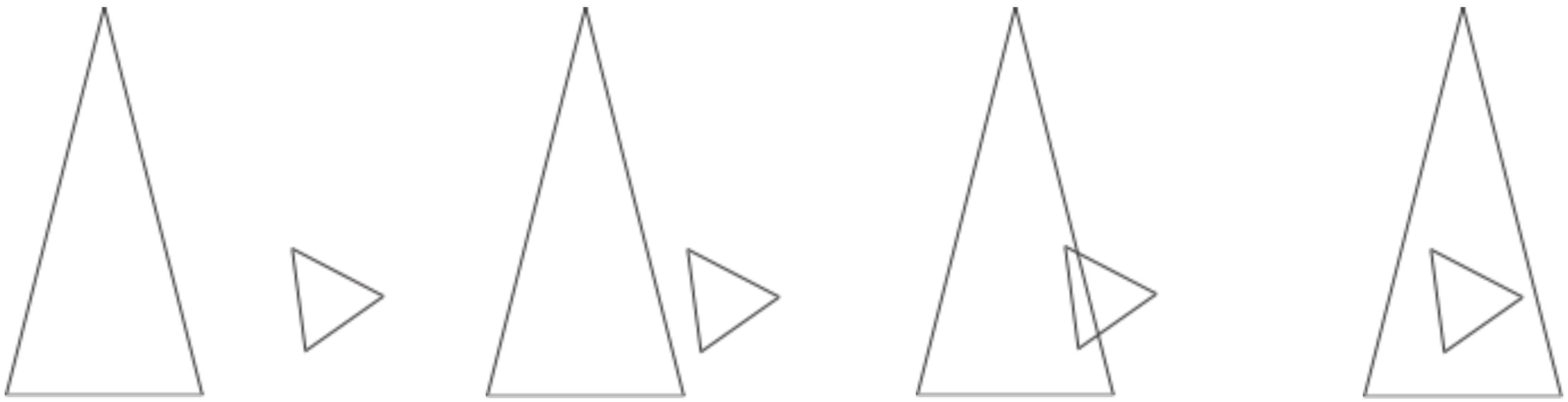
Is $\|m_1 - m_2\| \le r_1 + r_2$?

Triangle-Triangle



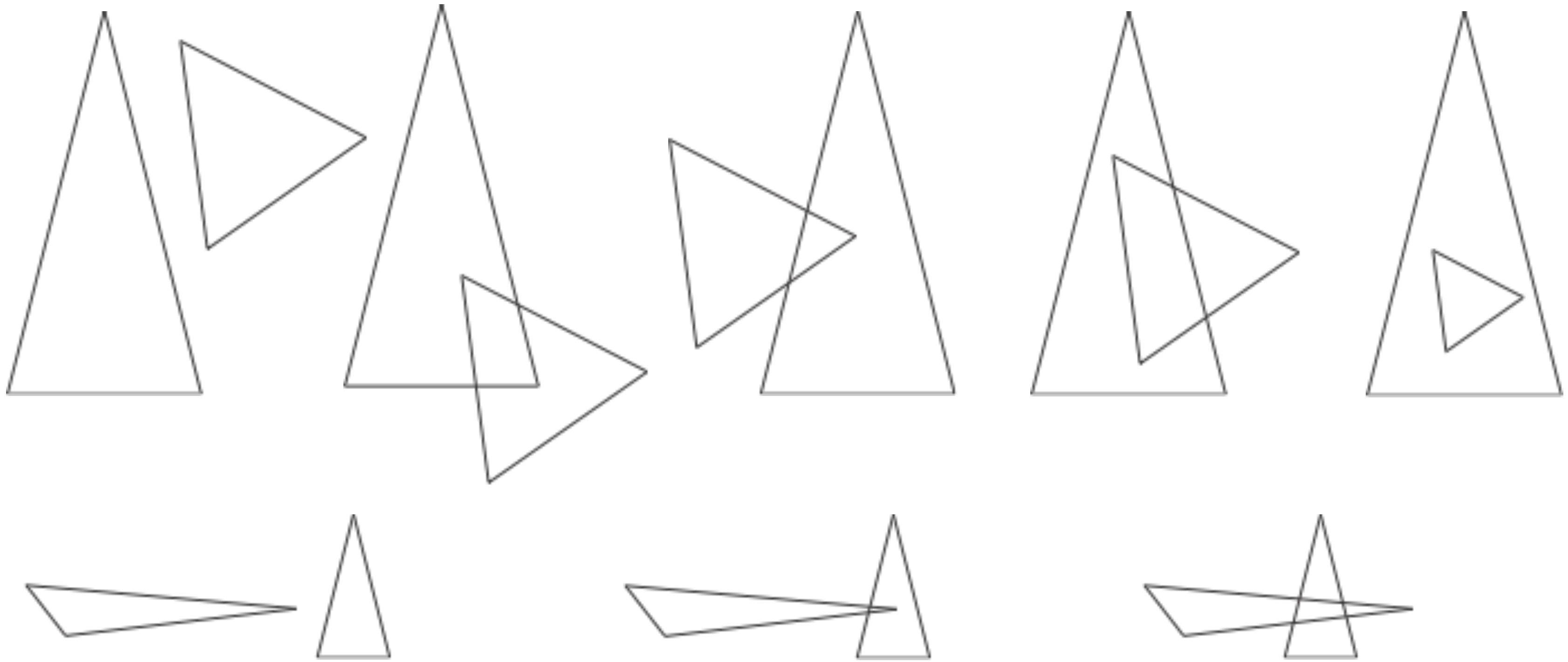
- Check for edge intersections
  - segment-segment intersection
    - line-line test & check parameters s,t

But . . . .

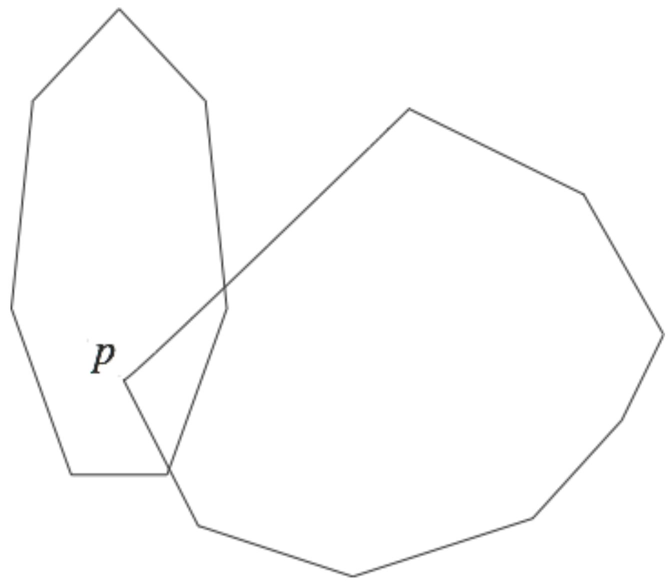


- Fails when one *inside* the other
  - Not actually a problem
  - If time steps are small enough

Half-Plane Method



- If vertex of A is inside B, intersection
  - assuming small time steps ...

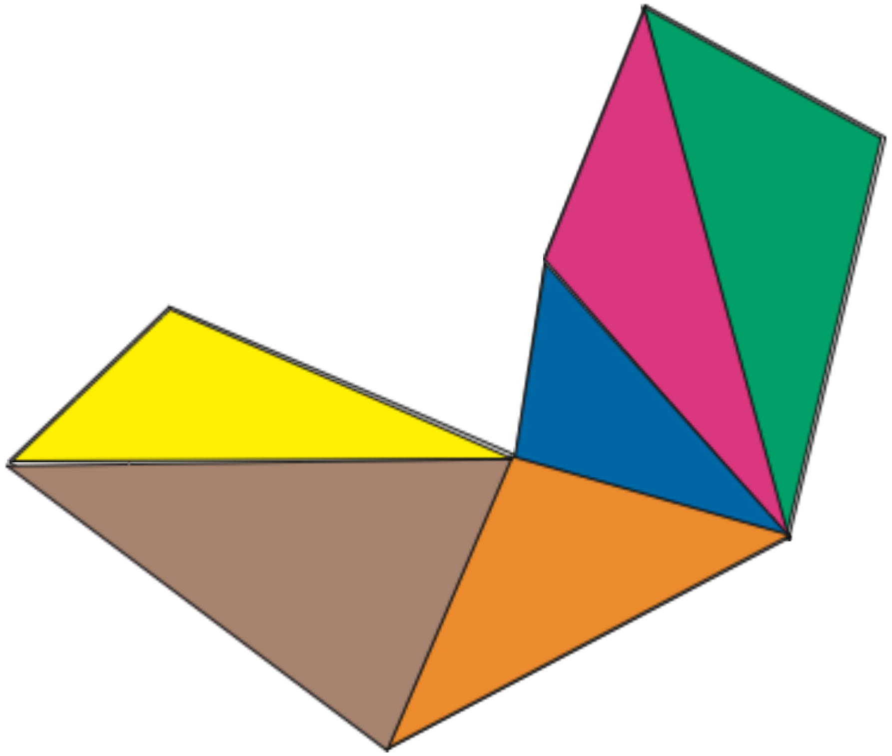


Convex Polygons

- Special case
  - Half-plane test still works
  - Points are consistently inside edges

Concave Polygons

- There is at least one corner with angle < π
- Cut it off, then iterate

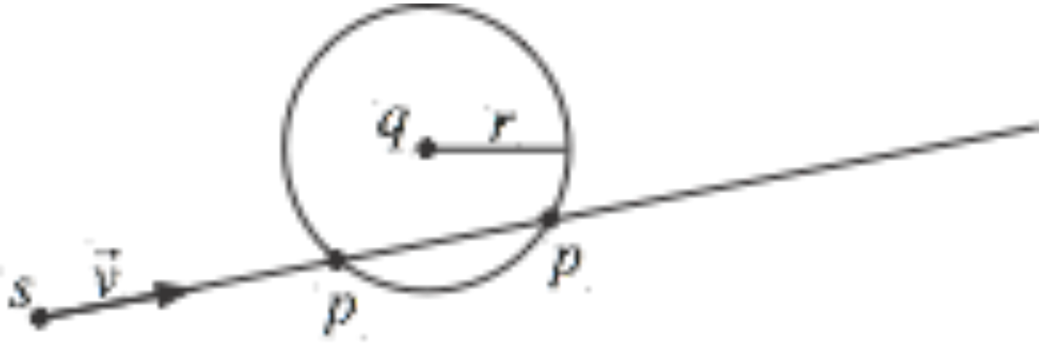


2.5D Intersections

- Line (Ray) - Sphere
- Line – Plane
- Line – Triangle
- Plane - Plane
- Triangle - Triangle

Line-Sphere Intersection

- Given a sphere $Sphere(q,r)$
- And a line $\overleftrightarrow{l} = s + \vec{v}t$
- Find point *p* at intersection
  - i.e. find *t*



UNIVERSITY OF LEEDS

Step 1

We know that:

$$p = s + \vec{v}t$$

and that:

$$(p - q)\bullet(p - q) = r^2$$

So we plug one into the other and get:

$$(s + \vec{v}t - q)\bullet(s + \vec{v}t - q) = r^2$$

We will simplify this by letting:

$$\vec{u} = s - q$$

And we get:

$$(\vec{u} + \vec{v}t)\bullet(\vec{u} + \vec{v}t) = r^2$$

$$(\vec{u} + \vec{v}t)\bullet(\vec{u} + \vec{v}t) = r^2$$

$$\vec{u}\bullet\vec{u} + 2\vec{u}\bullet\vec{v}t + \vec{v}\bullet\vec{v}t^2 = r^2$$

$$(\vec{v}\bullet\vec{v})t^2 + (2\vec{u}\bullet\vec{v})t + (\vec{u}\bullet\vec{u} - r^2) = 0$$

But this is a quadratic equation, so we solve:

$$A = \vec{v}\bullet\vec{v}$$

$$B = 2\vec{u}\bullet\vec{v}$$

$$C = \vec{u}\bullet\vec{u} - r^2$$

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

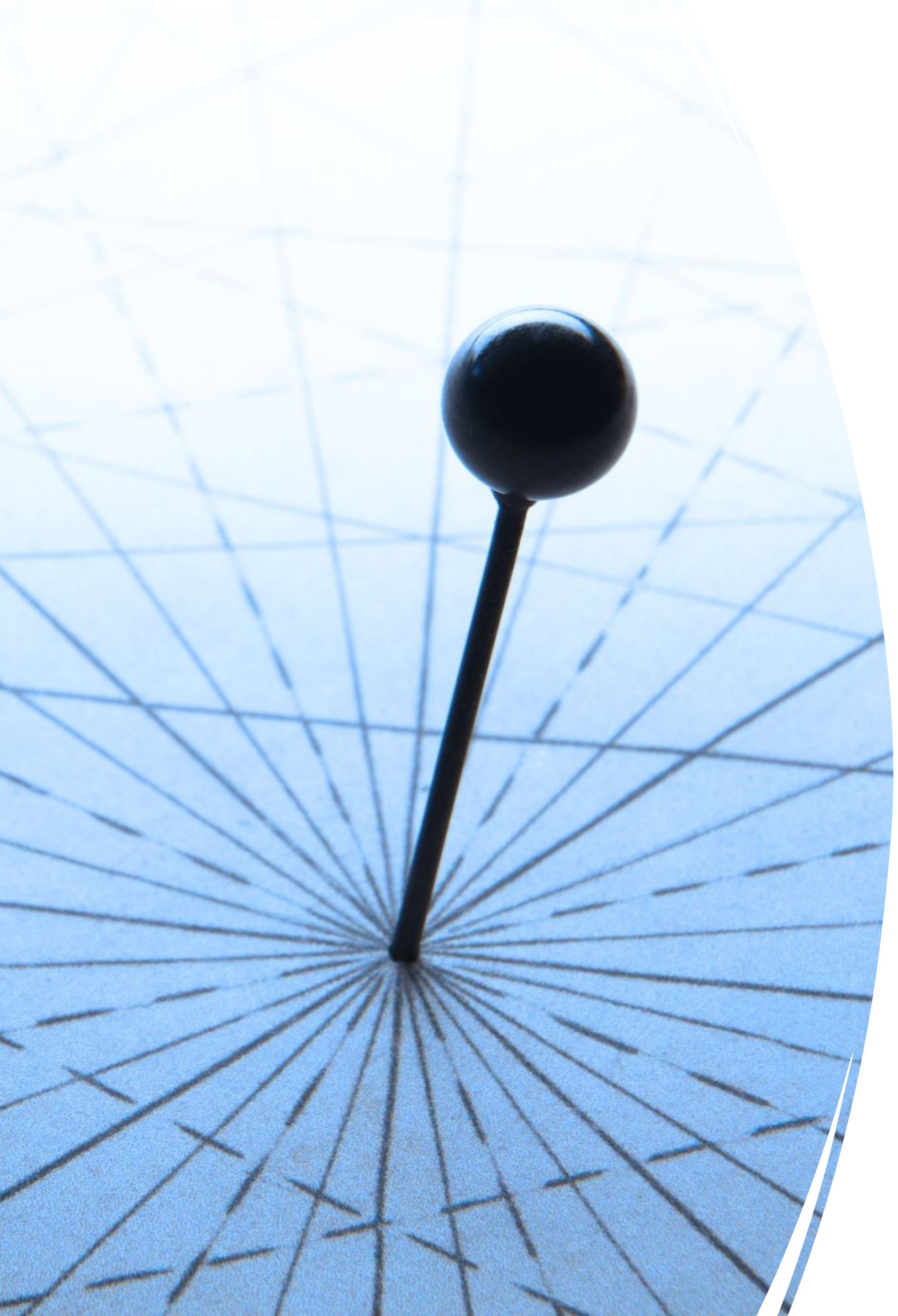
Step 2

Half-Space Test

- Generalised form of half-plane test

$$\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_x & q_x & r_x & s_x \\ p_y & q_y & r_y & s_y \\ p_z & q_z & r_z & s_z \end{vmatrix} = \begin{cases} - & s \text{ is to } \textit{left} \text{ of plane } pqr \\ 0 & \textit{on} \text{ line} \\ + & s \text{ is to } \textit{right} \text{ of plane } pqr \end{cases}$$

- Doesn't give you distance directly
- Because it uses an unnormalised normal

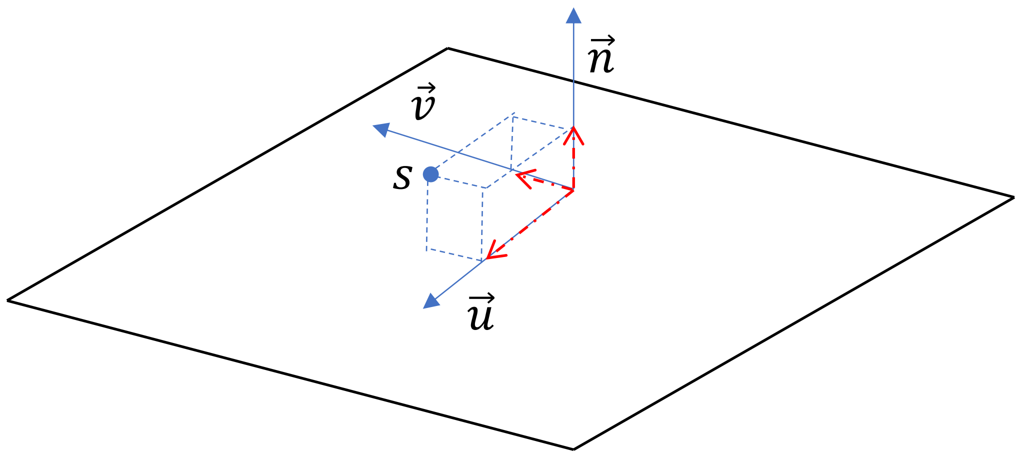


Planar Coordinate System

- Given plane defined by point p, vectors $\vec{u}, \vec{v}$
- Normalise $\vec{u}$
- Compute $\vec{n} = \vec{u} \times \vec{v}$ and normalise
- Compute $\vec{w} = \vec{n} \times \vec{u}$ and normalize
- $\vec{u}, \vec{w}, \vec{n}$ are now an orthonormal basis
- p is the origin with respect to the plane
- So we can convert other coords onto it

Converting to Planar CS

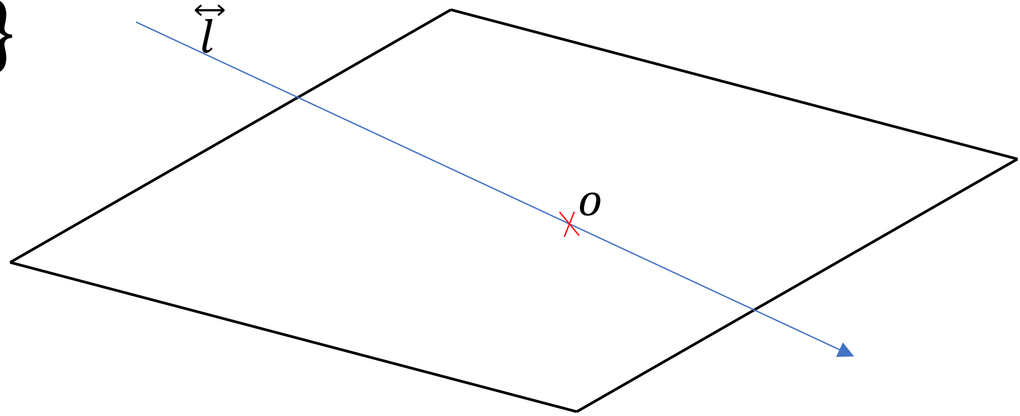
- Assume you have a plane $\Pi = \{p, \vec{u}, \vec{w}, \vec{n}\}$
- And a point s
- Let $\vec{s} = s - p$
- Then project $\vec{s}$ onto $\vec{u}, \vec{w}, \vec{n}$
  - $\vec{s}' = (\vec{s} \cdot \vec{u}, \vec{s} \cdot \vec{w}, \vec{s} \cdot \vec{n})$ gives s' in the PCS
  - $s' = \begin{bmatrix} \vec{u} \\ \vec{w} \\ \vec{n} \end{bmatrix} \vec{s}$ (i.e. $\vec{u}, \vec{w}, \vec{n}$ are rows of matrix R_Π)



UNIVERSITY OF LEEDS

Line – Plane Intersection

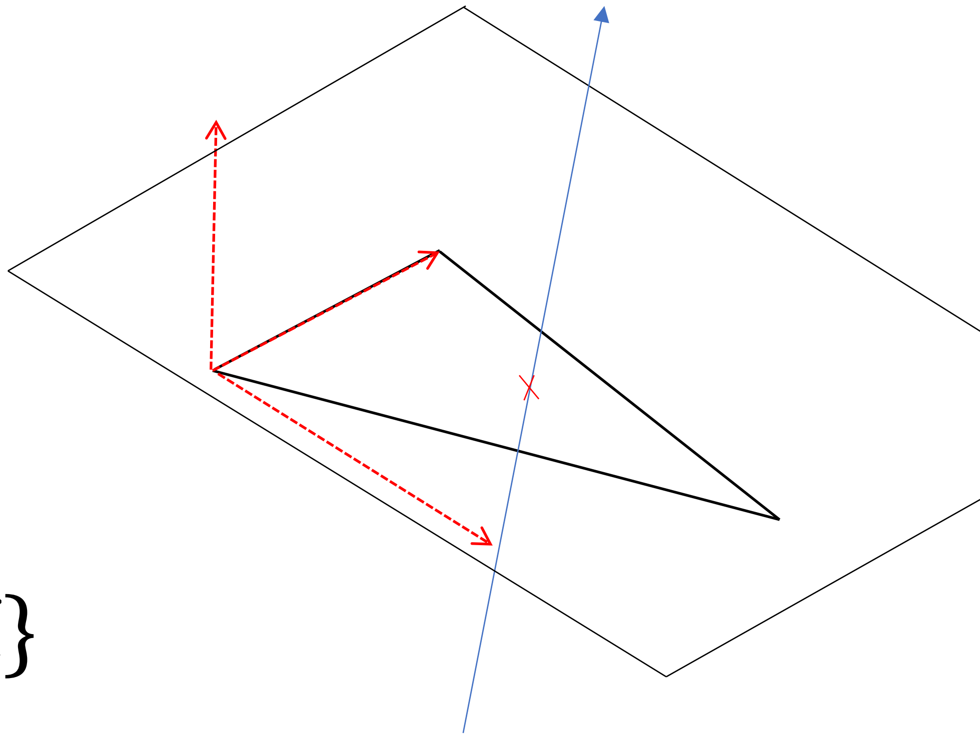
- Assume you have a plane $\Pi = \{p, \vec{u}, \vec{w}, \vec{n}\}$
- And a line $\overleftrightarrow{l} = \{s, \vec{l}\}$
- Find $s' = R_\Pi(p - s) = (s_u, s_w, s_n)$ in PCS
- Find $\vec{l}' = R_\Pi \vec{l} = \ (l_u, l_w, l_n) = (\vec{l} \cdot \vec{u}, \vec{l} \cdot \vec{w}, \vec{l} \cdot \vec{n})$
- Then l_n is the rate the line approaches Π
- And $t = \frac{s_n}{l_n} = \ \frac{(p-s)\cdot\vec{n}}{\vec{l}\cdot\vec{n}}$ is when they intersect
  - At point $o = s + \vec{l}t$



UNIVERSITY OF LEEDS

Line – Triangle Intersection

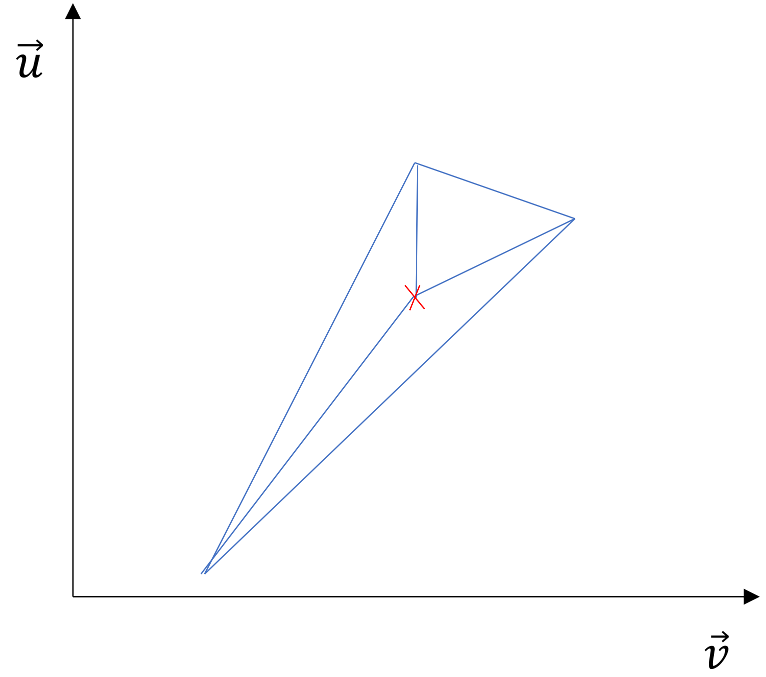
- Assume you have a triangle (p, q, r)
- And a line $\overleftrightarrow{l} = \{s, \vec{l}\}$
- Construct vectors $\vec{u} = q - p, \vec{v} = r - p$ on
- Use these to construct plane $\Pi = \{p, \vec{u}, \vec{w}, \vec{n}\}$
- Find point $o = s + \vec{l}\,\frac{(s-p)\cdot\vec{n}}{\vec{l}\cdot\vec{n}}$
- Convert to PCS with $o_\Pi = R_\Pi(o - p)$
- Do the same with p_Π, q_Π, r_Π



UNIVERSITY OF LEEDS

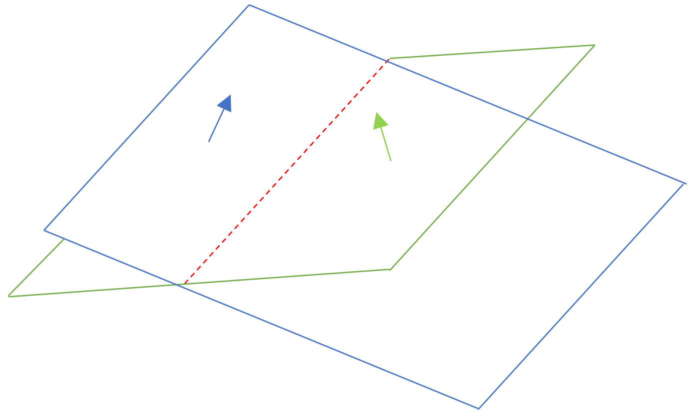
Testing on the Plane

- Notice that $o_n = p_n = q_n = r_n = 0$
- So we can just look at the u,w coordinates
- And now we can do barycentric interpolation
- And test whether the point o_n is in $\Delta\, pqr$
- There are obviously many optimisations
  - Which we may look at next term



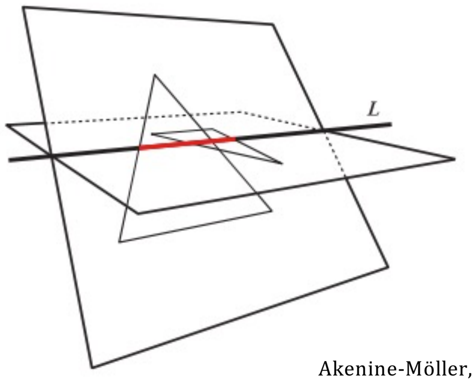
Plane-Plane Intersections

- Given planes $\Pi_1 = \{p_1, \vec{u}_1, \vec{w}_1, \vec{n}_1\}, \Pi_2 = \{p_2, \vec{u}_2, \vec{w}_2, \vec{n}_2\}$
- Then $\vec{v} = \vec{n}_1 \times \vec{n}_2$ is the vector of the shared line
- and $\vec{l} = \vec{n}_1 \times \vec{v}$ is on Π_1 perpendicular to it
- and line $\overleftrightarrow{l} = \{p_1, \vec{l}\}$ will intersect with Π_2
- Then $o = p_1 + \vec{l}\,\frac{(p_1 - p_2)\cdot\vec{n}_2}{\vec{l}\cdot\vec{n}_2}$ is on the shared line
- I.e. the line is $o + \vec{v}t$



UNIVERSITY OF LEEDS

Triangle-Triangle Intersections



Akenine-Möller, 3d ed.

- Use the triangles to construct two planes
- Use the plane-plane test to find the line
- Find the intersections with each triangle
- Use the 1D test for intersection!
- Again, there are *many* optimisations

3D Intersections

- Box - Box (AABB)
- Sphere - Sphere
- Tetrahedron - Tetrahedron
- Polyhedron - Polyhedron
- Cylinders & Capsules

3D AAB - AAB

- Axis-Aligned Boxes
- Same as 2D
  - Project to individual axes
  - Then do line segment tests

3D Spheres

- Same as 2D Circles
- or 1D Segments (2d form)

$$\text{Is } \left\| m_1 - m_2 \right\| \le r_1 + r_2?$$

3D Tetrahedra

- Same as 2D Triangles
- Use Half-Space Test
- Test against each face
  - with consistent orientation

3D Polyhedra

- Convex Polyhedra
  - Half-Space Test
- Concave Polyhedra
  - Decompose into Tetrahedra
  - Hard problem

Bounding Primitives

Polyhedra tests are expensive

Especially for large complex objects

Commonest case: no intersection

So use cheap test first:

- Bounding Spheres
- Axis-Aligned Bounding Boxes

Higher Order Surfaces

- The equations get much harder
- And may not have closed-form solutions
- Substitute numerical root-finding
- And use a bounding primitive first
- Or tessellate the surface at high resolution

Images by

- S9: Sirisvisual