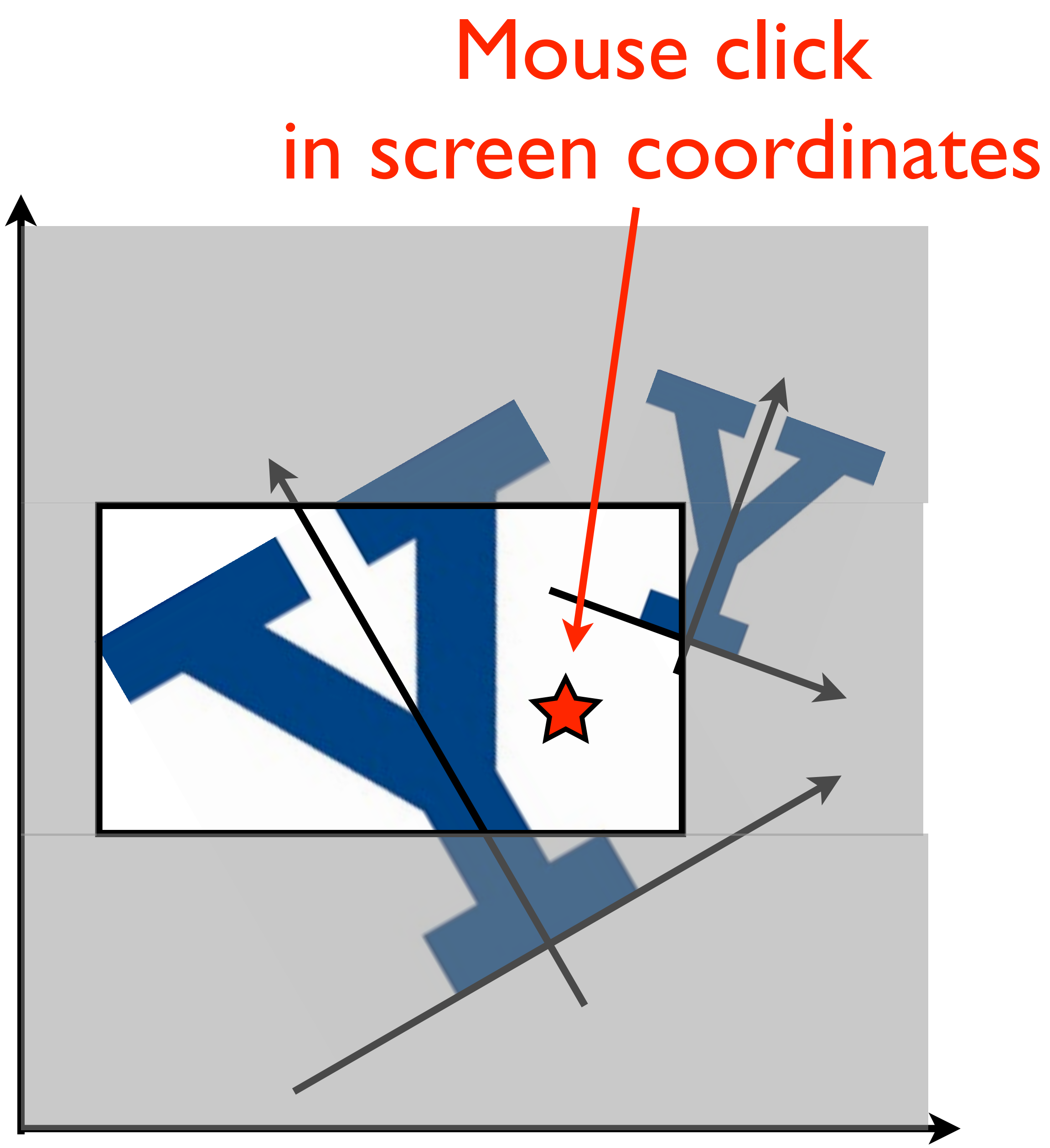
2D Selection Geometry

CS 355: Interactive Graphics and Image Processing

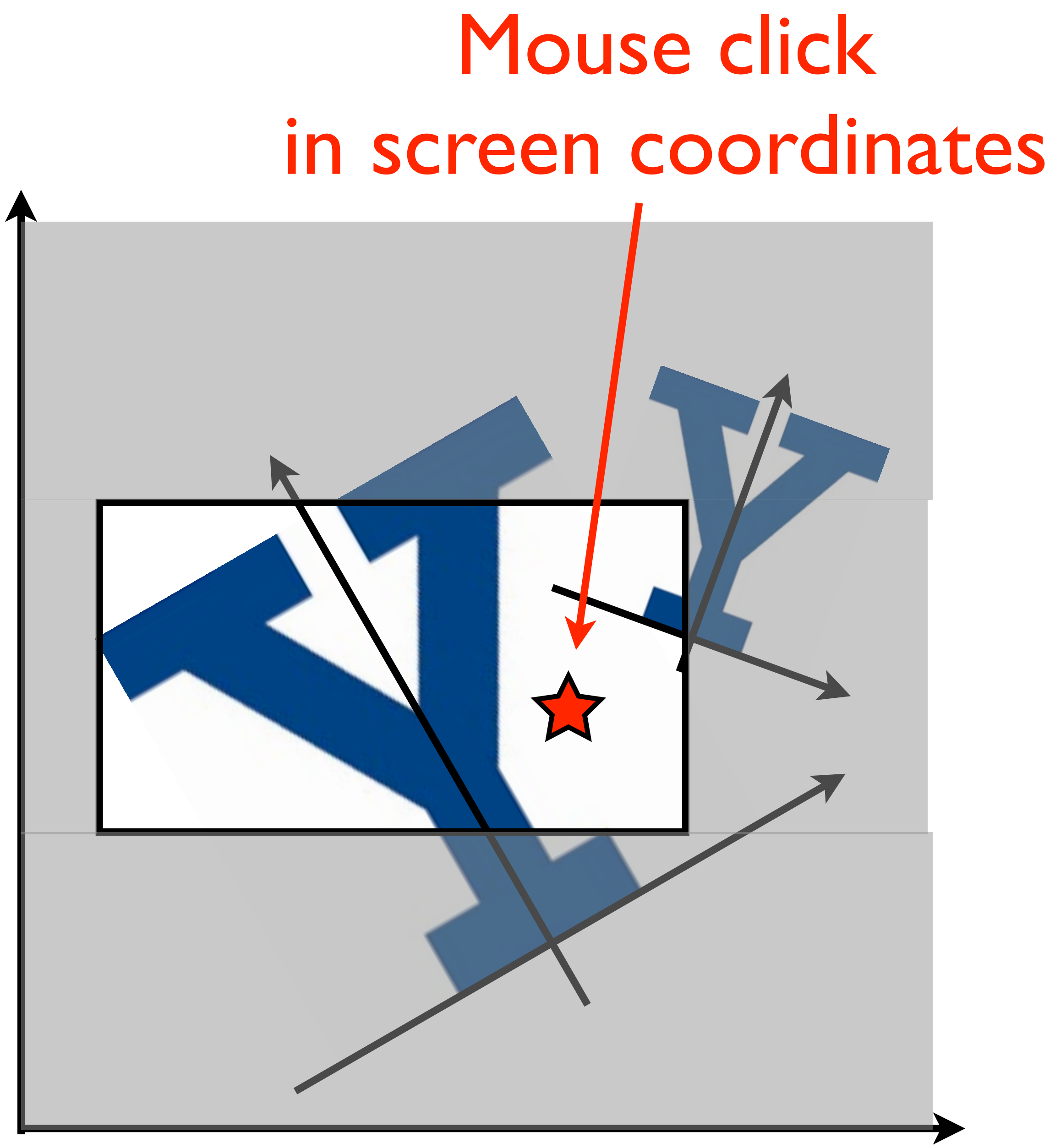
Selection

- User clicks on the screen, determine what was clicked on
- *This is the opposite of drawing*
- Turns into geometric test
  - Point in a shape
  - Point near a shape



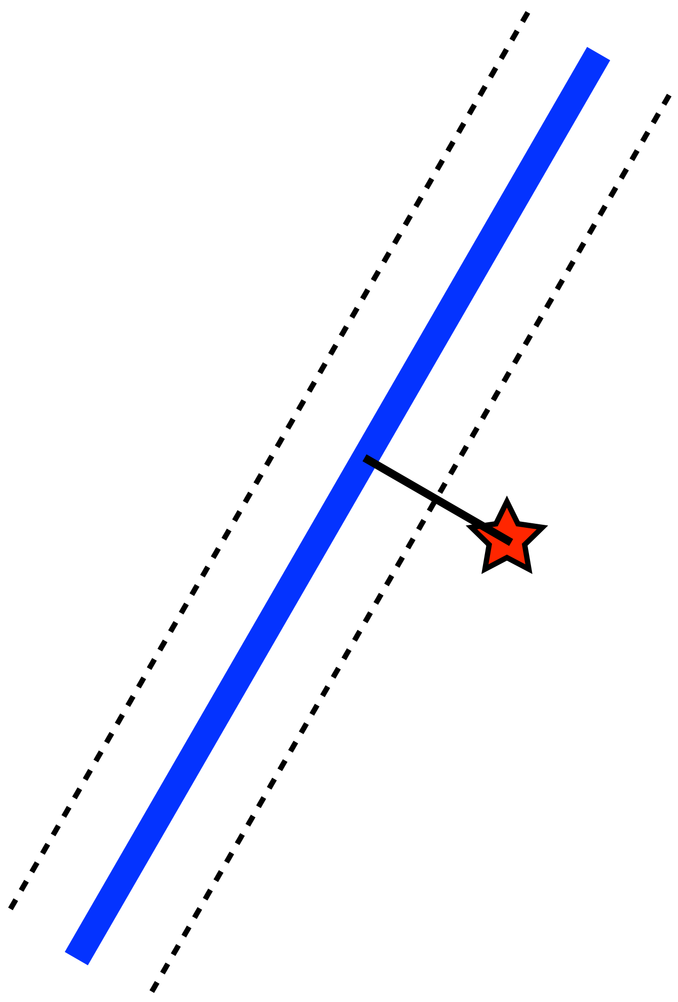
Convert Between Spaces

- Before testing, convert to appropriate space
  - Screen to world (if applicable)
  - World to object (if applicable)
  - Test in object space

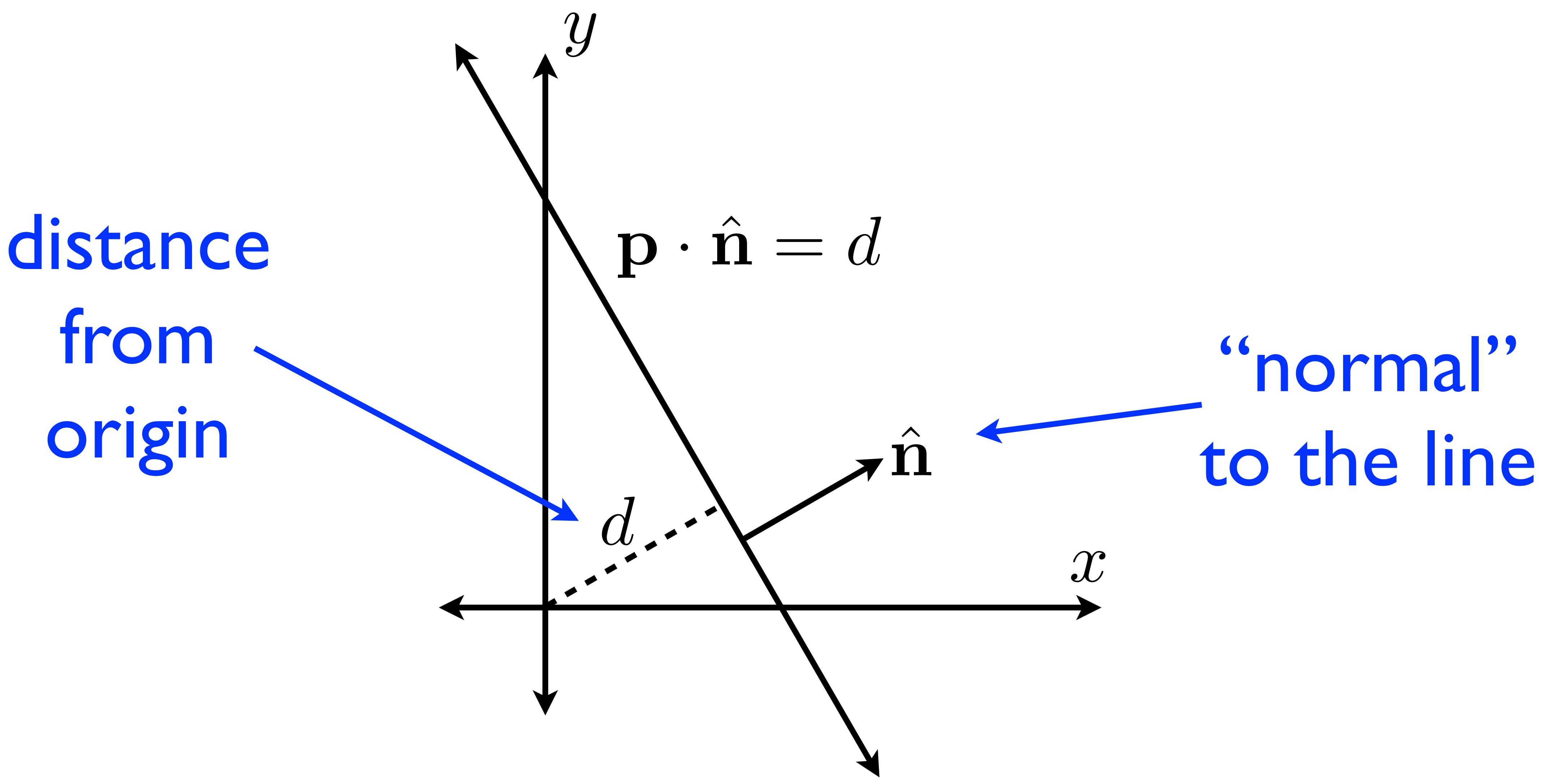


Lines

- Hard to click right on an infinitely thin line
- Test to see if point is *near enough* to the line
- Point-to-line distance (different depending on line representation)



Implicit Representation



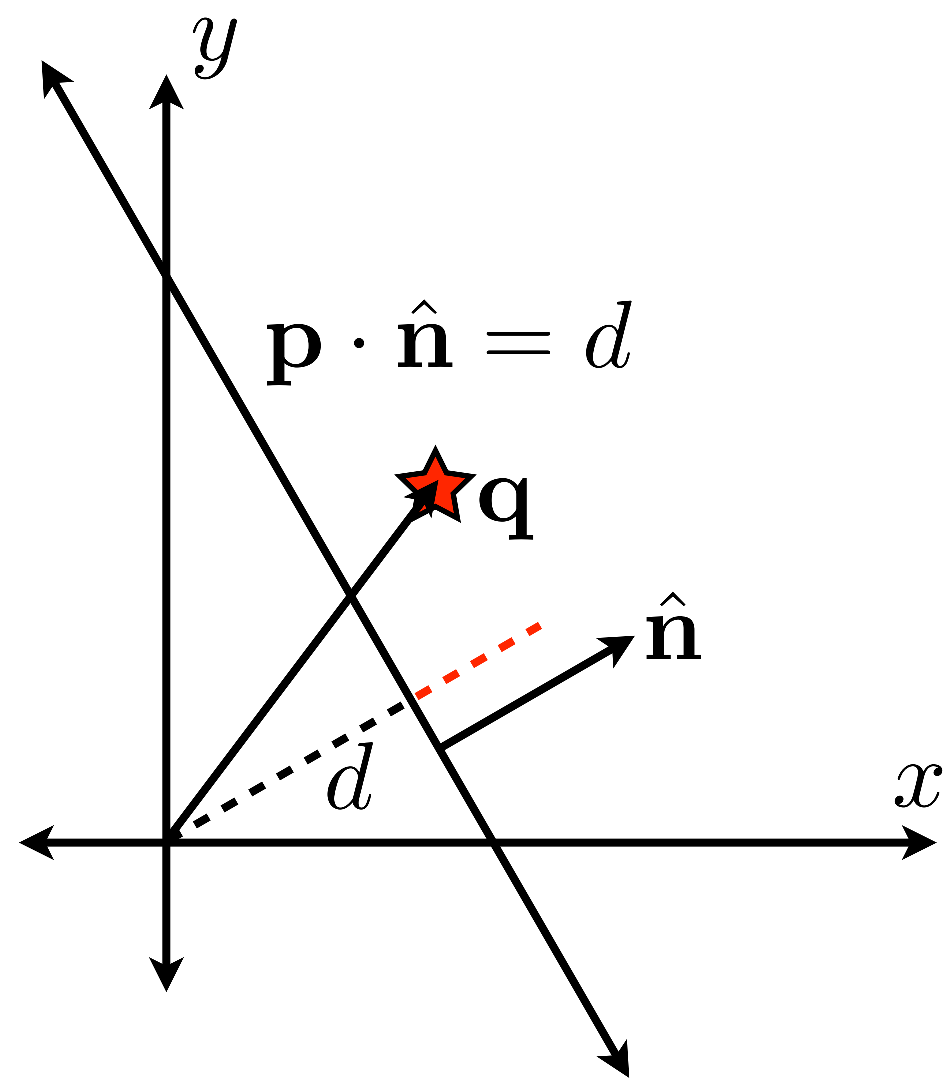
Distance to a Line

- Points **p** on the line L satisfy this constraint:

$$\mathbf{p} \cdot \hat{\mathbf{n}} = d$$

- Distance from point **q** to to the line L:

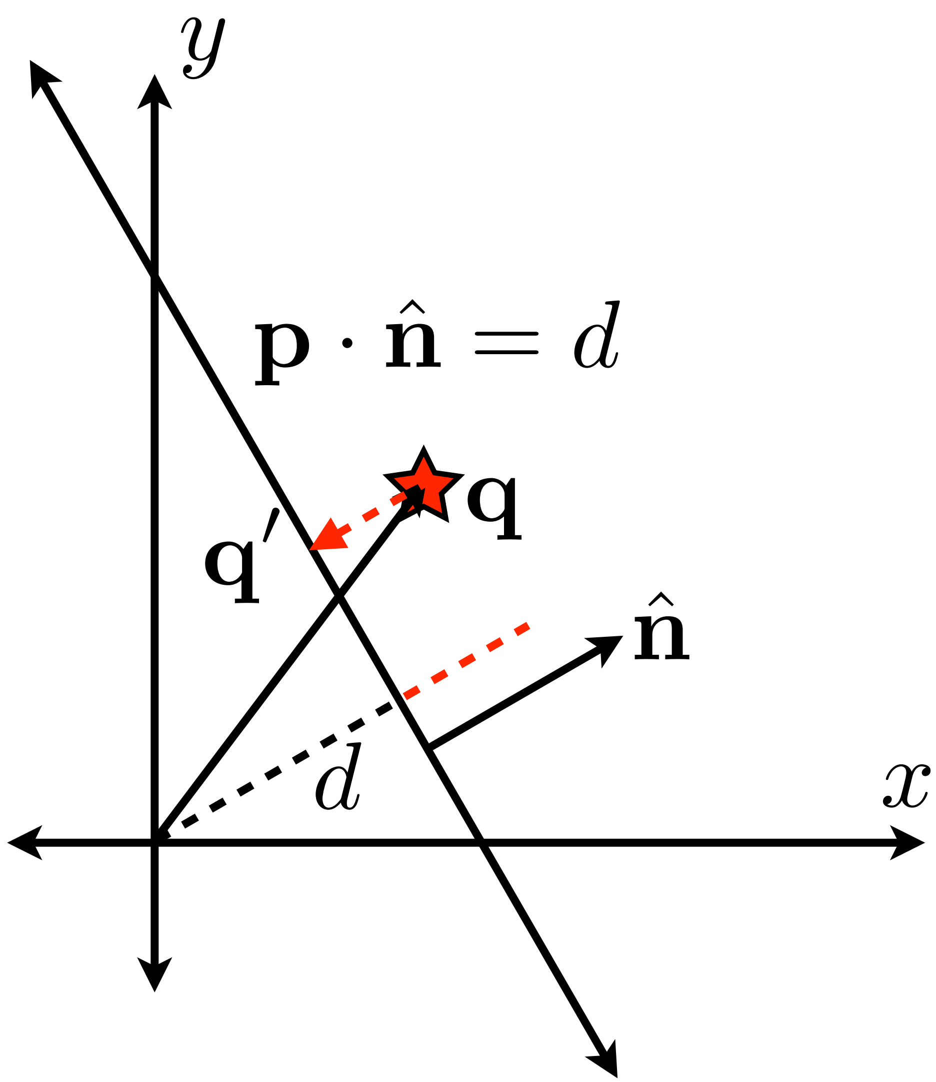
$$|\mathbf{q} \cdot \hat{\mathbf{n}} - d|$$



Closest Point to a Line

- To get the closest point on the line L to point **q**, go back along the normal direction:

$$\mathbf{q}' = \mathbf{q} + (d - \mathbf{q} \cdot \hat{\mathbf{n}})\ \hat{\mathbf{n}}$$



Aside: Lines and Distance

- You've seen before:

$$\underbrace{a}_{\text{normal}}x + \underbrace{b}_{\text{normal}}y + \underbrace{c}_{\text{negative distance}} = 0$$

- Normal and distance:

$$\mathbf{p} \cdot \hat{\mathbf{n}} = d$$

$$\mathbf{p} = \begin{bmatrix} x \\ y \end{bmatrix} \quad \hat{\mathbf{n}} = \begin{bmatrix} a \\ b \end{bmatrix}$$

$$d = -c$$

This is really what you've seen years ago, just in linear algebra form

Aside: Lines and Distance

- On the line:

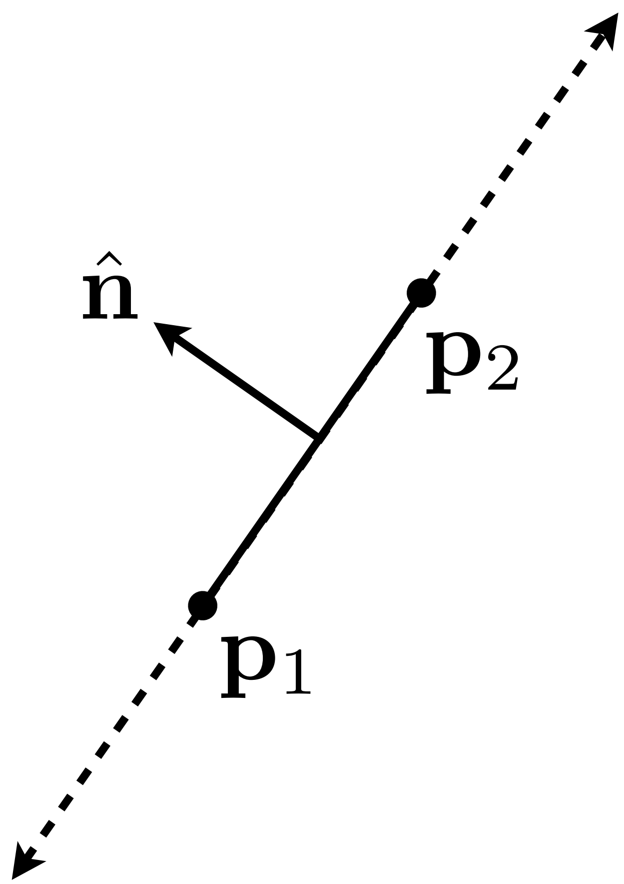
$$ax + by + c = 0$$

- Distance to the line:

$$\text{distance} = |ax + by + c|$$

Calculating Normals

- If given two points defining a line, what is the normal?
  - Vector between points:
    $$\mathbf{p}_2 - \mathbf{p}_1$$
  - Normalized:
    $$\frac{\mathbf{p}_2 - \mathbf{p}_1}{\|\mathbf{p}_2 - \mathbf{p}_1\|}$$
  - Perpendicular:
    $$\hat{\mathbf{n}} = \frac{(\mathbf{p}_2 - \mathbf{p}_1)_\perp}{\|\mathbf{p}_2 - \mathbf{p}_1\|}$$



Actually one of two normals

Perpendicular Vectors

- Vector:

$$\mathbf{v} = \left[\begin{array}{c} x \\ y \end{array} \right]$$

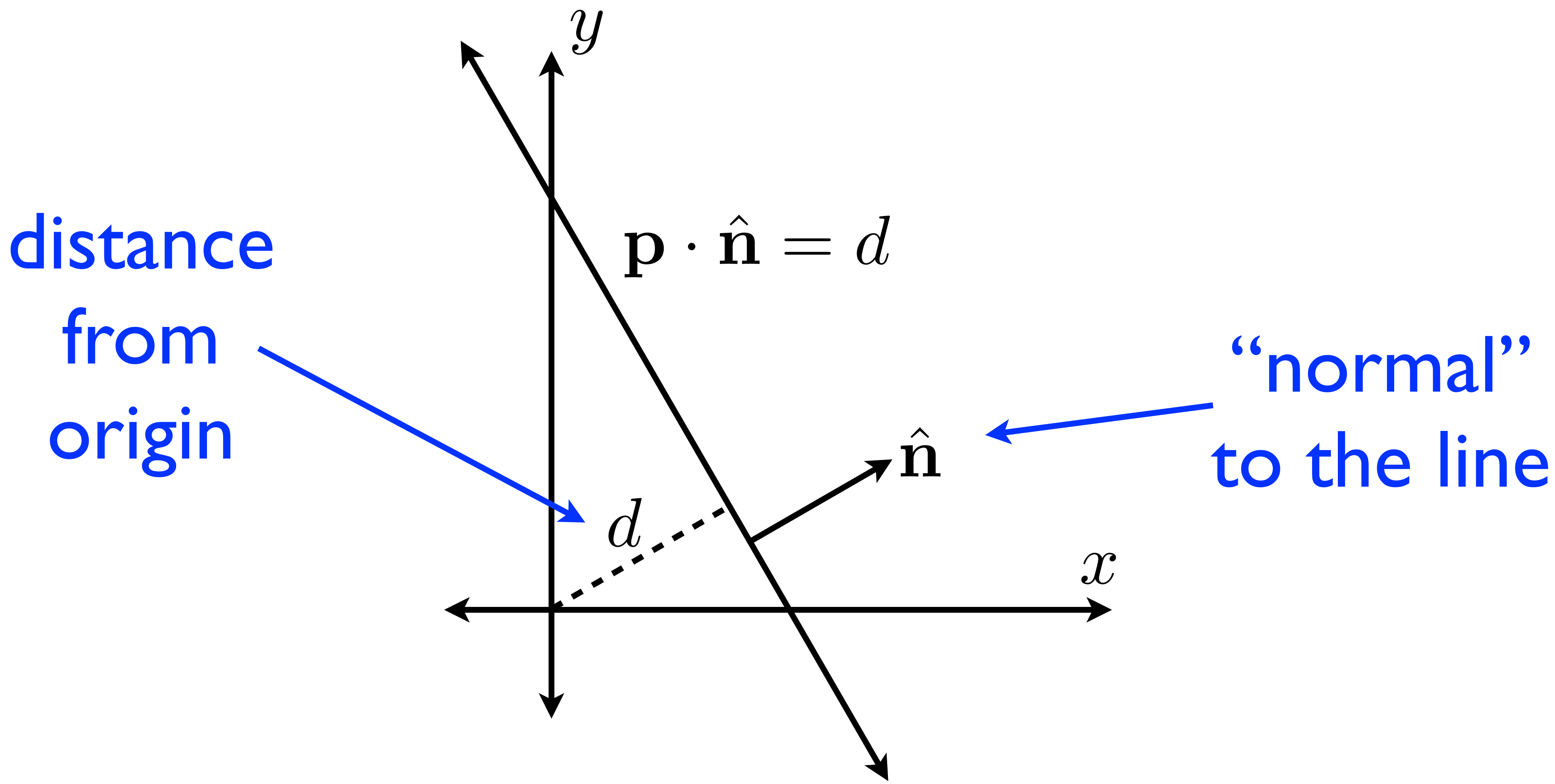
- Its perpendicular:

$$\mathbf{v}_\perp = \left[\begin{array}{c} -y \\ x \end{array} \right]$$

$$\mathbf{v} \cdot \mathbf{v}_\perp = \left[\begin{array}{c} x \\ y \end{array} \right] \cdot \left[\begin{array}{c} -y \\ x \end{array} \right] = 0$$

To get a perpendicular vector, swap x and y and negate one of the two

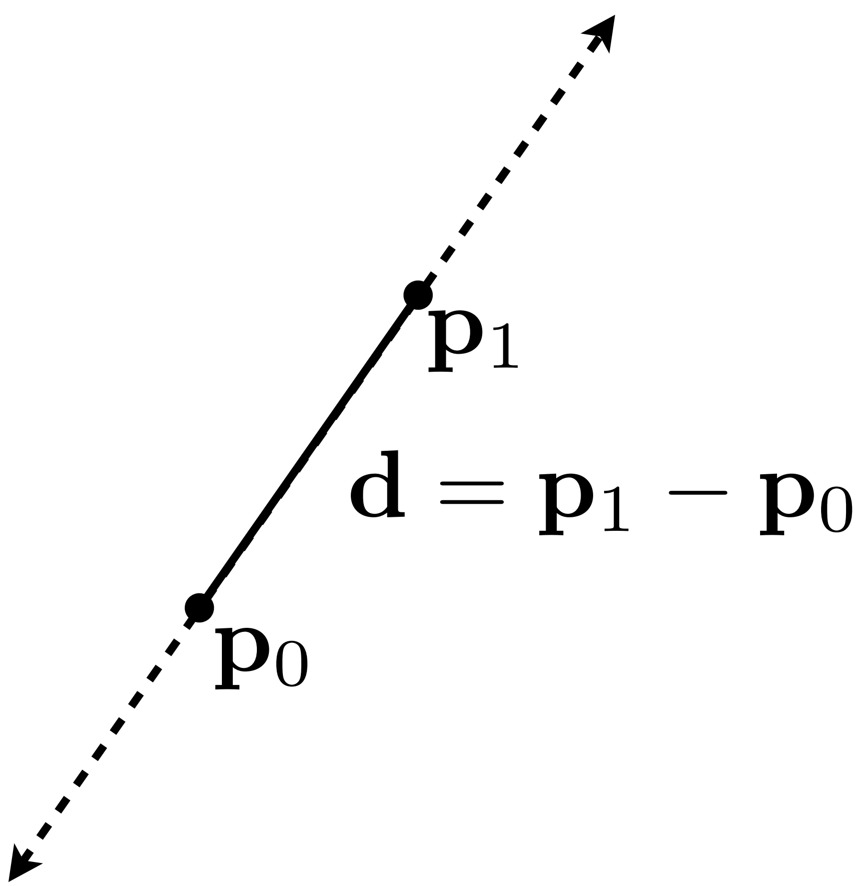
Implicit Representation



$$d = \mathbf{p}_1 \cdot \hat{\mathbf{n}} = \mathbf{p}_2 \cdot \hat{\mathbf{n}}$$

$$\hat{\mathbf{n}} = \frac{(\mathbf{p}_2 - \mathbf{p}_1)_\perp}{\|\mathbf{p}_2 - \mathbf{p}_1\|}$$

Parametric Representation



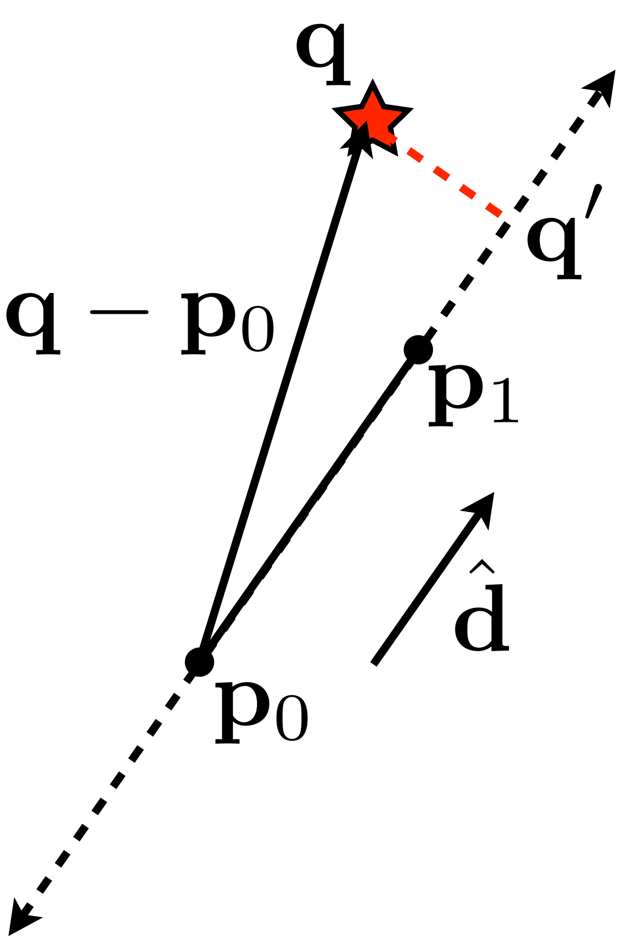
$$\mathbf{p}_0 + t\ \mathbf{d}$$

Line: $-\infty < t < \infty$

Line segment: $0 \leq t \leq 1$

Ray: $0 \leq t < \infty$

Distance to Line



$\mathbf{p}_0 + t\,\hat{\mathbf{d}}$

For rays,

$t \geq 0$

For segments,

$0 \leq t \leq \|\mathbf{p}_1 - \mathbf{p}_0\|$

At what value of t is the line closest to **q**?

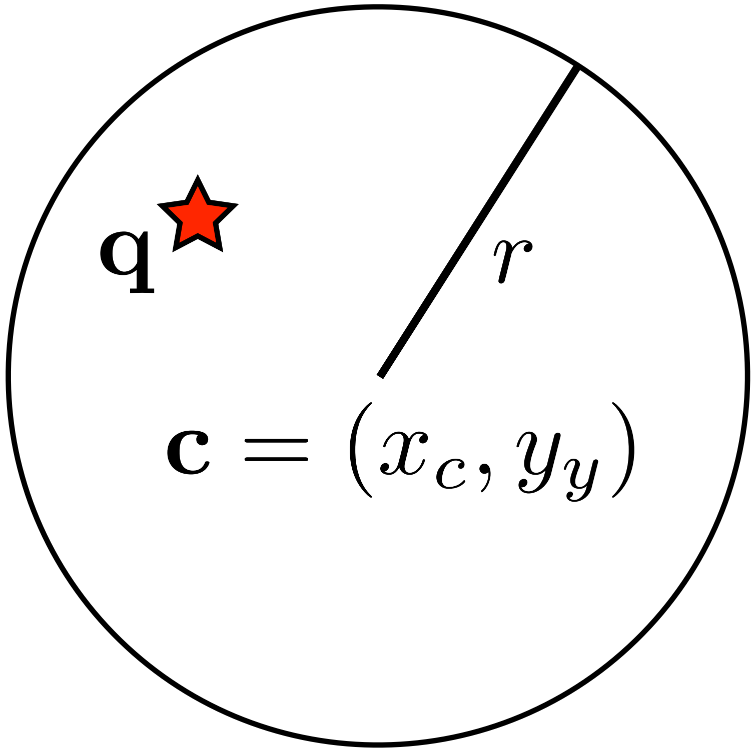
$$\hat{\mathbf{d}} = \frac{\mathbf{p}_1 - \mathbf{p}_0}{\|\mathbf{p}_1 - \mathbf{p}_0\|}$$

$$t = (\mathbf{q} - \mathbf{p}_0) \cdot \hat{\mathbf{d}}$$

$$\mathbf{q}' = \mathbf{p}_0 + ((\mathbf{q} - \mathbf{p}_0) \cdot \hat{\mathbf{d}})\,\hat{\mathbf{d}}$$

$$\text{distance} = \|\mathbf{q} - \mathbf{q}'\|$$

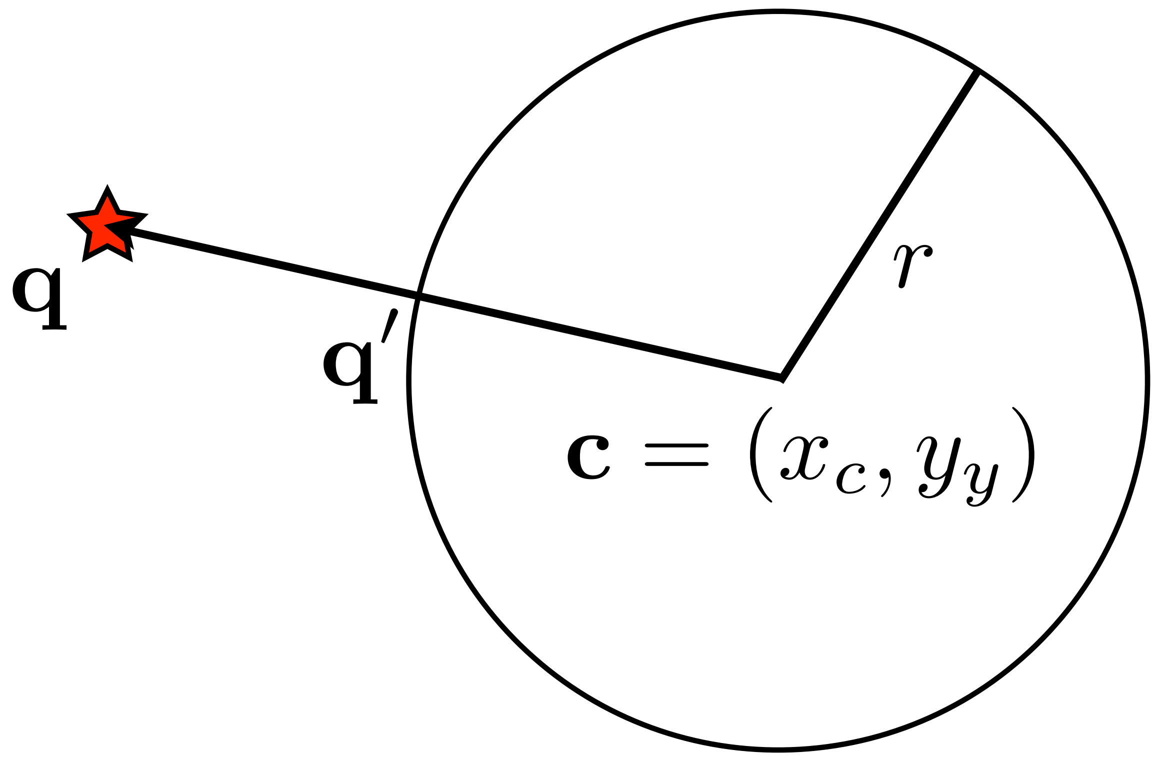
Point in Circle



$$\|\mathbf{q} - \mathbf{c}\| \leq r$$

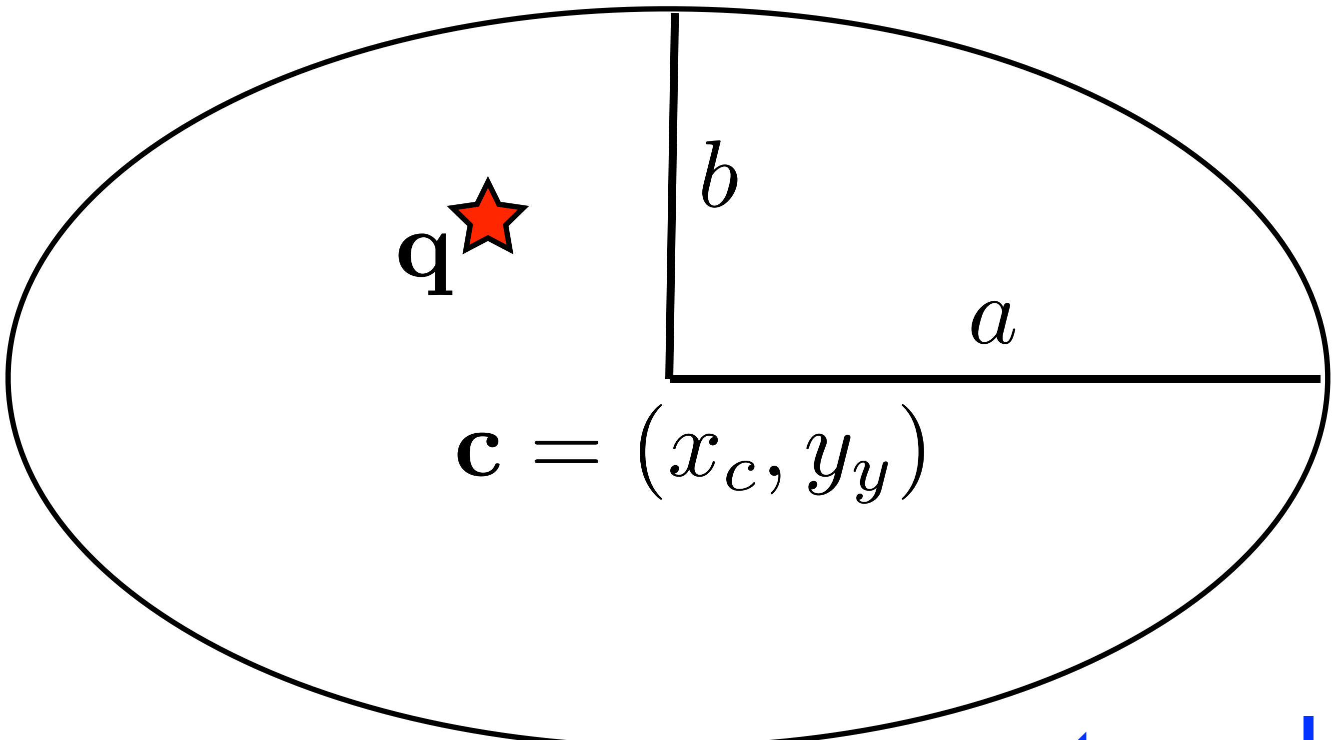
$$(q_x - c_x)^2 + (q_y - c_y)^2 \leq r^2$$

Closest Point on Circle



$$\mathbf{q}' = \mathbf{c} + r \; \frac{\mathbf{q} - \mathbf{c}}{\|\mathbf{q} - \mathbf{c}\|}$$

Point in Ellipse (Oval)

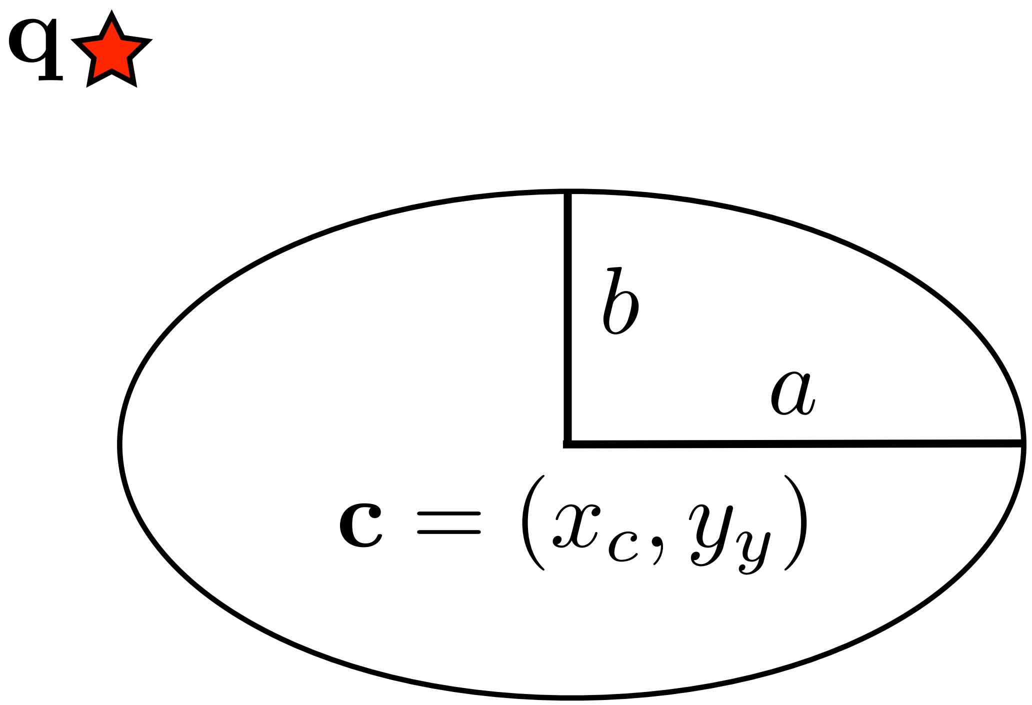


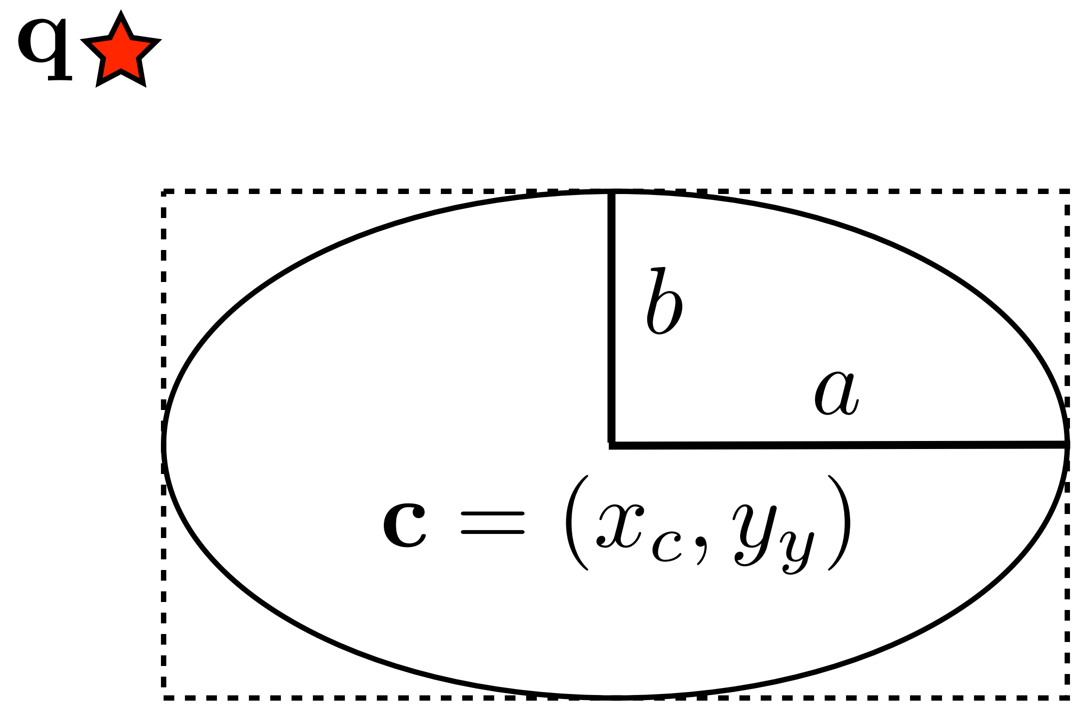
translate!

$$\left(\frac{q_x - c_x}{a}\right)^2 + \left(\frac{q_y - c_y}{b}\right)^2 \leq r^2$$

scale!

Bounding Boxes





Hard test:

$$\left(\frac{q_x - c_x}{a}\right)^2 + \left(\frac{q_y - c_y}{b}\right)^2 \leq r^2$$

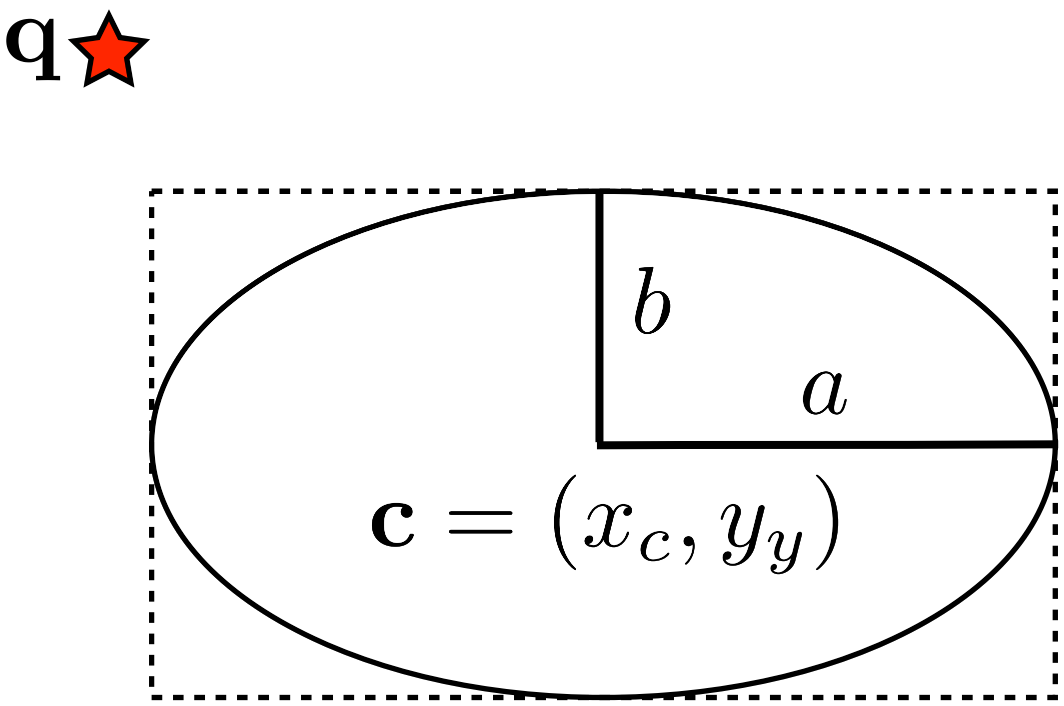
Easy test:

$$|q_x - c_x| \leq a$$

$$|q_y - c_y| \leq b$$

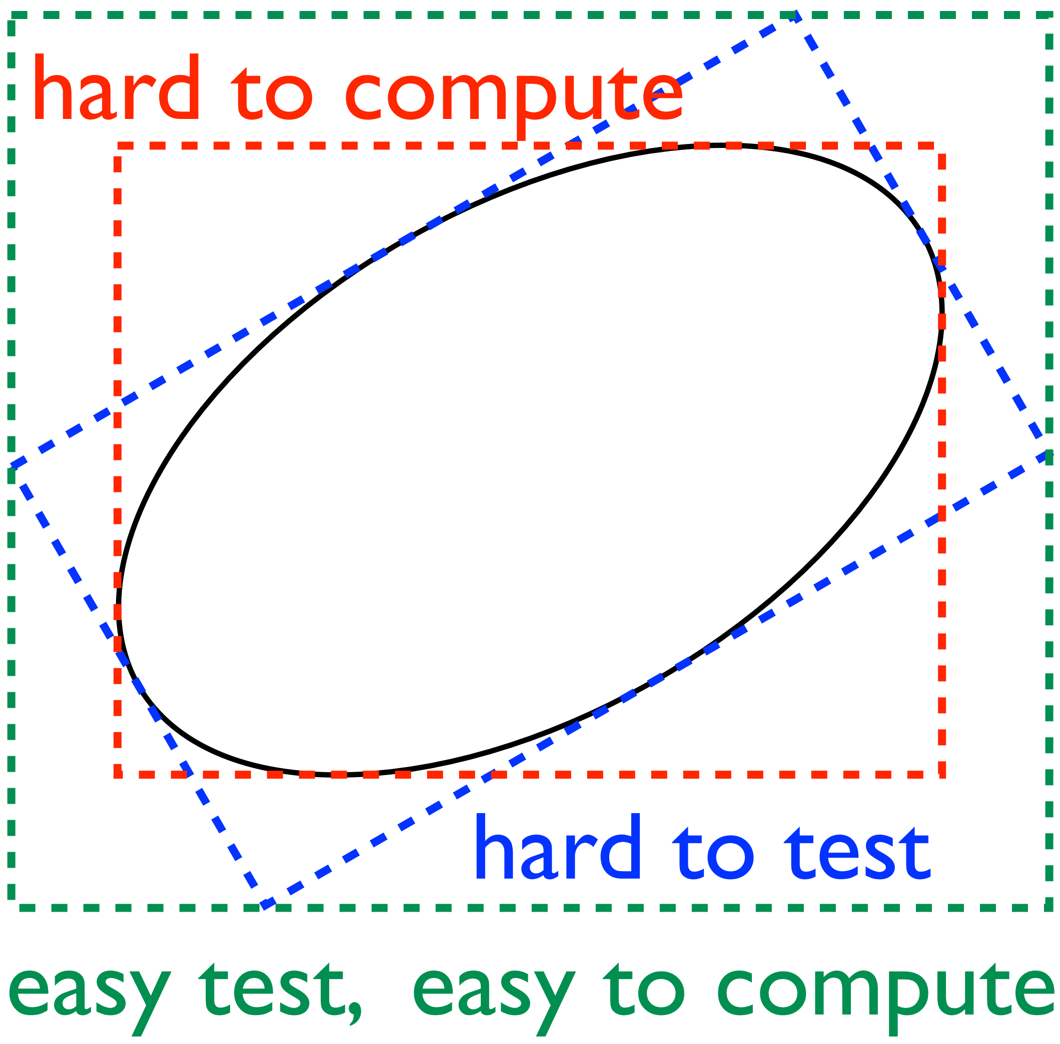
Bounding Boxes

- Idea: use bounding box tests as a "quick reject"
- If passes, then spend time on more complex tests
- The more complex the test, the bigger the win



Bounding Boxes

- Remember: to test, first convert to object space
  - Translate
  - Rotate
  - Scale (possibly)
- Use bounding box test first, but...



AABBs

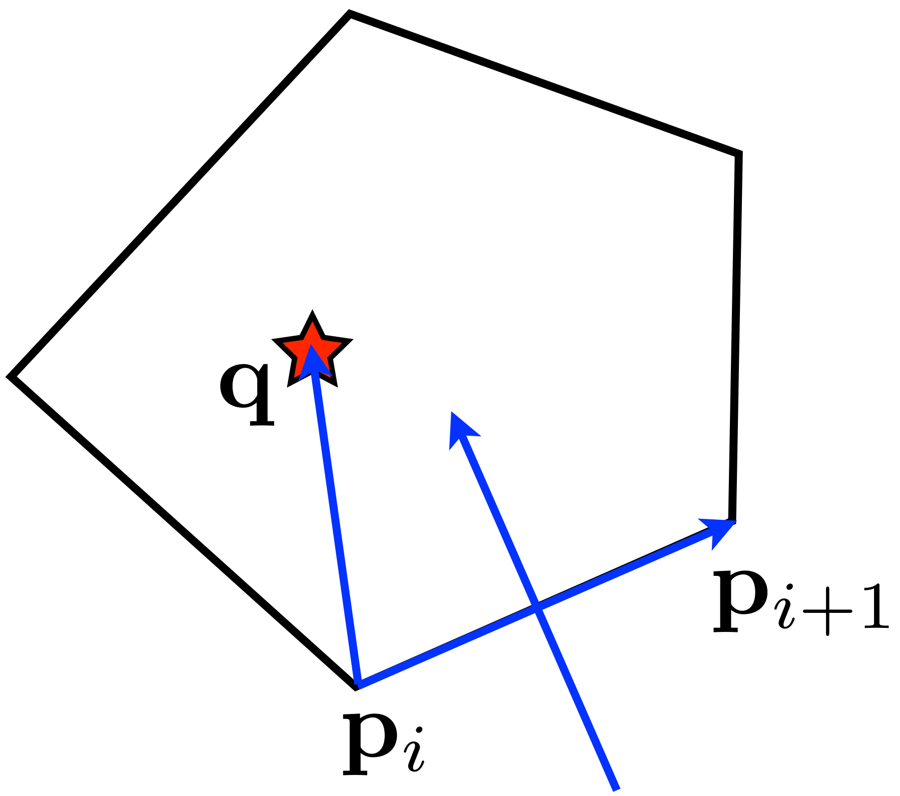
- Fastest quick-reject tests use AABBs: *axis-aligned bounding boxes*
- Don’t have to be tight—can be loose if easier to calculate
- Tighter is more efficient at rejecting, but a little loose is usually close enough

Convex Polygons

- In 2D: for all edges, the point is on the same side of the edge
- Walk around the polygon (in order) and test

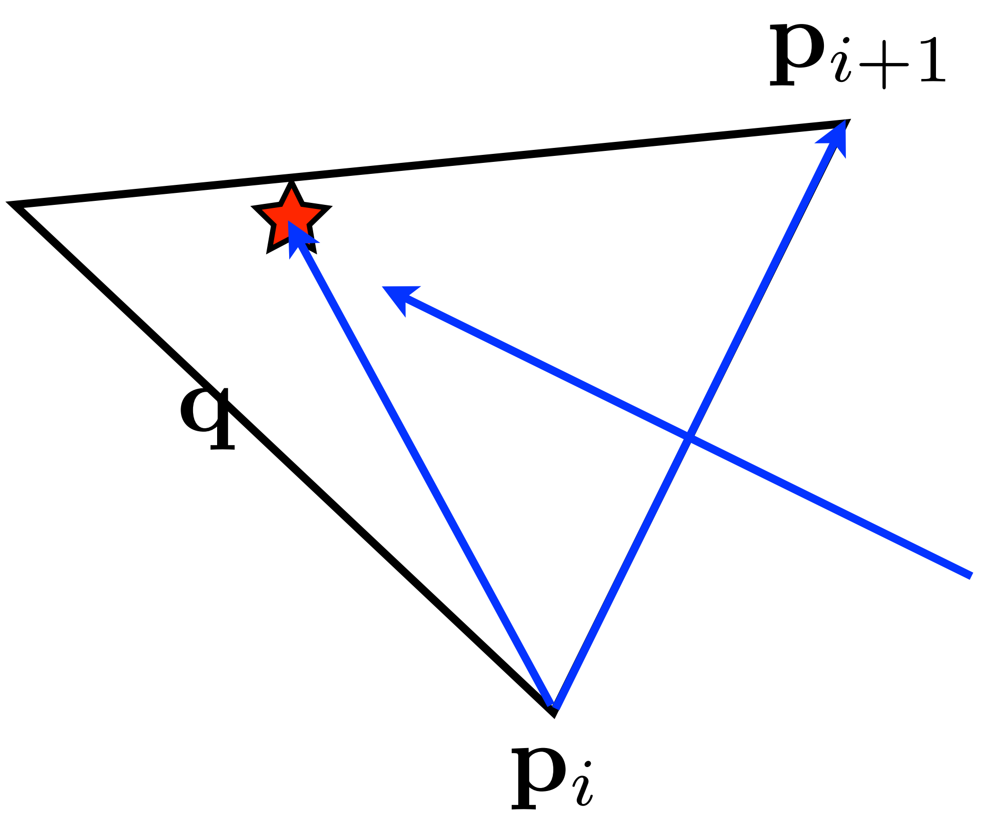
$$(\mathbf{q} - \mathbf{p}_i) \cdot (\mathbf{p}_{i+1} - \mathbf{p}_i)_\perp > 0$$

- just be consistent with ordering, perpendiculars



Triangles

- In 2D: for all edges, the point is on the same side of the edge
- Walk around the triangle (in order) and test



$$(\mathbf{q} - \mathbf{p}_0) \cdot (\mathbf{p}_1 - \mathbf{p}_0)_\perp > 0$$
$$(\mathbf{q} - \mathbf{p}_1) \cdot (\mathbf{p}_2 - \mathbf{p}_1)_\perp > 0$$
$$(\mathbf{q} - \mathbf{p}_2) \cdot (\mathbf{p}_0 - \mathbf{p}_2)_\perp > 0$$

(or all negative, if you don't care about order)

Polygon Bounding Boxes

- Bounding boxes for polygons are really easy
- Just min / max tests over all of the vertices' *x* and *y* coordinates

$$\min(x_i)$$

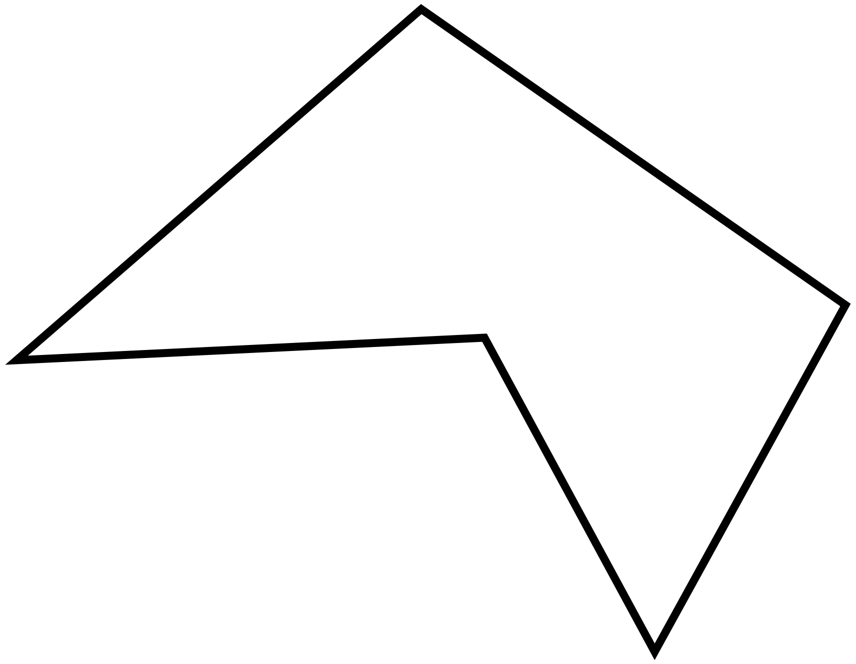
$$\min(y_i)$$

$$\max(x_i)$$

$$\max(y_i)$$

Other Shapes

- Lots of other selections tests for other shapes
- Lots of other intersection tests for various shapes, especially 3D



Coming up...

- Introduction to matrices
- Matrix transformations
  - Forward
  - Inverse