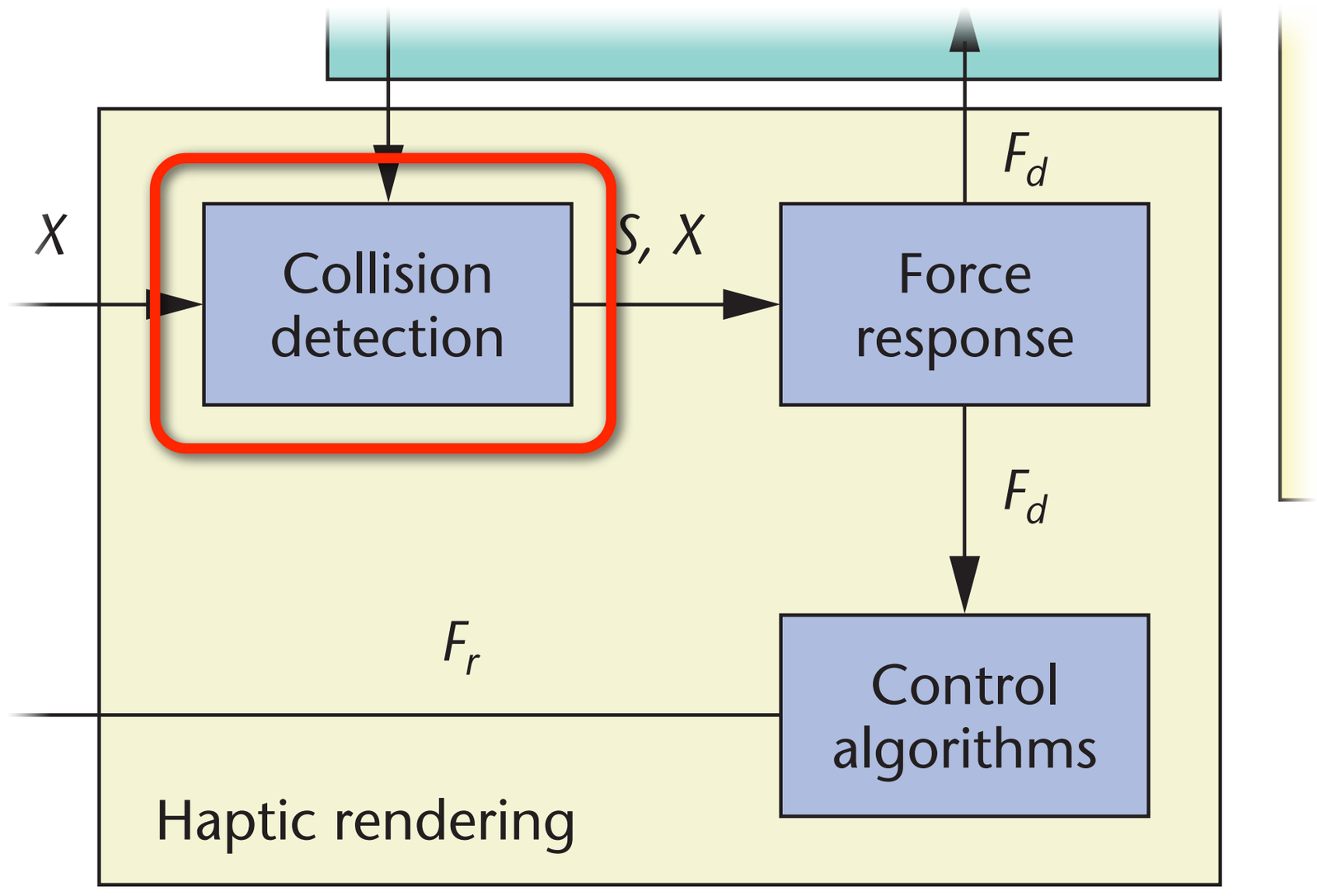
CS277 - Experimental Haptics
Lecture 6

# Collision Detection I

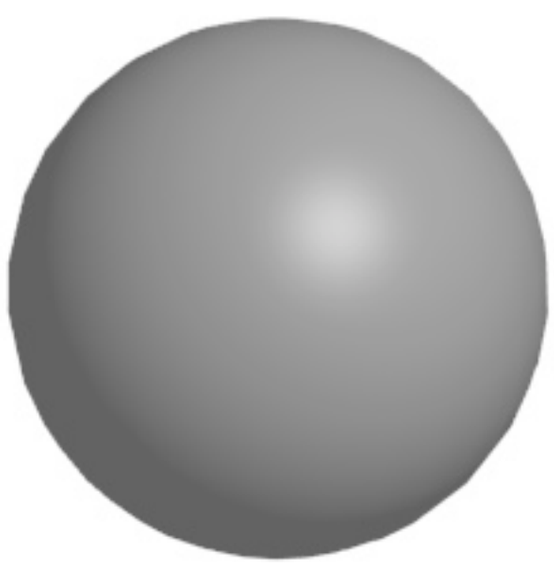
# Motivation

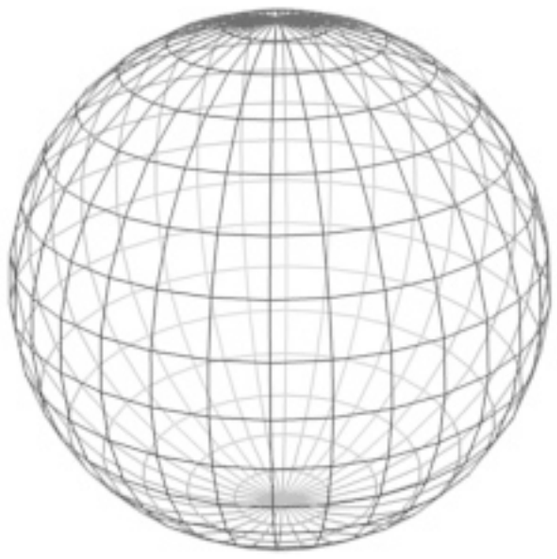

$F_d$
$X$
Collision detection
$S$, $X$
Force response
$F_d$
$F_r$
Control algorithms
Haptic rendering

# Problem Definition

- We seek *efficient* algorithms to answer the following queries:
  - Intersection query (do objects overlap?)
  - Contact manifolds (set of contact points)
  - Penetration depth / intersection volume
  - Separation distance
- Difficulty increases as we move down...

# Geometric Representations



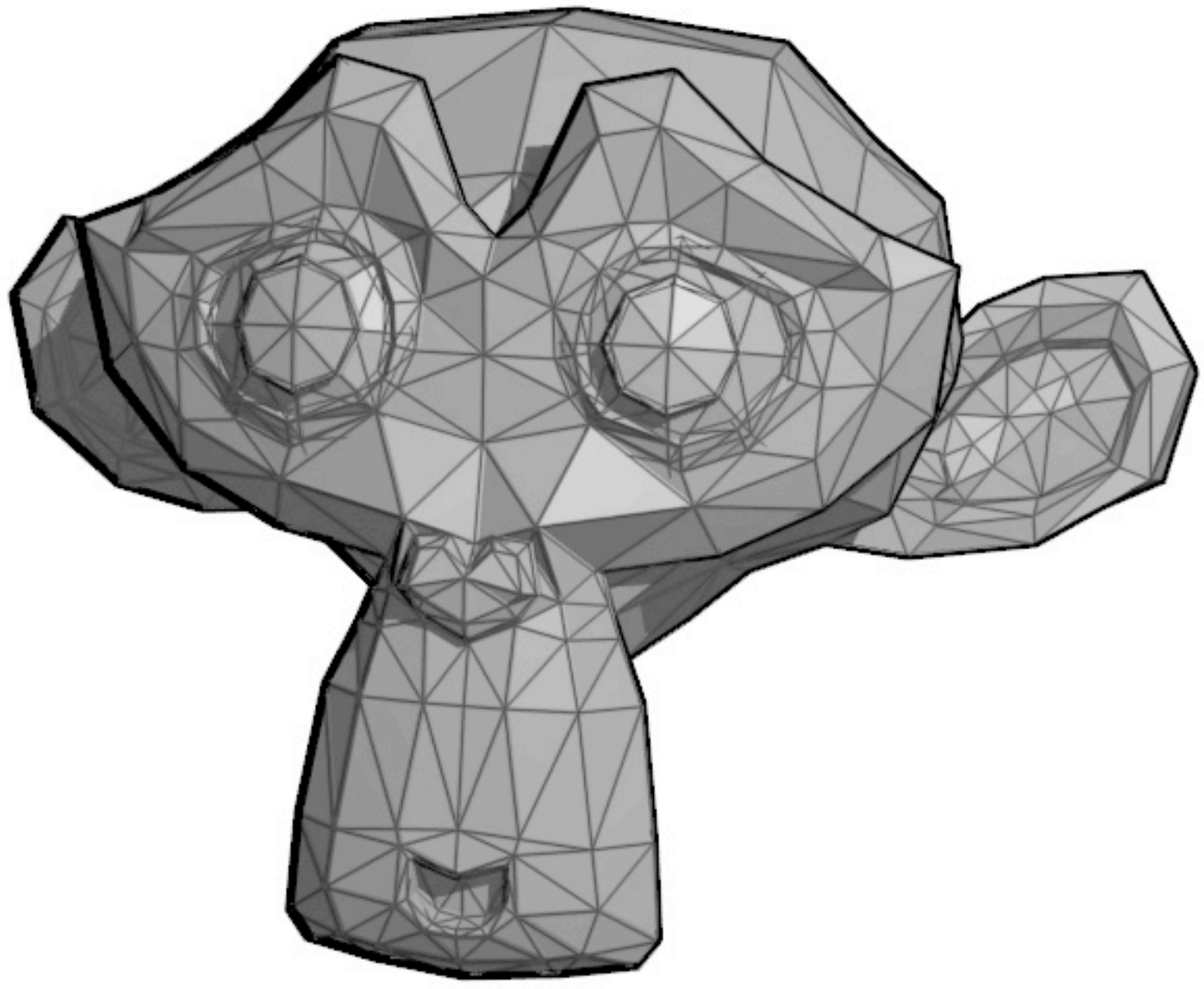


## Many different ways to describe the same object

# Surface Representations

- Implicit surface: $S(x, y, z) = 0$
- Parametric surface: $P(u, v)|u, v \in \mathcal{D}$
- Point-sampled surface (point cloud)
- Polygonal mesh:
  - Triangle mesh
  - Quadrilateral (quad) mesh
- ... any other ones you can think of?

# Triangle Meshes



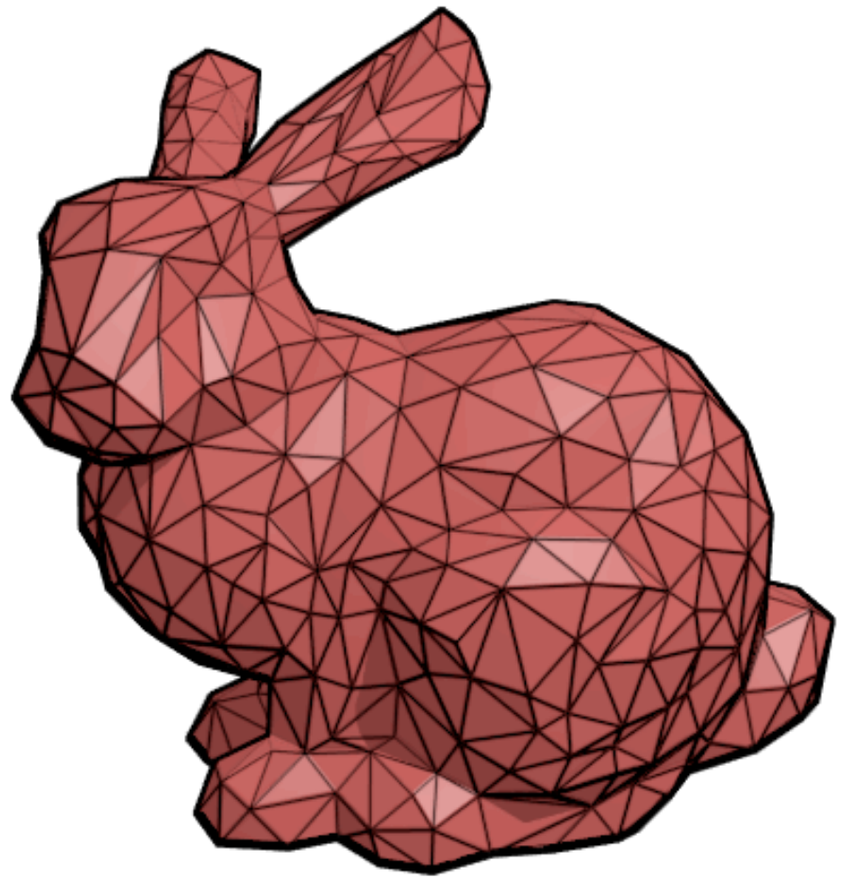
Why is this the most popular representation?

# Terminology

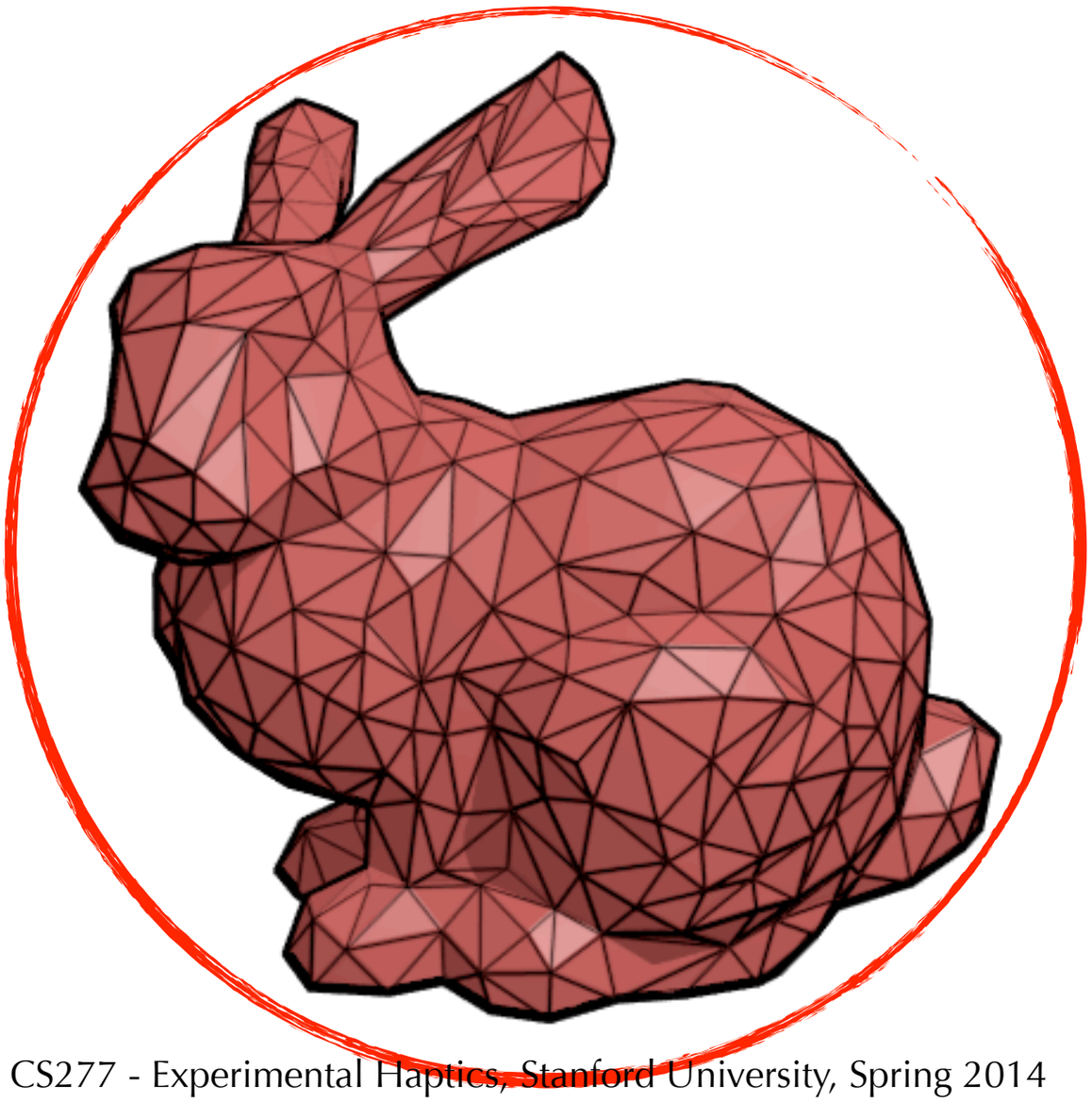
- Objects are composed of primitive shapes

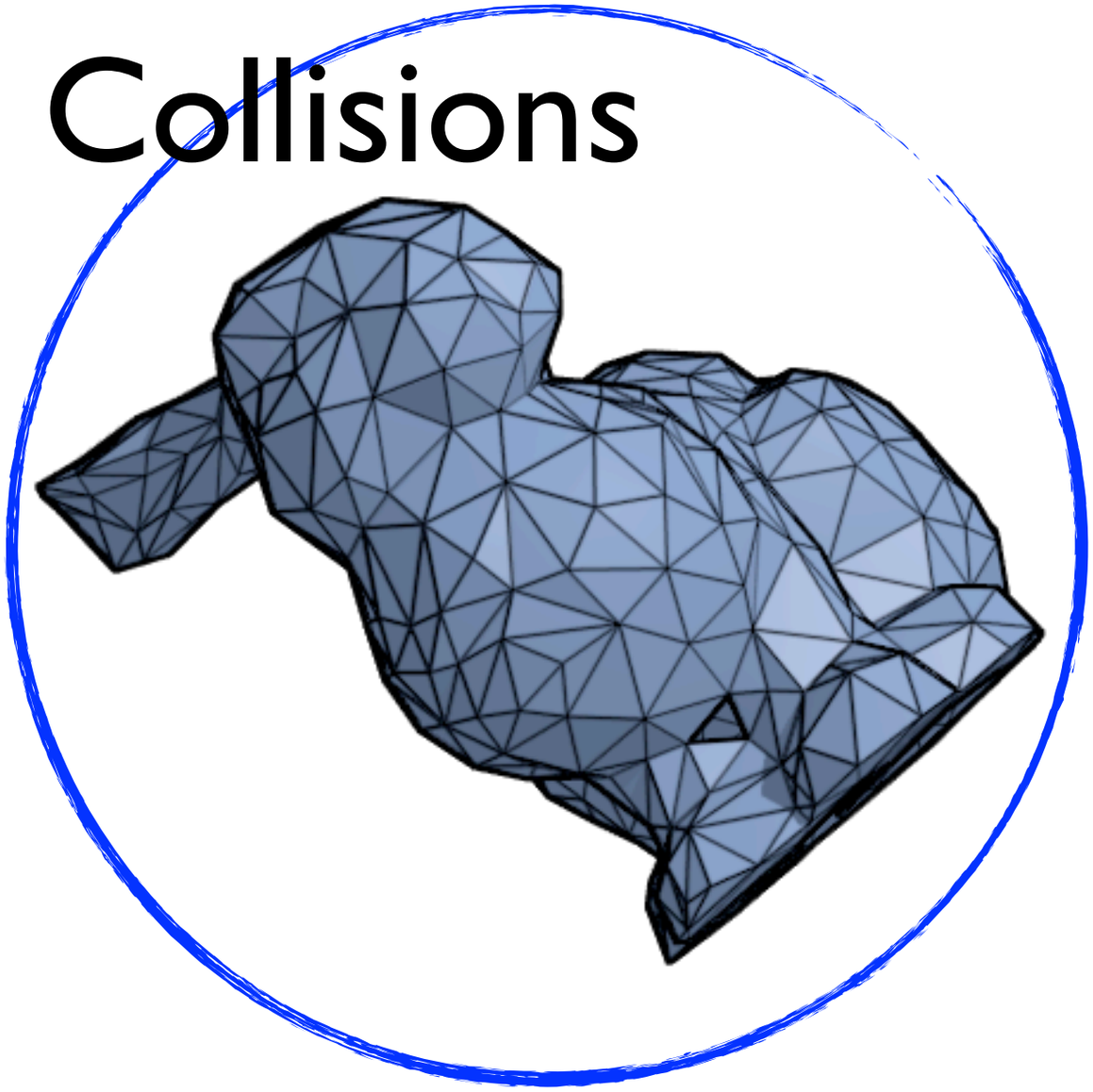
- **Broad phase**
  - Which objects are in a vicinity?

- **Narrow phase**
  - Does the geometry intersect?



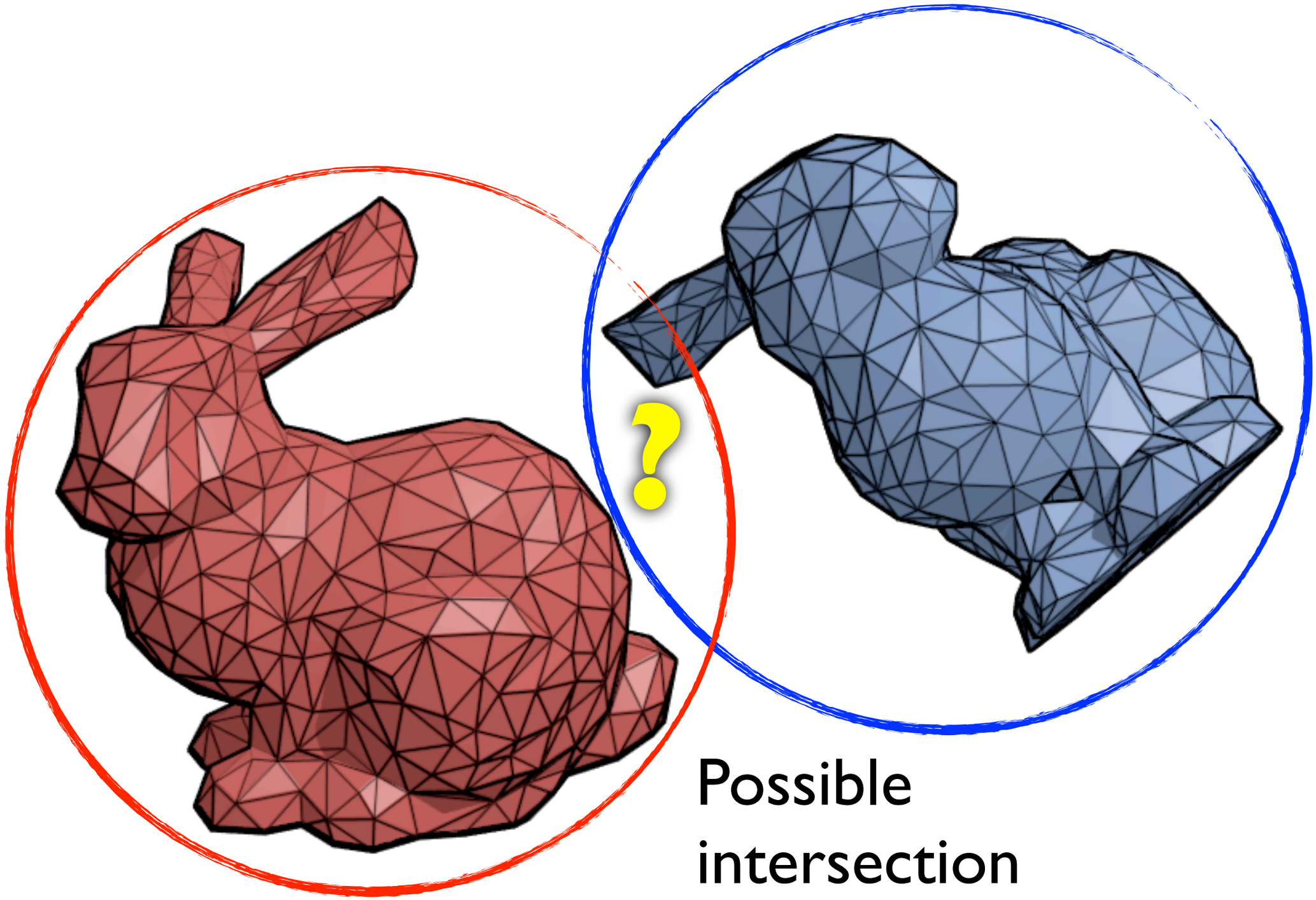
# Broad Phase Collisions





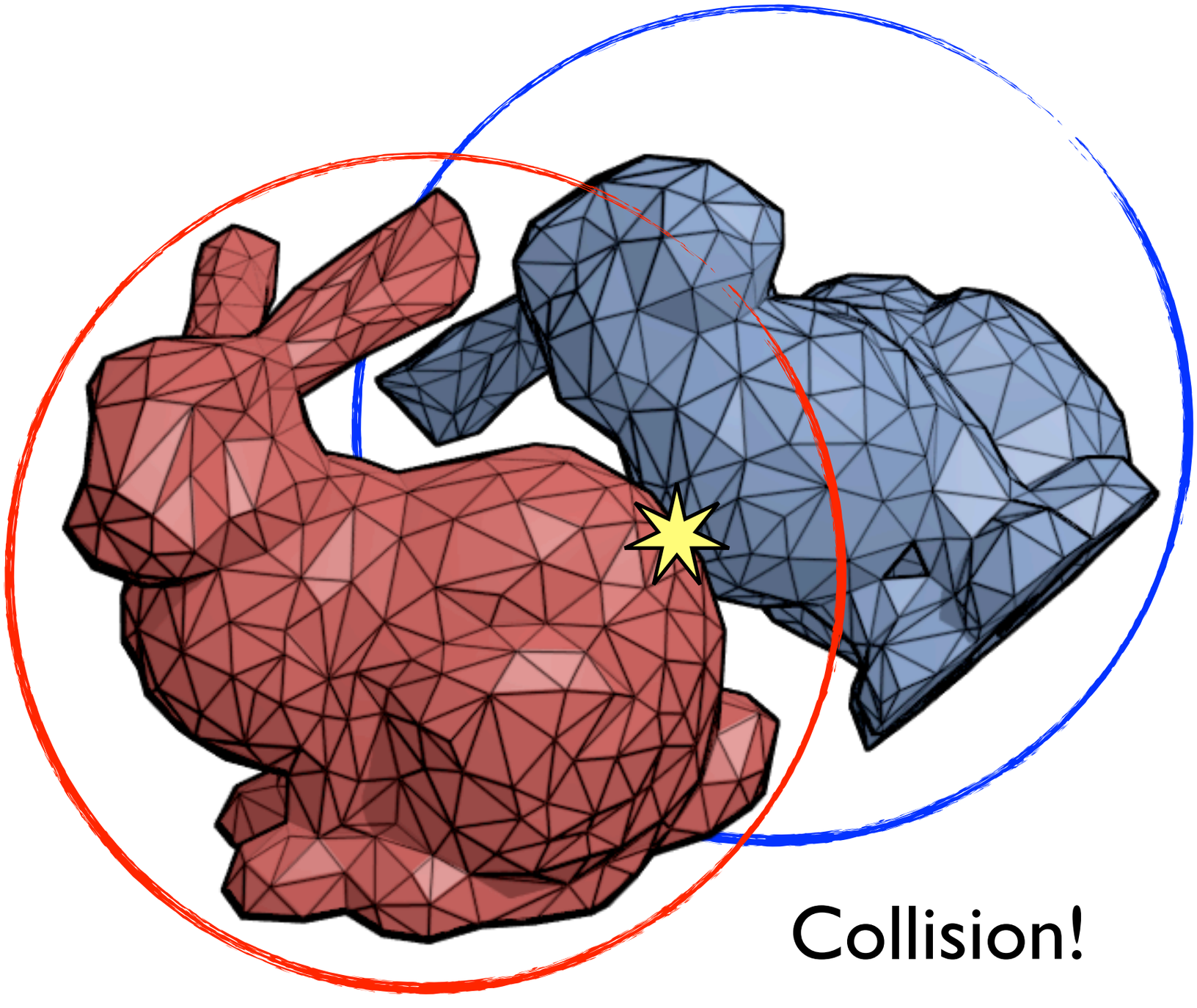
No possibility
of intersection

# Broad Phase Collisions

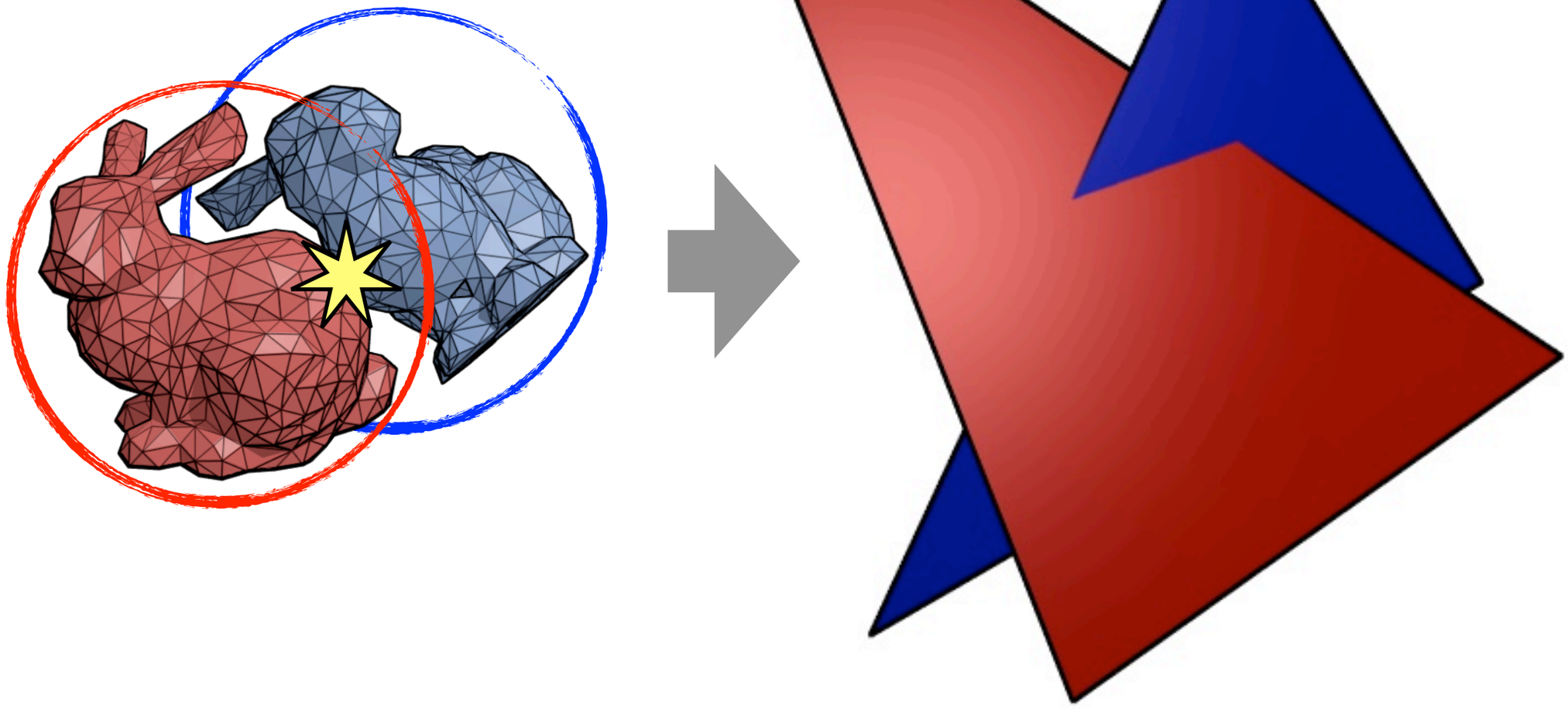


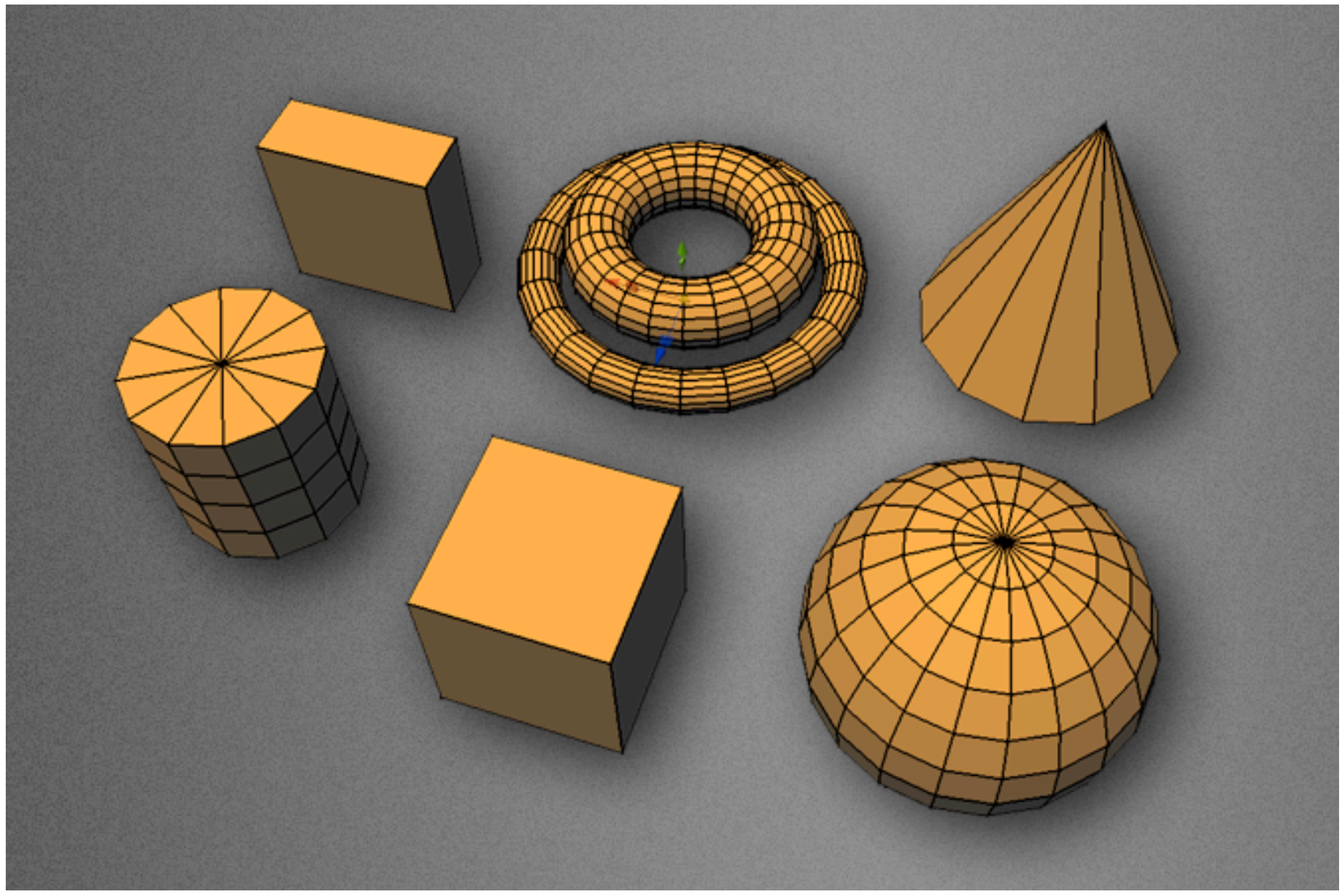

Possible intersection

# Narrow Phase Collisions



# Today's Lecture

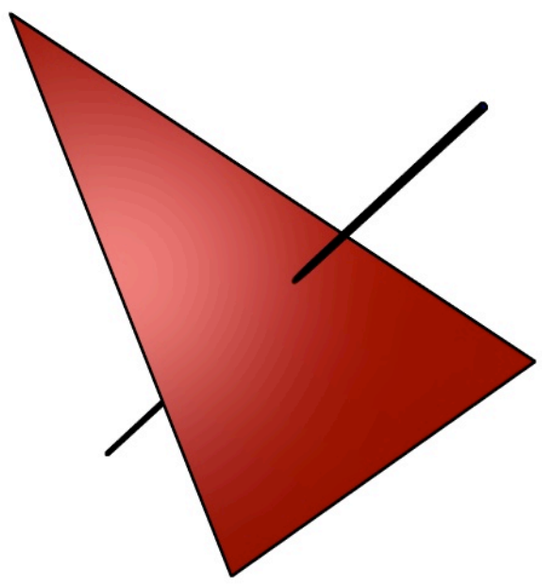


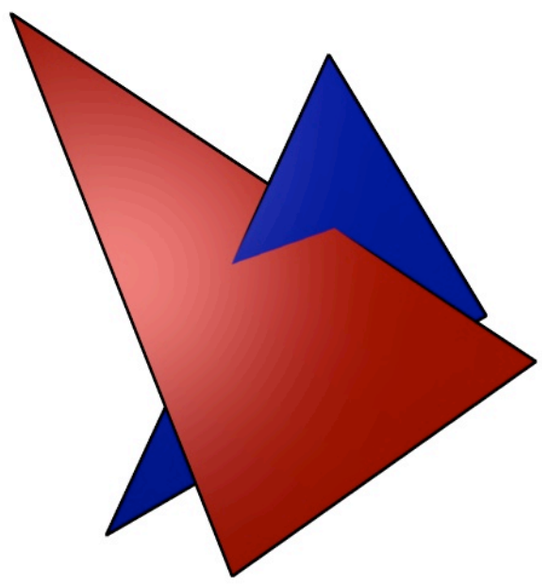


# Primitive Tests

# Tests for Meshes

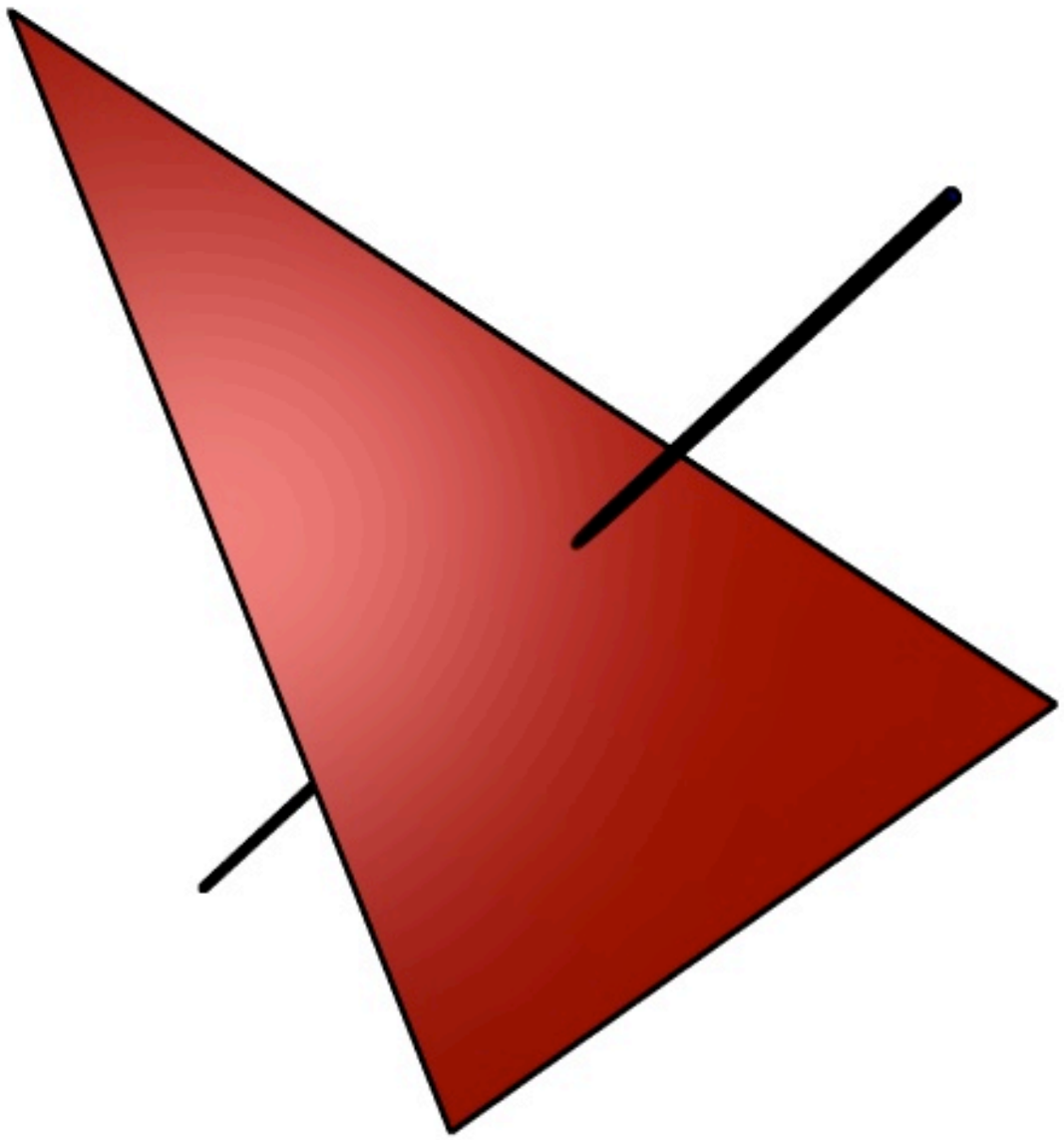
- The two most common collision queries for haptic rendering of polygonal meshes:
  - line segment-triangle intersection test
  - triangle-triangle intersection test



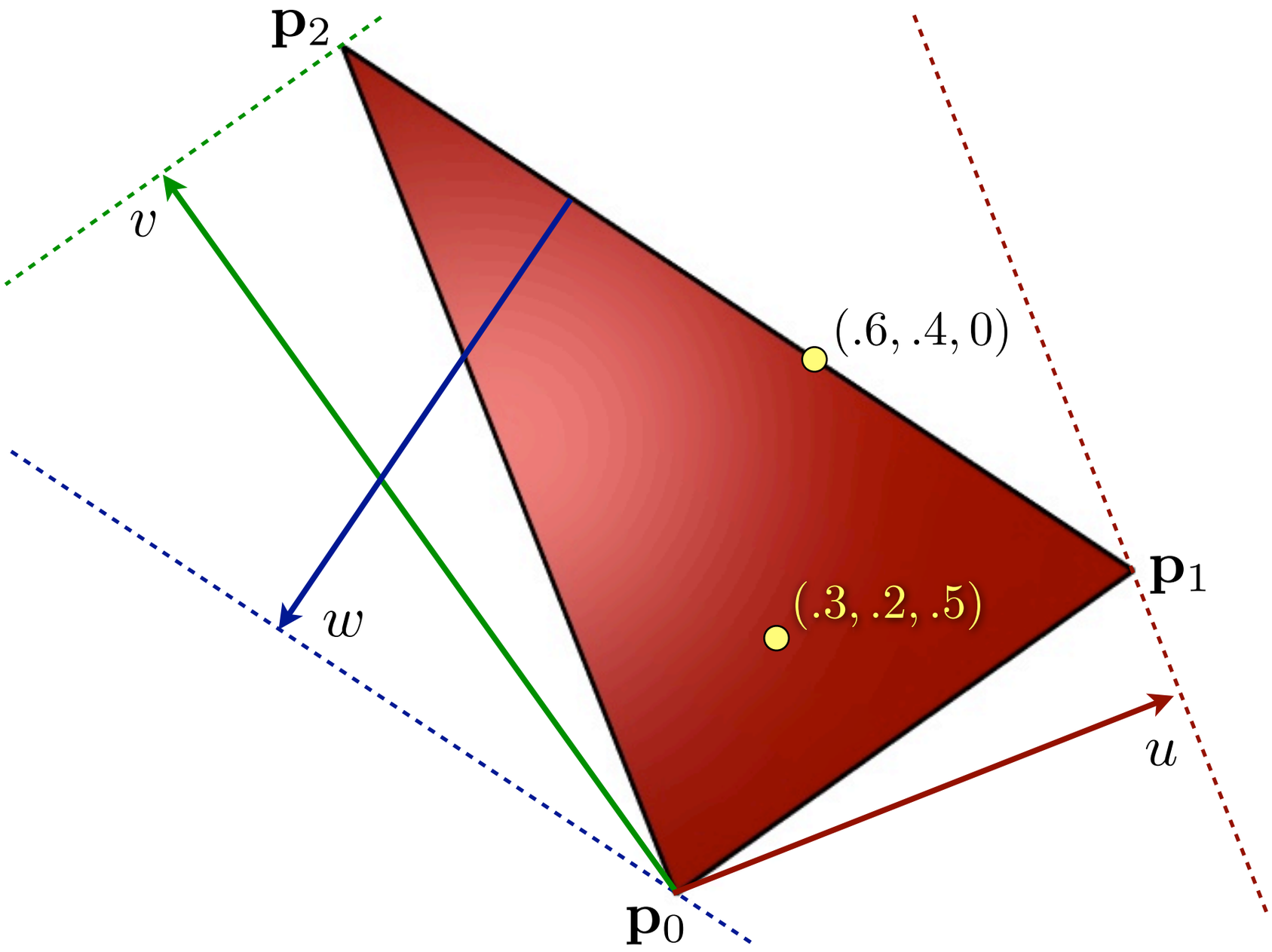


# Ray-Triangle Intersection

- Find intersection between line and plane
- Discard if point is outside segment range
- Use barycentric coordinates to determine if the point is inside the triangle

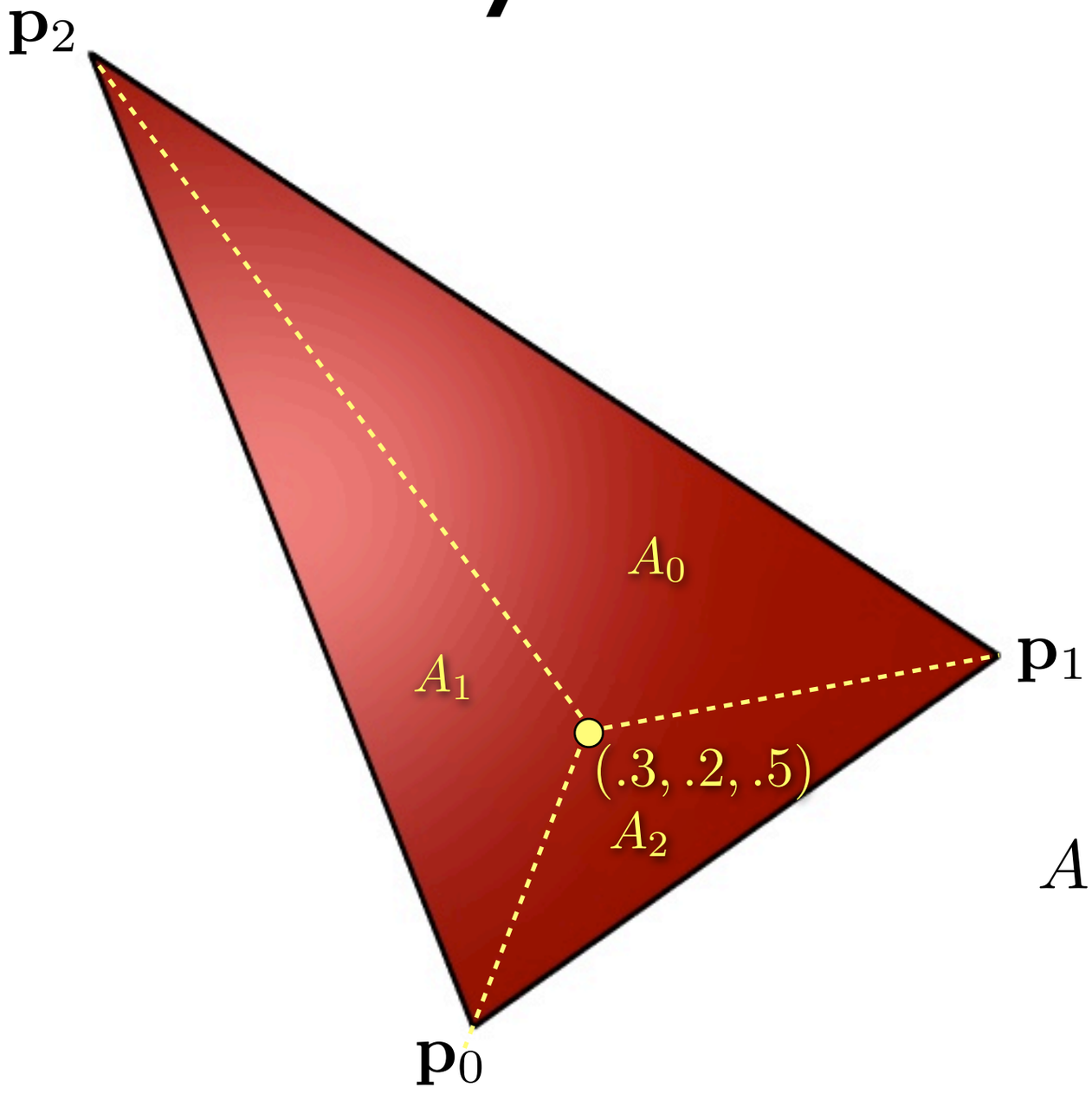


# Barycentric Coordinates




$$\mathbf{f}(u, v) = (1 - u - v)\mathbf{p}_0 + u\mathbf{p}_1 + v\mathbf{p}_2$$

# Barycentric Coordinates




$$u = \frac{A_1}{A}$$

$$v = \frac{A_2}{A}$$

$$A = \tfrac{1}{2}\,|(\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_0)|$$

$$\mathbf{f}(u,v) = (1-u-v)\mathbf{p}_0 + u\mathbf{p}_1 + v\mathbf{p}_2$$

# A Direct Approach

A ray: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$

A triangle: $\mathbf{f}(u, v) = (1 - u - v)\mathbf{p}_0 + u\mathbf{p}_1 + v\mathbf{p}_2$

Ray-triangle intersect: $\mathbf{o} + t\mathbf{d} = (1 - u - v)\mathbf{p}_0 + u\mathbf{p}_1 + v\mathbf{p}_2$

Rearrange terms: $\begin{pmatrix} -\mathbf{d} & \mathbf{p}_1 - \mathbf{p}_0 & \mathbf{p}_2 - \mathbf{p}_0 \end{pmatrix} \begin{pmatrix} t \\ u \\ v \end{pmatrix} = \mathbf{o} - \mathbf{p}_0$

Solve for t, u, and v ...

# Cramer's Rule

- Given the set of linear equations

$$\begin{pmatrix} & & \\ \mathbf{a} & \mathbf{b} & \mathbf{c} \\ & & \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{d}$$

- Write the determinant of the matrix

$$\det(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}$$

- Then

$$x = \frac{\det(\mathbf{d}, \mathbf{b}, \mathbf{c})}{\det(\mathbf{a}, \mathbf{b}, \mathbf{c})} \qquad y = \frac{\det(\mathbf{a}, \mathbf{d}, \mathbf{c})}{\det(\mathbf{a}, \mathbf{b}, \mathbf{c})} \qquad z = \frac{\det(\mathbf{a}, \mathbf{b}, \mathbf{d})}{\det(\mathbf{a}, \mathbf{b}, \mathbf{c})}$$

# A Direct Approach

Our equation:

$$\begin{pmatrix} -\mathbf{d} & \mathbf{p}_1 - \mathbf{p}_0 & \mathbf{p}_2 - \mathbf{p}_0 \end{pmatrix} \begin{pmatrix} t \\ u \\ v \end{pmatrix} = \mathbf{o} - \mathbf{p}_0$$

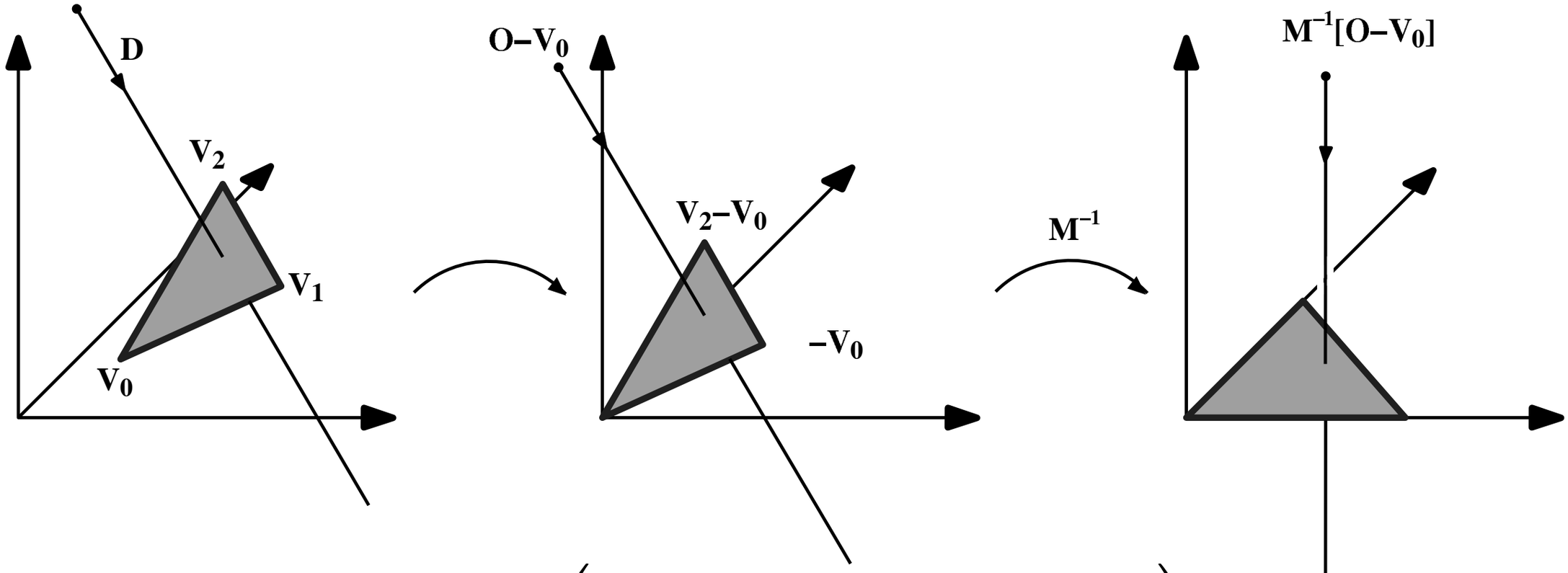
Applying Cramer's rule:

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{\det(-\mathbf{d}, \mathbf{e}_1, \mathbf{e}_2)} \begin{pmatrix} \det(\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2) \\ \det(-\mathbf{d}, \mathbf{s}, \mathbf{e}_2) \\ \det(-\mathbf{d}, \mathbf{e}_1, \mathbf{s}) \end{pmatrix}$$

where $\mathbf{e}_1 = \mathbf{p}_1 - \mathbf{p}_0,\ \mathbf{e}_2 = \mathbf{p}_2 - \mathbf{p}_0,\ \mathbf{s} = \mathbf{o} - \mathbf{p}_0$

## Check t, u, v within intervals!

# Geometric Interpretation

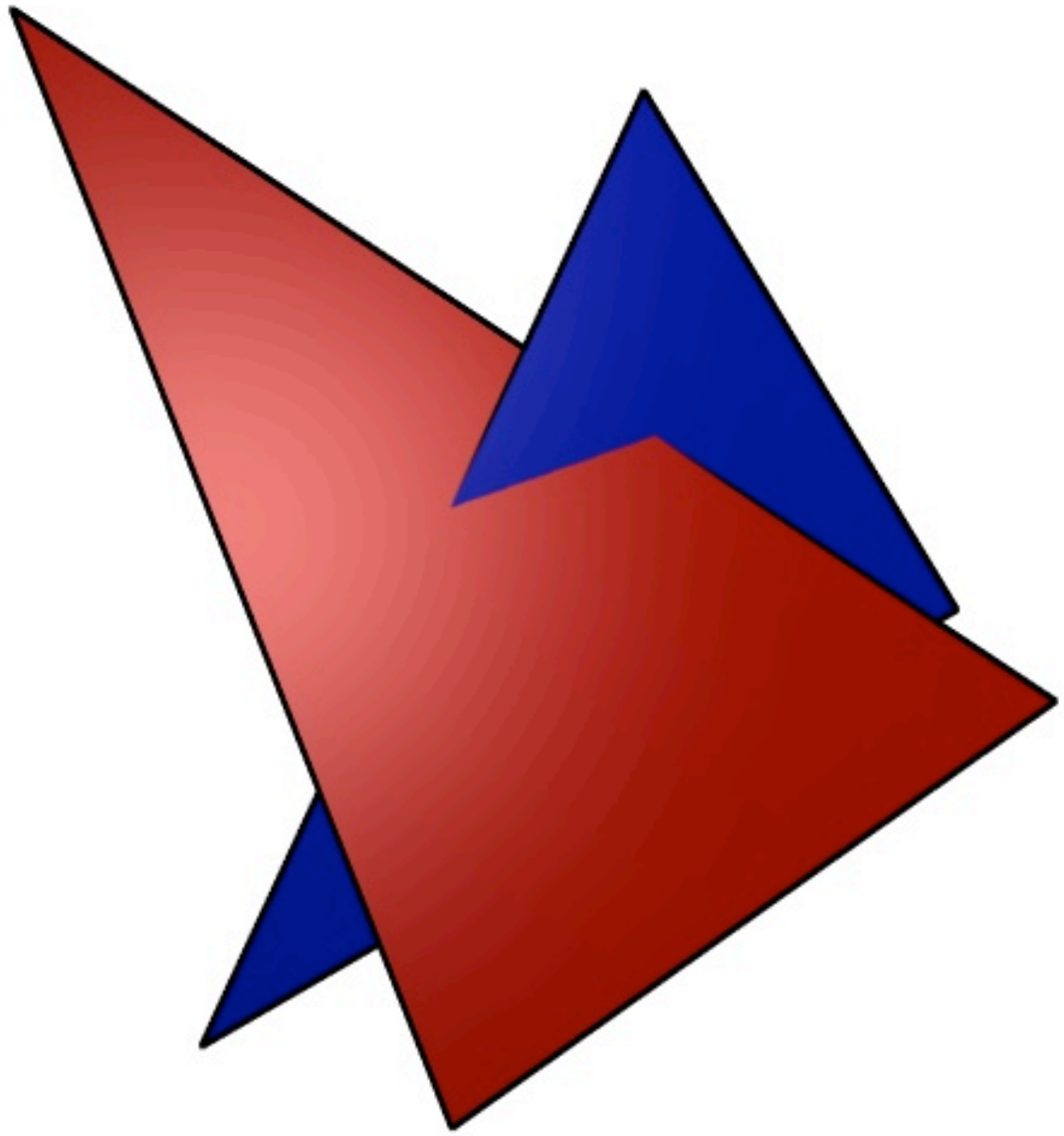



$$\mathbf{M} = \begin{pmatrix} -\mathbf{d} & \mathbf{p}_1 - \mathbf{p}_0 & \mathbf{p}_2 - \mathbf{p}_0 \end{pmatrix}$$

[from T. Möller & B. Trumbore, *Journal of Graphics Tools*, 1997.]

# Triangle-Triangle Intersection

- Triangles A and B may intersect if they cross each other's plane
  - Test A's vertices against B's plane, and vice versa for rejection
  - Test for interval overlap along the line of intersection



# Half-Plane Test

$$
\begin{aligned}
[\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}] &= \begin{vmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \\ 1 & 1 & 1 & 1 \end{vmatrix} \\
&= (\mathbf{d}-\mathbf{a}) \cdot ((\mathbf{b}-\mathbf{a}) \times (\mathbf{c}-\mathbf{a}))
\end{aligned}
$$

- Geometric interpretation:
  - This tests which side of the plane defined by triangle **abc** the point **d** is on

# Half-Plane Test

- Given two triangles:

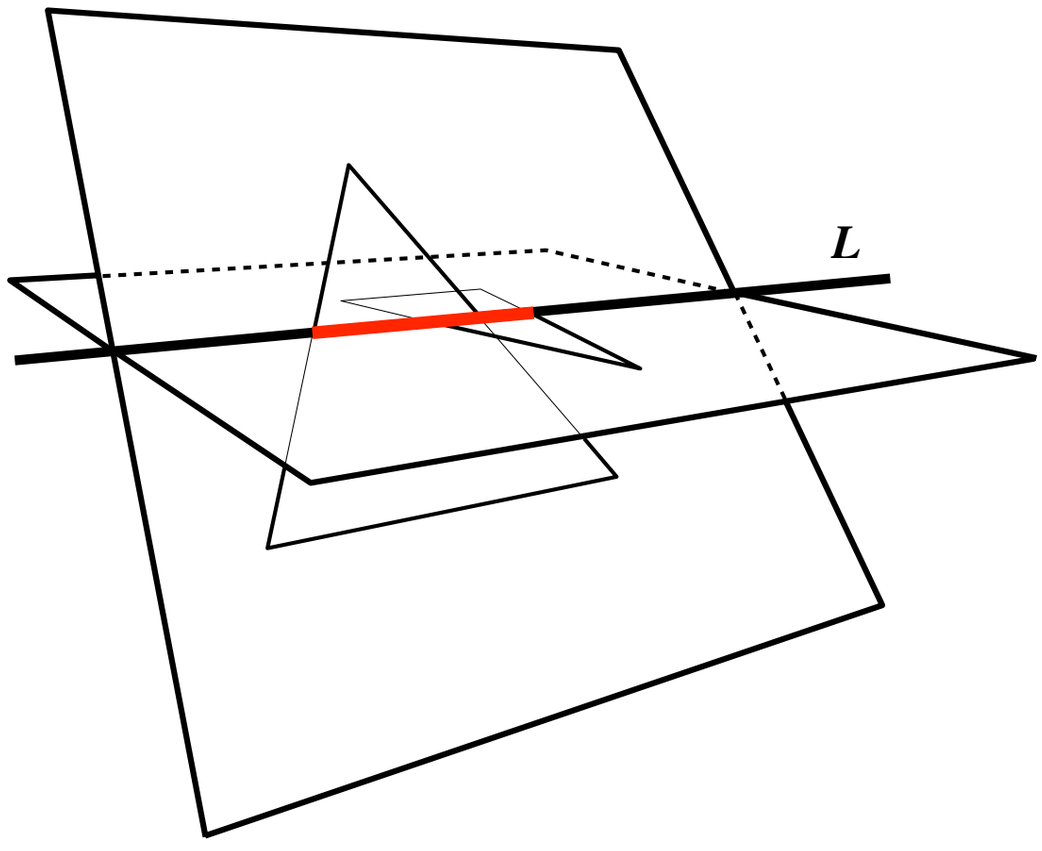
$$\triangle \mathbf{p}_1\mathbf{q}_1\mathbf{r}_1 \quad \text{and} \quad \triangle \mathbf{p}_2\mathbf{q}_2\mathbf{r}_2$$

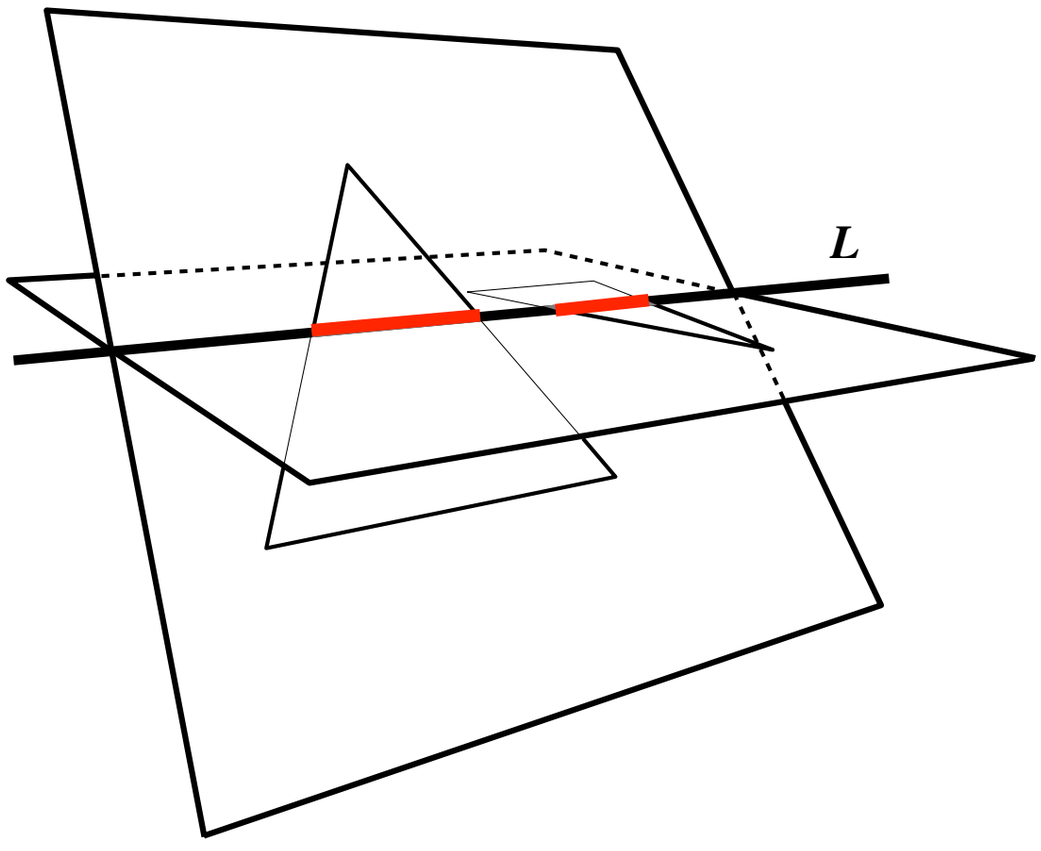
  - We can first perform the half-plane test on triangle one:

$$[\mathbf{p}_2, \mathbf{q}_2, \mathbf{r}_2, \mathbf{p}_1] \quad [\mathbf{p}_2, \mathbf{q}_2, \mathbf{r}_2, \mathbf{q}_1] \quad [\mathbf{p}_2, \mathbf{q}_2, \mathbf{r}_2, \mathbf{r}_1]$$

  - Then symmetrically perform the half-plane test on the other triangle...

# Intersection of Intervals




[from T. Möller, *Journal of Graphics Tools*, 1997.]

# Interval Intersection Test

- Intervals on line L are

$$I_1 = [i, j] \quad I_2 = [k, l]$$

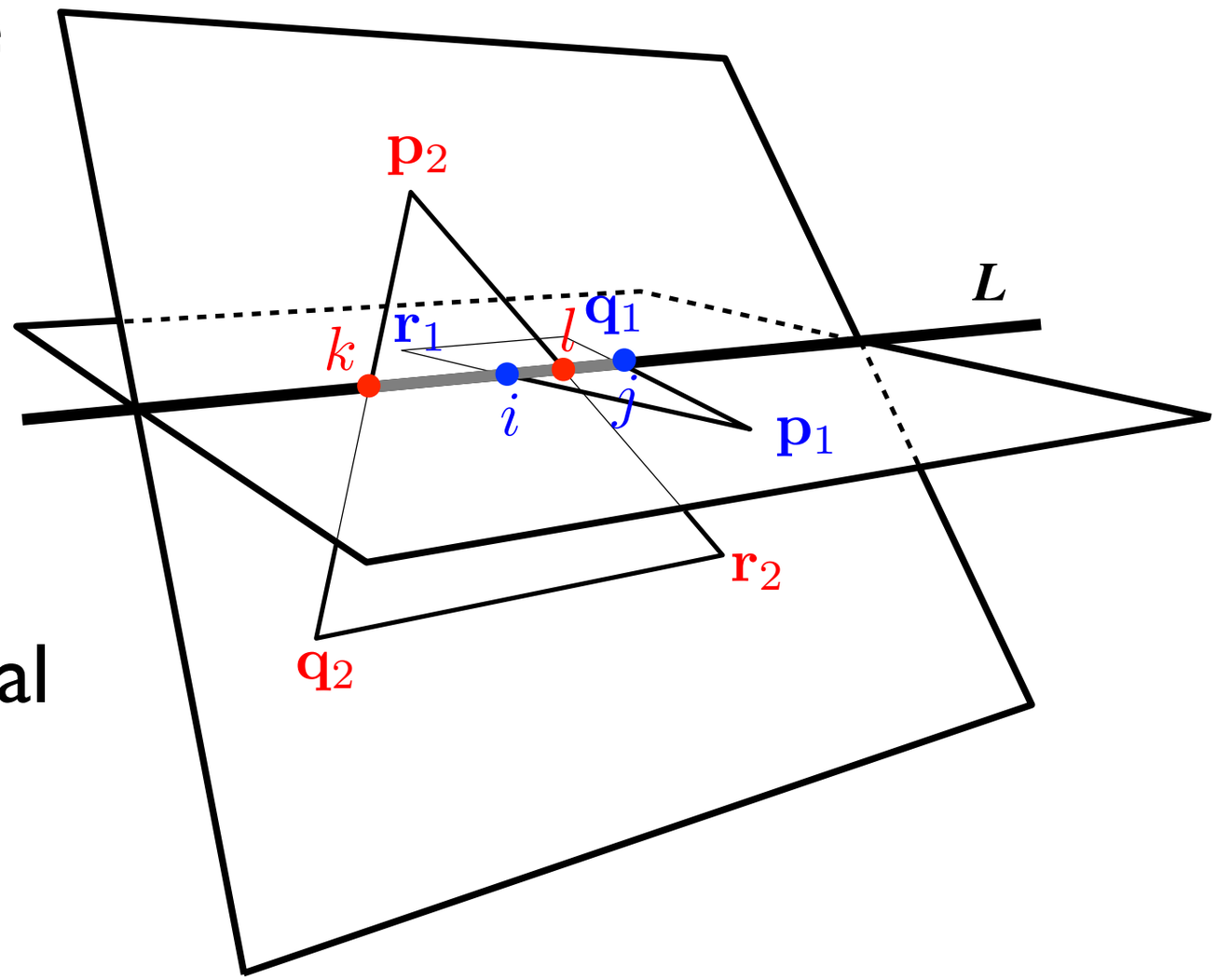
- Intervals overlap if

$$k \leq j \ \text{ and } \ i \leq l$$

- Perform two additional determinant tests:

$$[\mathbf{p}_1, \mathbf{q}_1, \mathbf{p}_2, \mathbf{q}_2]$$

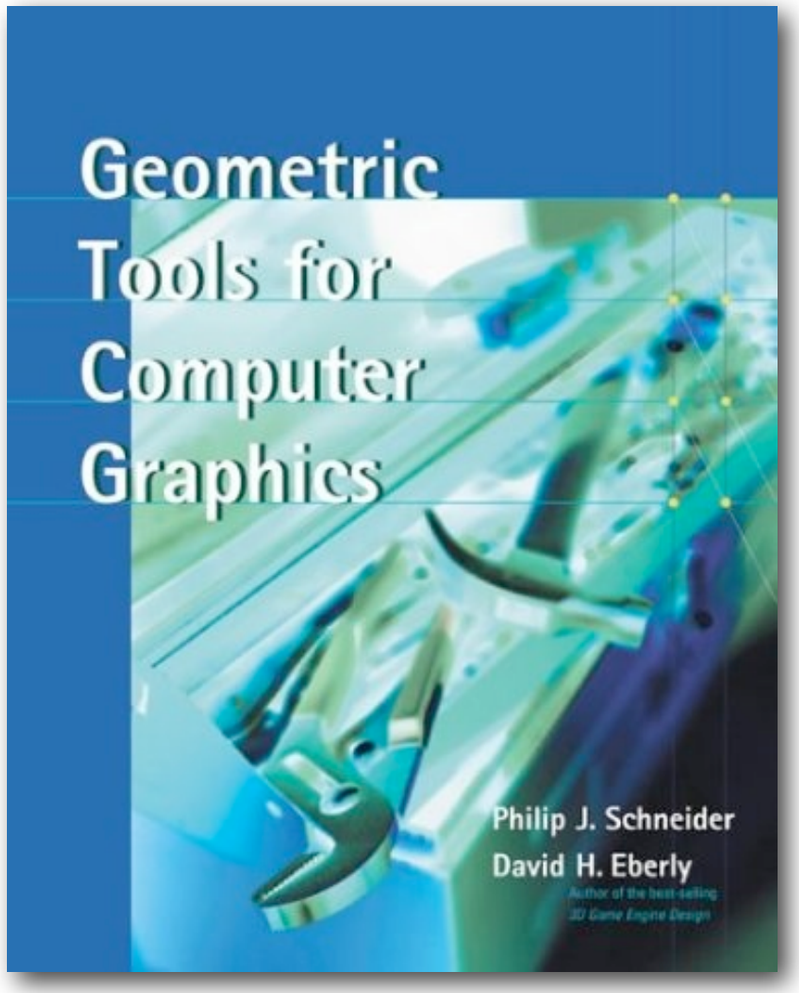
$$[\mathbf{p}_1, \mathbf{r}_1, \mathbf{r}_2, \mathbf{p}_2]$$

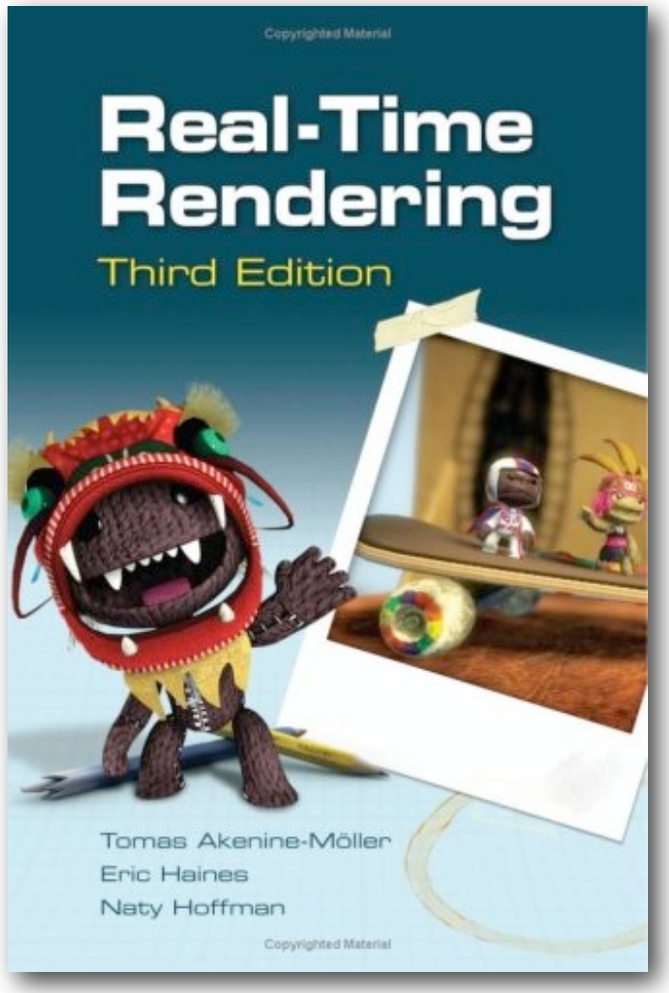


# Triangle-Triangle Summary

- Compute three 4x4 determinants to test first triangle against the second's plane
- If triangle intersects the plane, perform the symmetric test using three more determinants
- If both triangles intersect the other's plane, perform interval overlap test on the intersecting line with two last 4x4 determinants
- What happens with co-planar triangles?!

# Two of My Favorite Books



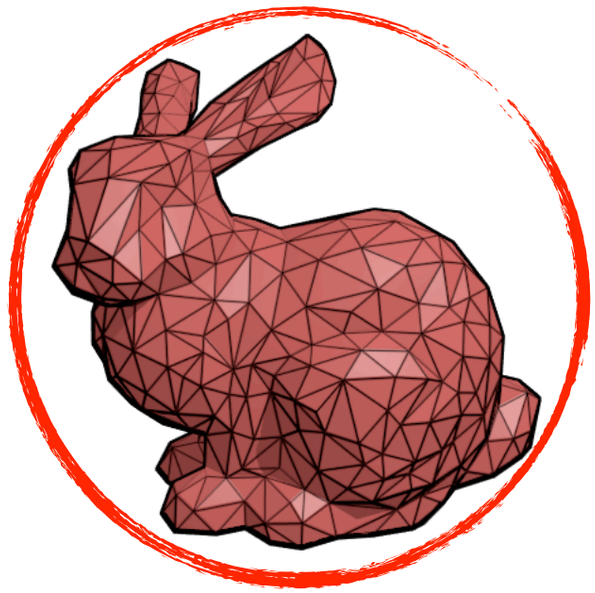

No need to memorize any of these algorithms!

# Some Easier Stuff...

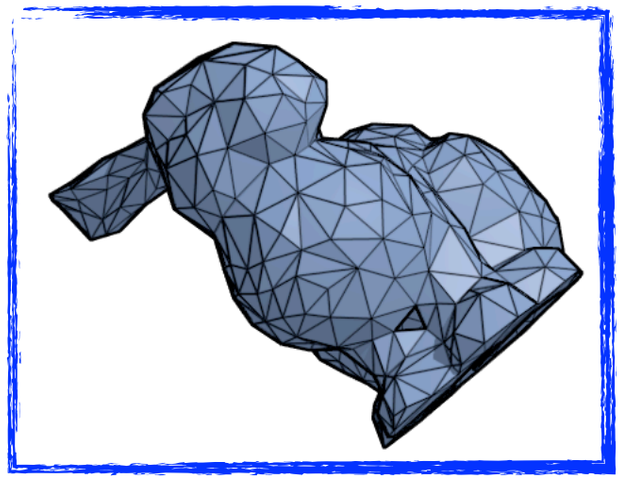
- Mesh geometry intersection tests are expensive, and must be performed for every triangle
- Collision detection can be sped up significantly by using *rejection tests* on *bounding volumes*

# Bounding Volumes

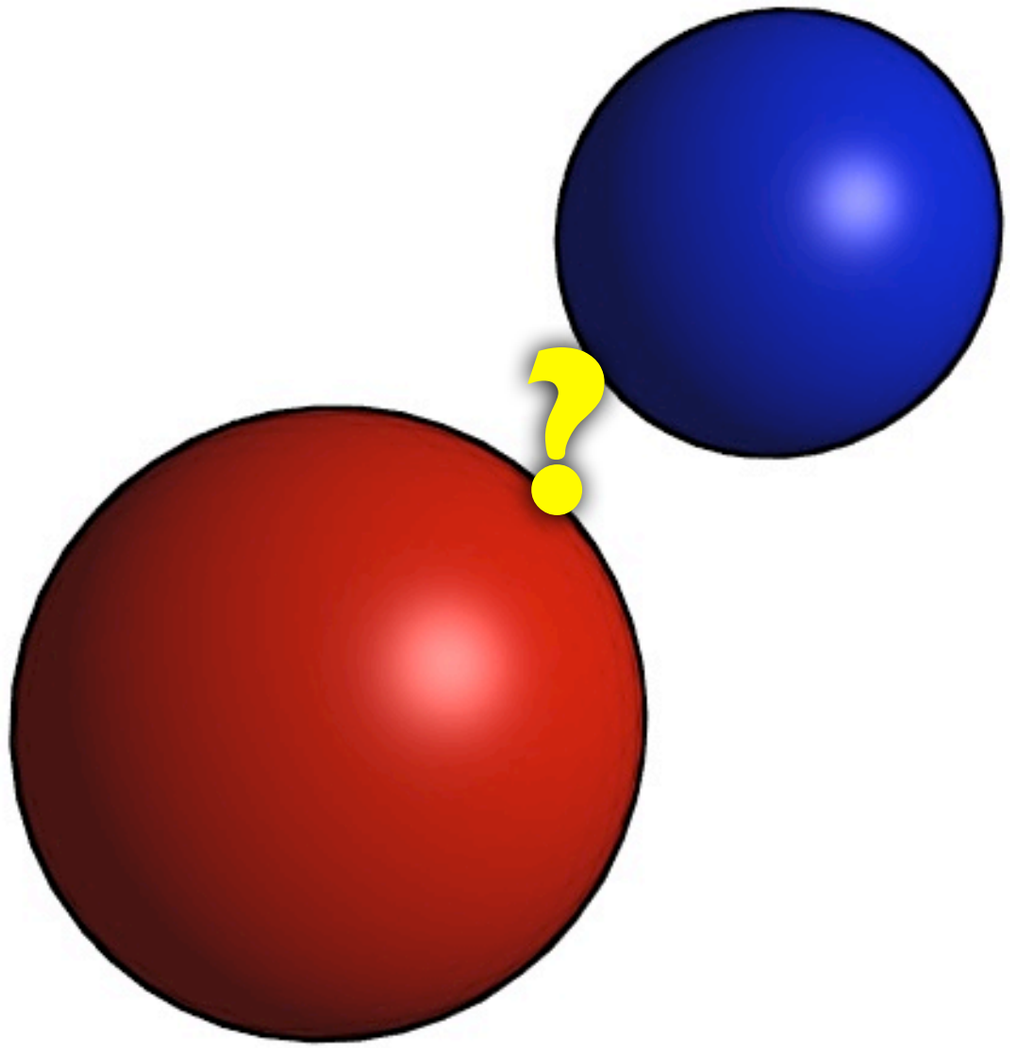
- Most common bounding volumes are **spheres** and **boxes**



- Two most common collision queries:
  - Sphere-sphere intersection
  - Box-box intersection



# Sphere-Sphere Intersection

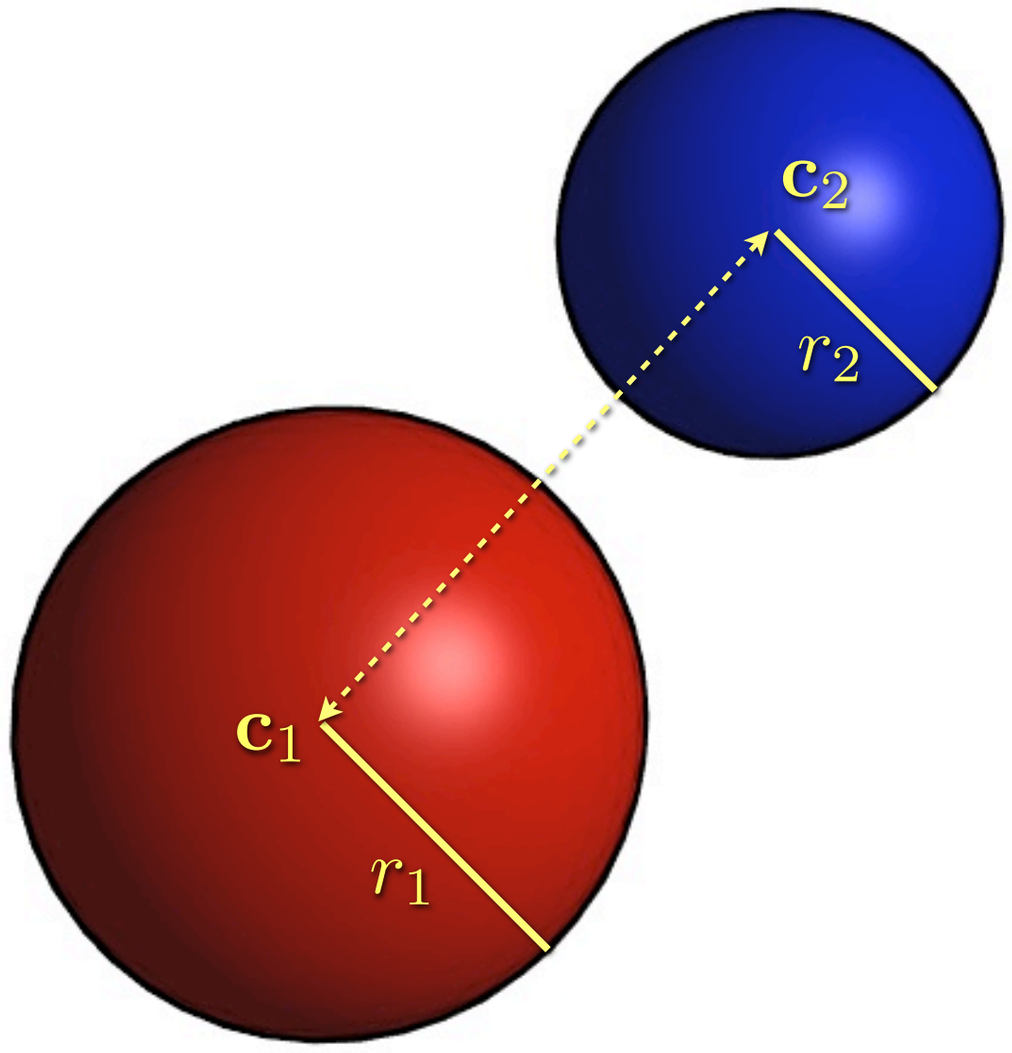


Easiest one in the book!

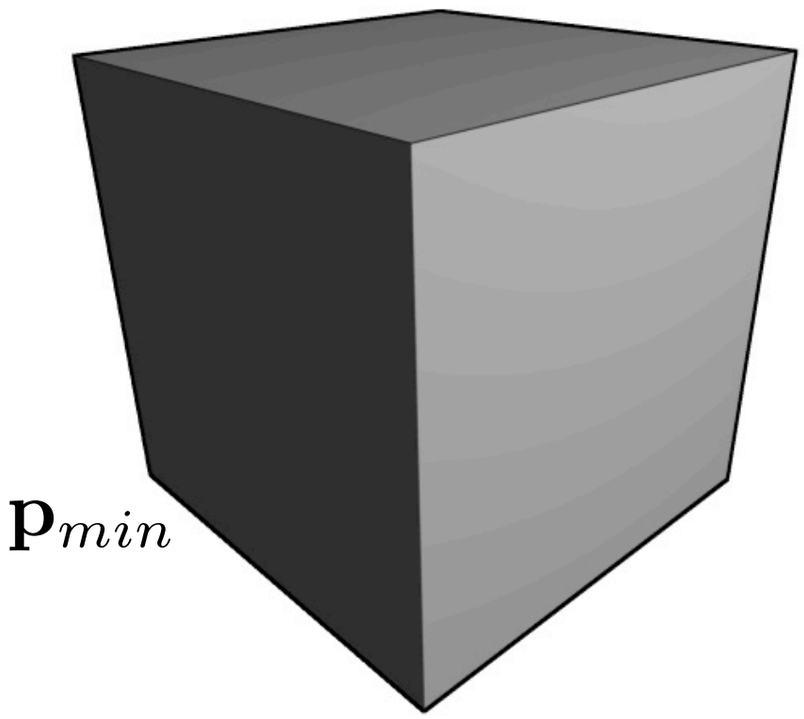
# Sphere-Sphere Intersection

- Two spheres intersect if the separation between their centers is less than the sum of their radii:

$$||\mathbf{c}_1 - \mathbf{c}_2|| < r_1 + r_2$$



# Box-Box Intersection



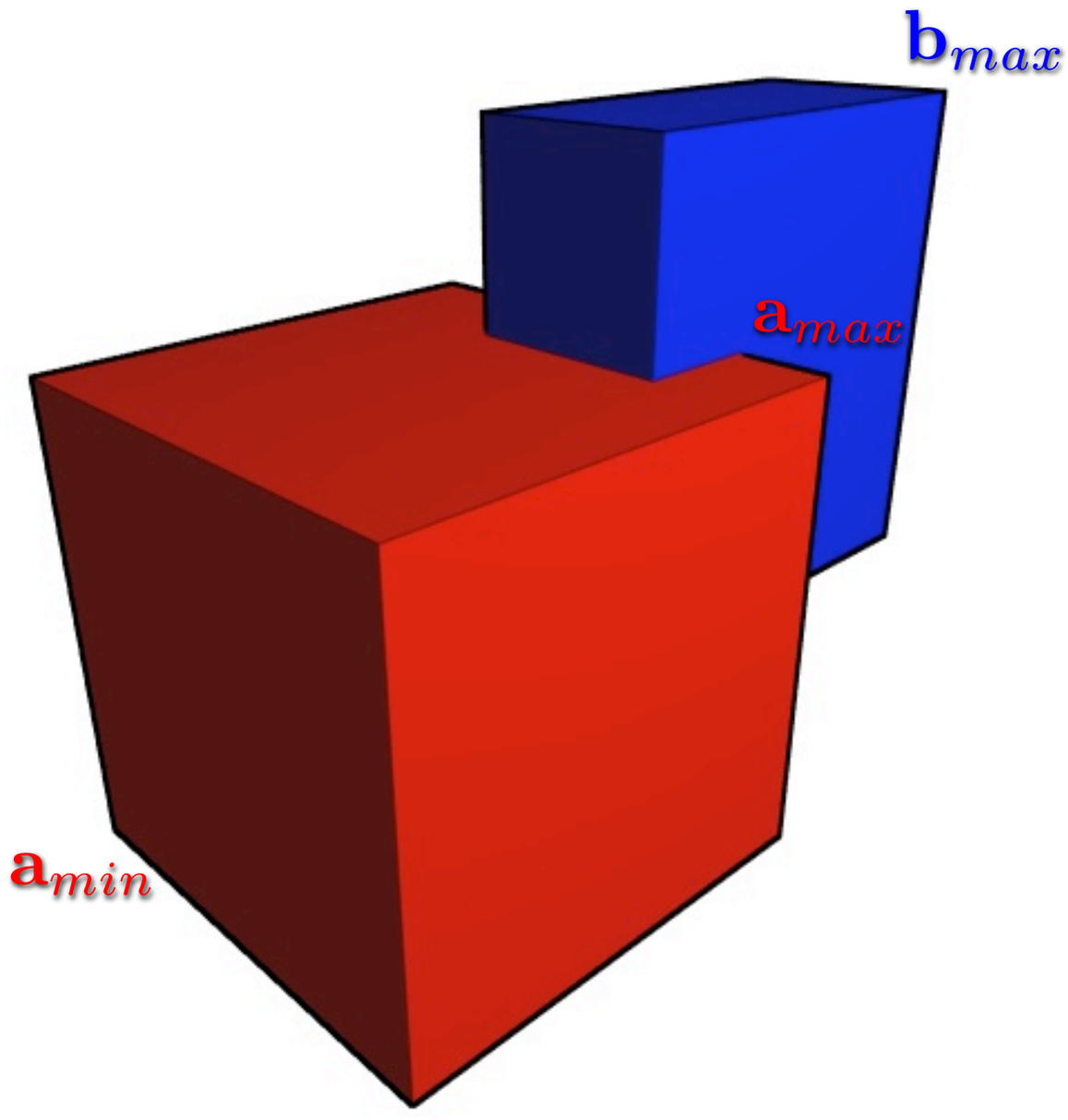

- An axis-aligned box is represented by lower (minimum coordinate) and upper (maximum coordinate) vertices
- How do we detect intersection of boxes?

# Box-Box Intersection

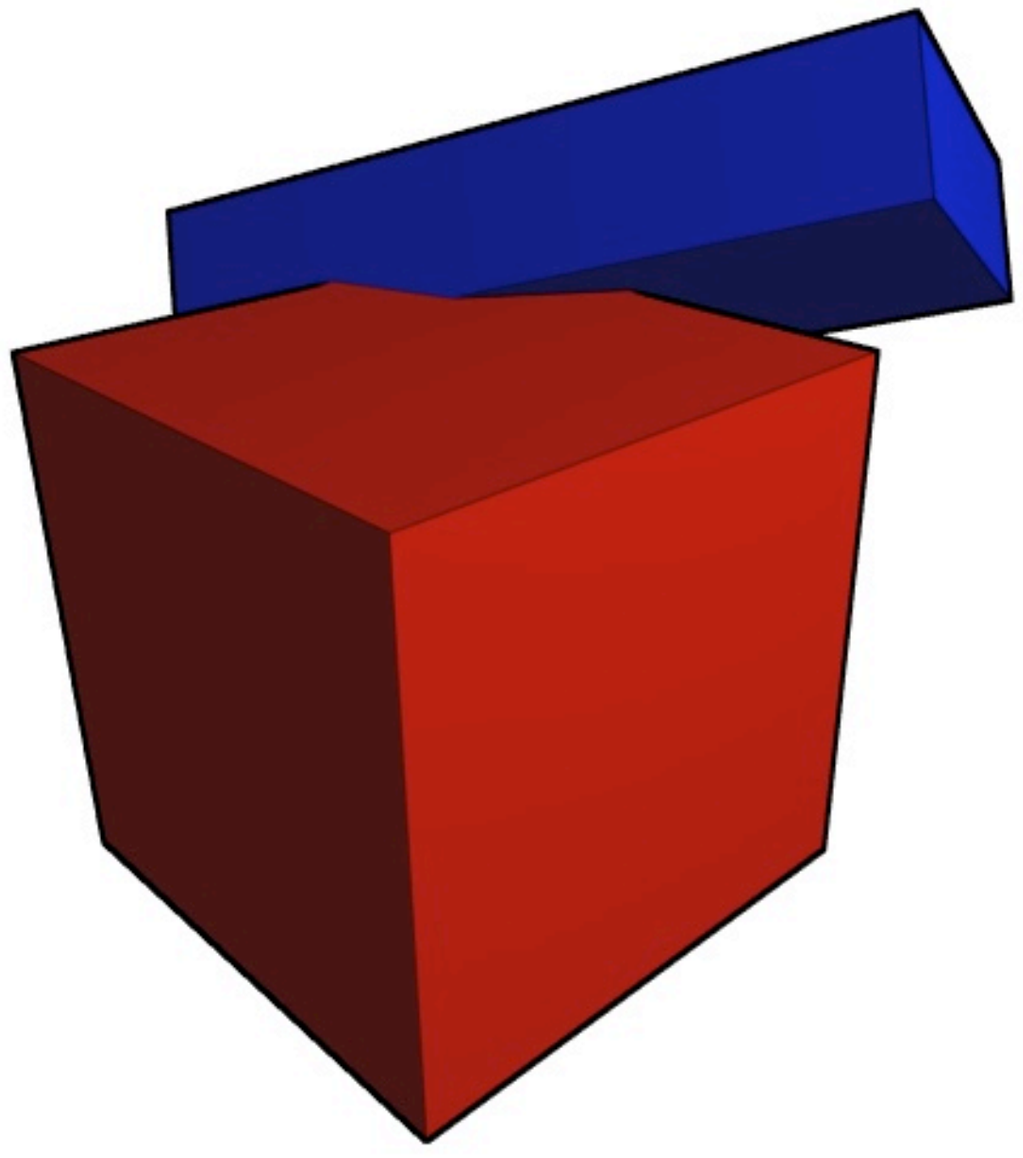
- Two axis-aligned boxes intersect if the lower coordinate of each box is bounded by the upper coordinate of the other:

$$\mathbf{a}_{min} < \mathbf{b}_{max}$$

$$\mathbf{b}_{min} < \mathbf{a}_{max}$$



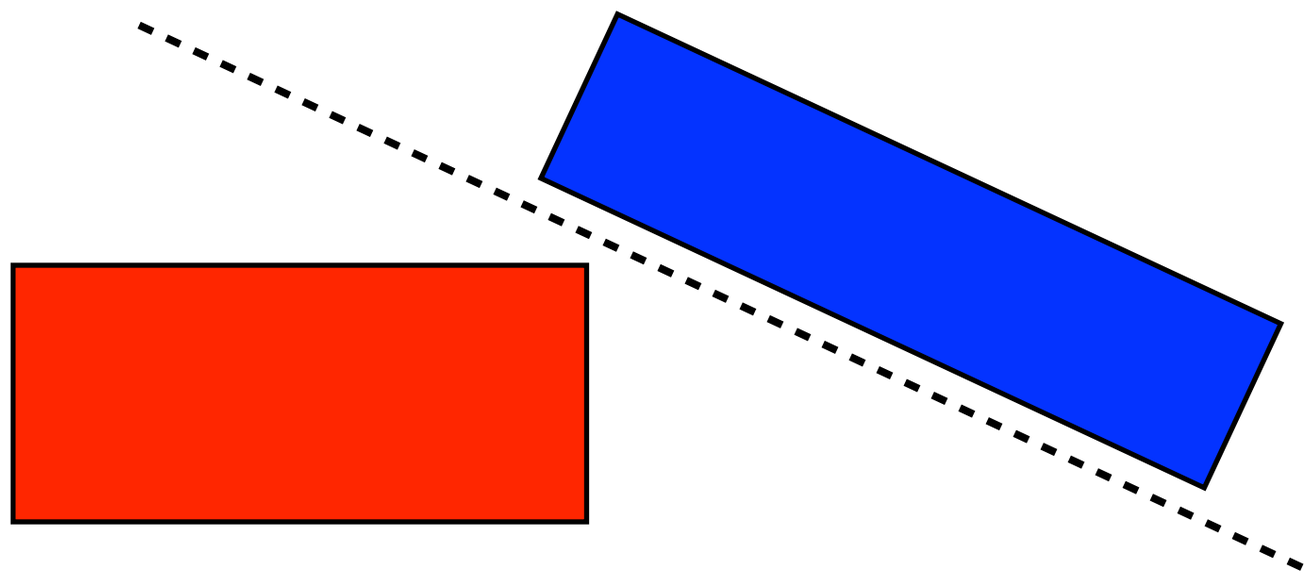
# Oriented Box Intersection



How do we test for this kind of box intersection?

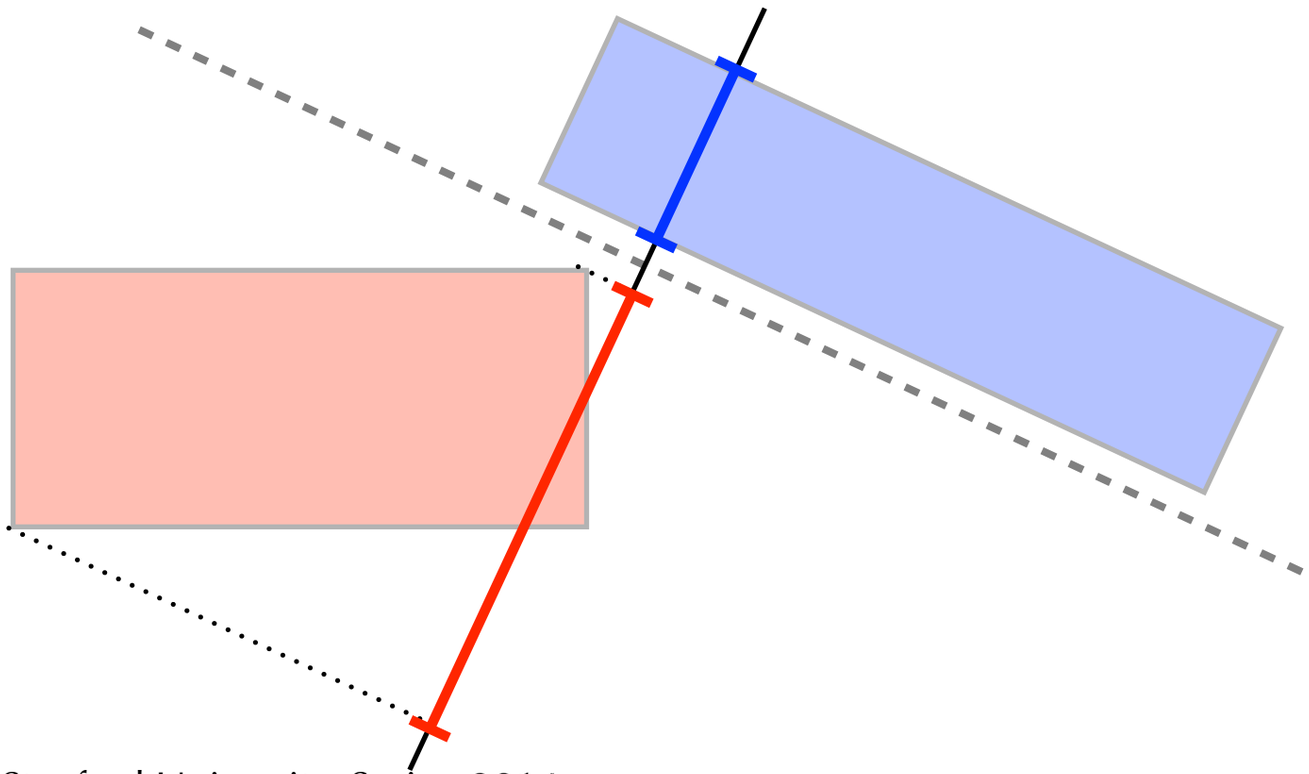
# Separating Hyperplane Theorem

- Two convex polytopes can be separated by a hyperplane if and only if they are disjoint
- For disjoint polyhedra, there exists a separating plane parallel to a face on either polyhedron, or an edge selected from each polyhedron (why?)

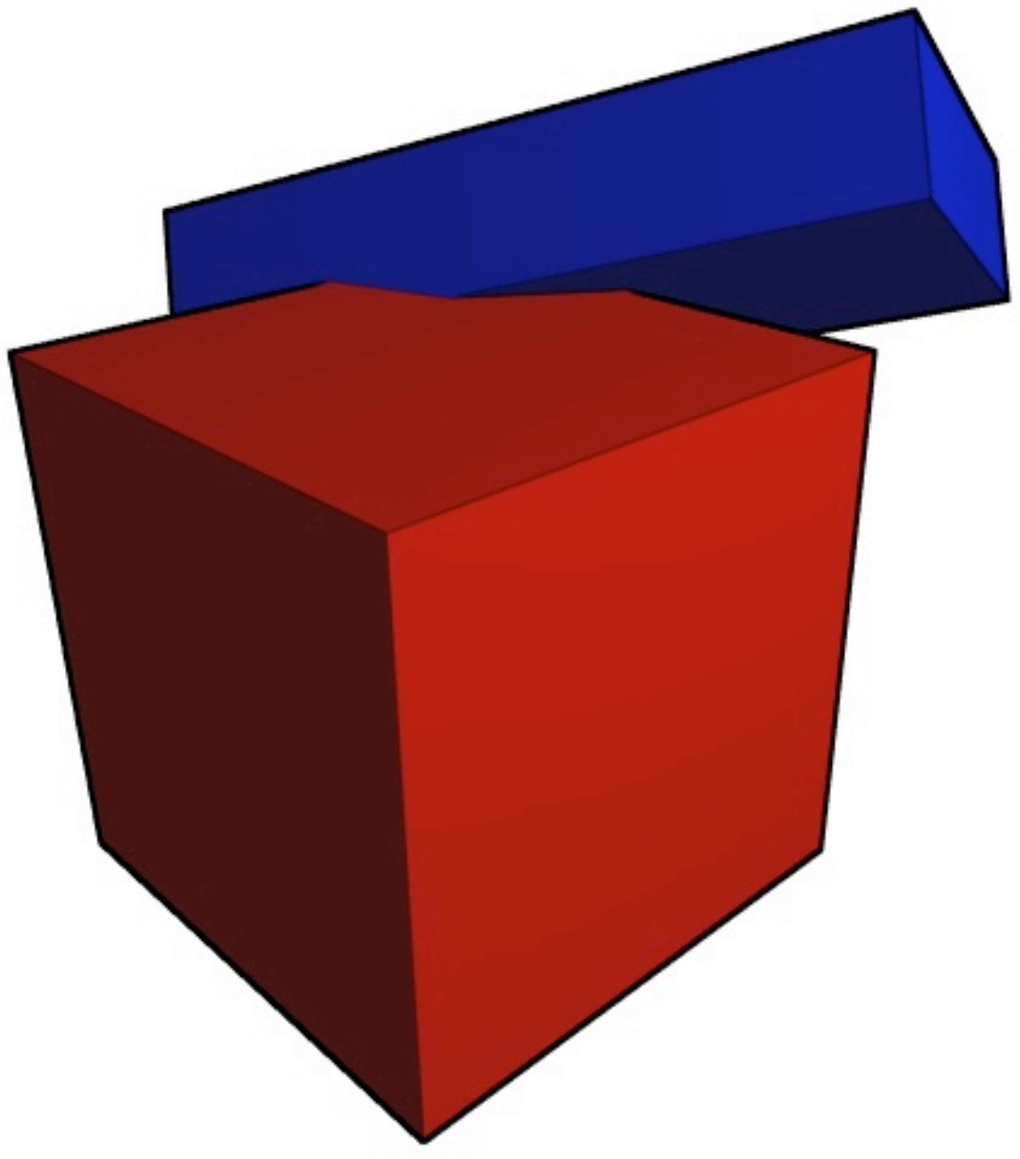


# Separating Axis Test

- Project all vertices onto the normal of the separating plane ("separating axis")
- Projections from each polytope form an interval
- Polytopes are disjoint if intervals are disjoint

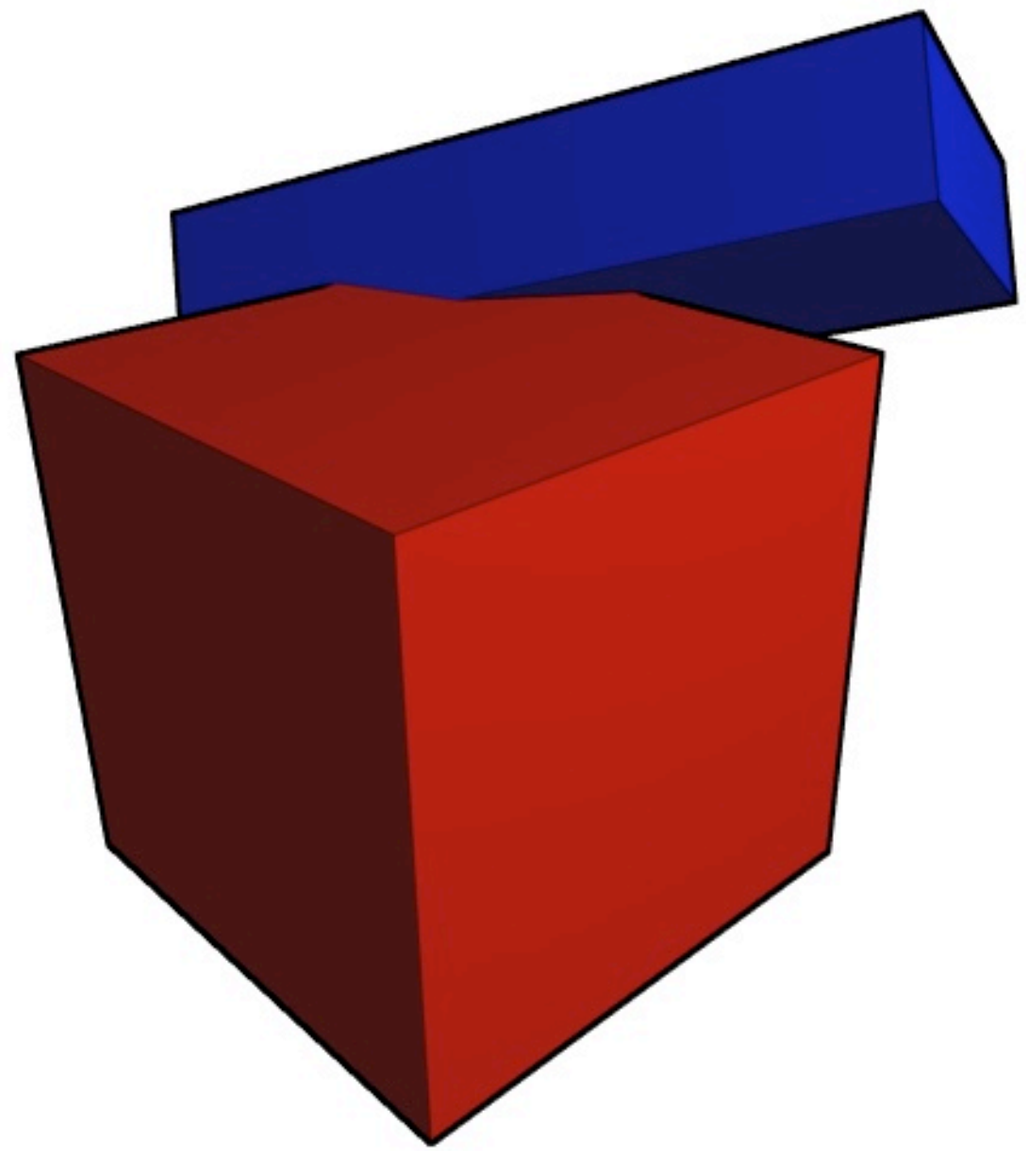


# Oriented Box Intersection

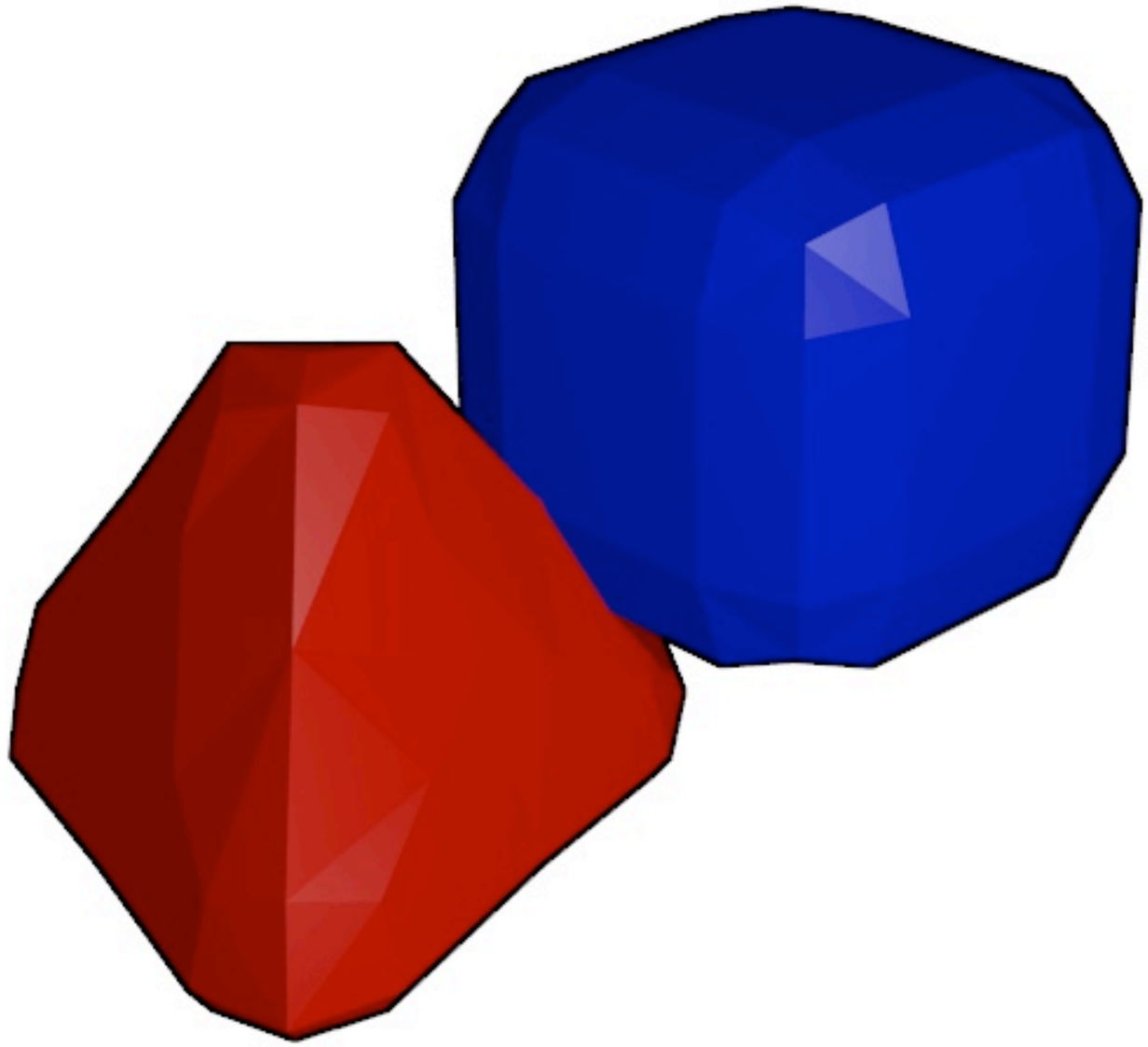


# Oriented Box Intersection

- Perform separating axis test on every possible axis:
  - 3 axes (faces of box A)
  - 3 axes (faces of box B)
  - 3 x 3 = 9 axes from pairs of box edges
- Total: 15 separating axis tests



# Convex Polyhedra



How would we test shapes like these?

# Intersection of Convex Polyhedra

- Use the separating hyperplane theorem:
  - How do we determine what the plane/axis is?

- Use a feature tracking approach:
  - Polyhedra will collide when their separation distance vanishes
  - Can we tell which elements are about to collide?

# Linear Programming

- Maximize

$$\mathbf{c}^T\mathbf{x}$$

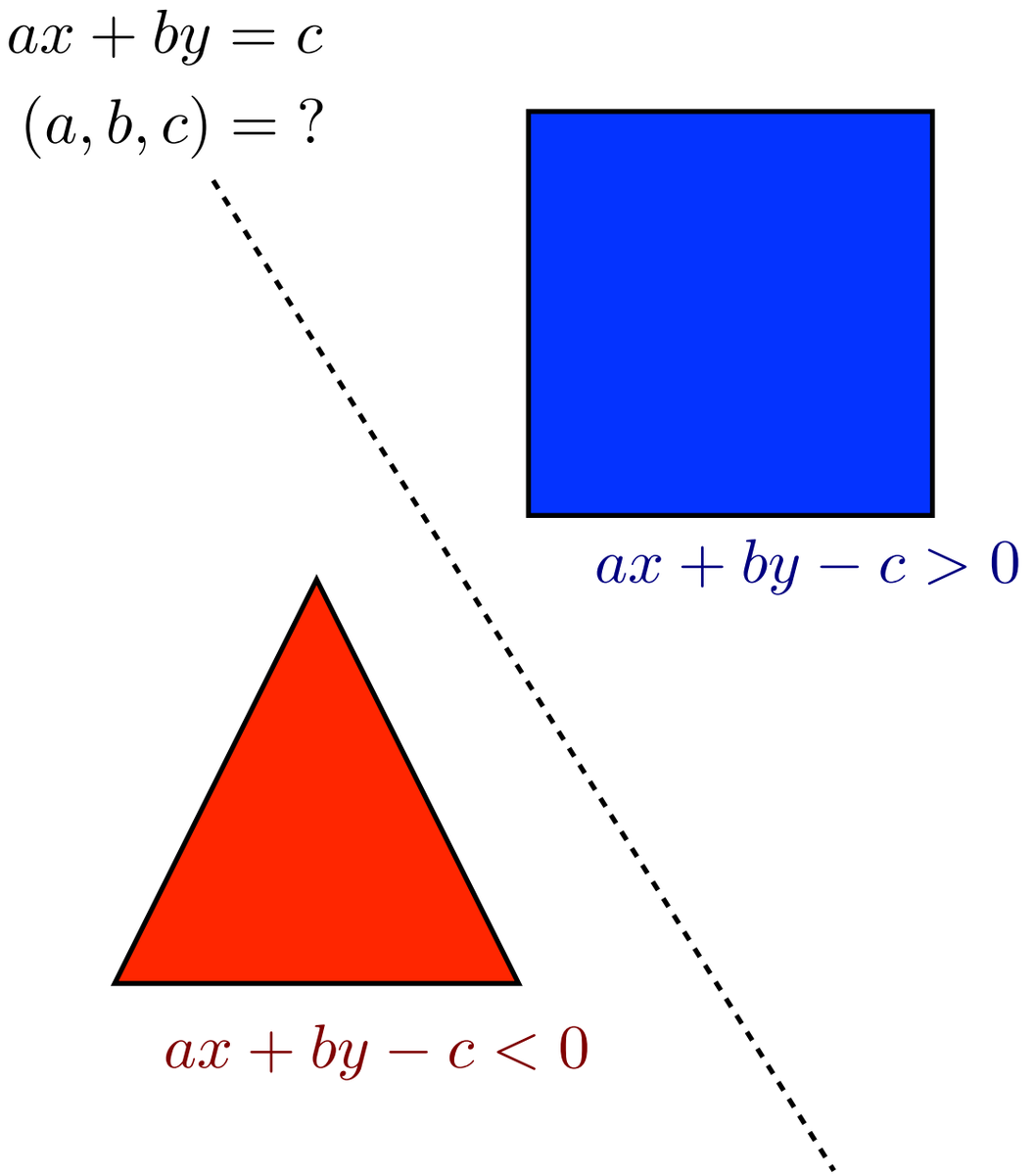
- Subject to the linear constraints

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$

- where **x** is a vector of *n* unknowns, **c** is a vector of coefficients, and **A** and **b** are constraints which define an *n*-dimensional convex polytope
- Can detect infeasibility

# Linear Programming

- Four coefficients of the separating plane are linear programming variables
- Constrain all vertices of polyhedron A to one side of plane and vertices of polyhedron B to the other
- LP tells us whether or not a separating plane exists
- Expected linear time

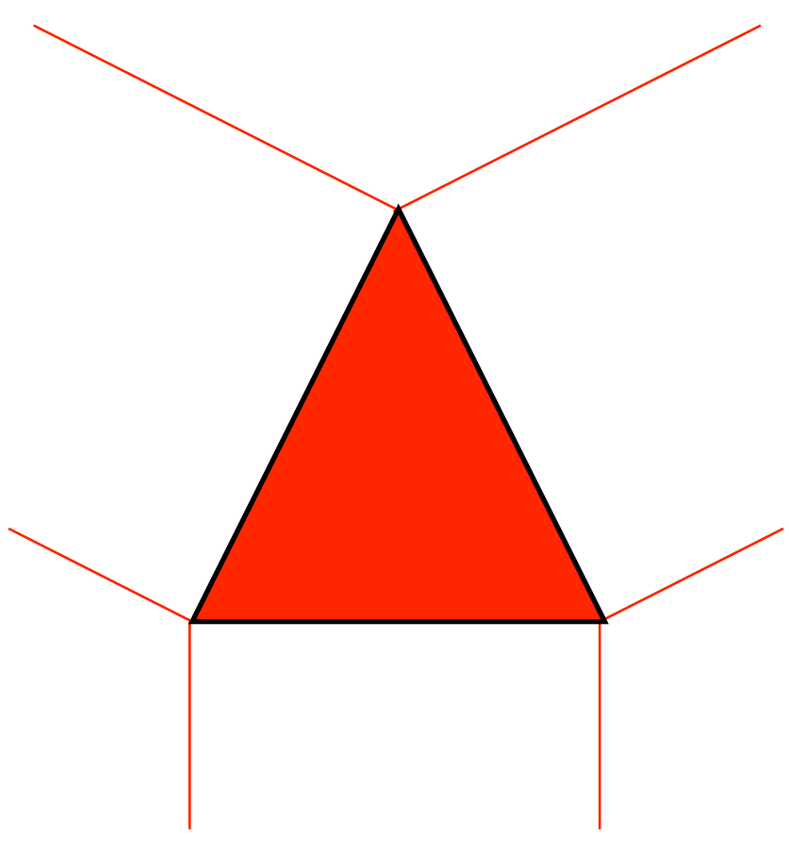


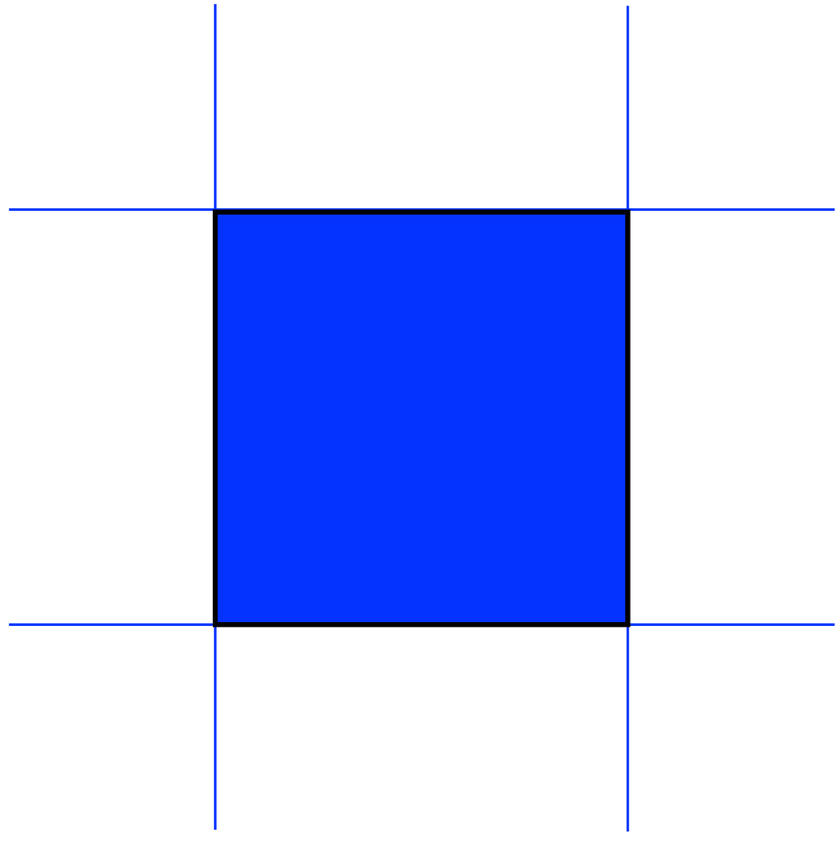
# Closest Feature Tracking

- Proposed by Lin & Canny (1991)
- Observe that when contact occurs, it will occur between the two closest points
  - Two contact configurations provide features that determine the closest points
  - Can use *Voronoi regions* to track the closest features on non-overlapping convex polyhedra

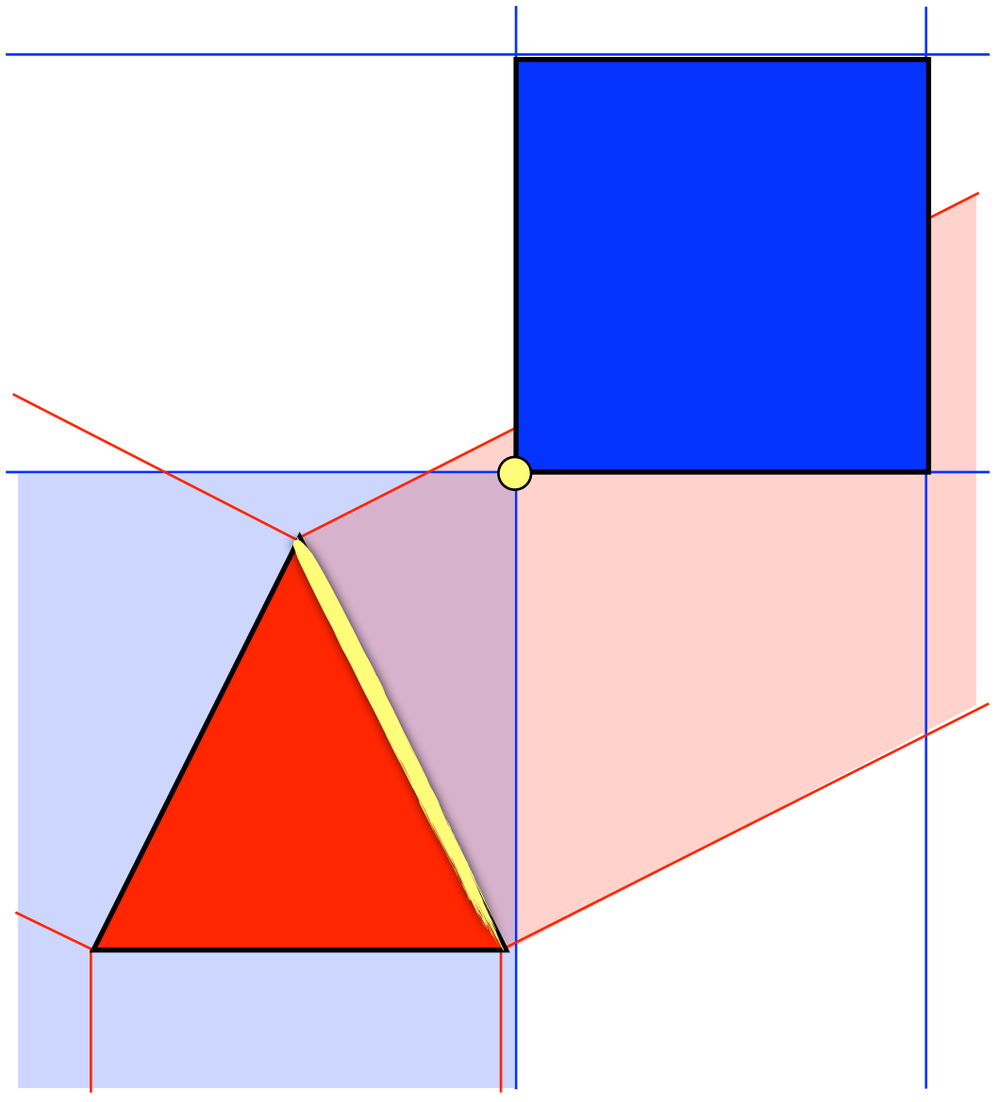
# Voronoi Regions

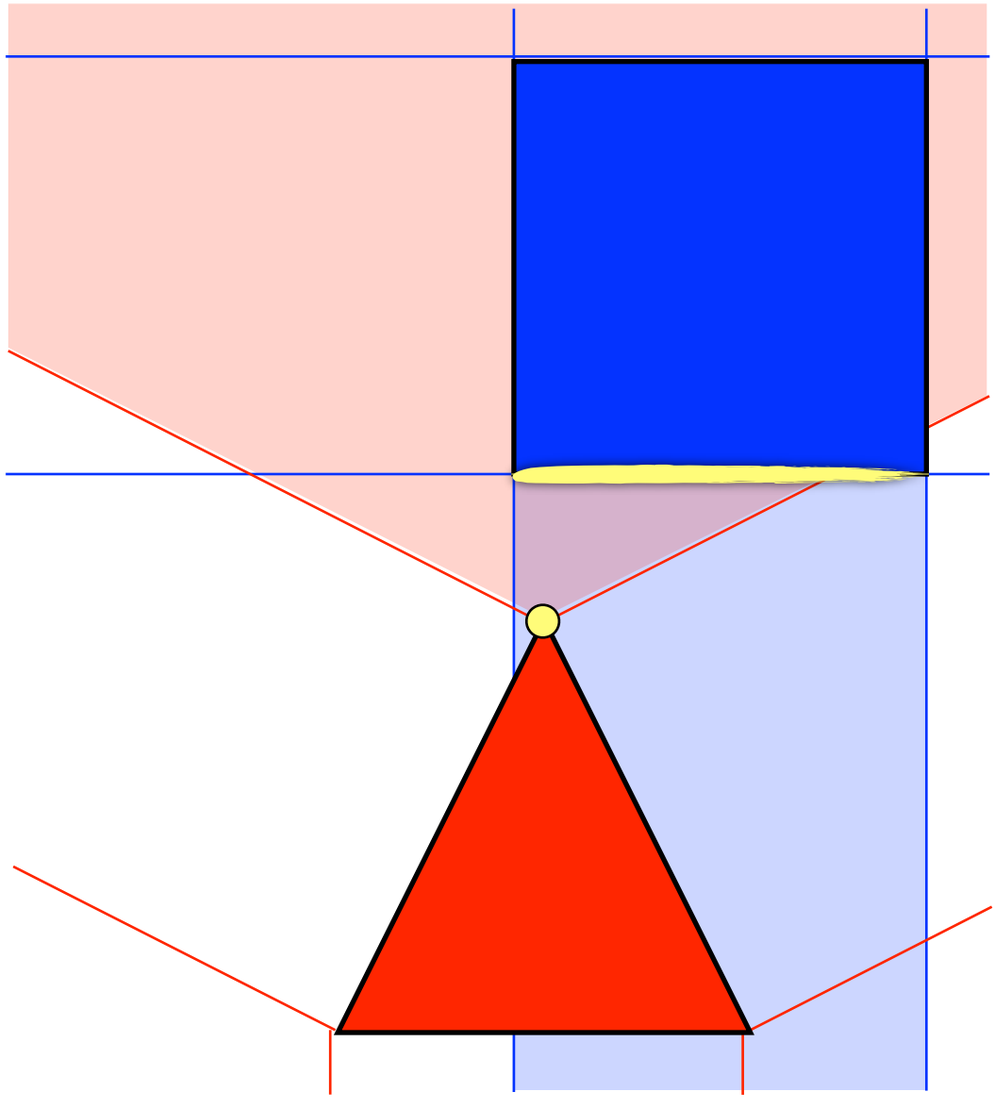
- Defines a convex region of space for each feature containing the set of points closer to it than any other feature





# Feature Tracking





Walk along outside as objects change position...

# Lin-Canny Algorithm

- Computes minimum separation distance between closest pair of features
- Takes advantage of *temporal coherence*
- What is the expected running time of this algorithm?

# Lin-Canny Algorithm

- Performance can be near **constant** time once the tracking is initialized
- Can be quite difficult to implement!

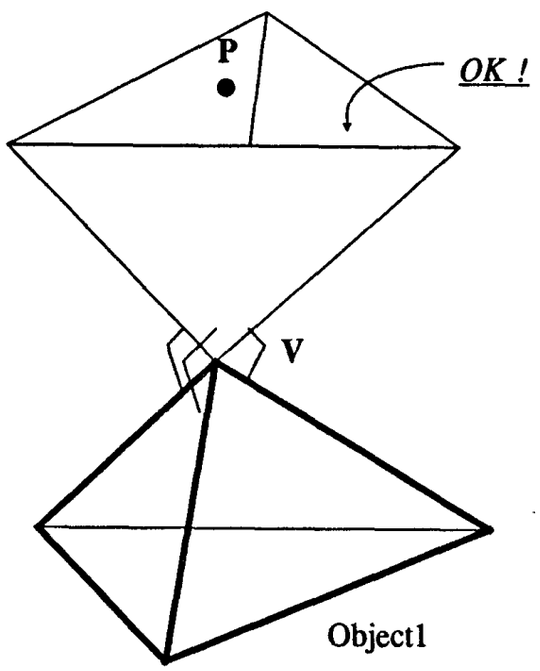



Figure 1: Point-Vertex Applicability Criterion

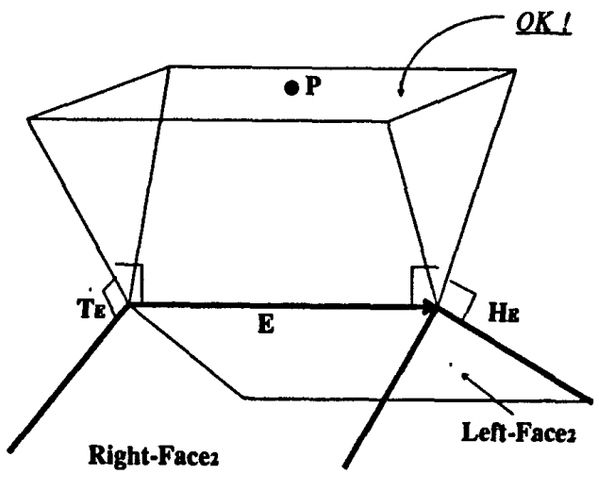



Figure 2: Point-Edge Applicability Criterion

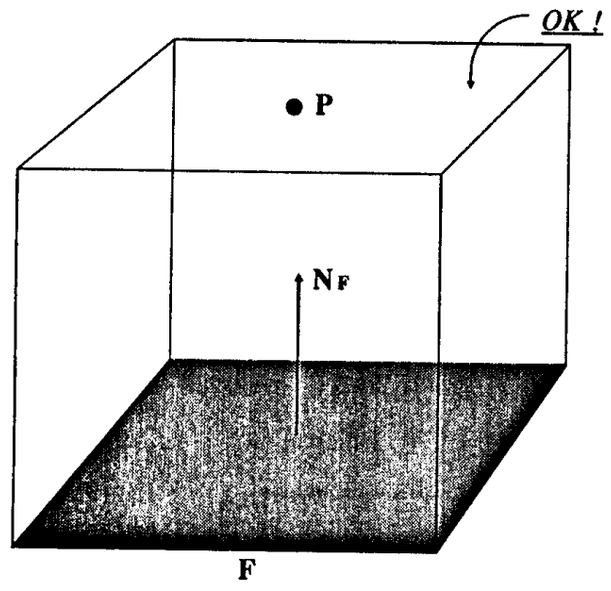



Figure 3: Vertex-Face Applicability Criterion

[from M. Lin & J. F. Canny, *Proc. IEEE Intl. Conf. on Robotics and Automation*, 1991.]

# Primitive Test Summary

- We covered fast intersection tests for:
  - Segment-triangle
  - Triangle-triangle
  - Sphere-sphere
  - Axis-aligned bounding box (AABB)
  - Oriented bounding box (OBB)
  - Convex polyhedra
- Look up others if needed!

# Summary

- We can test collision between two objects, but what happens if there are thousands?