# Lecture 12

# Collision detection and handling

# Projects

# Time to consider what to do!

# 2D collision detection

**Even in 2D, collisions is a non-trivial problem.**

- **Rectanges - easy**
- **Enclosing convex polygon(s)**
- **Pixel-level testing**

# Most fundamental collision tests

## Rectangles (axis aligned bounding boxes)

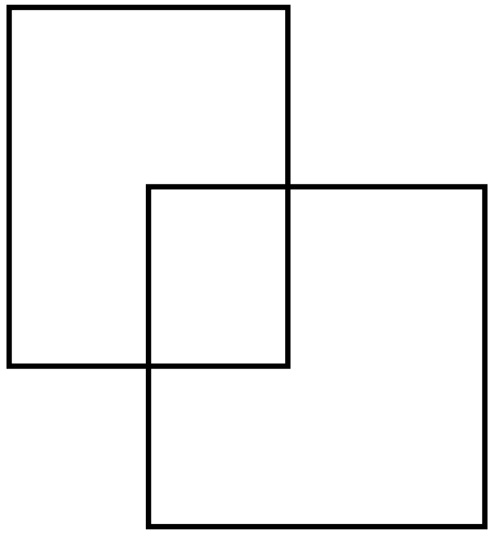
Two tests along each axis

a.left > b.right

b.left > a.right

a.top > b.bottom

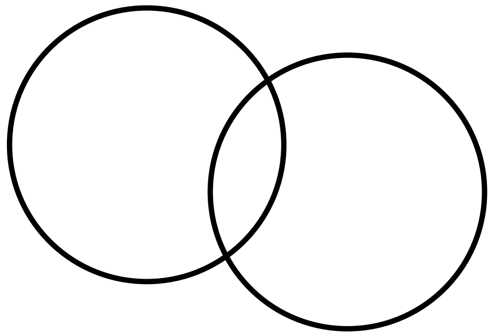
b.top > a.bottom



## Circles/Spheres

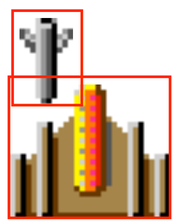
Sum of radii compared to distance between centers

$r_1+r_2 > \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2}$

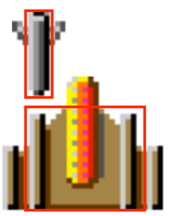


# 2D collision detection with rectangles

**Testing with the bounding box: often unsatisfactory. Too crude approximation!**



**Smaller rectangles make decent compromises.**



# Special cases

**When using pseudo-3D, sprites in perspective, the shape outline is not usable.**

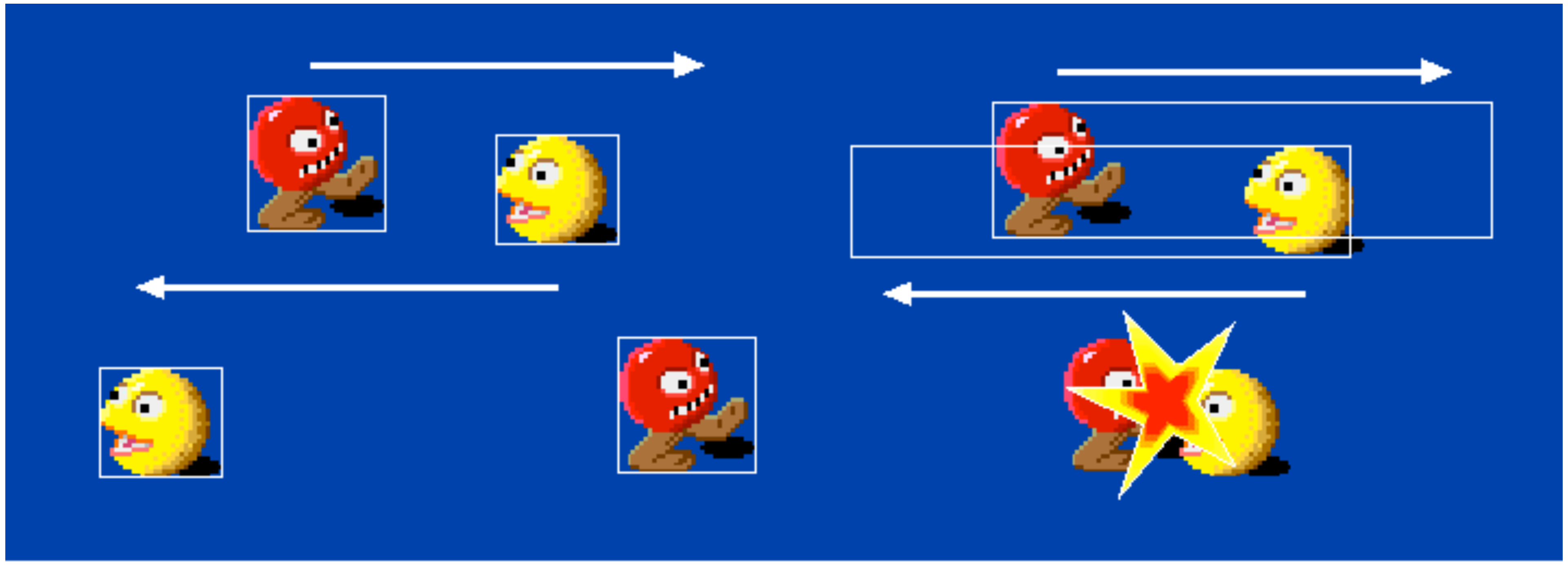
**Simple approach: Modified box:**

Wrong

Right

# Moving objects

**Fast moving objects may pass right through each other.**



Wrong Right

**An example of *temporal aliasing***

# Fast-moving objects

- Test with elongated shapes (sweeping)
- Test several times along the trajectory (multisampling)

# In 3D:

# Polyhedra-polyhedra collisions

## Naive method: Check all polygons in all objects against all polygons in all objects!

# Polyhedra-polyhedra collisions

**Practical methods:**

**1) Separate into broad phase and narrow phase**

**2) Simplified shape(s) matching a complex shape**

# Broad phase - narrow phase

**Really three phases:**

**Global phase**

**Bounding shapes phase**

Broad phase

**Narrow phase**

# Bounding shapes phase

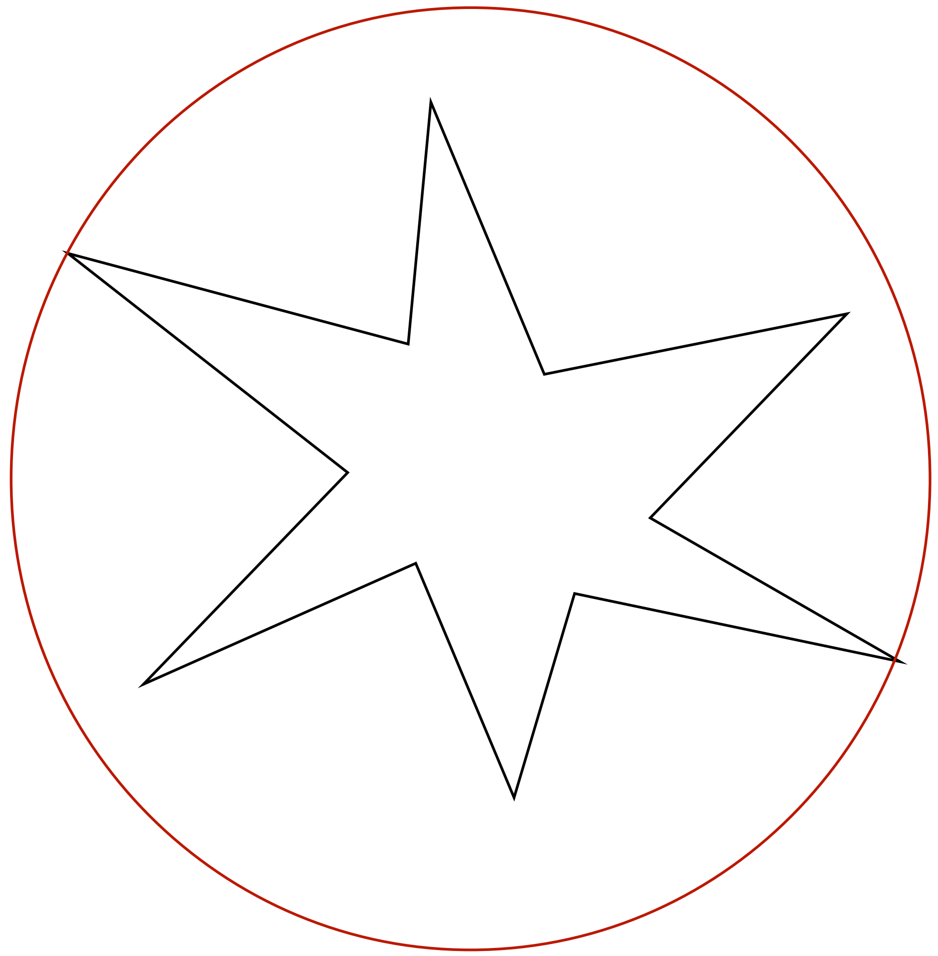
**Make simple checks to see if objects are so close that we should test in detail!**

**Use simple bounding shapes!**

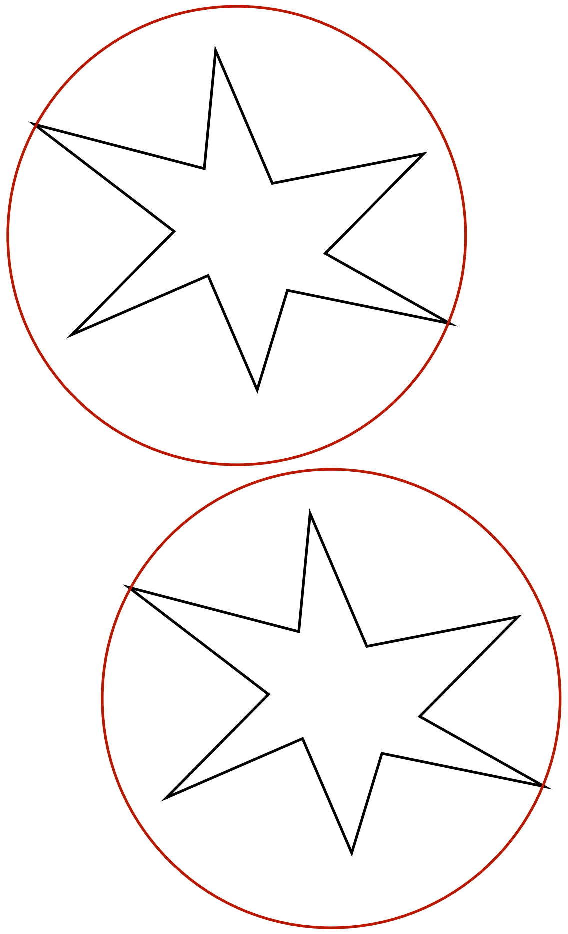
**Typical shapes:**

- **Sphere**
- **AABB (axis aligned bounding box)**
- **OBB (oriented bounding box)**

# Sphere

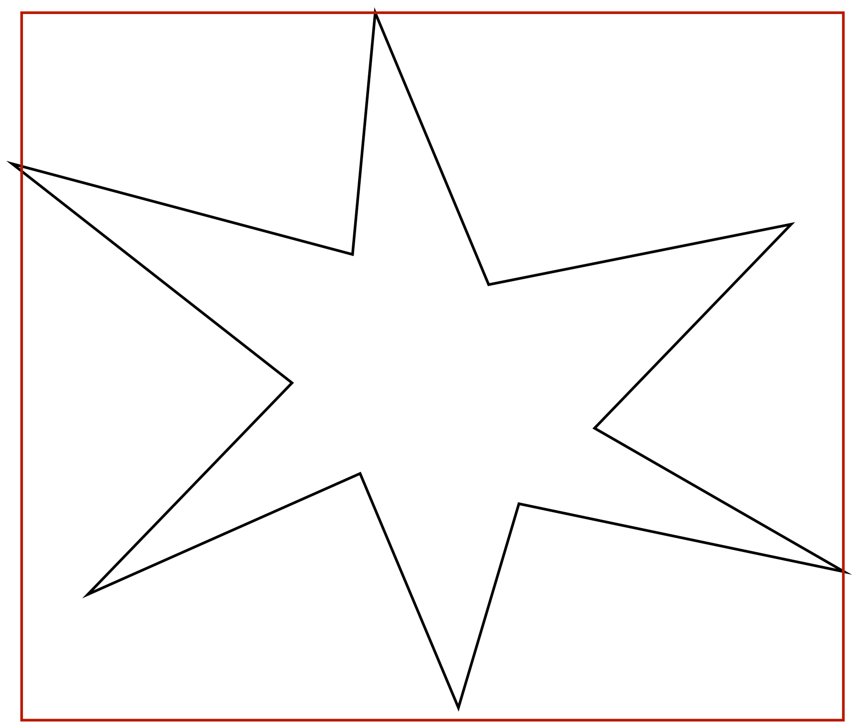


# Sphere

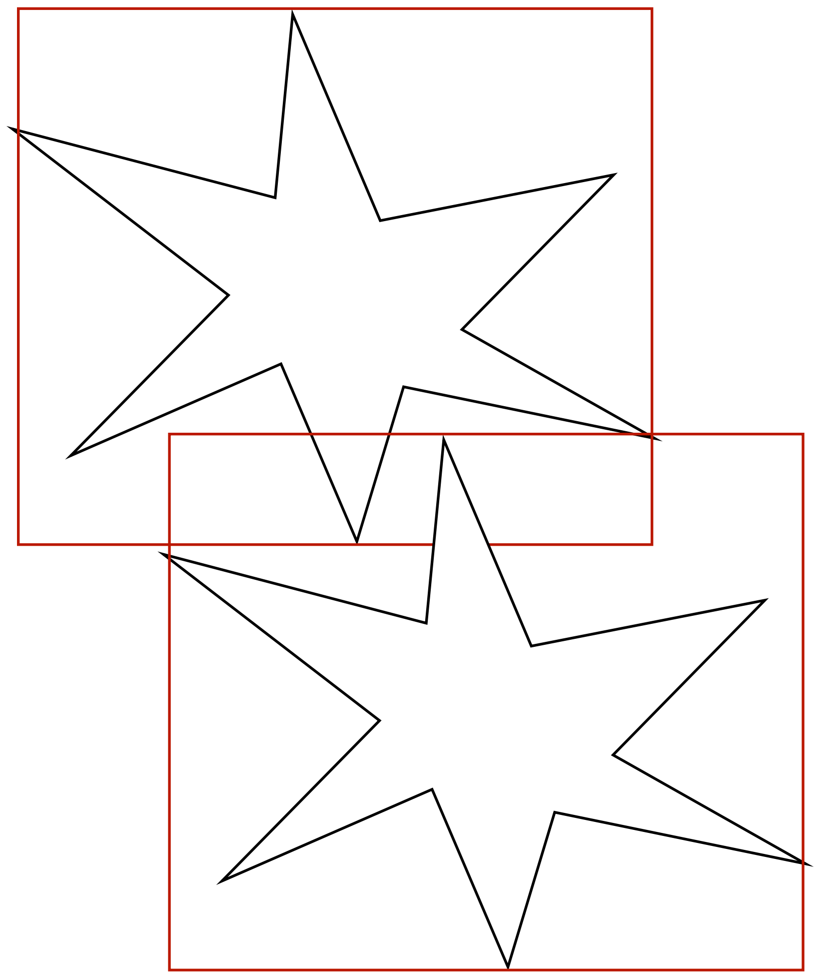


- **Very simple tests. $d^2 < (r_1 + r_2)^2$**
- **Calculate radius once and for all. Never changes for rigid objects. Loop through all vertices, pick biggest distance to center.**
- **Rotation allowed!**
- **Good fit for compact objects.**
- **No flat surfaces, object can not rest on each other.**

# AABB



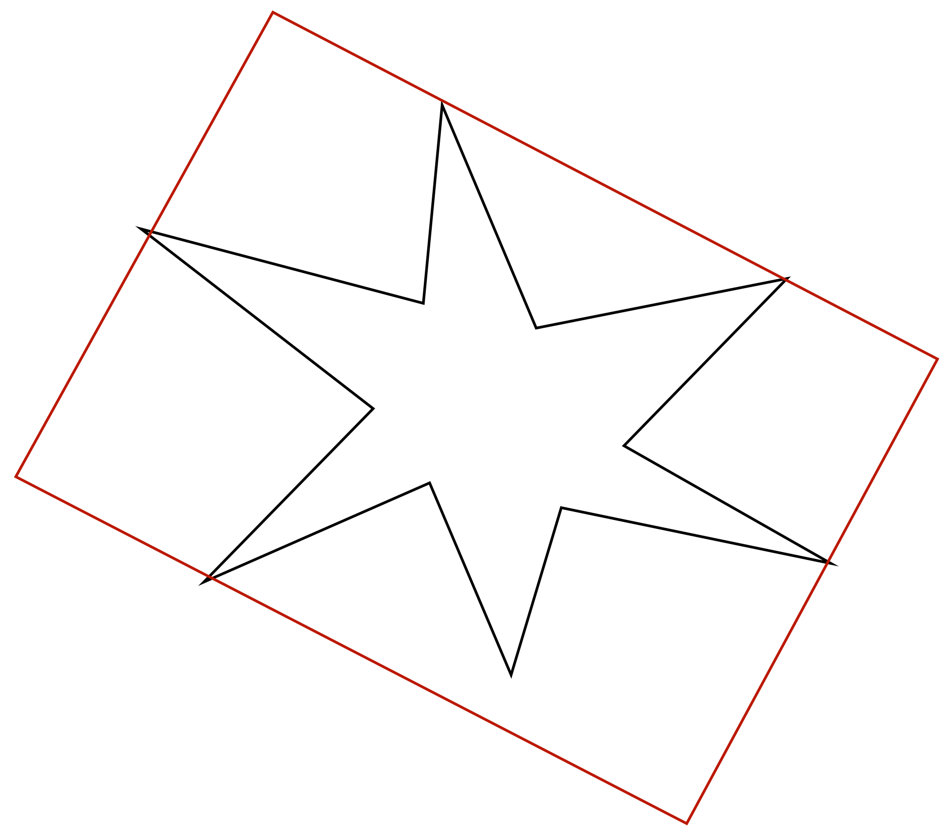
**Axis aligned bounding box**



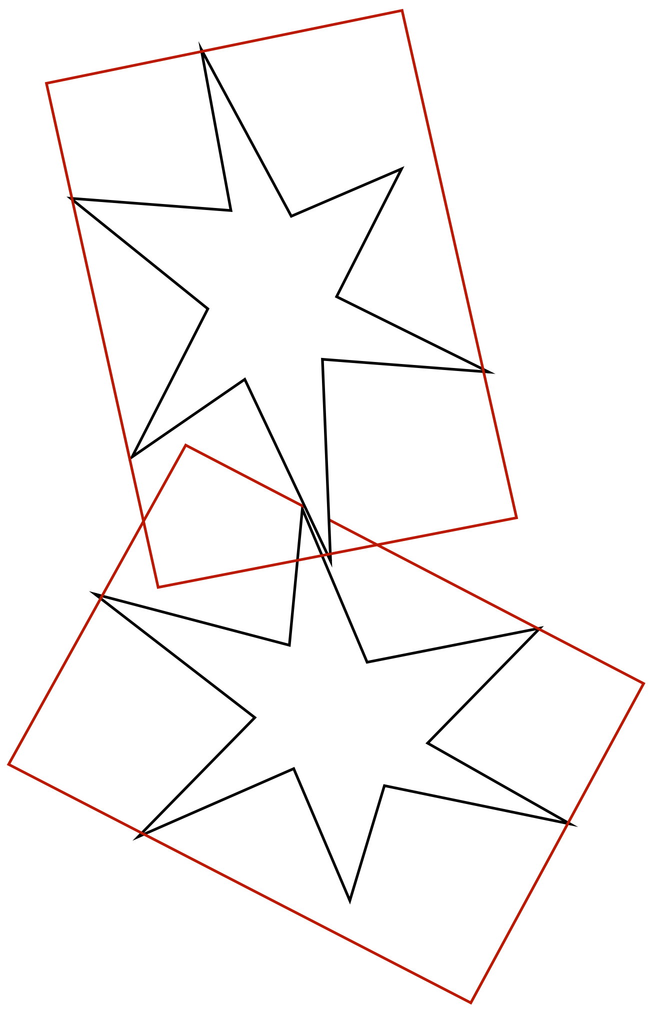
# AABB

- Very simple tests. 6 tests, like the 2D case.
- Calculate by finding max and min along each axis.
- Can't be rotated freely!
1) Recalculate on rotation.
2) Make new AABB from rotated vertices of AABB.
- Not very good fit on many objects.

# OBB
# Oriented bounding box

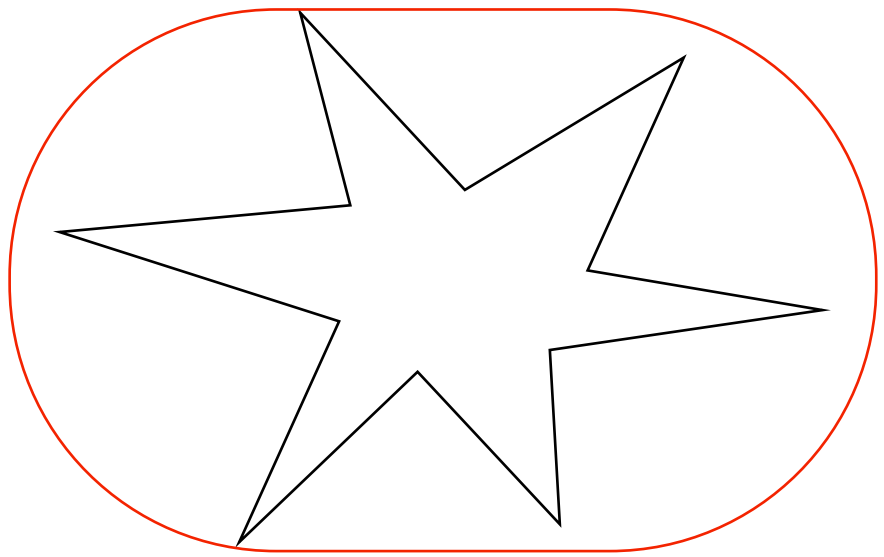


# OBB
# Oriented bounding box

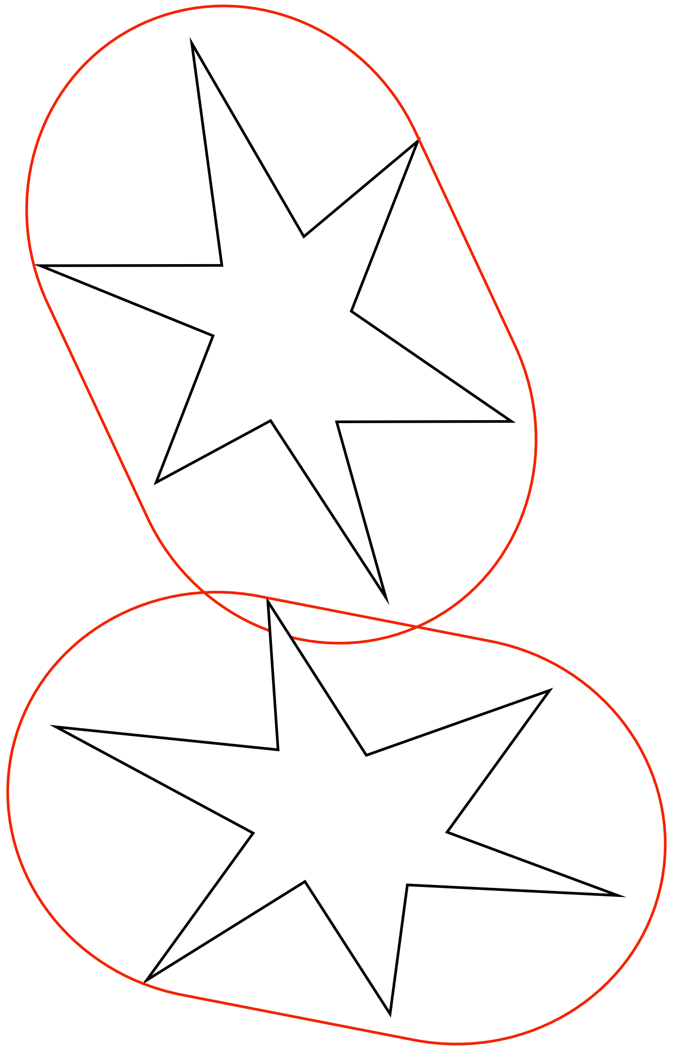


- **Good fit, you can find smallest possible box. Few false hits.**
- **Complicated tests.**
- **Rotation allowed.**

# Capsule

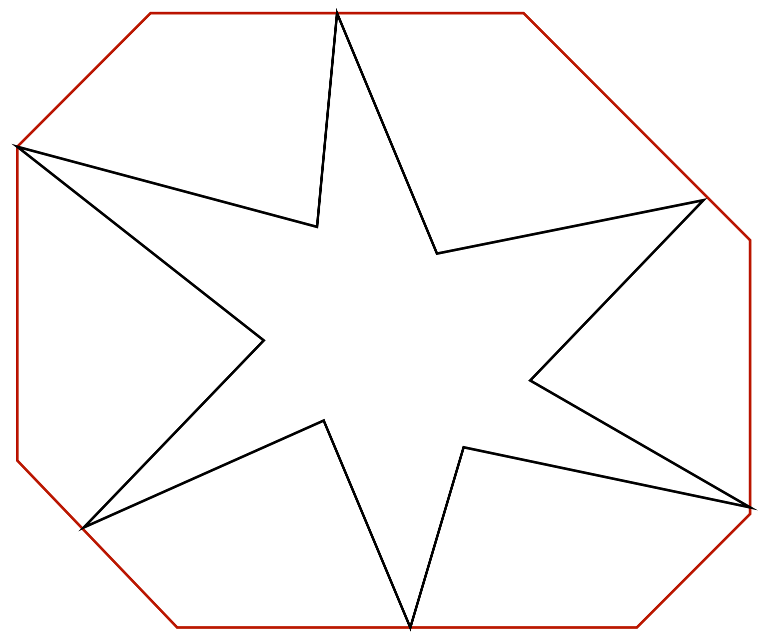


# Capsule
## Cylinder + 2 half spheres



- Good fit, you can find smallest possible box. Few false hits.
- Fairly complicated tests - but easier than OBB.
- Rotation allowed.

# k-DOP
## Discrete oriented polytope with k sides

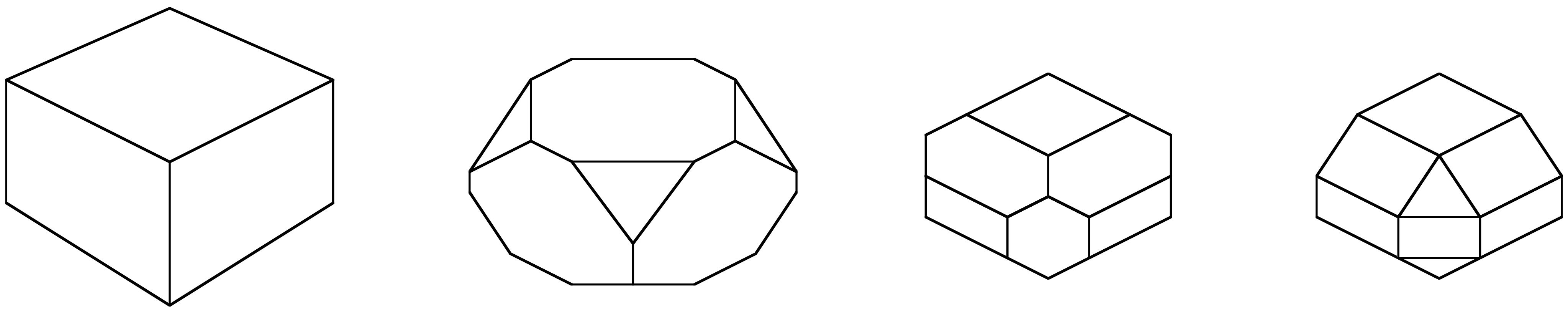


**Like AABB but you can cut off corners and sides**
**Fairly simple tests**
**Fewer false hits**

# k-DOP
## Discrete oriented polytope with k sides

**Four kinds:**

**6-DOP (= AABB)**
**14-DOP (corners)**
**18-DOP (sides)**
**26-DOP (sides and corners)**



**6-DOP** **14-DOP** **18-DOP** **26-DOP**

# Balancing the broad and narrow phases

$$T = N_v C_v + N_p C_p$$

Simple broad phase - many false hits - high $N_p$.

Elaborate broad phase - few false hits - lower $N_p$ but high $C_v$.

# Narrow phase

## Separating Axis Theorem (SAT)

## Containment/intersection test

## Volume intersection

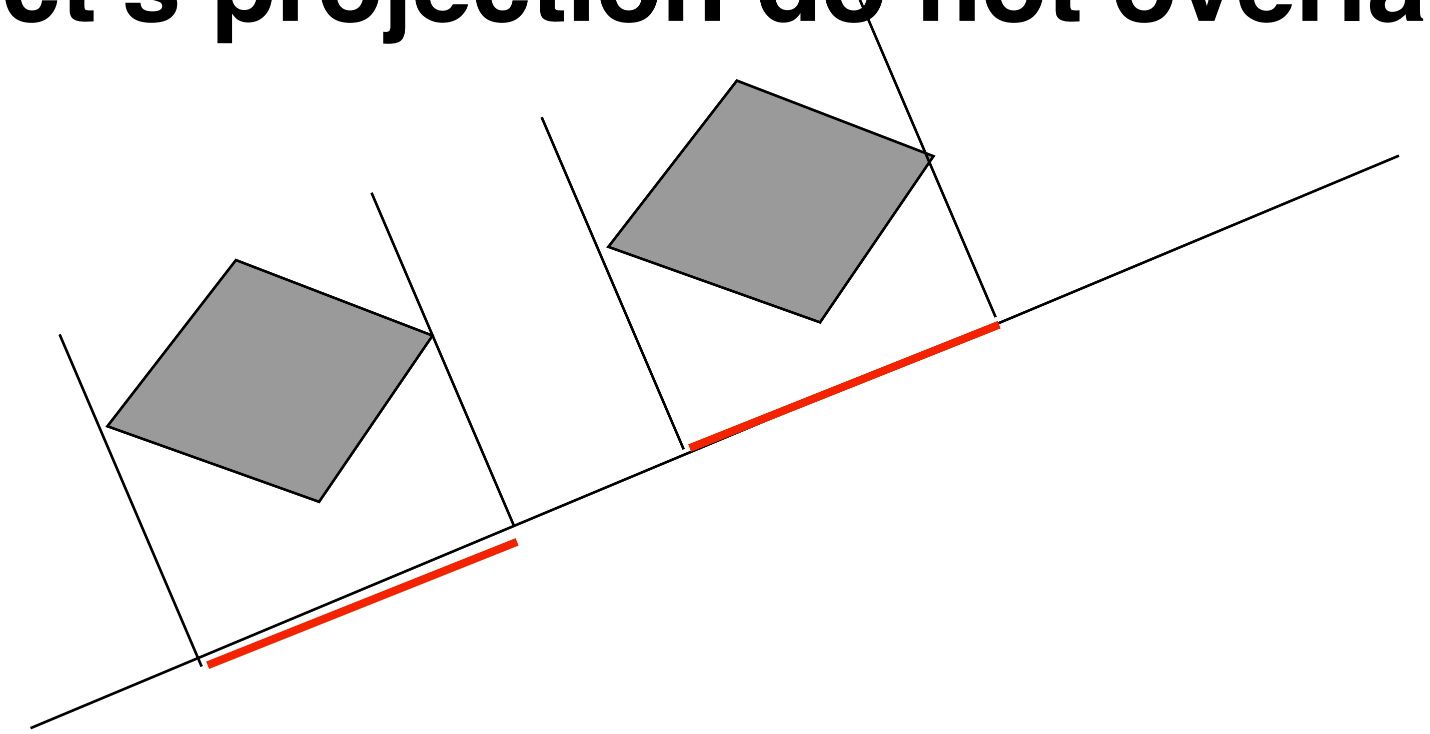
## Advanced methods: GJK

# Separating axis theorem (SAT)

## and its application on collision detection

# Separating axis theorem

**Two convex objects do *not* overlap if there exists a line (axis) onto which the two object's projection do not overlap**



# Really ”Separating plane”

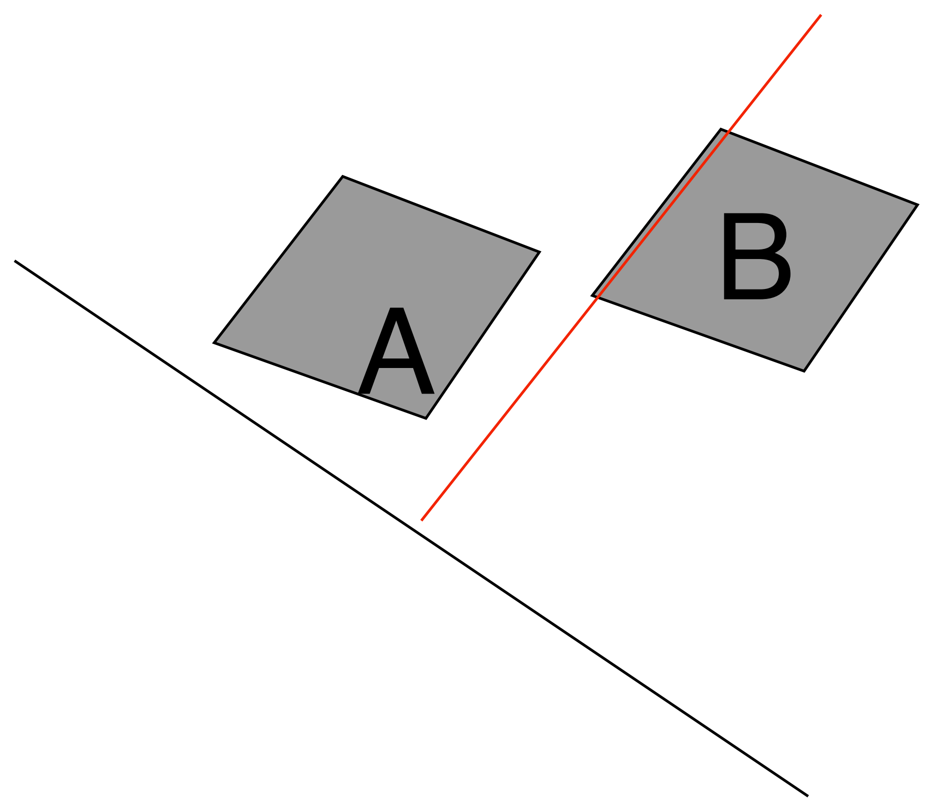
**Axis = normal vector of separating plane**

**Can you find a plane between them?**

**Problem: The number of possible planes/ axes to test are *infinite*!**

# SAT in practice

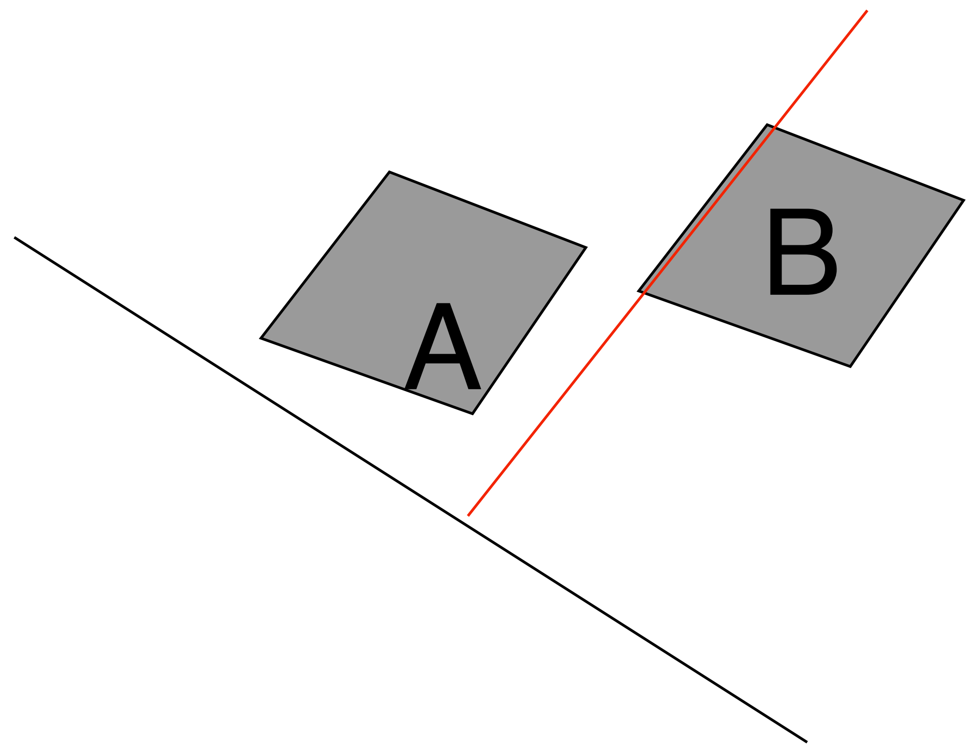
**For polyhedra models, use the *faces* of the models!**



**Fairly simple in the 2D case**

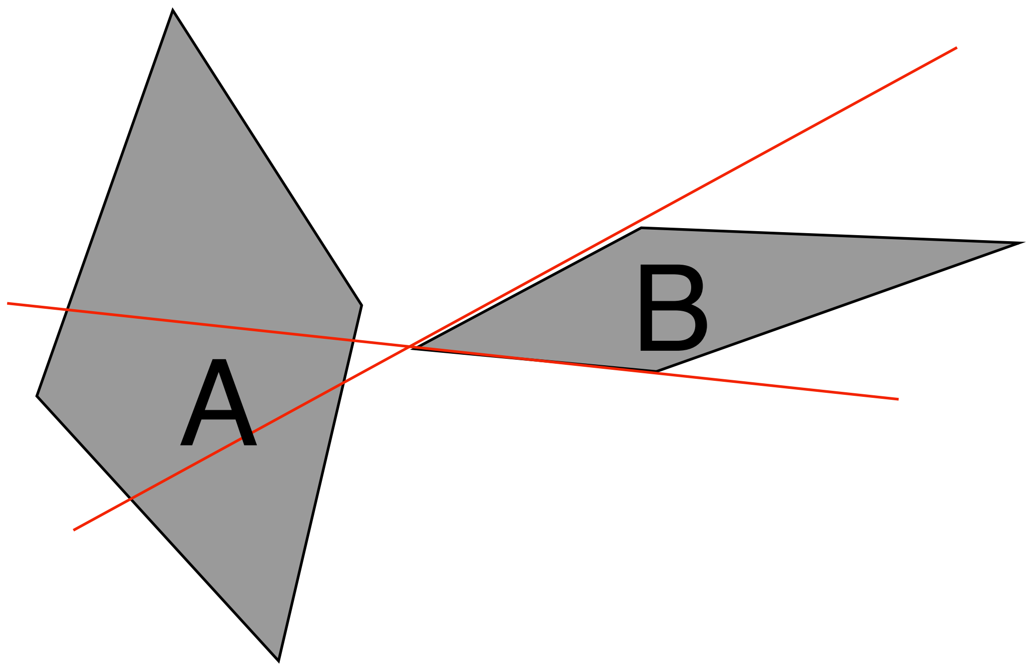
**For every face in B, find plane equation**

**Test all vertices in A with dot product!**



# But you must go both ways!

**There are configurations where one model has no single face that will exclude the other!**



# 2D SAT algorithm

**for all faces in A**
**let a be a vertex in A**
**hit = false**
**for all vertices b in B**
**diff = n • a - n • b**
**if diff > 0 then**
**hit = true**
**if not hit then**
**return false**

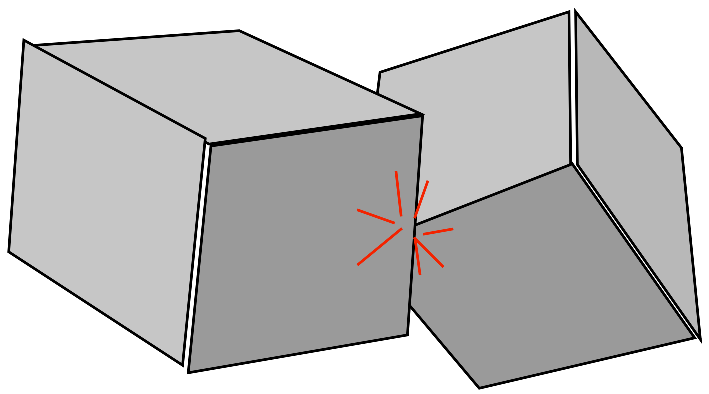
**for all faces in B**
**(same algorithm with A and B reversed)**

**return true**

# SAT in 3D

**Not as simple!**

**Two object meeting at an edge may fail the test!**



**We must add tests with more planes.**

# SAT in 3D

**We must add tests with more planes. Which ones?**

**Cross product of an edge in a with an edge in b = additional potential separating plane**

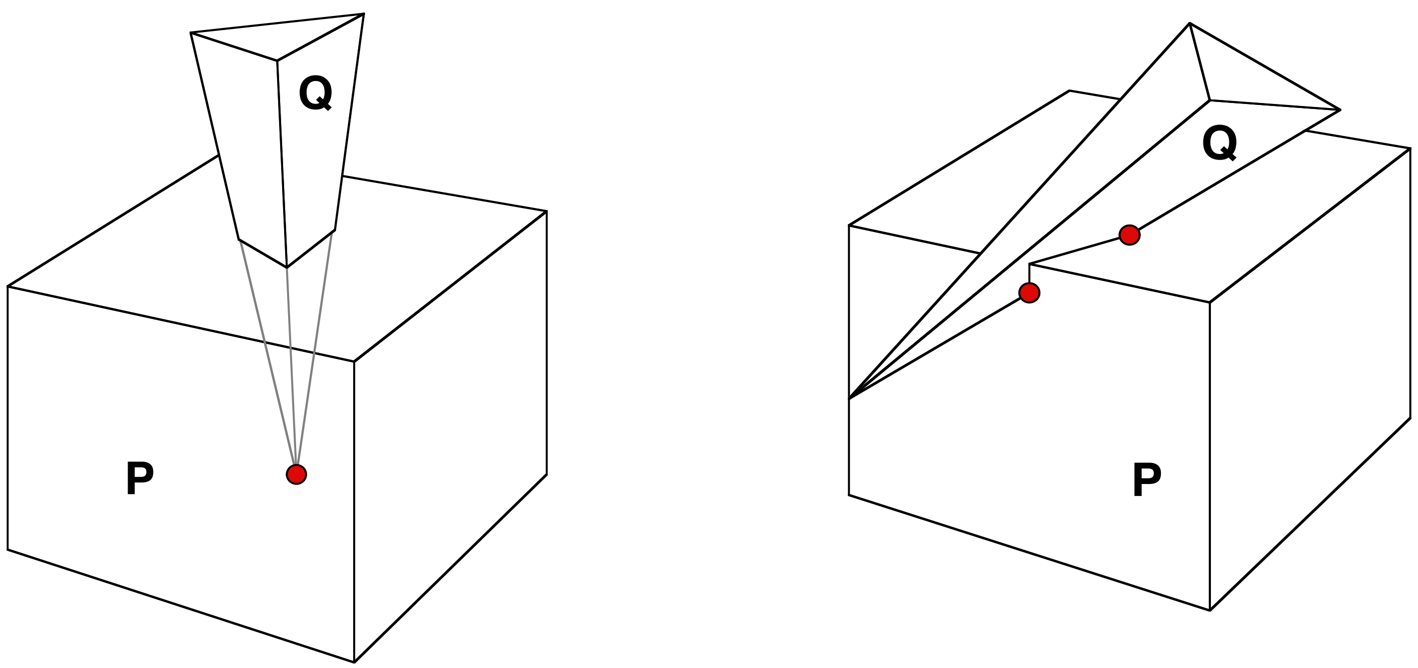
**All edges in a * all edges in b!**

**=> Complete SAT on complex 3D objects is *expensive!***

# So how can SAT be useful?

- Simplify. Skip some tests, accept some false hits.
- Use SAT on simplified bounding shapes.
- Remember separating axis from previous test!

# Containment/intersection



# Containment/Intersection

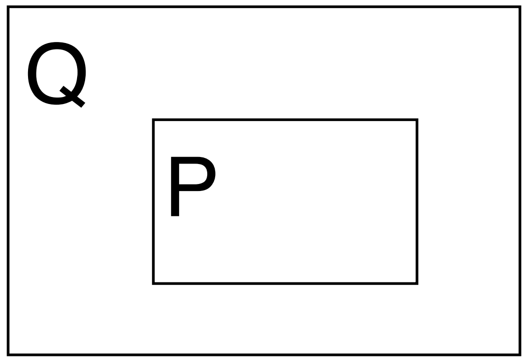
**Exact test between two polyhedra, Q and P.**

**Two tests are needed:**

**1) Is any vertex in Q contained in P or vice versa?**
**2) Does any edge of Q penetrate a face of P?**

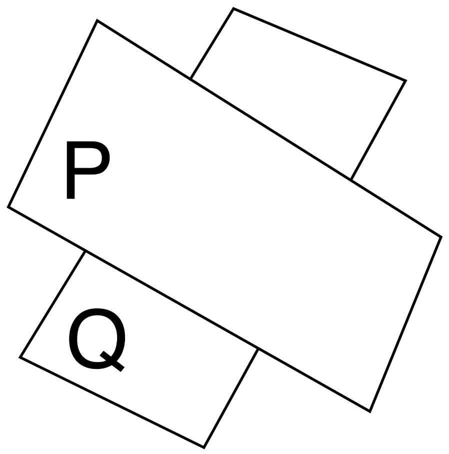
# The need for all tests

1) in both directions:



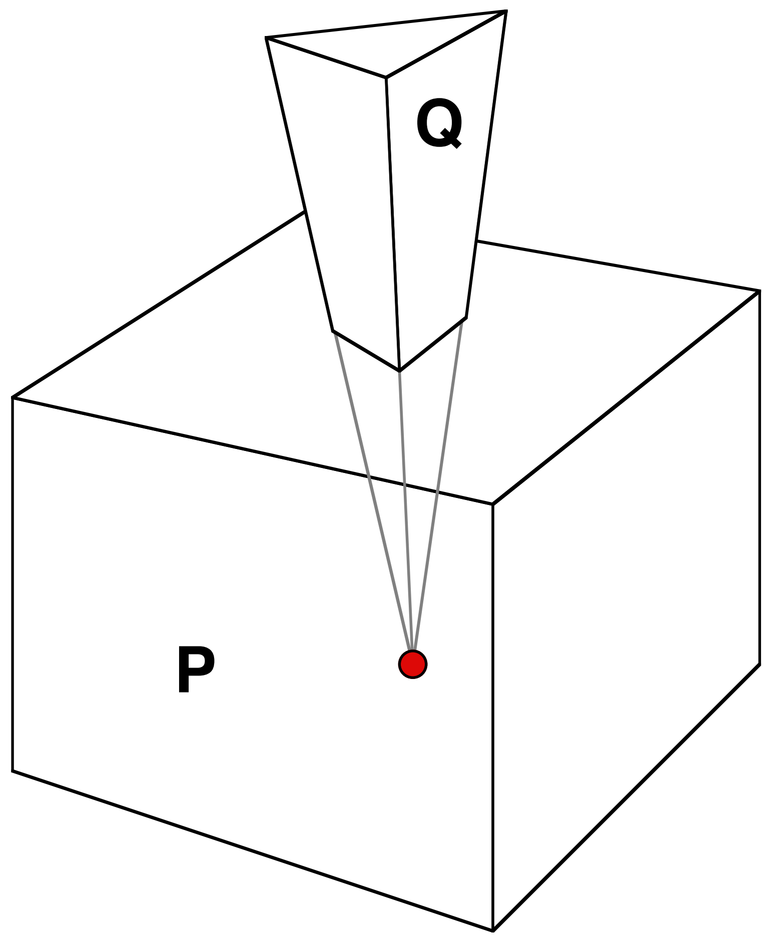

Q contains points in P, P contains no points in Q

2) when 1) fails:



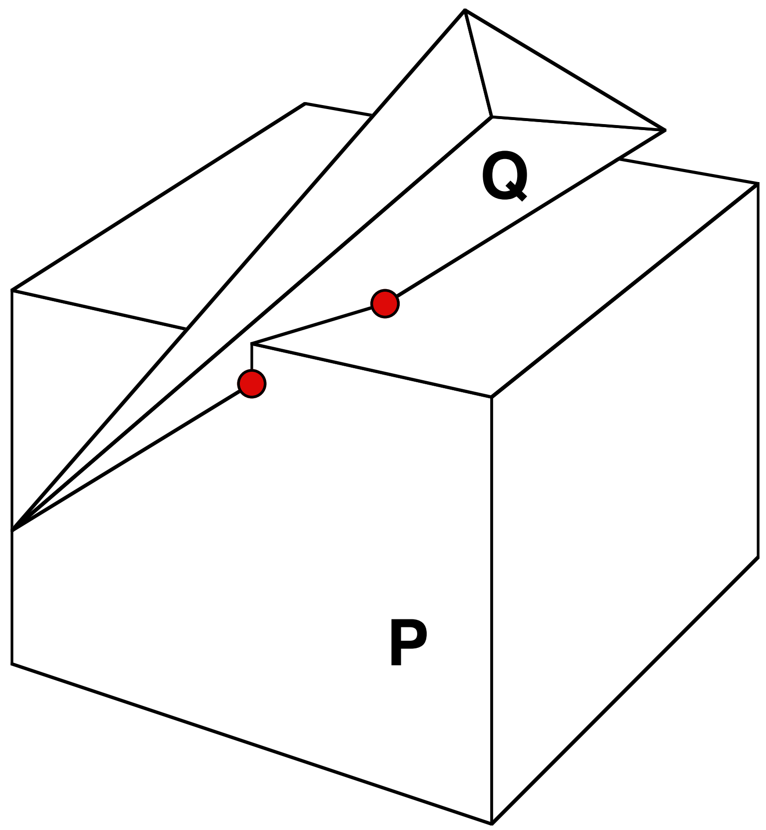

Neither P nor Q contain points of each other, but still overlap!

# Test 1:




**A point in Q is inside P**

# Test 2:

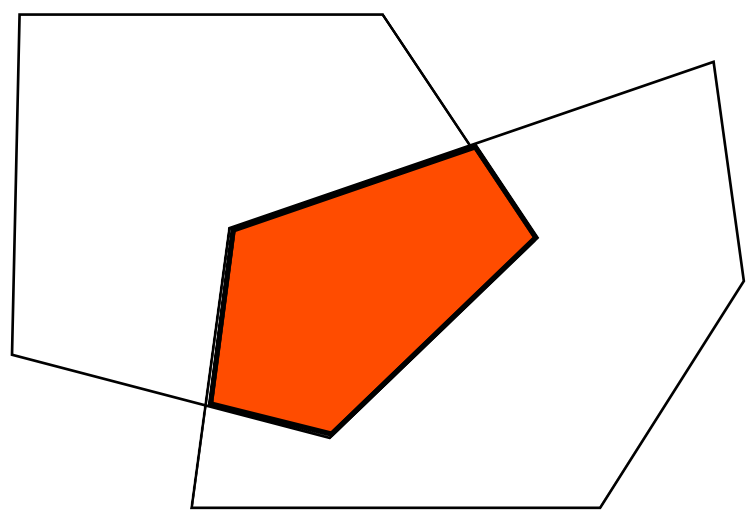



**An edge of Q intersects sides of P**

# Volume intersection

**Split P by any plane in Q, repeat. When all planes have been used, the intersection has been calculated.**

**Can be fairly fast if a good first plane is chosen.**



# Comparison

**Full SAT**

$V_p*P_q+P_p*V_q + Ep*Eq*Vp + Ep*Eq*Vq$ O($N^3$)

**Containment/intersection:**

$V_p*P_q+P_p*V_q + E_p*P_q+P_p*E_q$ O($N^2$)

**Volume intersection:**

$\sum k^i Pq \approx P_q \log P_p$ O(NlogN)

**where i = 0 to $P_p$ and k < 1.**

# Advanced methods

**Volume intersection can sometimes simplify by starting ”right”.**

**Advanced methods take this even further. Remember last time, reduce test to very few. Just test the ”right test”.**

# Problem: Convex shapes only!

**Solution:**

**1) Use the convex hull**

**May be very different from original**

**2) Decompose objects into convex parts.**

**Impossible/impractical for many shapes.**

# HACD

## Hierarchical Approximate Convex Decomposition