Collision Detection

What is Collision Detection?

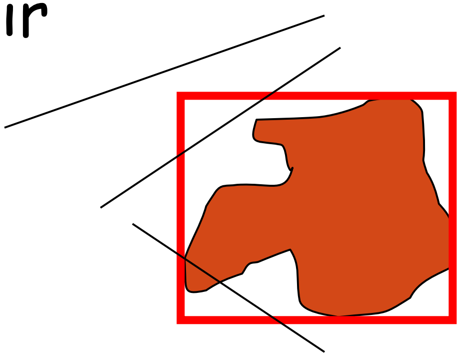
- Given two geometric objects, determine if they overlap.
- Typically, at least one of the objects is a set of triangles.
  - Rays/lines
  - Planes
  - Polygons
  - Frustums
  - Spheres
  - Curved surfaces

When is it useful?

- Often in simulations
  - Objects moving/deforming – find when they hit something else, (then react)
- Other examples
  - Ray tracing speedup
  - Culling objects in regions
- Usually, needs to be fast
  - Applied to lots of objects, often in real-time applications

Bounding Volumes

- Key idea:
  - Surround the object with a (simpler) bounding object (the bounding volume).
  - If something does not collide with the bounding volume, it does not collide with the object inside.
  - Often, to intersect two objects, first intersect their bounding volumes

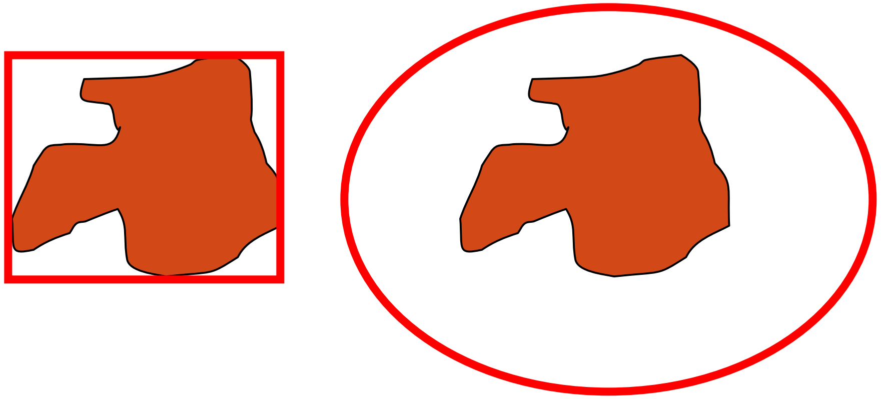


Choosing a Bounding Volume

- Lots of choices, each with tradeoffs

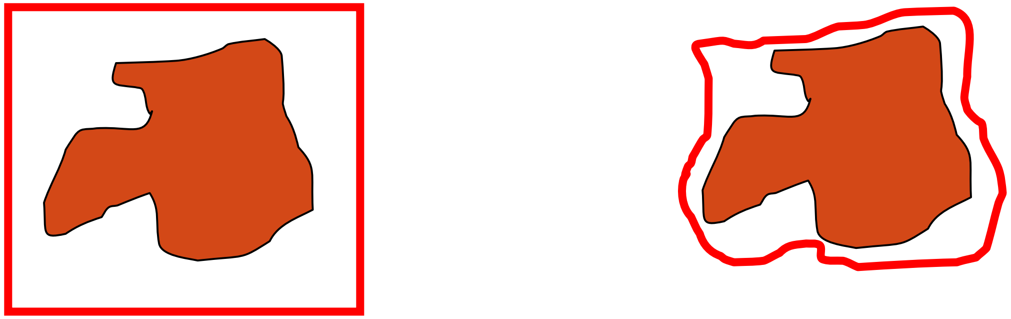
Choosing a Bounding Volume

- Lots of choices, each with tradeoffs
- Tighter fitting is better
  - More likely to eliminate "false" intersections



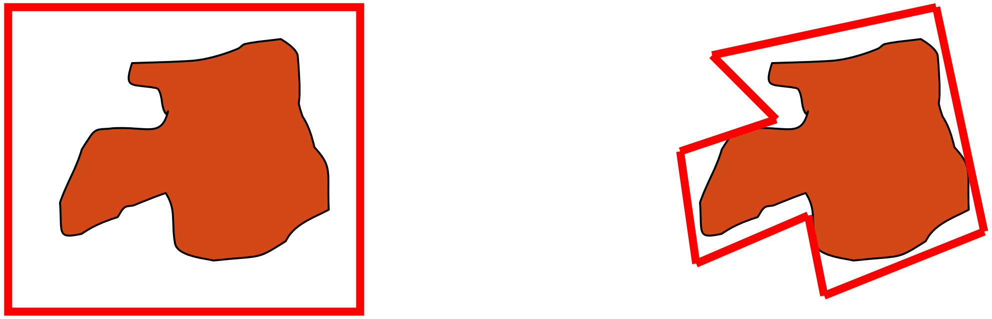
Choosing a Bounding Volume

- Lots of choices, each with tradeoffs
- Tighter fitting is better
- Simpler shape is better
  - Makes it faster to compute with



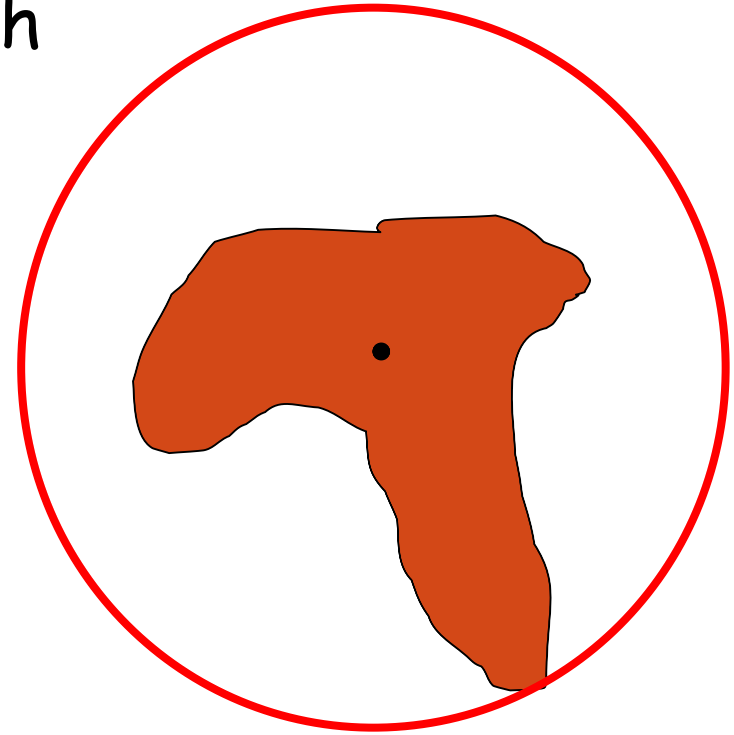
Choosing a Bounding Volume

- Lots of choices, each with tradeoffs
- Tighter fitting is better
- Simpler shape is better
- Convex is usually better
  - Gives simpler shape, easier computation



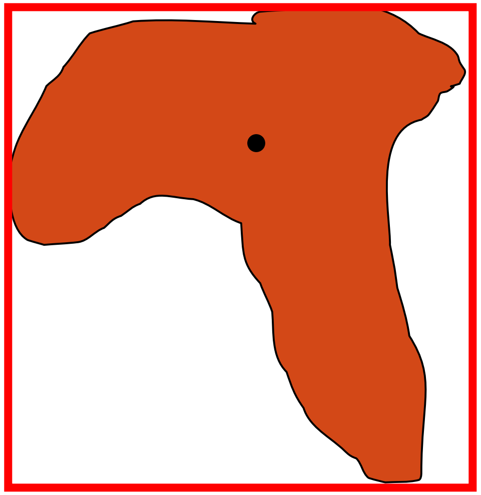
Common Bounding Volumes: Sphere

- Rotationally invariant
- Usually fast to compute with
- Store: center point and radius
  - Center point: object's center of mass
  - Radius: distance of farthest point on object from center of mass.
- Often not very tight fit



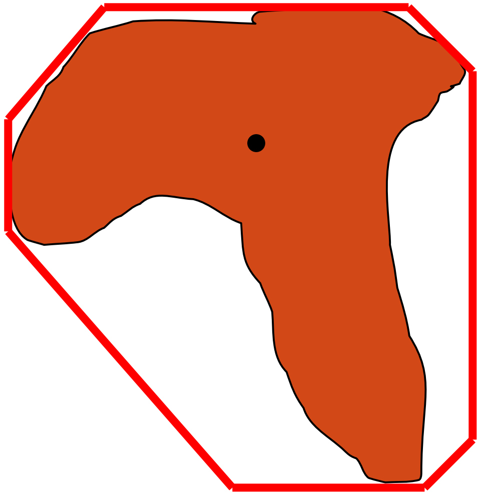
Common Bounding Volumes: Axis Aligned Bounding Box (AABB)

- Very fast to compute with
- Store: max and min along x,y,z axes.
  - Look at all points and record max, min
- Moderately tight fit
- Must update after rotation
  - Unless: using a loose box that encompasses the bounding sphere → invariance to the object's global rotation



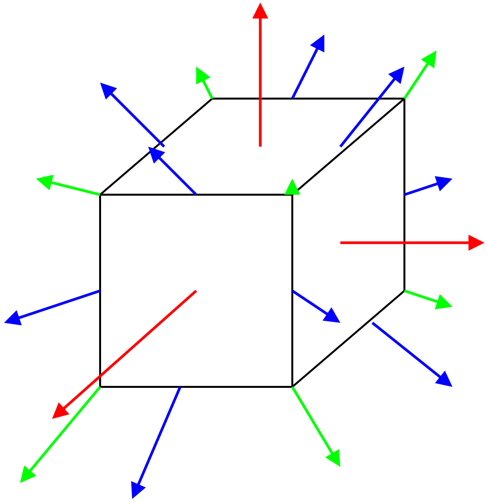
Common Bounding Volumes: k-dops

- k-Discrete Oriented Polytopes
- Same idea as AABBs, but use more axes.
- Store: max and min along fixed set of axes.
  - Need to project points onto other axes.
- Tighter fit than AABB, but also a bit more work.



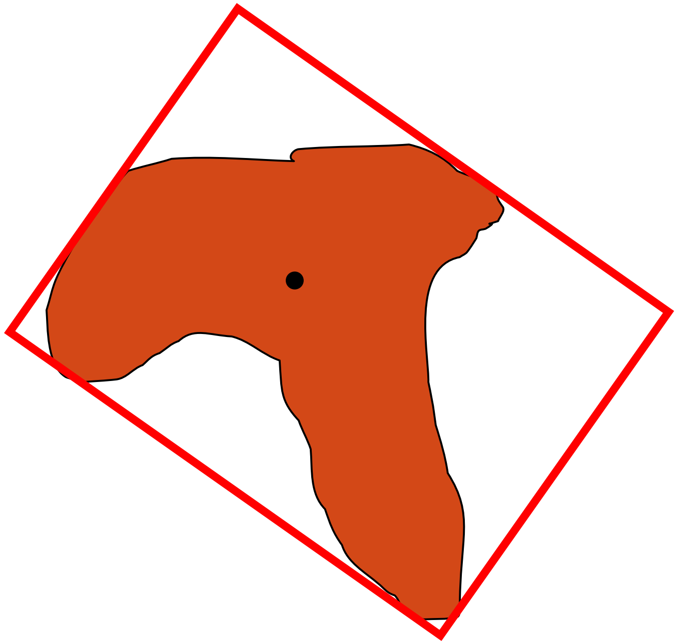
Choosing axes for k-dops

- Common axes: consider axes coming out from center of a bounding cube:
- Through faces: 6-dop
  - same as AABB
- Faces and vertices: 14-dop
- Faces and edge centers: 18-dop
- Faces, vertices, and edge centers; 26-dop
- More than that not really helpful
- Empirical results show 14 or 18-dop performs best.



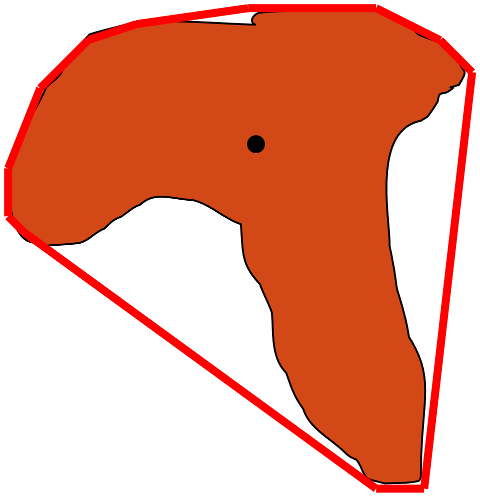
Common Bounding Volumes: Oriented Bounding Box (OBB)

- Store rectangular parallelepiped oriented to best fit the object
- Store:
  - Center
  - Orthonormal set of axes
  - Extent along each axis
- Tight fit, but takes work to get good initial fit
- OBB rotates with object, therefore only rotation of axes is needed for update
- Computation is slightly slower than for AABBs



Common Bounding Volumes: Convex Hull (CH)

- Very tight fit (tightest convex bounding volume)
- Slow to compute with
- Store: set of polygons forming convex hull
- Can rotate CH along with object.
- Can be efficient for some applications

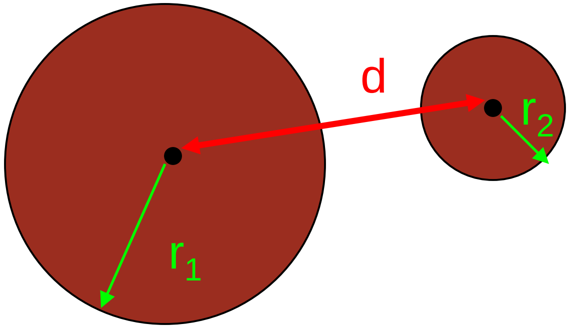


Testing for Collision

- Will depend on type of objects and bounding volumes.
- Specialized algorithms for each:
  - Sphere/sphere
  - AABB/AABB
  - OBB/OBB
  - Ray/sphere
  - Triangle/Triangle

Collision Test Example Sphere-Sphere

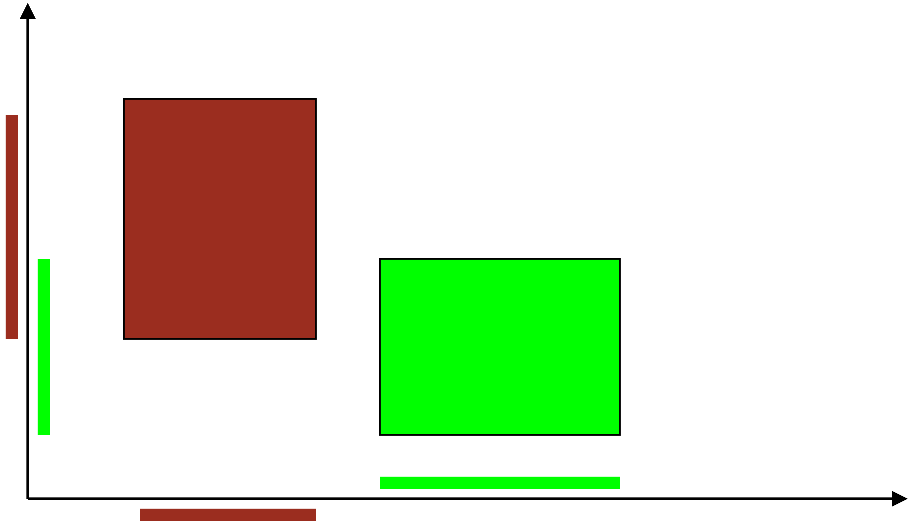
- Find distance between centers of spheres
- Compare to sum of sphere radii
  - If distance is less, they collide
- For efficiency, check squared distance vs. square of sum of radii



Collision Test Example
AABB vs. AABB

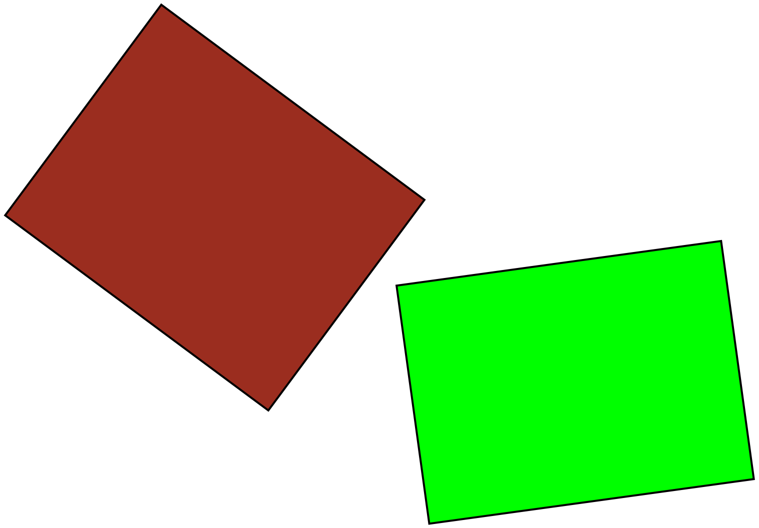
- Project AABBs onto axes
  - i.e. look at extents
- If overlapping on *all* axes, the boxes overlap.
- Same idea

  for k-dops.



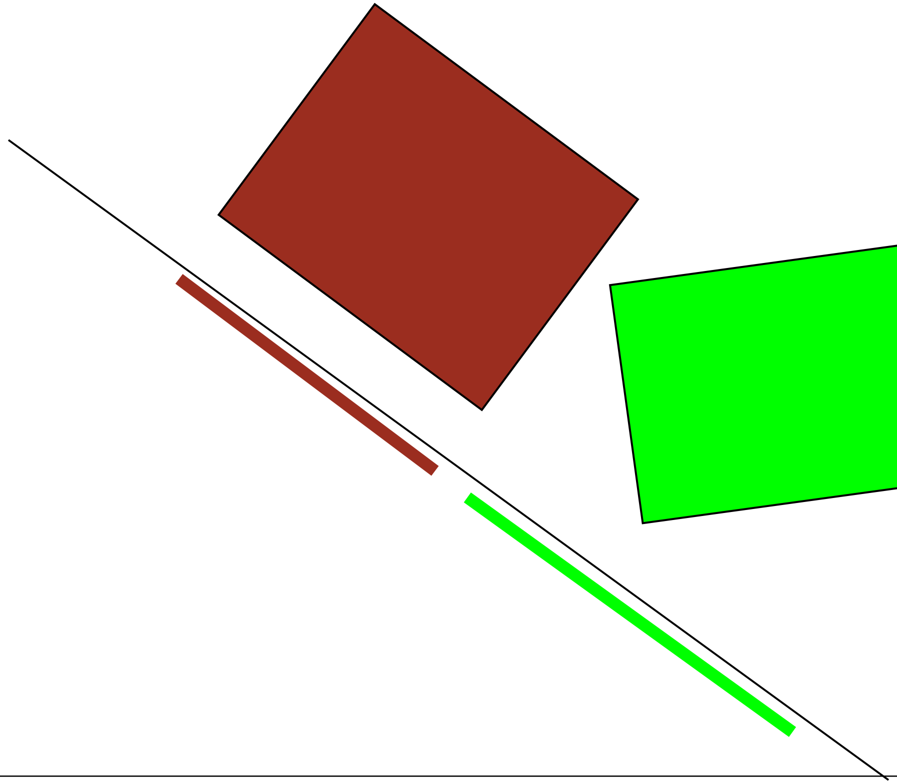
Collision Test Example
OBB vs. OBB

- Similar to overlap test for k-dops
- How do we find axes to test for overlap?



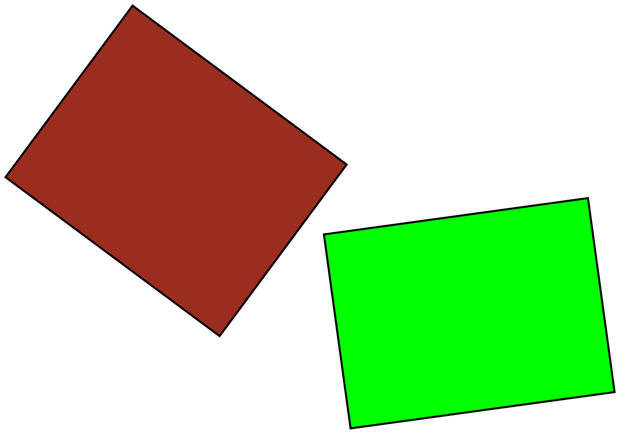
Separating Axis Theorem

- Two convex shapes do not overlap if and only if there exists an axis such that the projections of the two shapes do not overlap



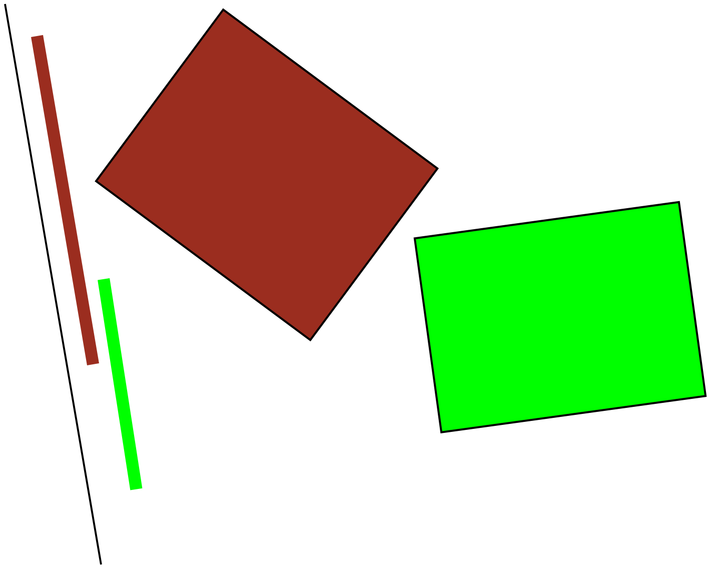
Enumerating Separating Axes

- 2D: check axis aligned with normal of each face
- 3D: check axis aligned with normals of each face and cross product of each pair of edges



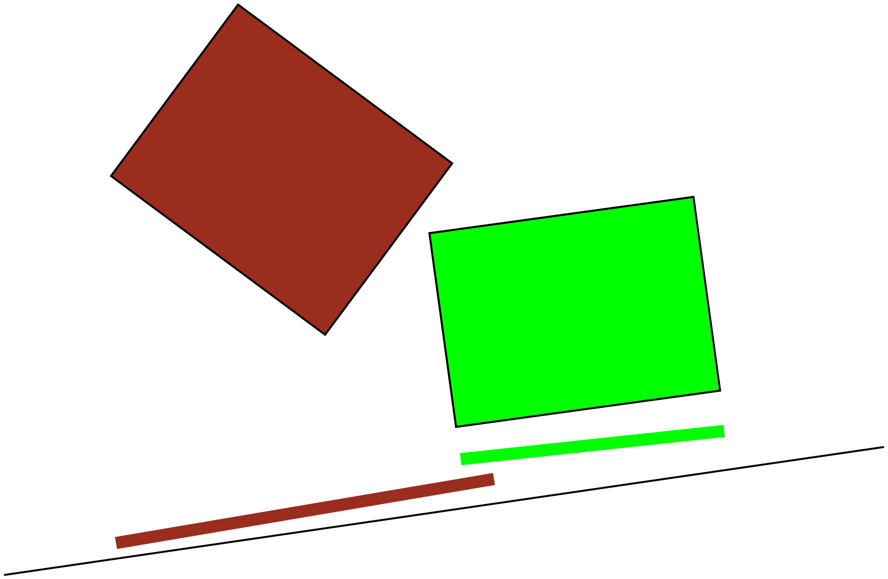
Enumerating Separating Axes

- 2D: check axis aligned with normal of each face
- 3D: check axis aligned with normals of each face and cross product of each pair of edges



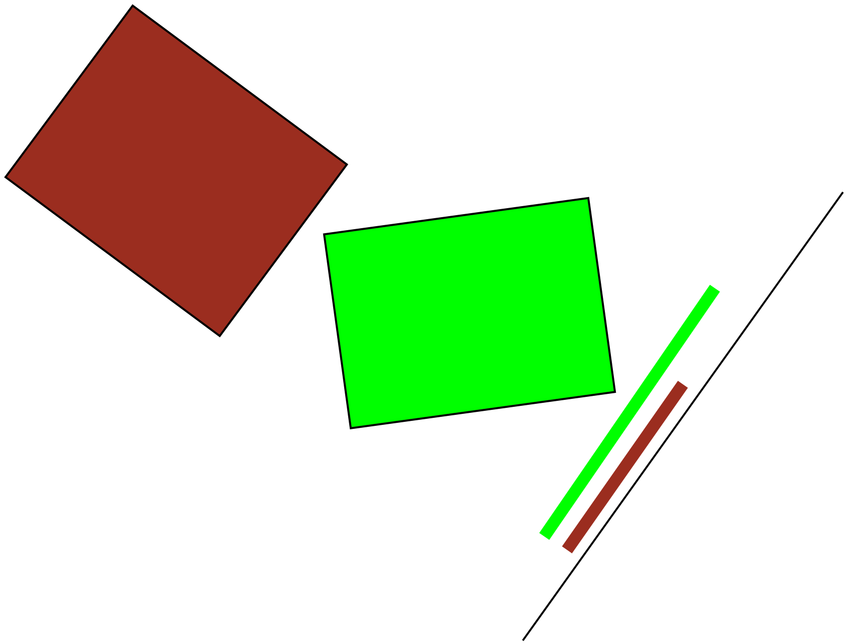
Enumerating Separating Axes

- 2D: check axis aligned with normal of each face
- 3D: check axis aligned with normals of each face and cross product of each pair of edges



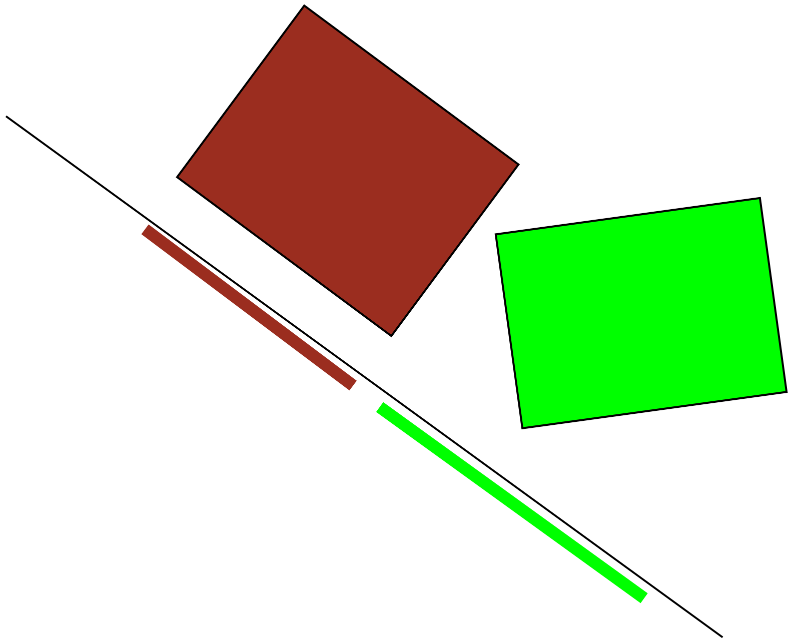
Enumerating Separating Axes

- 2D: check axis aligned with normal of each face
- 3D: check axis aligned with normals of each face and cross product of each pair of edges



Enumerating Separating Axes

- 2D: check axis aligned with normal of each face
- 3D: check axis aligned with normals of each face and cross product of each pair of edges

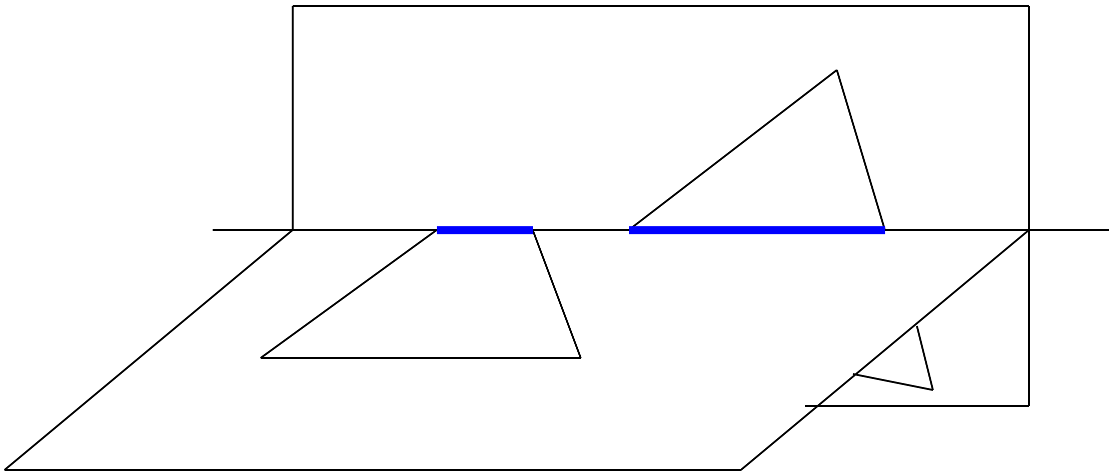


Collision Test Example Triangle-Triangle

- Mesh collision detection tests eventually reduce to this.
- Two common approaches. Both involve finding the plane a triangle lies in.
  - Cross product of edges to get triangle normal.
  - This is the plane normal [A B C] where plane is $Ax+By+Cz+D=0$
  - Solve for D by plugging in a triangle vertex

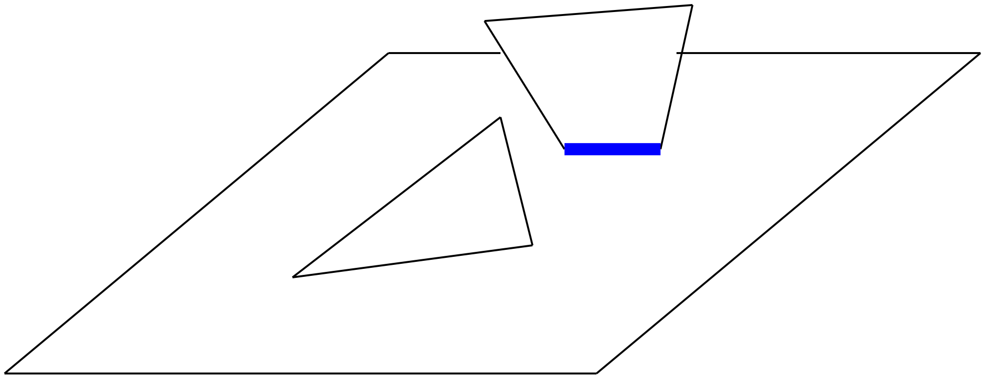
Triangle-Triangle Collision 1

- Find line of intersection between triangle planes.
- Find extents of triangles along this line
- If extents overlap, triangles intersect.



Triangle-Triangle Collision 2

- Intersect edges of one triangle with plane of the other triangle.
- 2 edges will intersect – form line segment in plane.
- Test that 2D line segment against triangle.



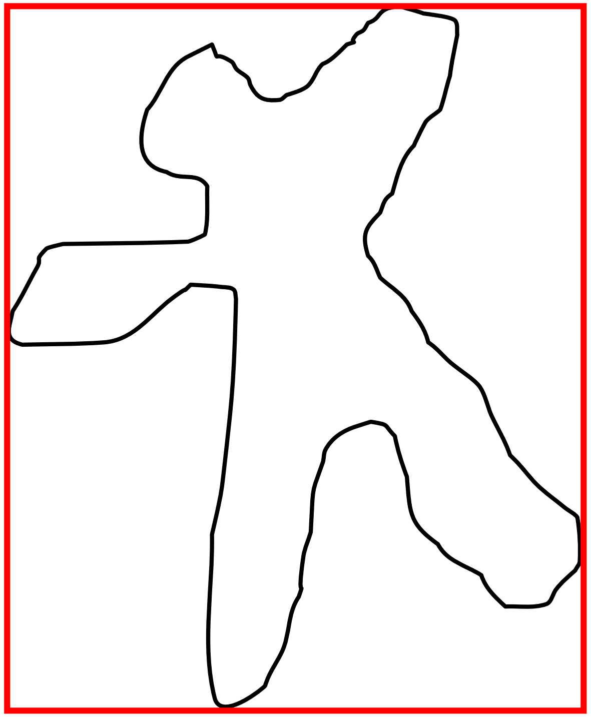
Bounding Volume Hierarchies

- What happens when the bounding volumes do intersect?
  - We must test whether the actual objects underneath intersect.
  - For an object made from lots of polygons, this is complicated.
  - So, we will use a bounding volume hierarchy

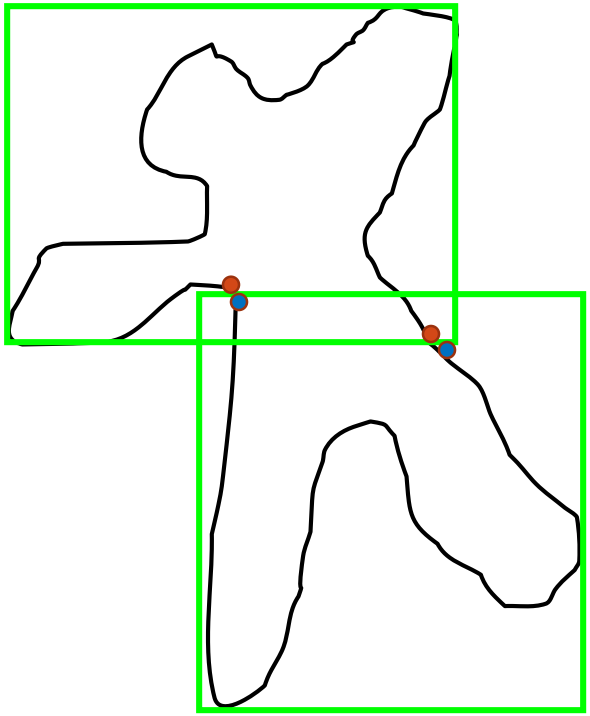
Bounding Volume Hierarchies

- Highest level of hierarchy – single BV around whole object
- Next level – subdivide the object into sub-parts.
  - Each part gets its own BV
- Continue recursively until only one triangle remains

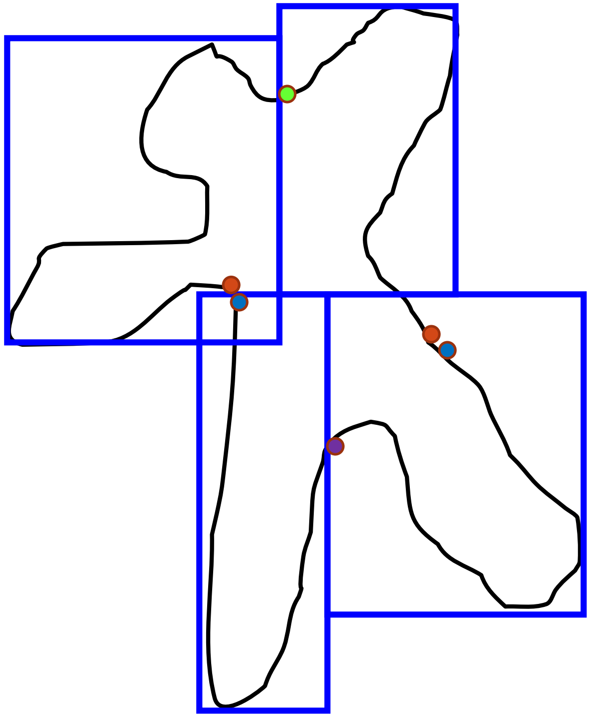
Bounding Volume Hierarchy Example



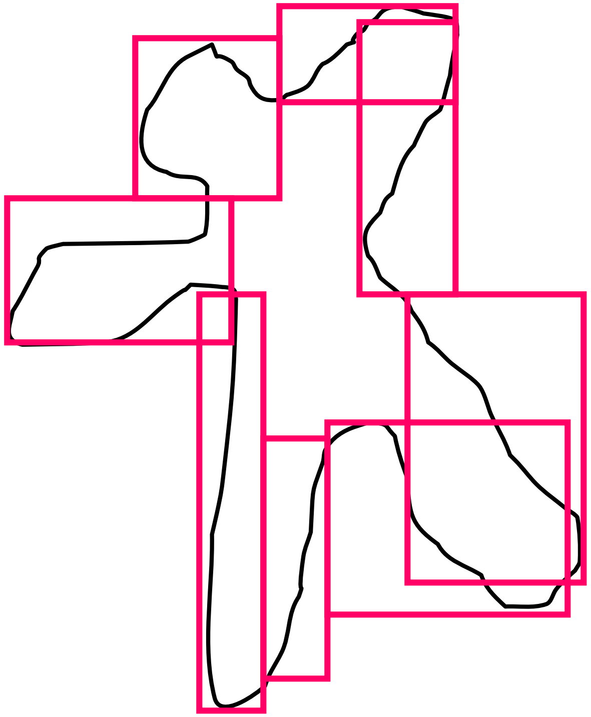
Bounding Volume Hierarchy Example



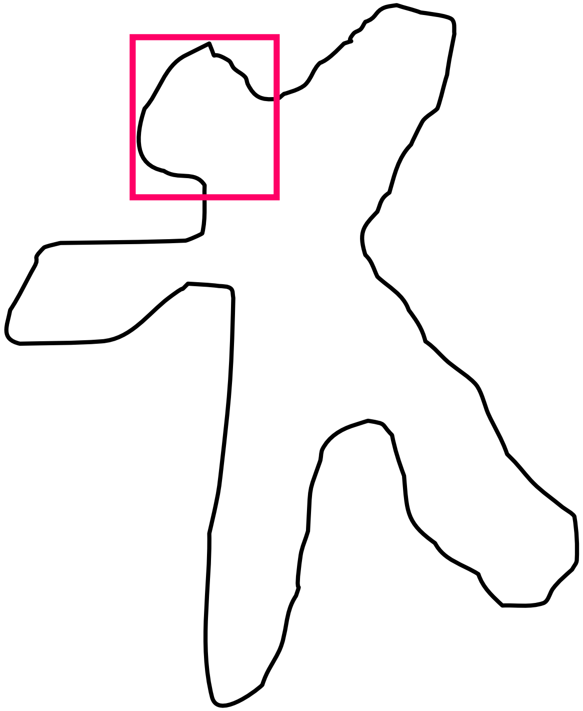
Bounding Volume Hierarchy Example



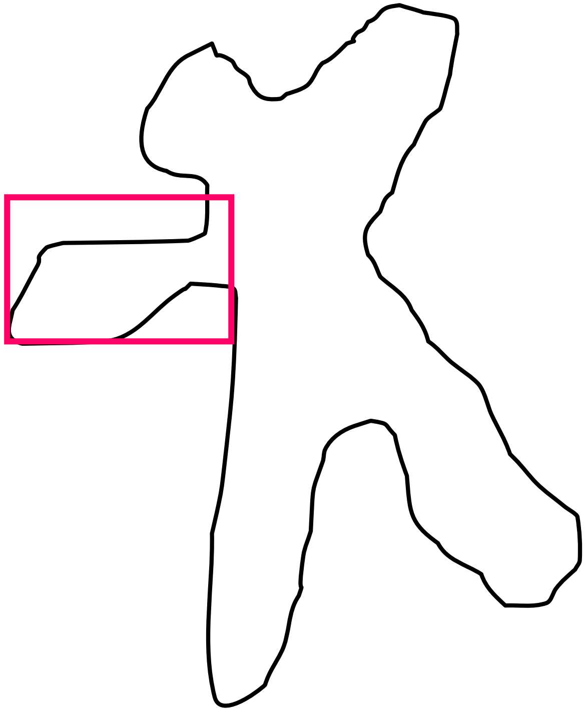
Bounding Volume Hierarchy Example



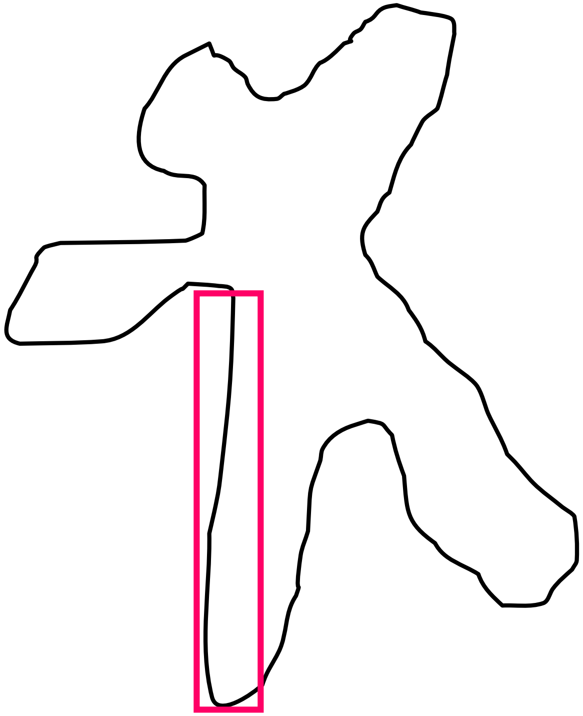
Bounding Volume Hierarchy Example



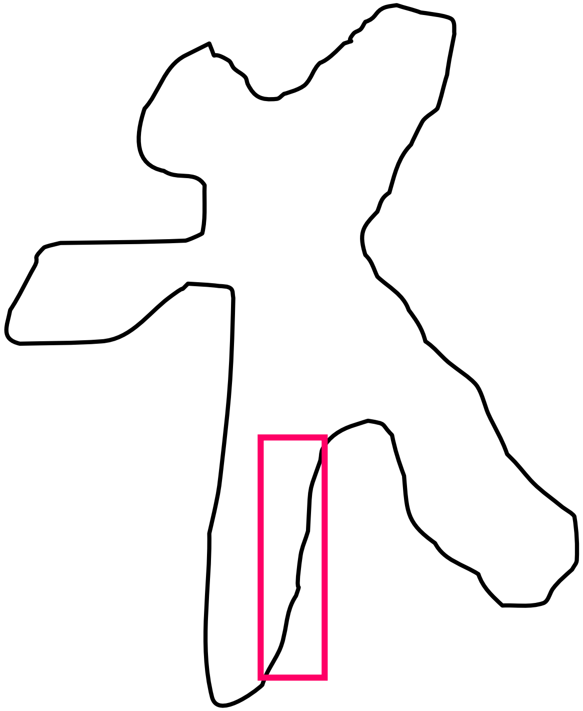
Bounding Volume Hierarchy Example



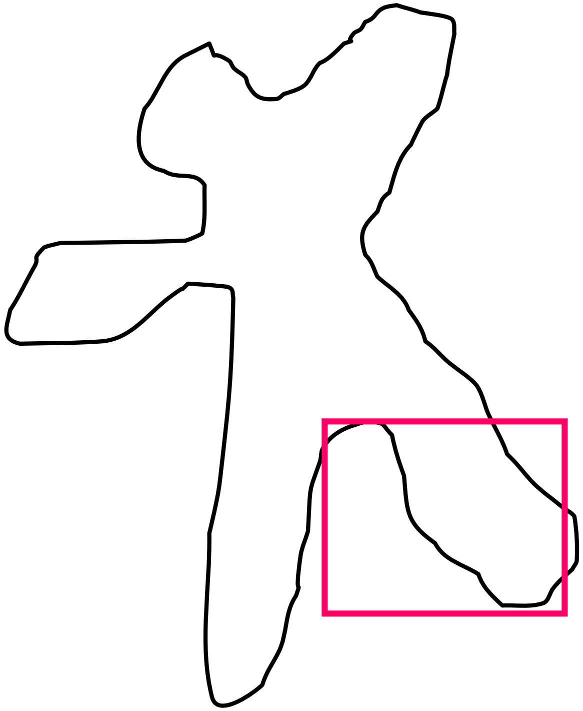
Bounding Volume Hierarchy Example



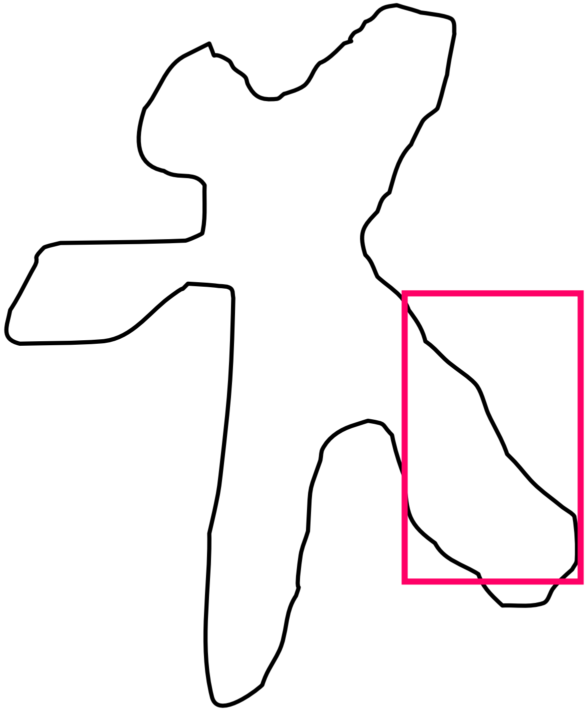
Bounding Volume Hierarchy Example



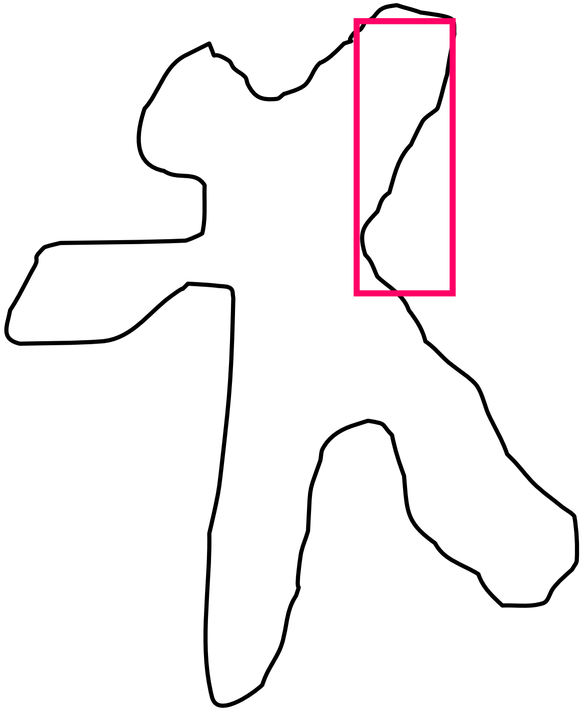
Bounding Volume Hierarchy Example



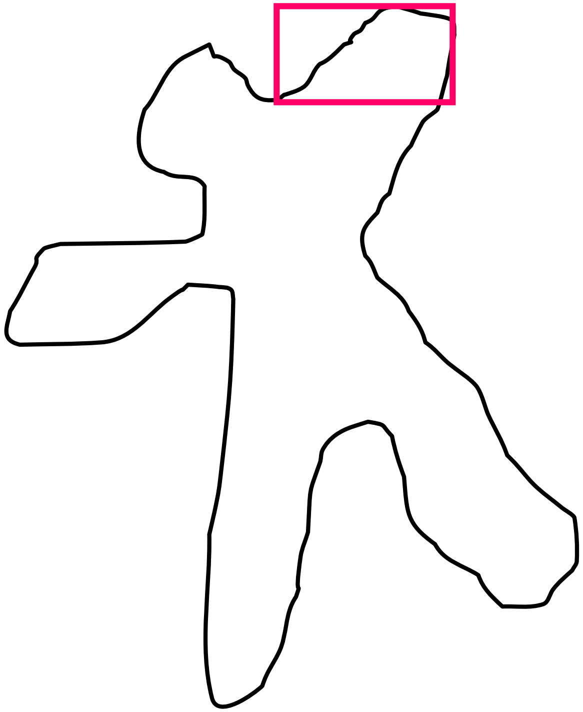
Bounding Volume Hierarchy Example



Bounding Volume Hierarchy Example



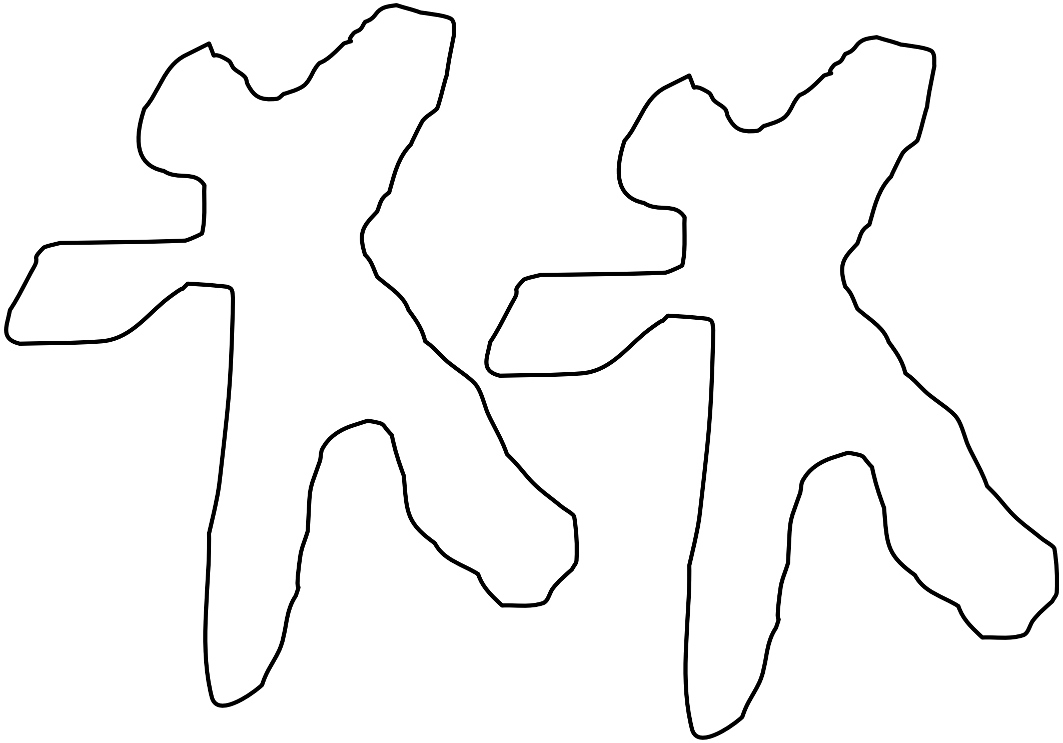
Bounding Volume Hierarchy Example



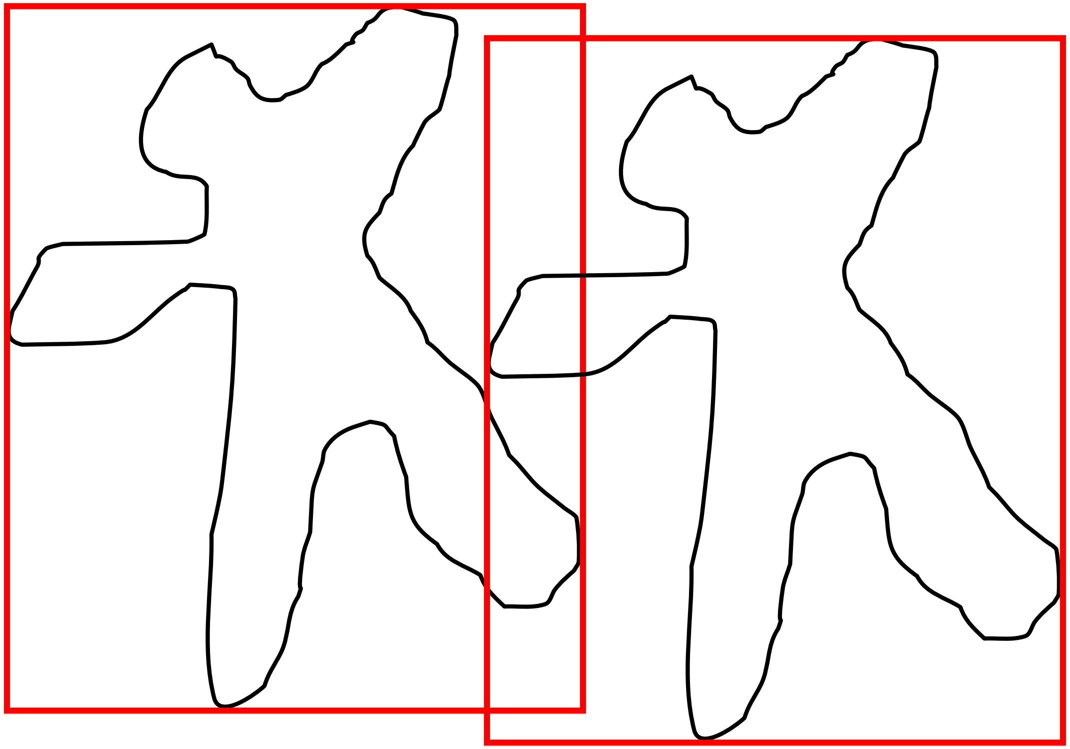
Intersecting Bounding Volume Hierarcies

- For object-object collision detection
- Keep a queue of potentially intersecting BVs
  - Initialize with main BV for each object
- Repeatedly pull next potential pair off queue and test for intersection.
  - If that pair intersects, put pairs of children into queue.
  - If no child for both BVs, test triangles inside
- Stop when we either run out of pairs (thus no intersection) or we find an intersecting pair of triangles

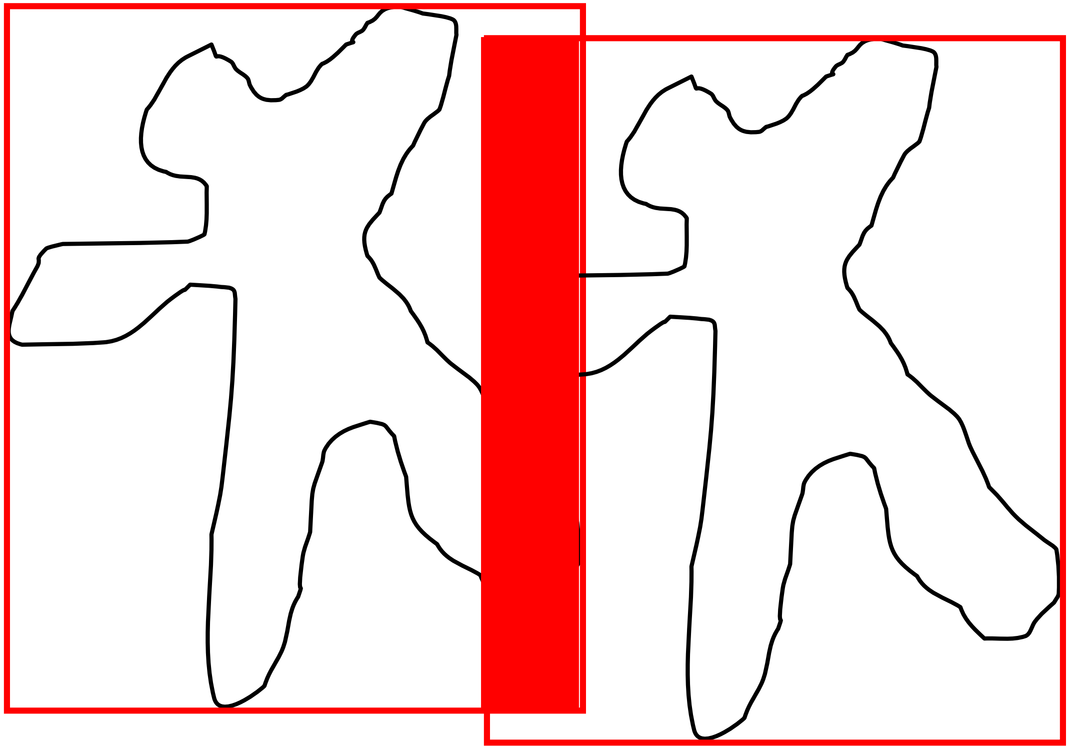
BVH Collision Test example



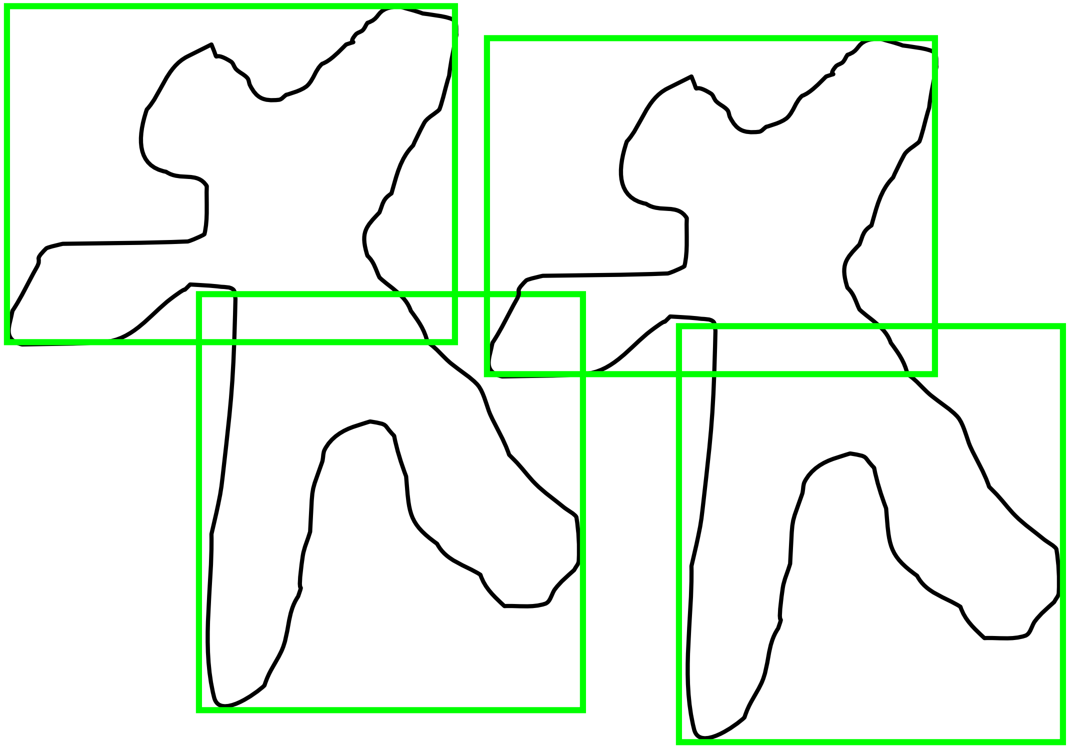
BVH Collision Test example



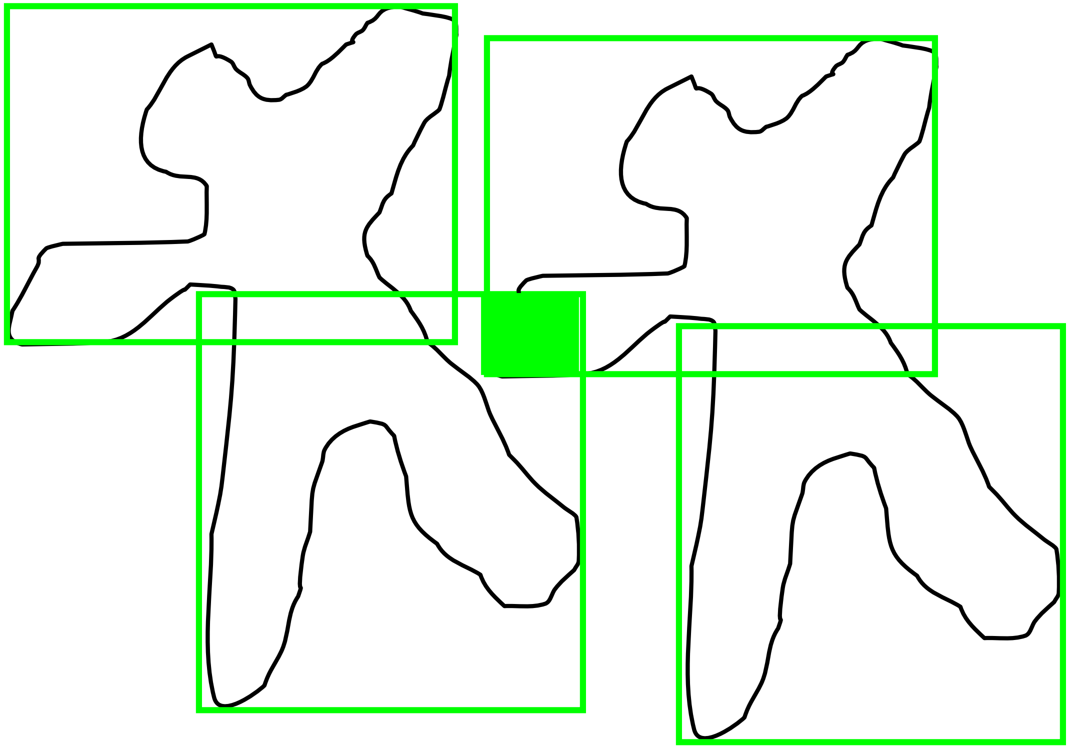
BVH Collision Test example



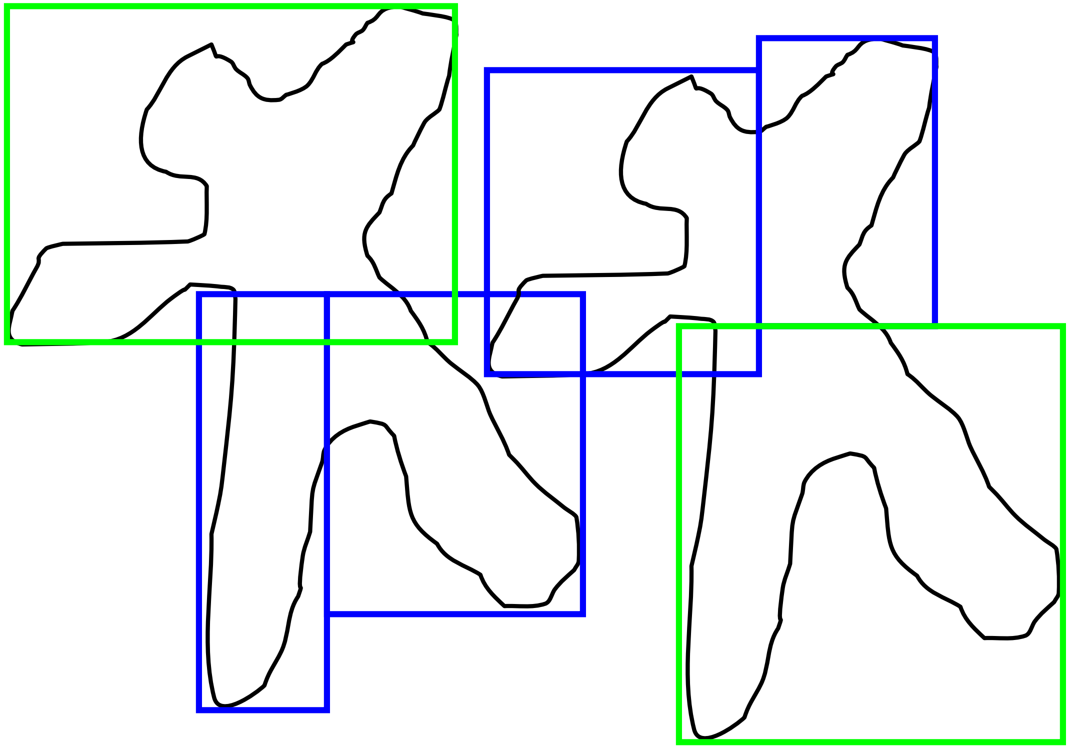
BVH Collision Test example



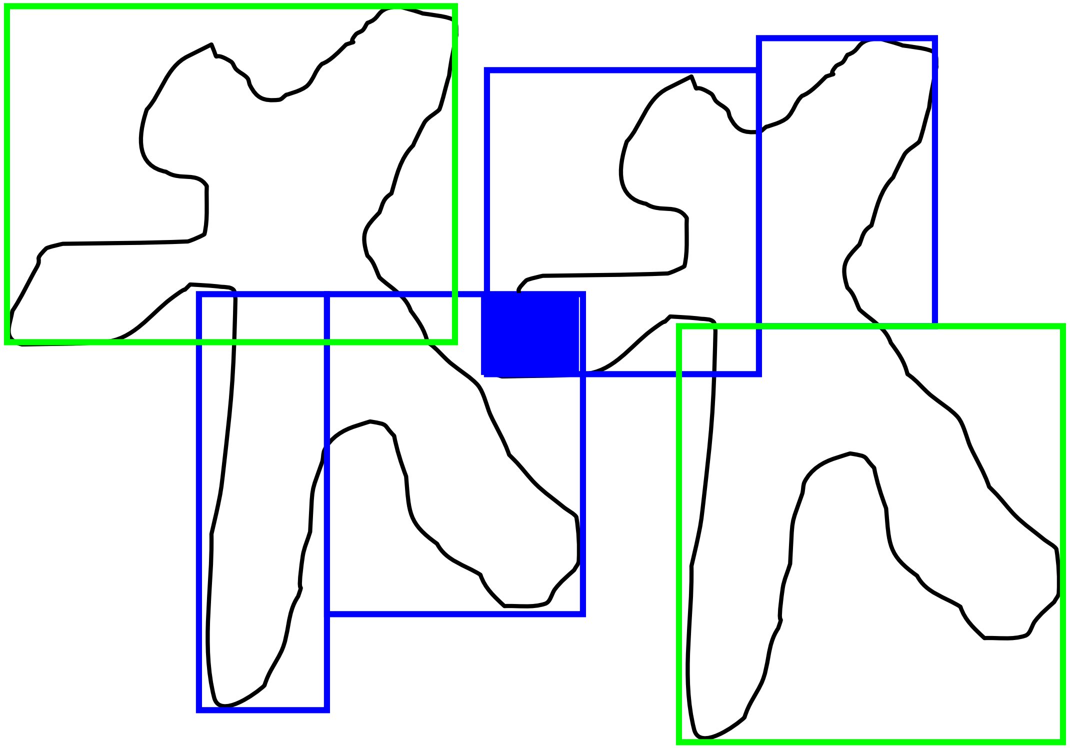
BVH Collision Test example



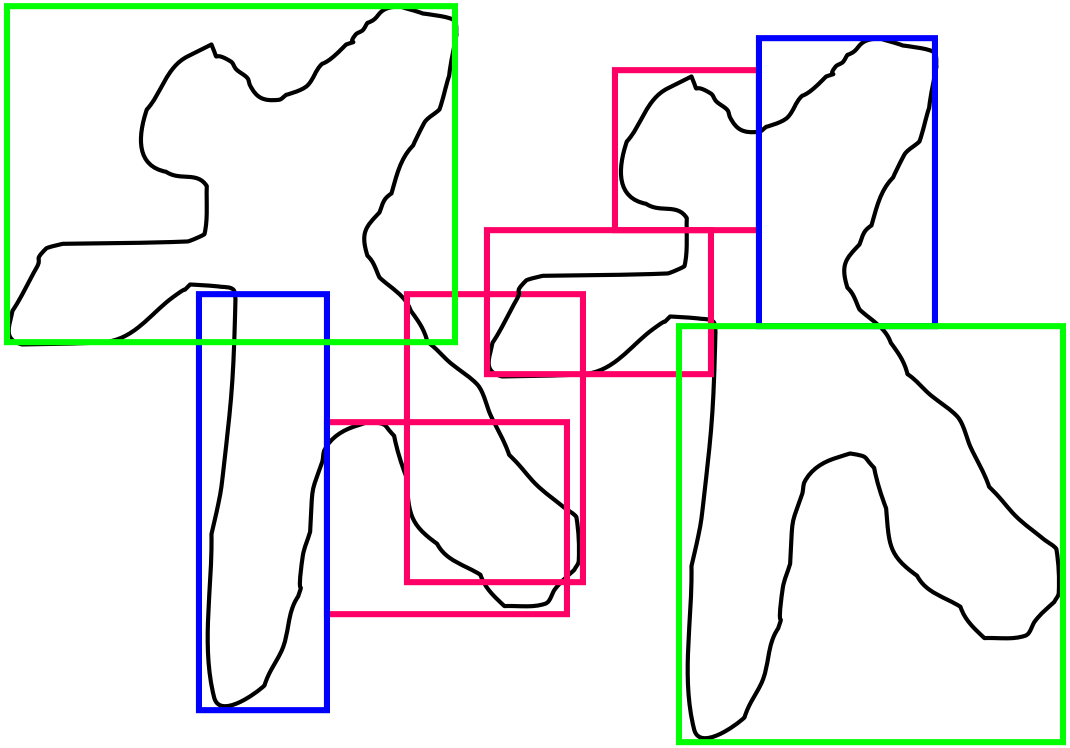
BVH Collision Test example



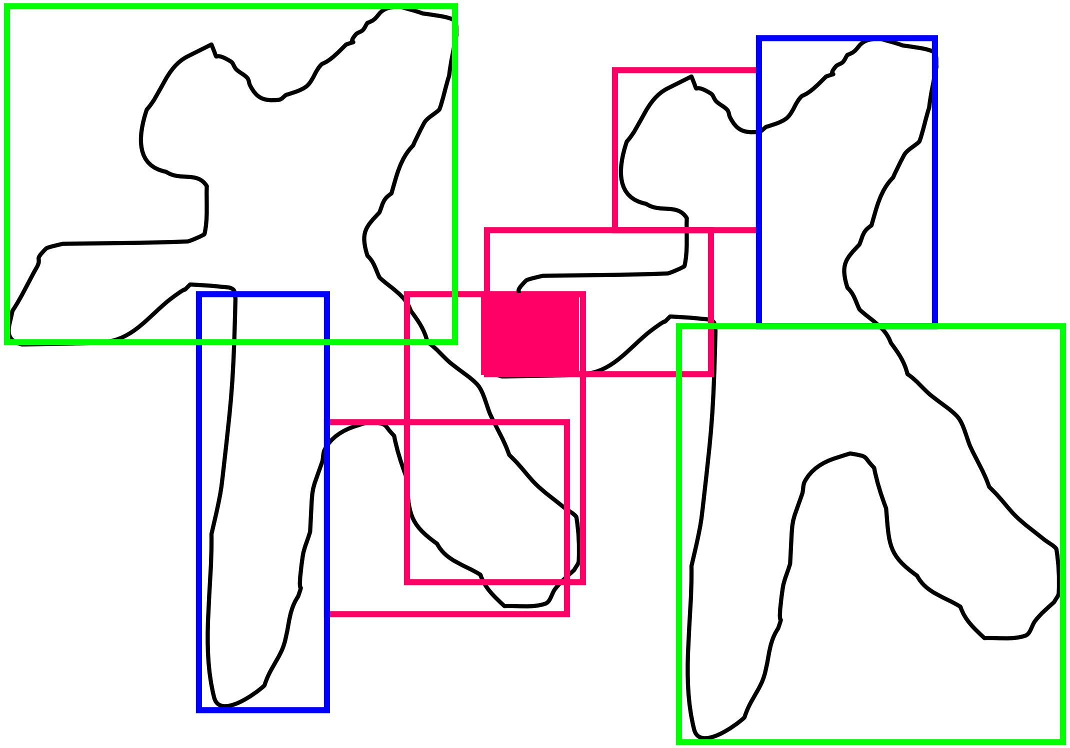
BVH Collision Test example



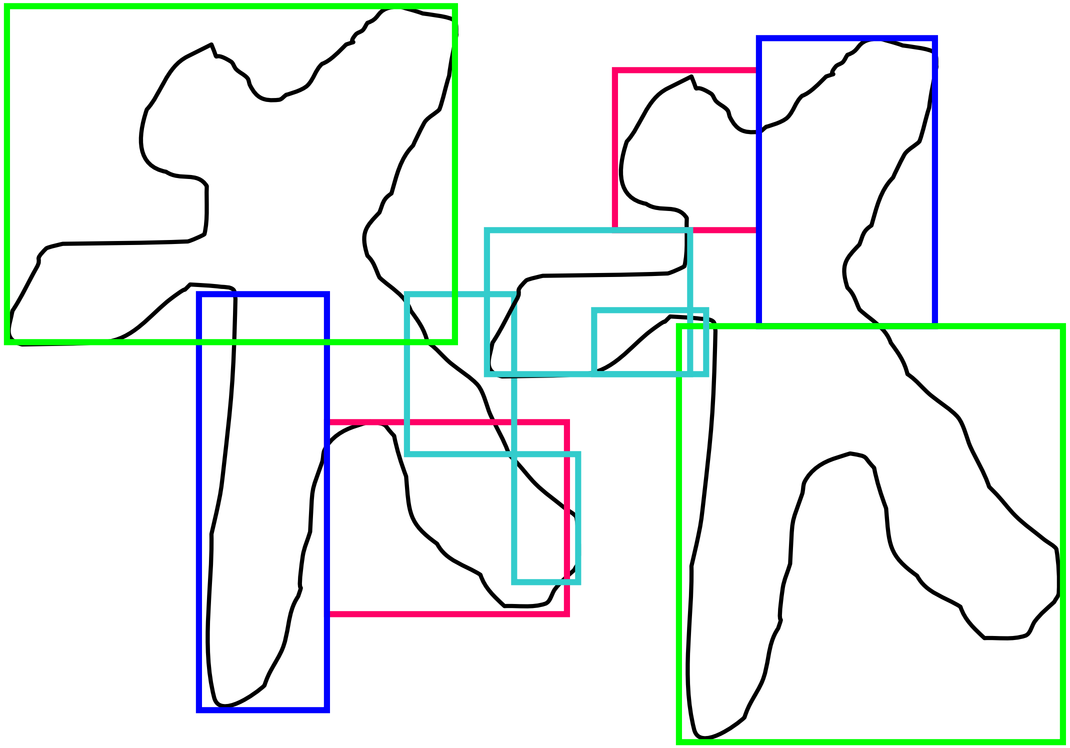
BVH Collision Test example



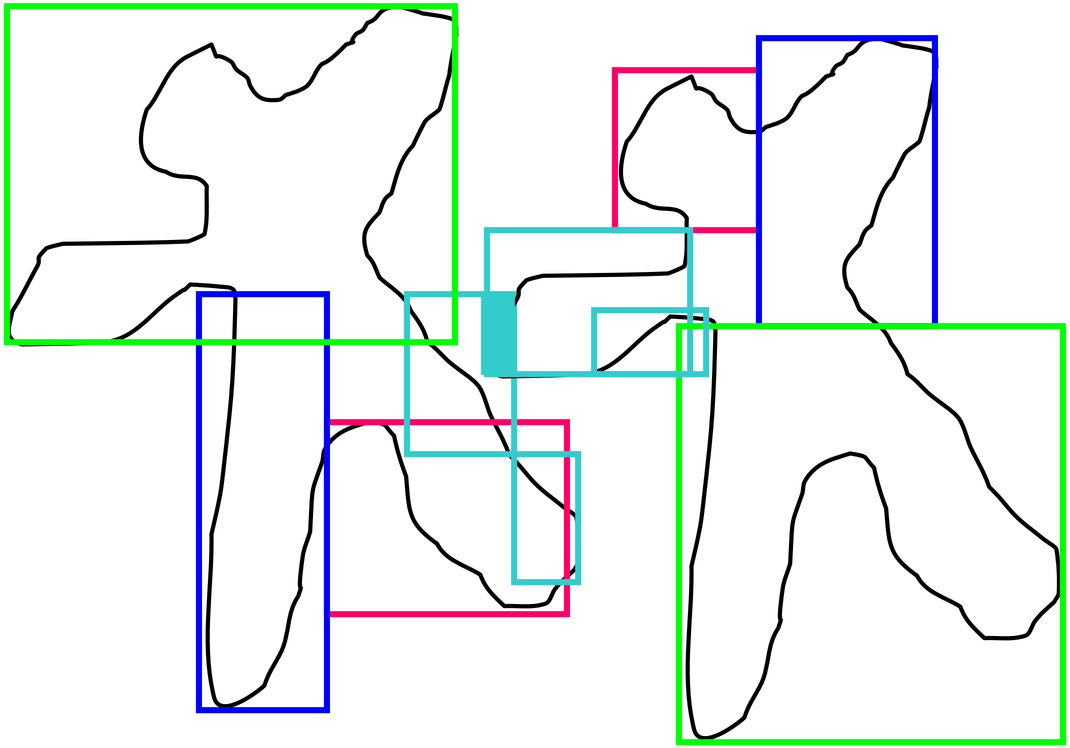
BVH Collision Test example



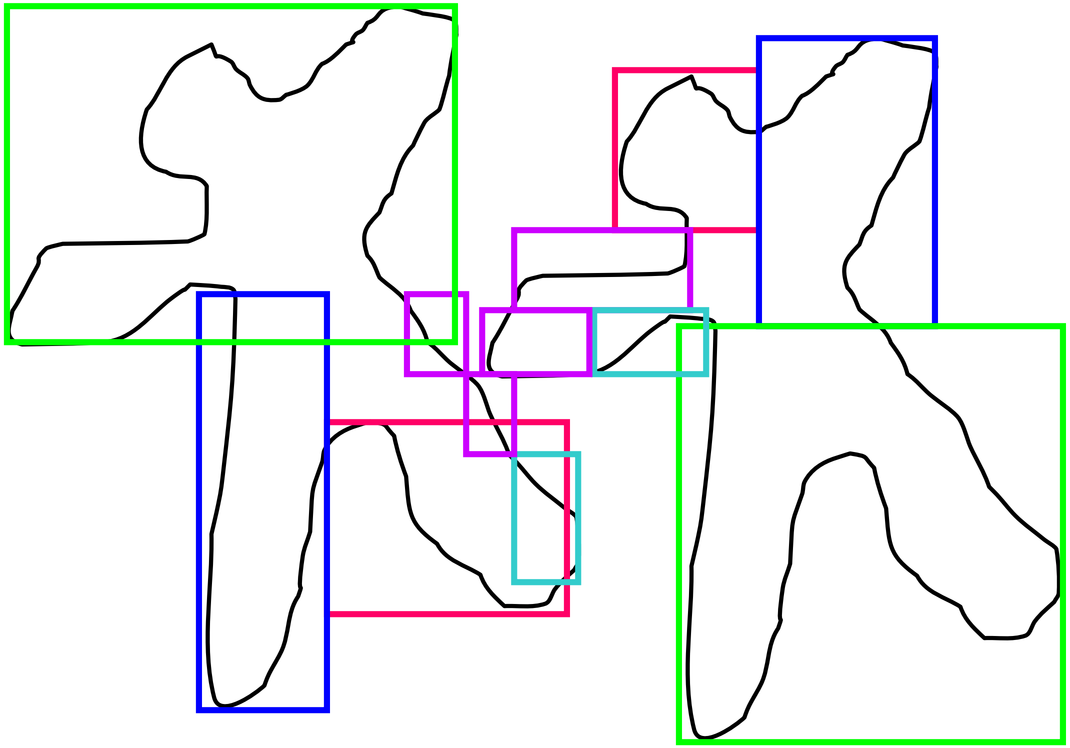
BVH Collision Test example



BVH Collision Test example



BVH Collision Test example



Broad Phase vs. Narrow Phase

- What we have talked about so far is the “narrow phase” of collision detection.
  - Testing whether two particular objects collide
- The “broad phase” assumes we have a number of objects, and we want to find out all pairs that collide.
- Testing every pair is inefficient

Broad Phase Collision Detection

- Form an AABB for each object
- Pick an axis
  - Sort objects along that axis
  - Find overlapping pairs along that axis
  - For overlapping pairs, check along other axes.
- Limits the number of object/object tests
- Overlapping pairs then sent to narrow phase

- Or
  - Consider constructing an octree for all these models