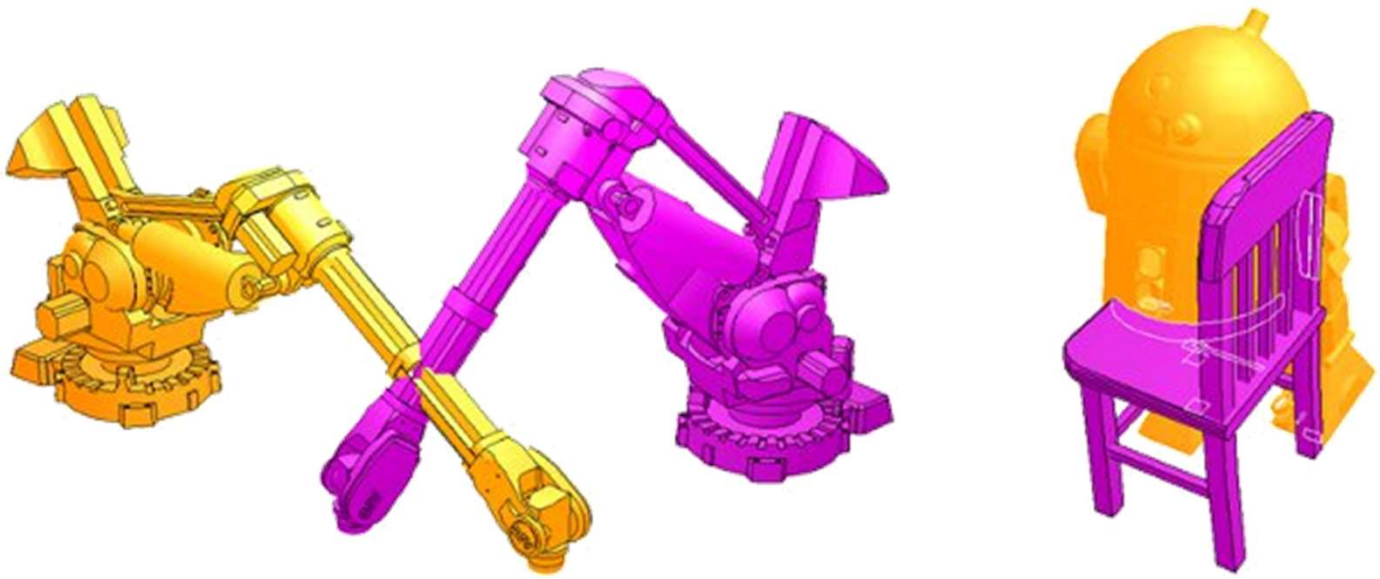
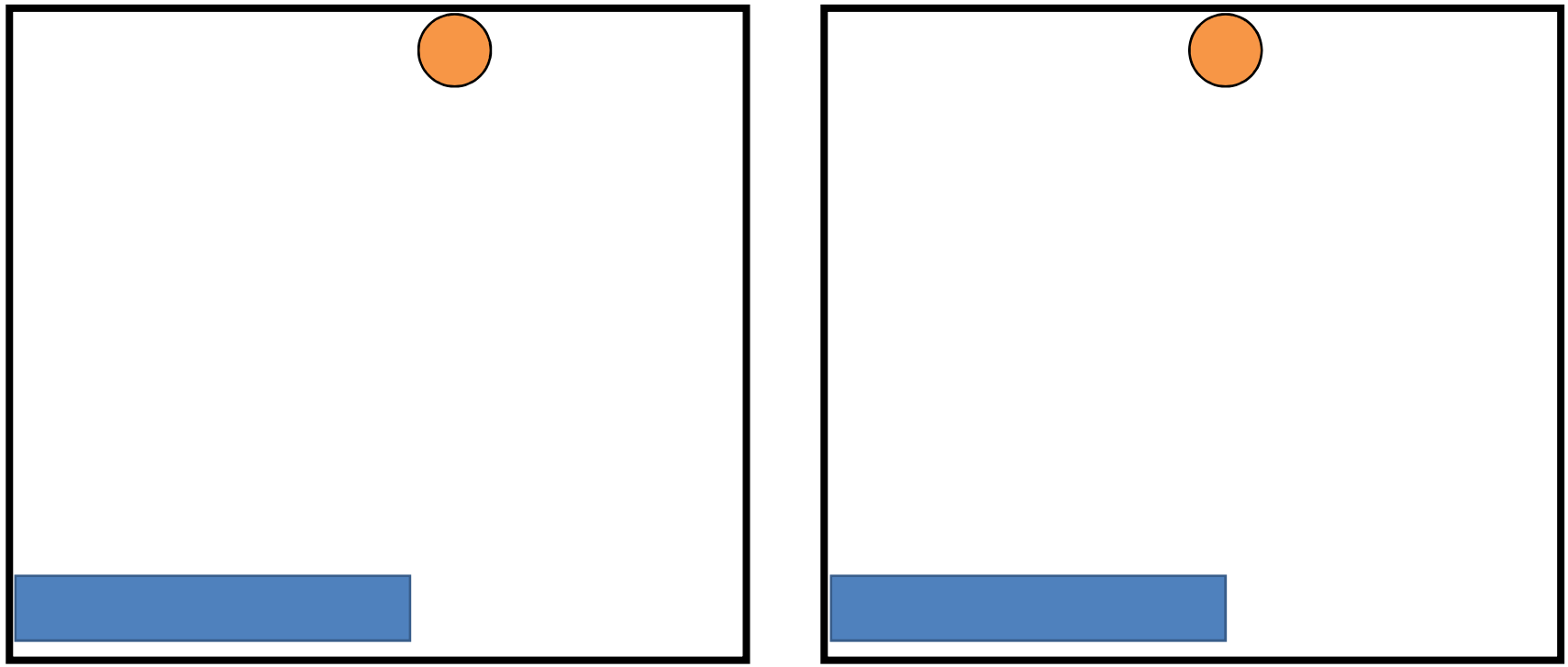
Handling Collisions



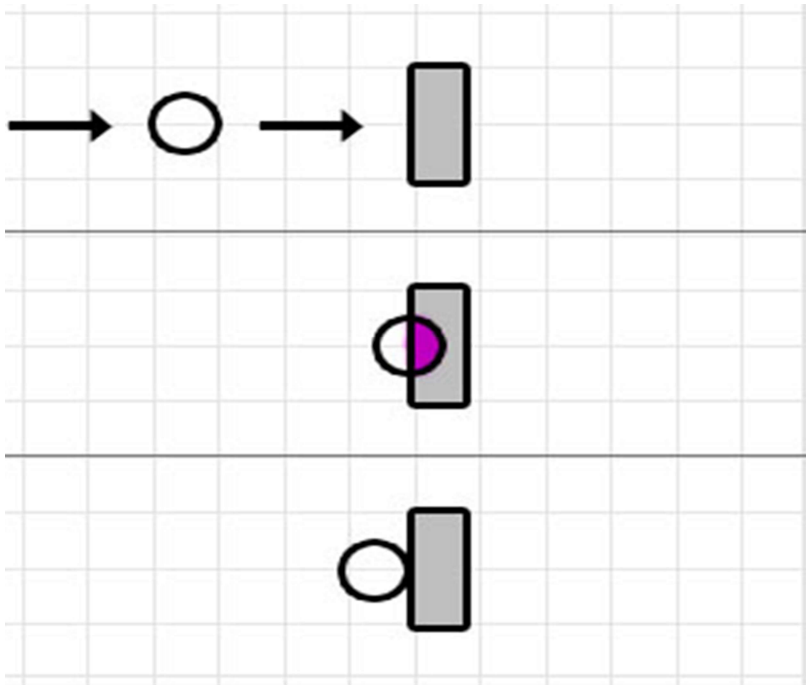
Why?

- Realisme / game play
  - Without objects pass through other objects



Three Major Parts

- Collision detection
  - Do the objects collide?
- Collision determination
  - Where do they collide?
- Collision response
  - What happens now?



Three Major Parts

- Collision detection
  - Do the objects collide?

Always needed

- Collision determination
  - Where do they collide?

Not always needed

- Collision response
  - What happens now?

Specific to application domain

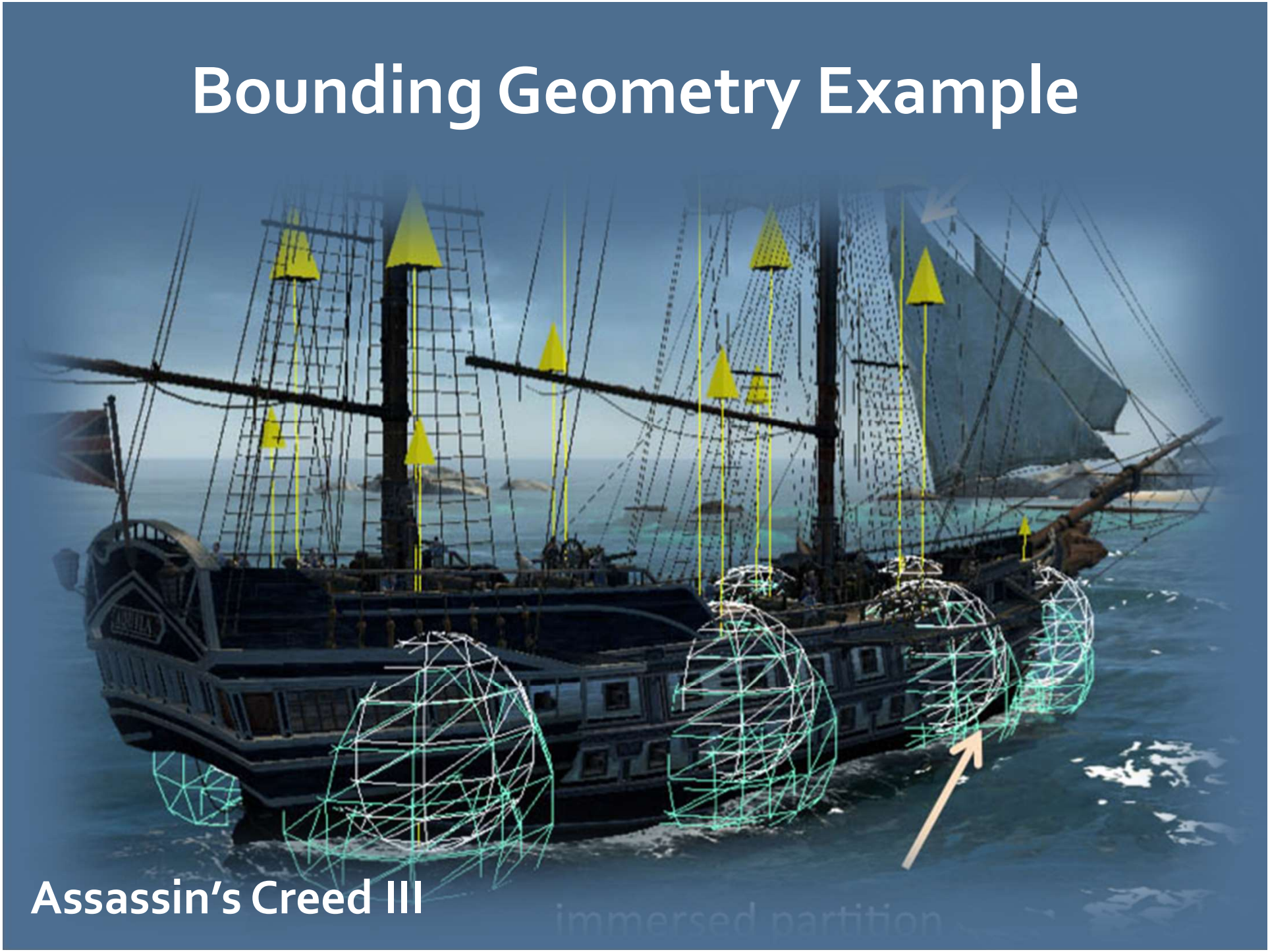
Collision Detection

Collision Detection

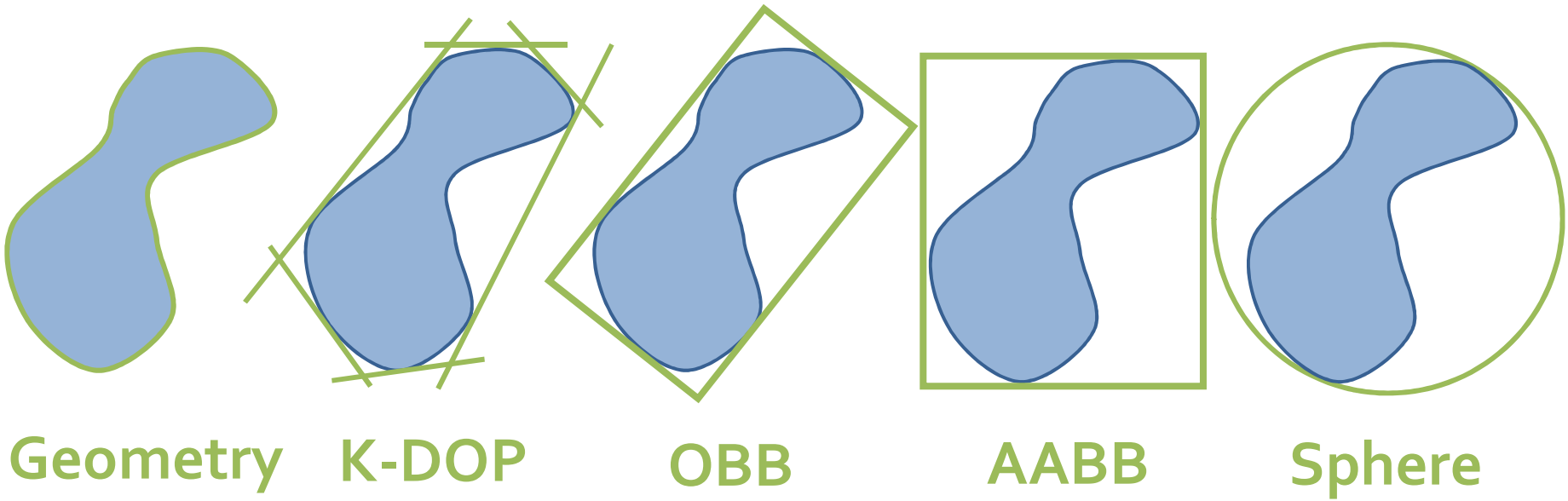
- Could check real geometry
  - Complex geometry means slow detection
  - Often not necessary
- Use simple approximation a.k.a. bounding geometry

Bounding Geometry Example

Bounding Geometry Example

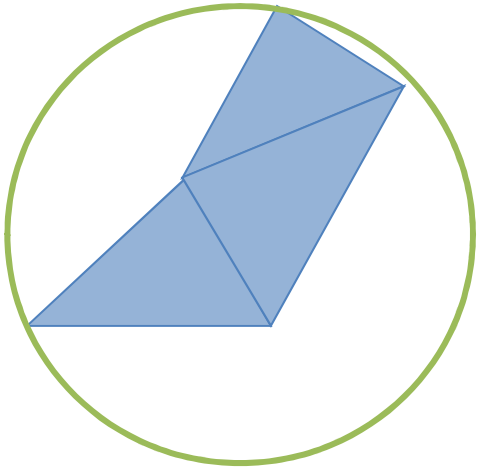


Bounding Geometry



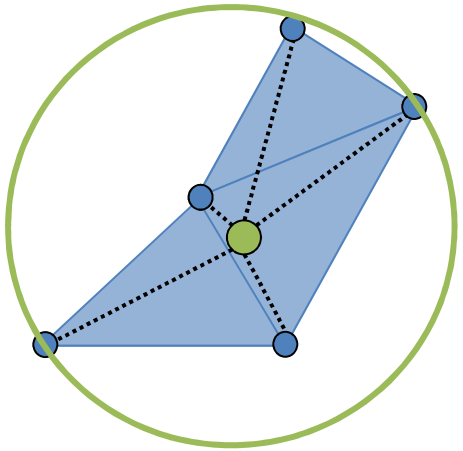
How do we find a bounding geometry?

- Calculation (again many algorithms)
- Artists defines the bounding geometry alongside the object



Bounding Sphere – Calculation

- Find the center
  - Average of all vertices
- Find radius
  - For all vertices: calculate max. distance to M
- In mathematics: minimal bounding sphere problem

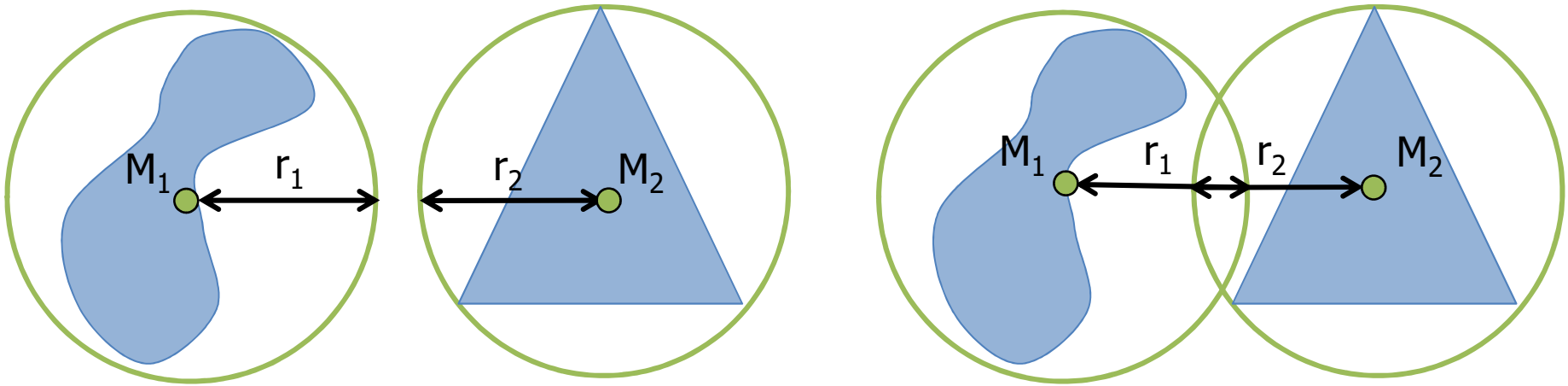


Bounding Sphere – Collision Detection

- Collision iff

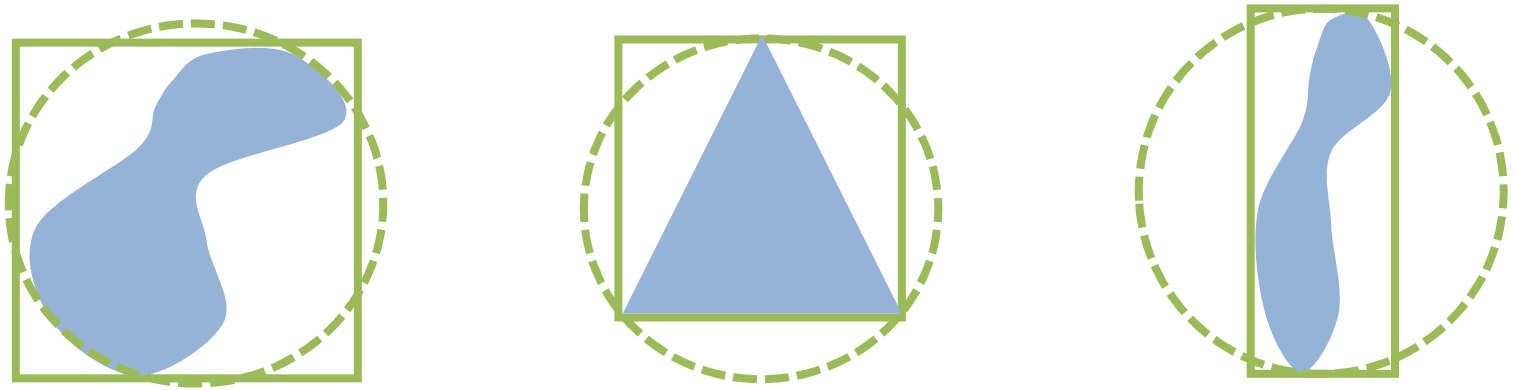
$$distance(M_1, M_2) < r_1 + r_2$$

$$\Leftrightarrow distance(M_1, M_2)^2 < (r_1 + r_2)^2$$

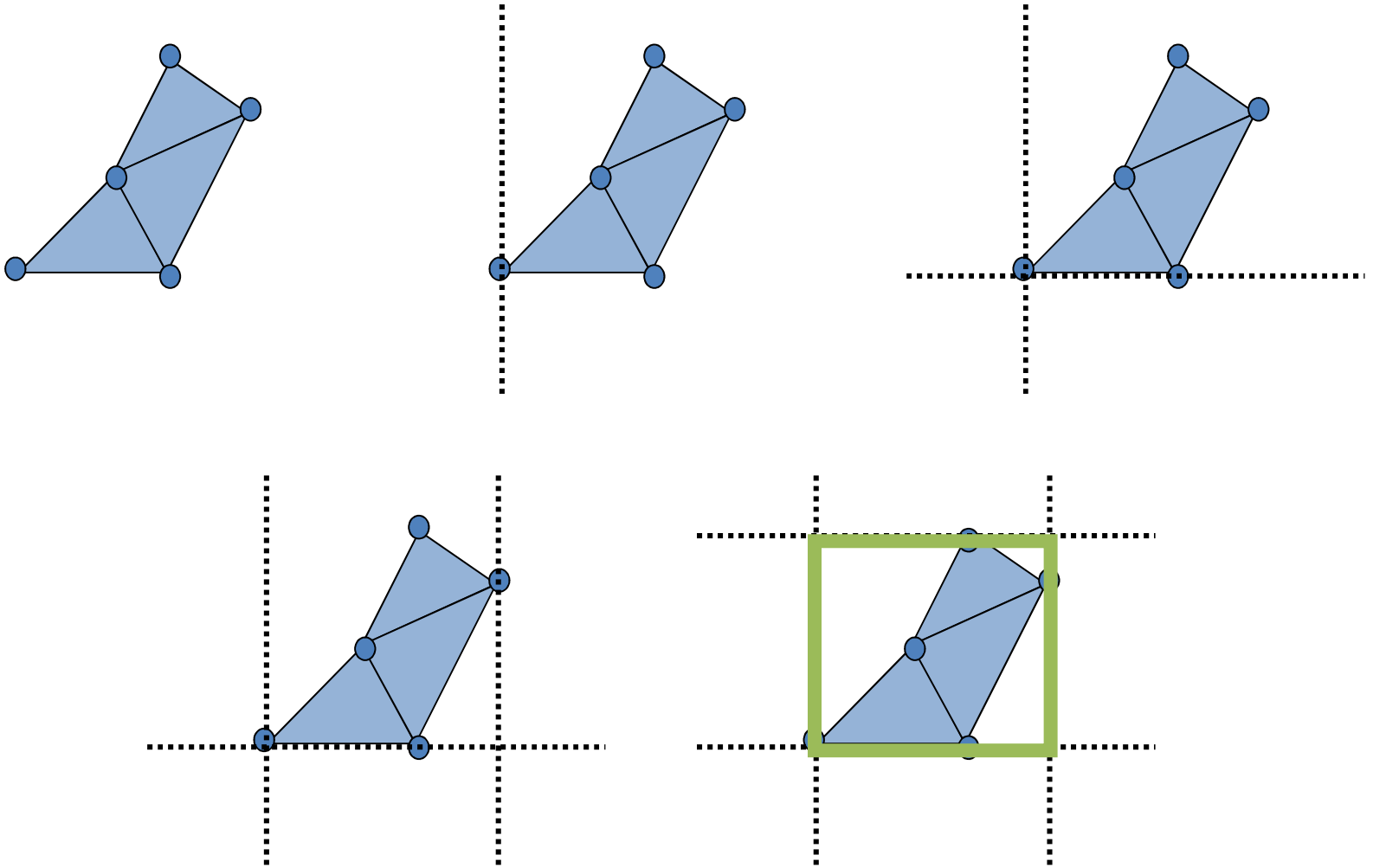


AABB-Algorithm

- Bounding-Spheres:
  - Efficient
  - Inaccurate
- Axis Aligned Bounding Boxes
  - Better fit for many objects
  - Only slightly more complicated

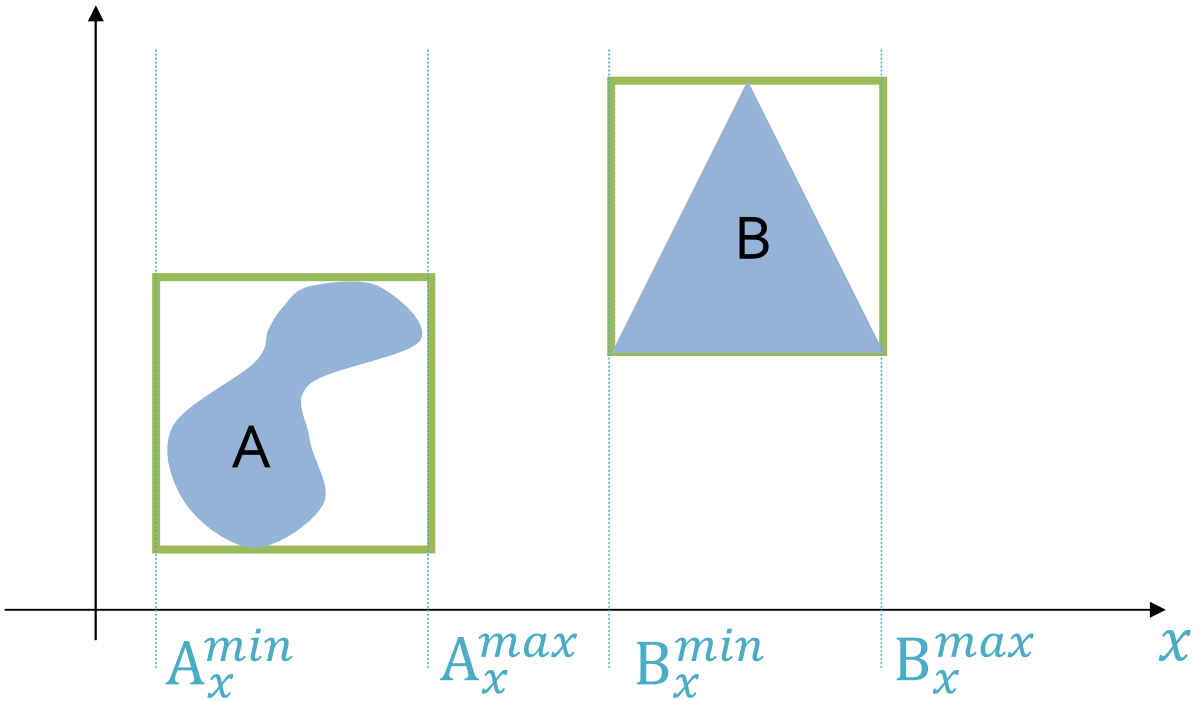


AABB – Calculation



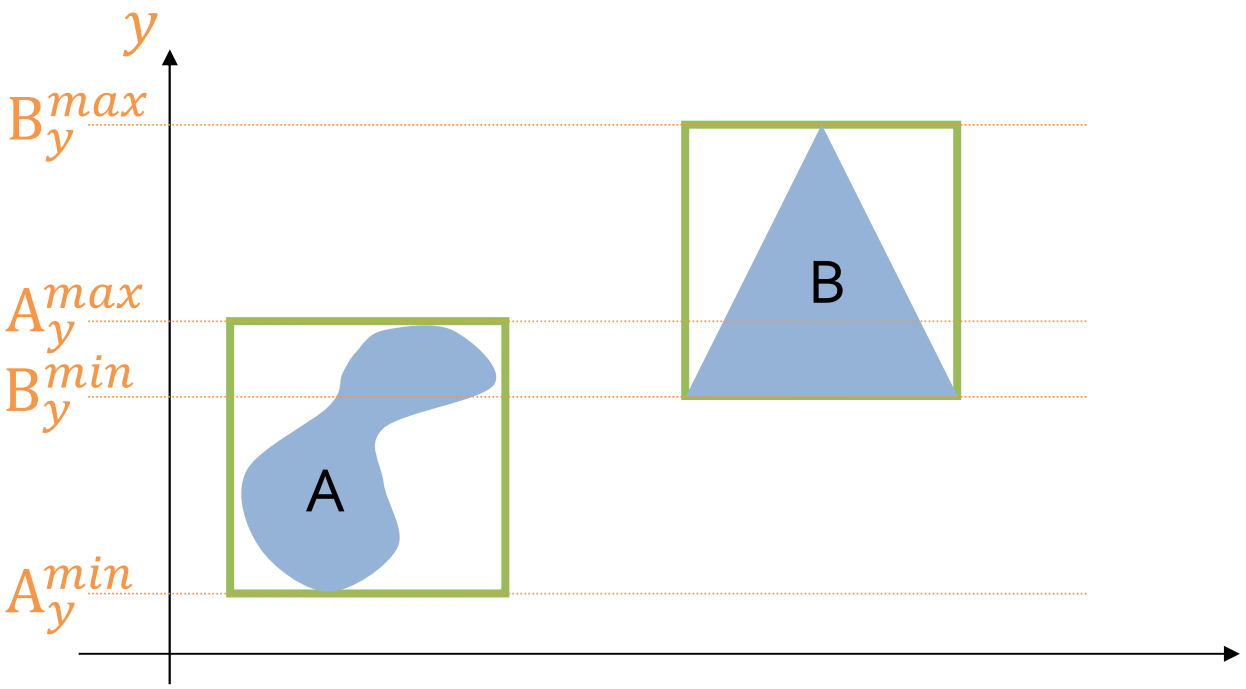
AABB-Algorithm

- No collision if
- $\left(\mathrm{A}_x^{min} > \mathrm{B}_x^{max}\right) or \left(\mathrm{B}_x^{min} > \mathrm{A}_x^{max}\right)$



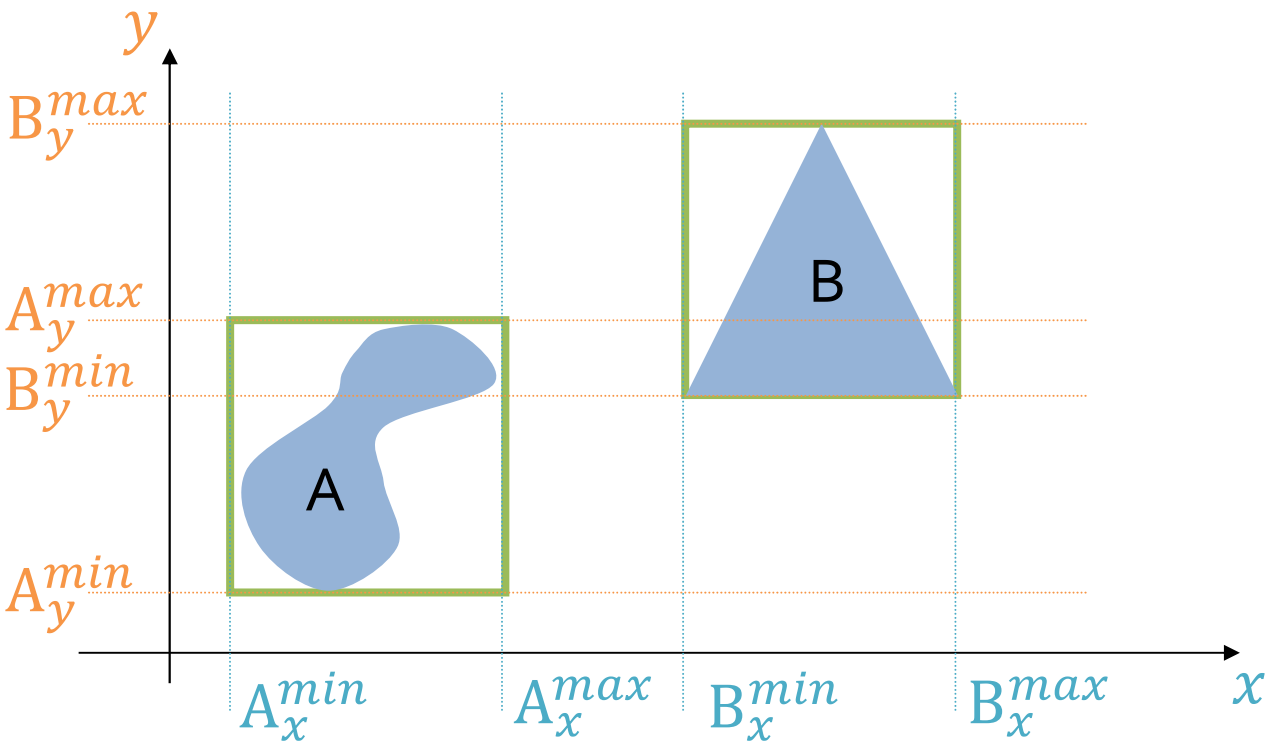
AABB-Algorithm

- No collision if
- $\left(\mathrm{A}_y^{min} > \mathrm{B}_y^{max}\right) or \left(\mathrm{B}_y^{min} > \mathrm{A}_y^{max}\right)$



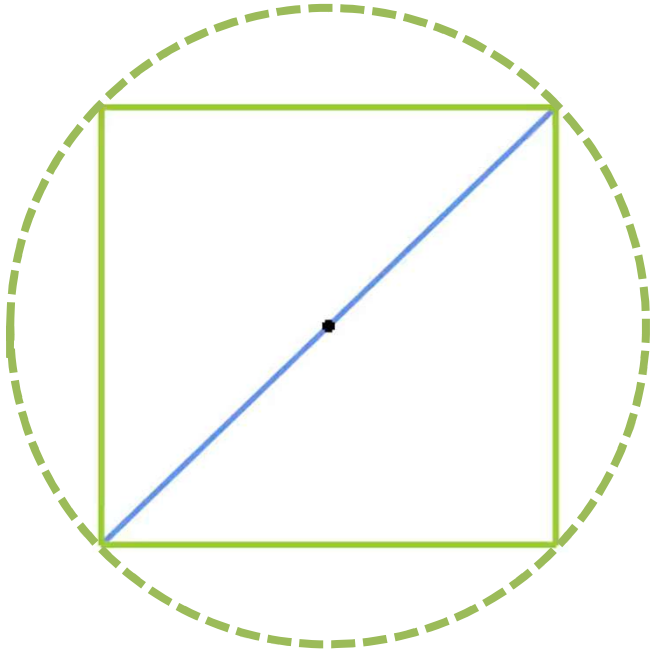
AABB-Algorithm

- No collision if
- $\exists i \in \{x, y\} | \left(\mathrm{A}_i^{min} > \mathrm{B}_i^{max}\right) or \left(\mathrm{B}_i^{min} > \mathrm{A}_i^{max}\right)$
  - Separating axis theorem (same for z)



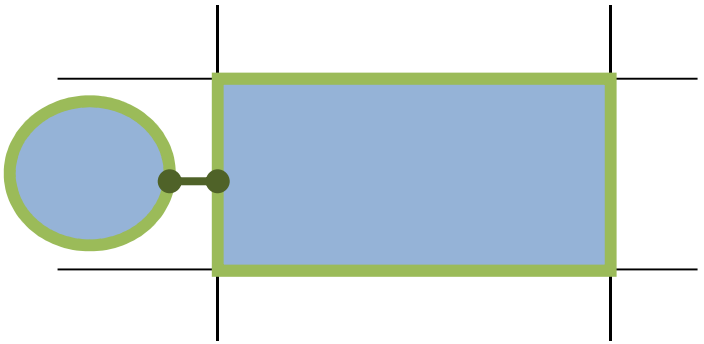
AABB - Problems

- While rotating an object, we have to recalculate the bounding box
- Bounding sphere avoids calculation; Why?

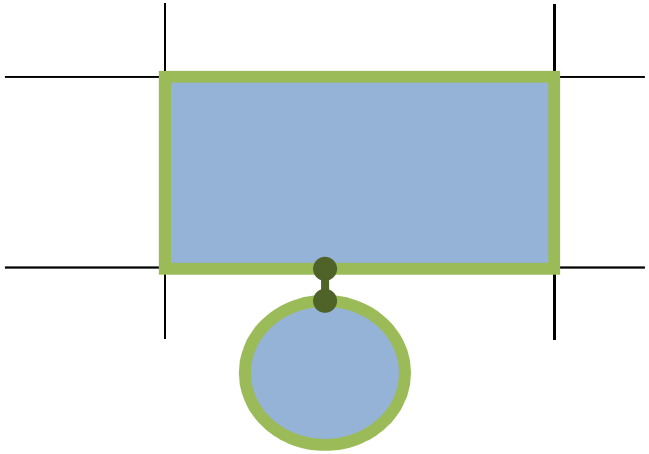


Sphere-Box Intersection

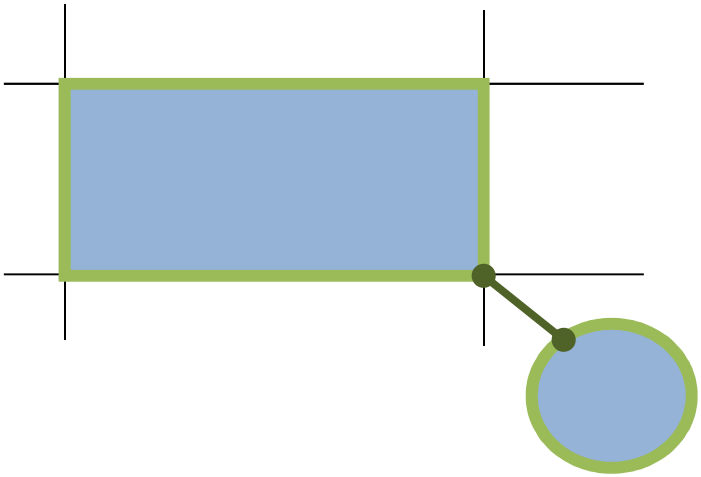
(1)



(2)



(3)

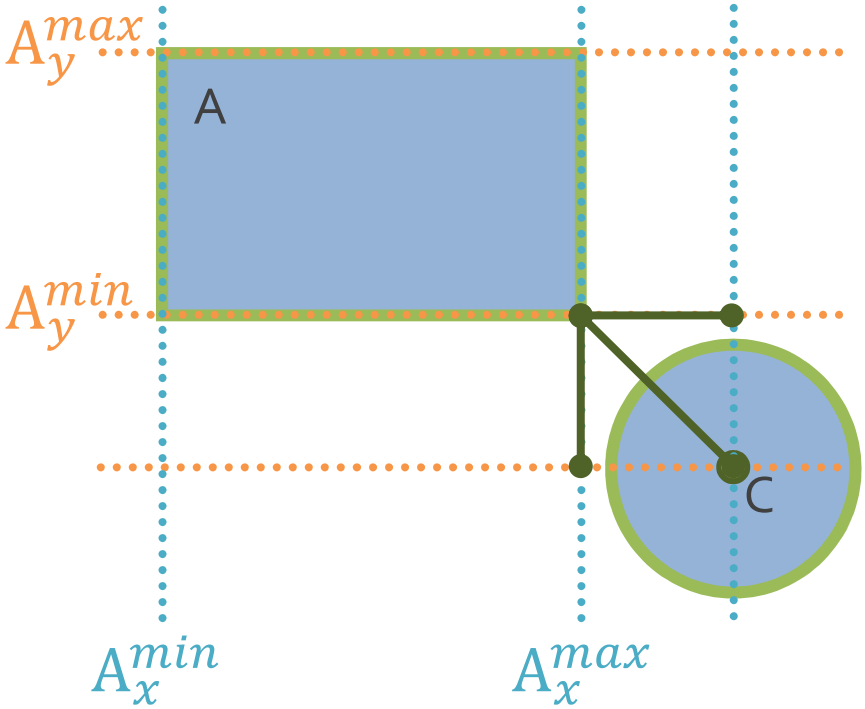


(4)

3D?

Sphere-Box Intersection

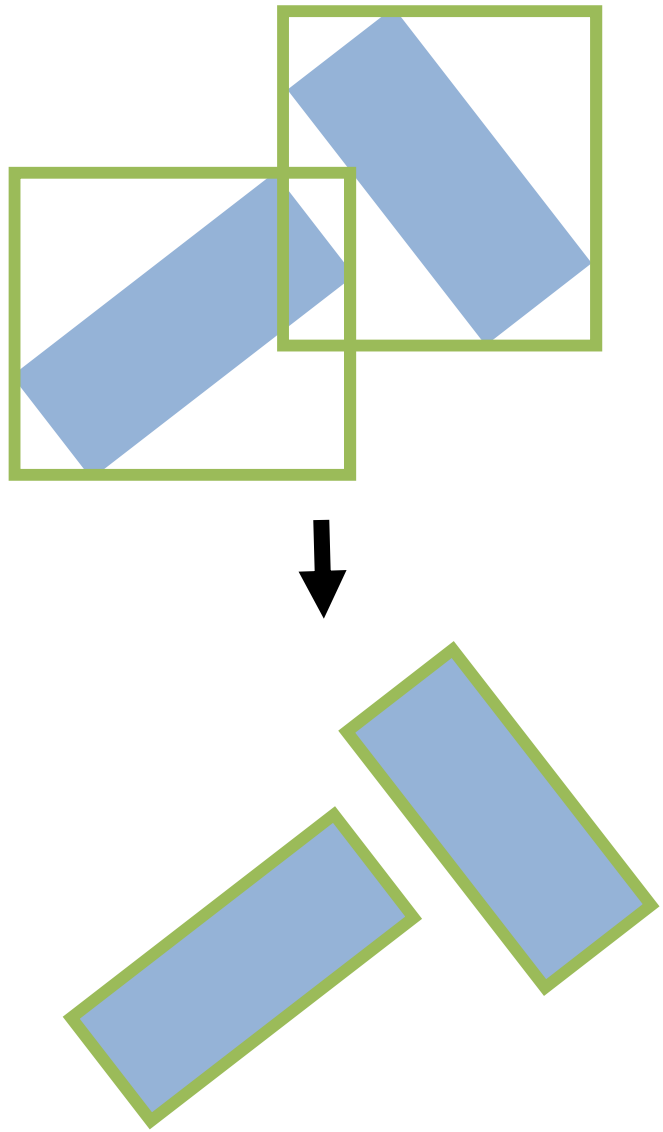
- General: distance > o
- Idea: Coordinate-wise Euclidean distance




```
d = 0
for each i ∈ {x, y, z}
{
  if (C_i < A_i^min)
    d = d + (C_i - A_i^min)^2
  else if (C_i > A_i^max)
    d = d + (C_i - A_i^max)^2
}
if (d > r^2)
  return DISJOINT
else
  return OVERLAP
```

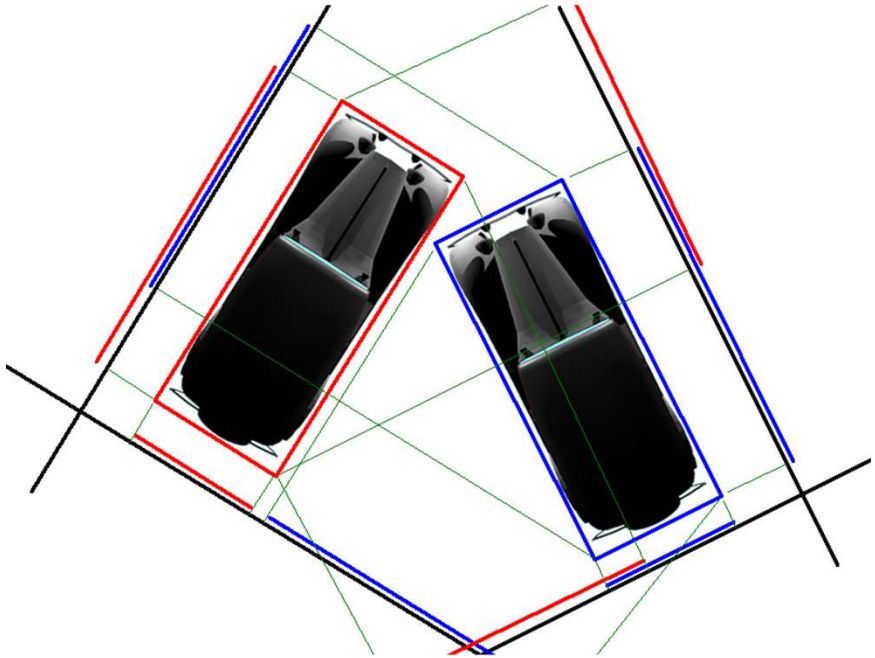
Oriented Bounding Box

- Rotation is no problem
- More complicated to calculate than AABB



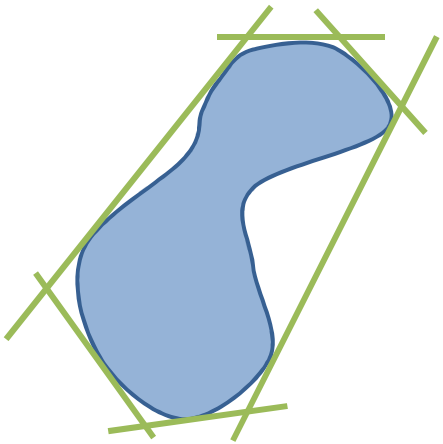
Oriented Bounding Box

- Rotation is no problem
- More complicated to calculate than AABB
- Separating axis theorem still works
- More information
  - www.gamasutra.com
  - Game Prog Gems (I, II, III)



k-DOP

- k-Discrete Oriented Polytop
- OBB and AABB are 6-DOPs
- Optimal bounding boxes
- If convex separating axis theorem applies

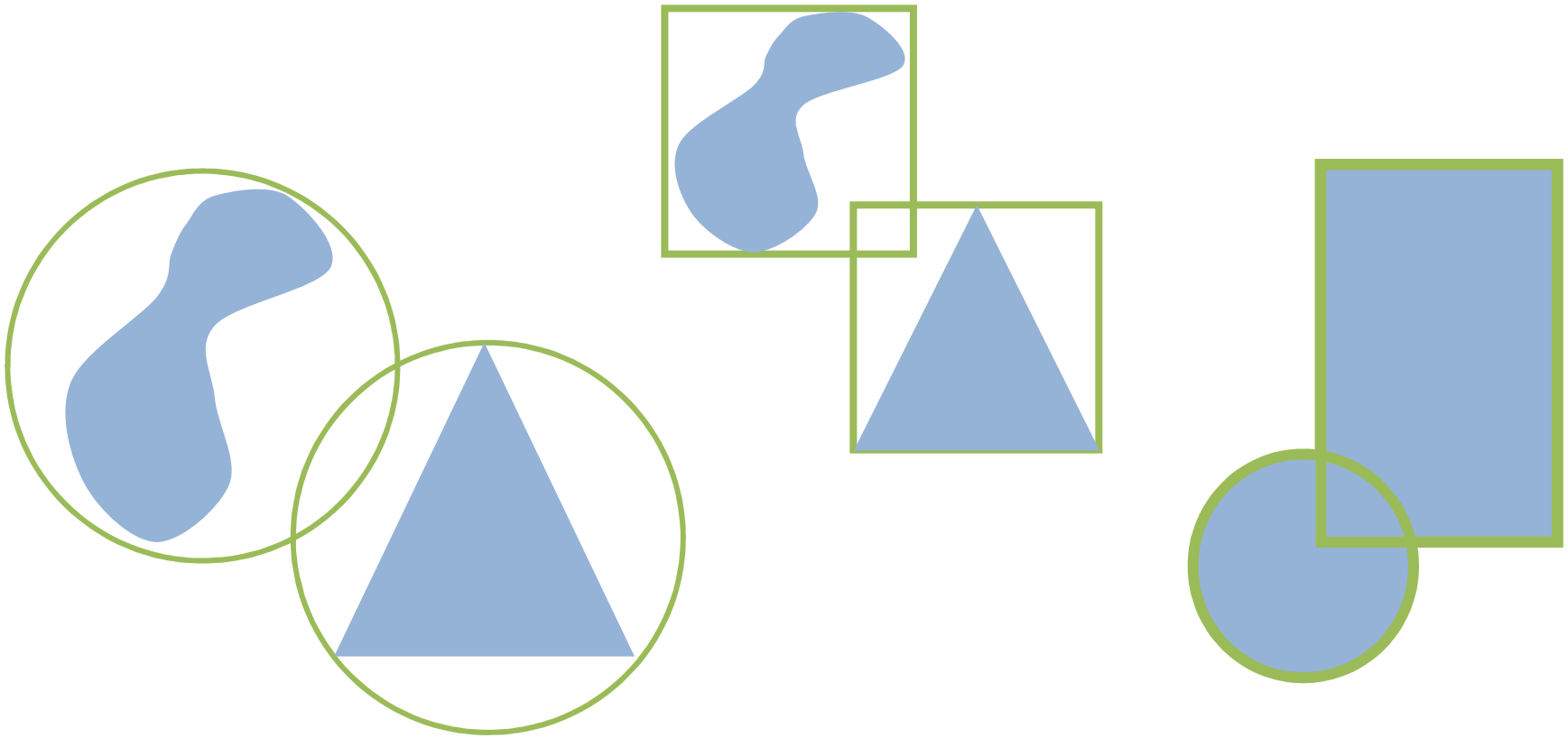


6-DOP

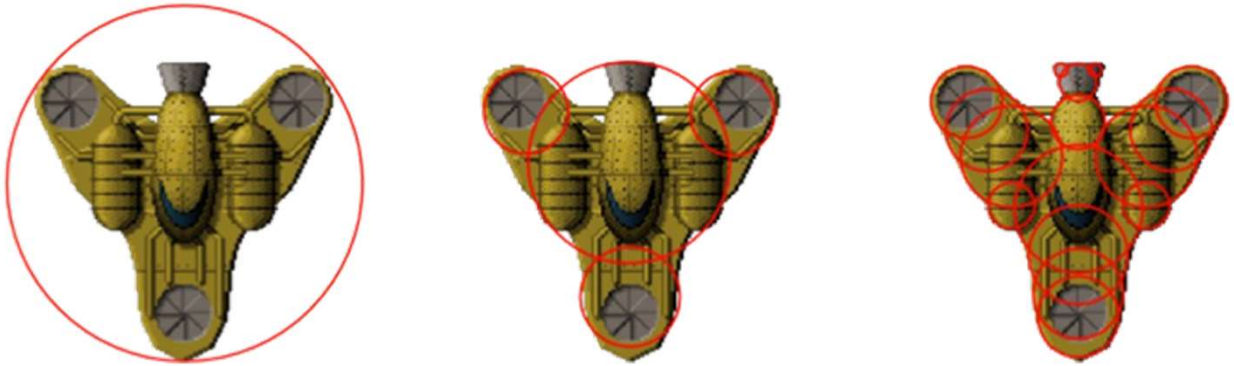
7-DOP

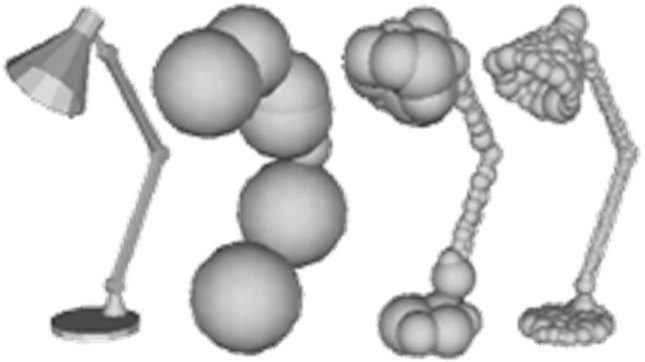
Collision Detection

- Many specialized algorithms for specific geometry
  www.realtimerendering.com/intersections.html



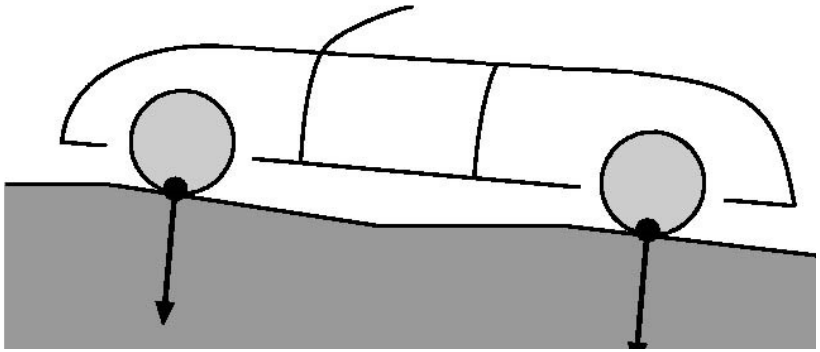
Object Bounding Hierarchy





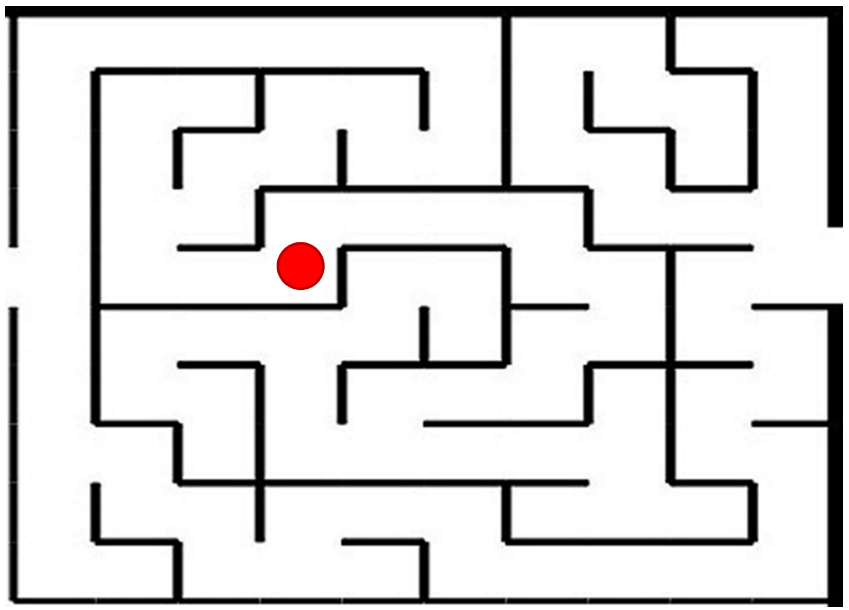
Collision Detection with Rays

- E.x.: car on road, player on terrain
- Test all triangles of all wheels against road geometry
- Often approximation good enough
- Idea: approximate complex object with set of rays



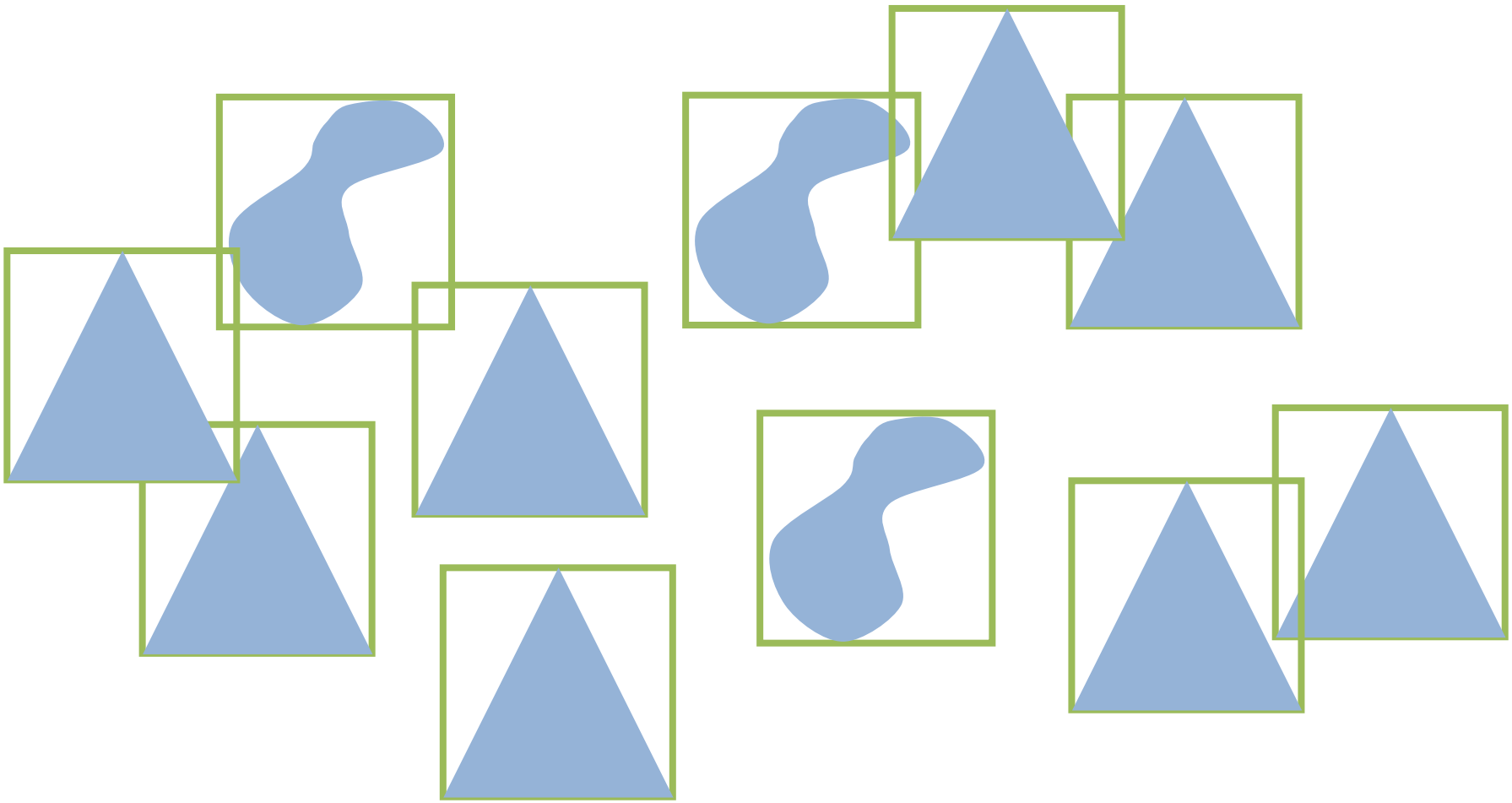
Another Simplification

- Turn 3D into 2D operations
- Example: maze (many first person shooters)
- Approximate player by circle
- Test circle against lines of maze



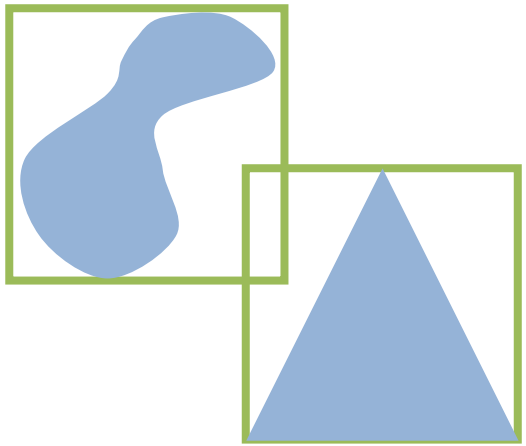
How many collision tests?

- Check each object with every other object $\frac{N \cdot (N-1)}{2} \approx N^2$



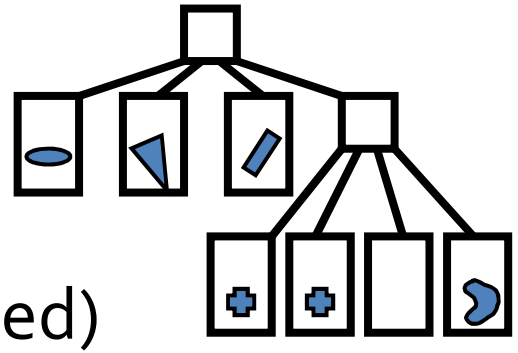
Handling High Numbers of Objects

- Two phases
- Broad Phase (use spacial data structure for speed)
  - Grids
  - Spatial subdivisions hierarchies
  - Sweep and prune
- Narrow Phase
  - Pairwise collision testing

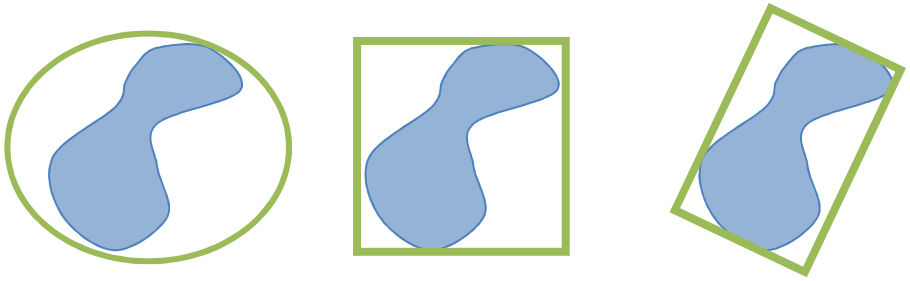


Phases

- Broad Phase (use spacial data structure for speed)
  - Grids
  - Spatial subdivisions hierarchies
  - Sweep and prune



- Narrow Phase (real object is intersected)
  - Bounding objects
  - Point-Line
  - Point-Triangle
  - Triangle-Triangle
  - …

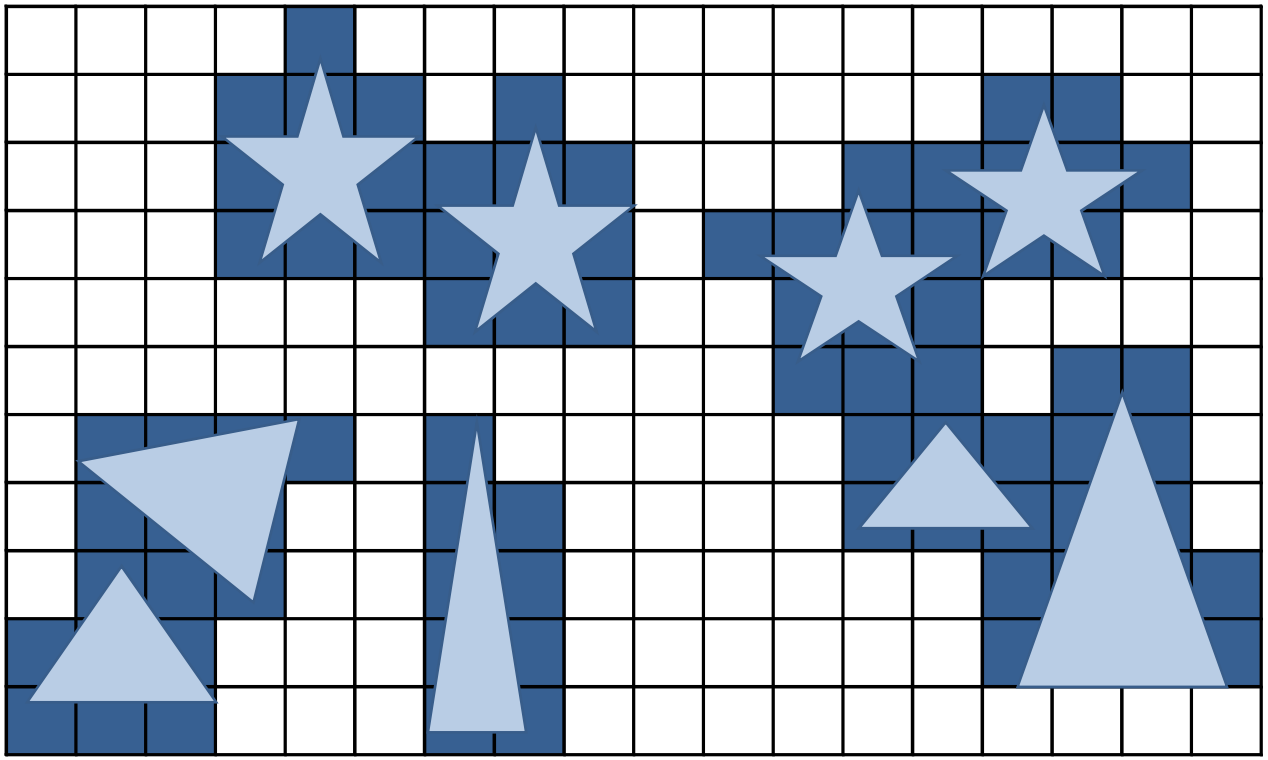


Broad Phase

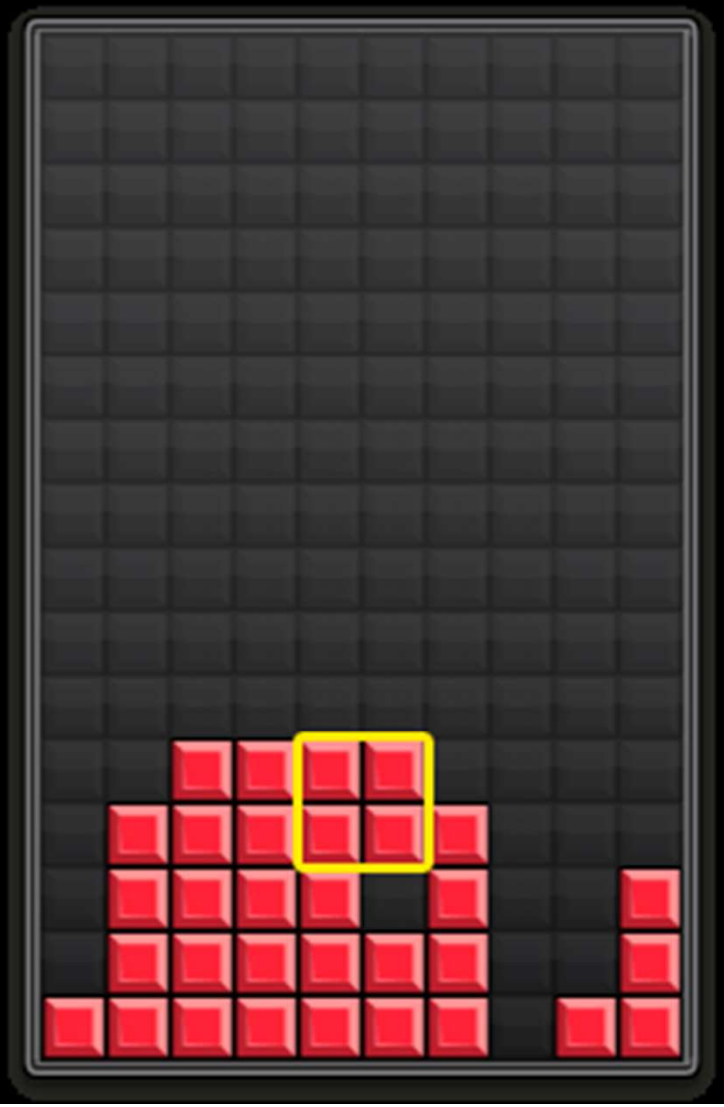
Handling High Numbers of Objects

- Regular subdivion
- Hierarchical subdivision

Regular Subdivision – 2d grid

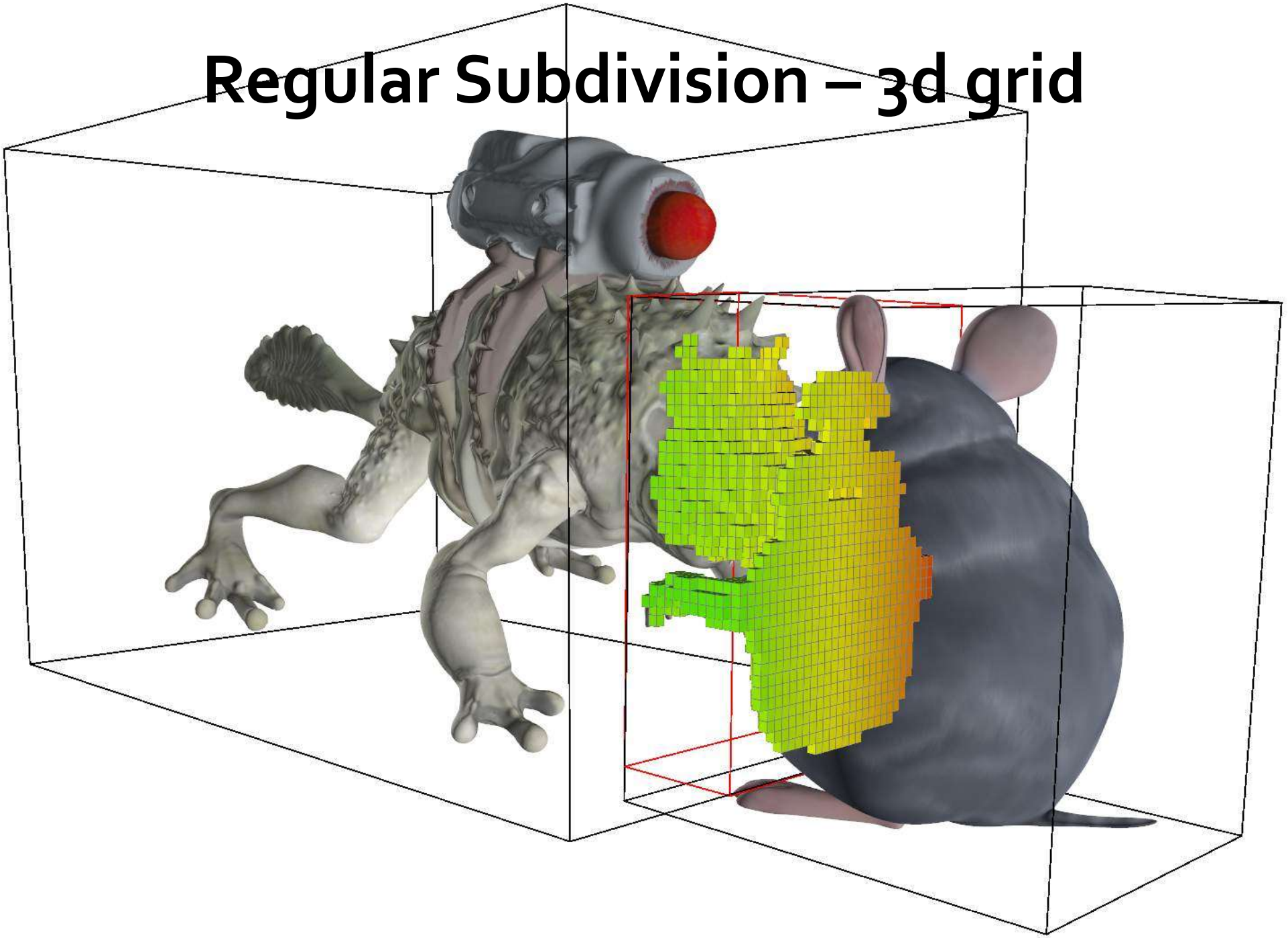


Regular Subdivision – 2d grid



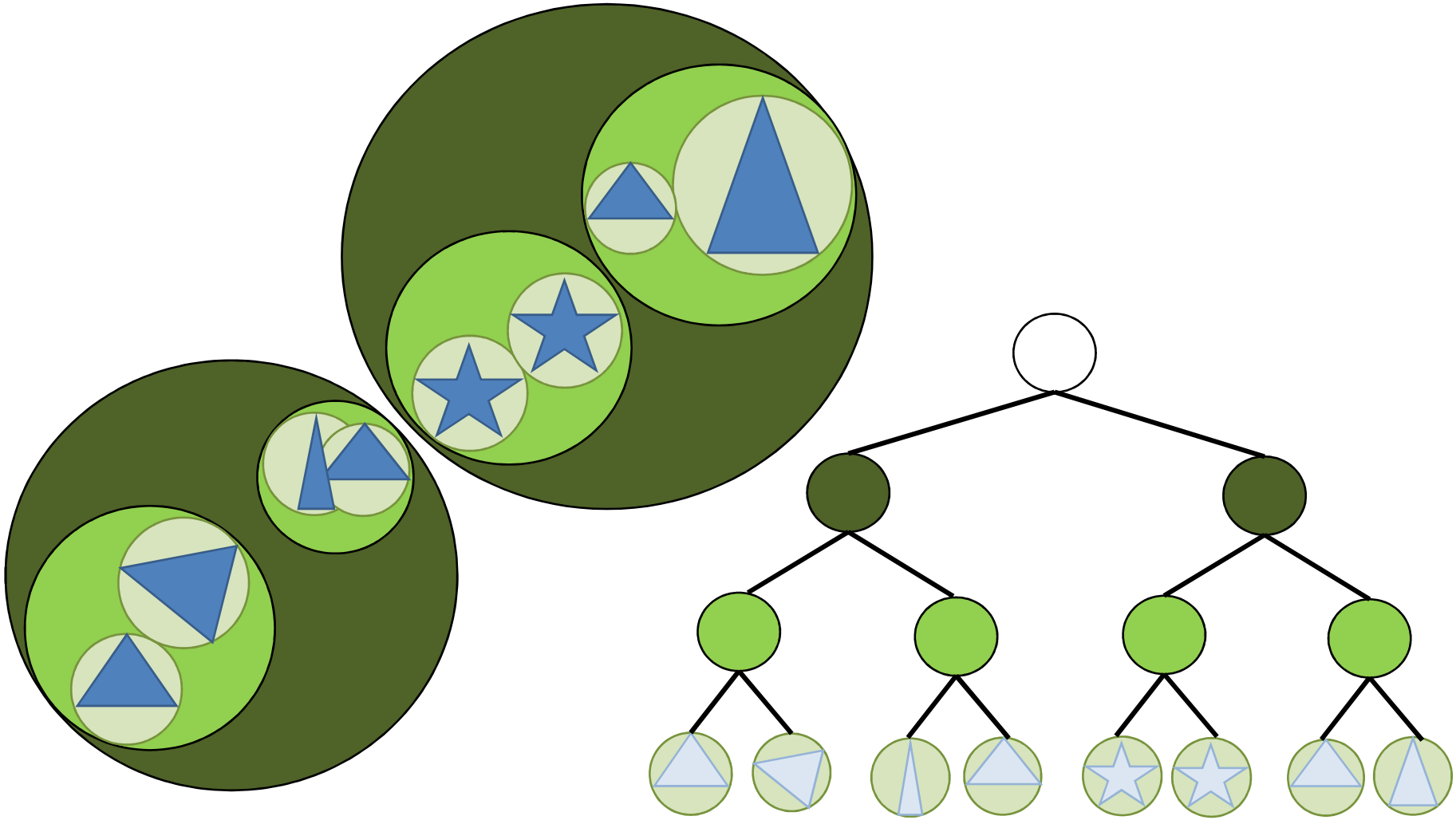
```
[[0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,1,1,0,0,0,0,0,0],
 [0,1,1,1,0,0,1,0,0,0],
 [0,1,1,1,1,0,1,0,0,1],
 [0,1,1,1,1,1,1,0,0,1],
 [1,1,1,1,1,1,1,0,1,1]]
```

Regular Subdivision – 3d grid



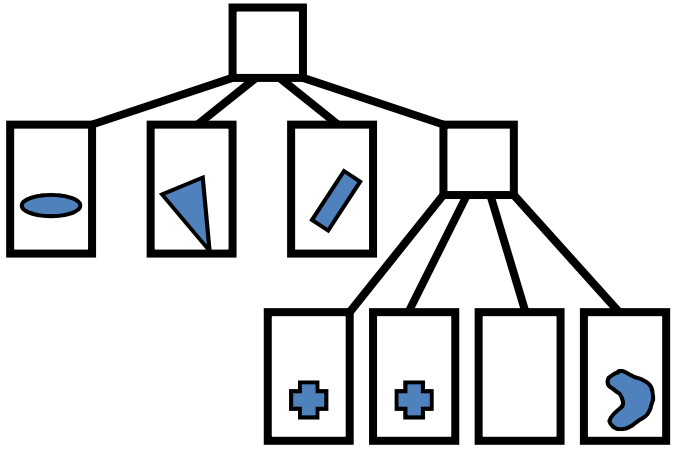
Hierarchical subdivision – BVH

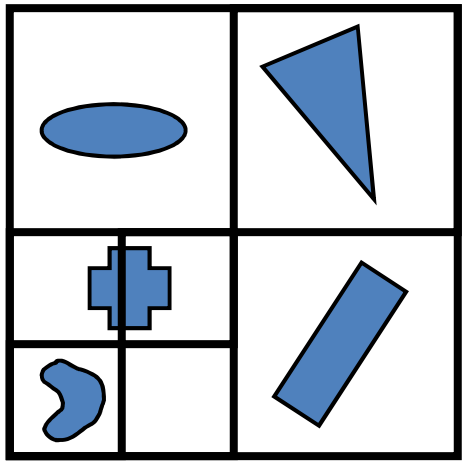
- Bounding Volume Hierarchy = BVH



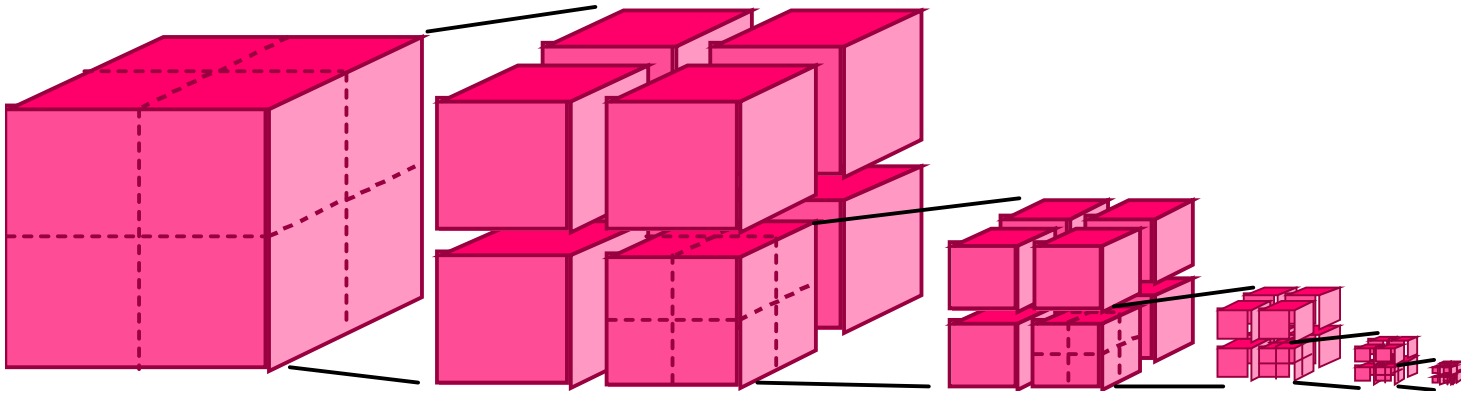
Quad/Octrees

- Quadtree (2D)

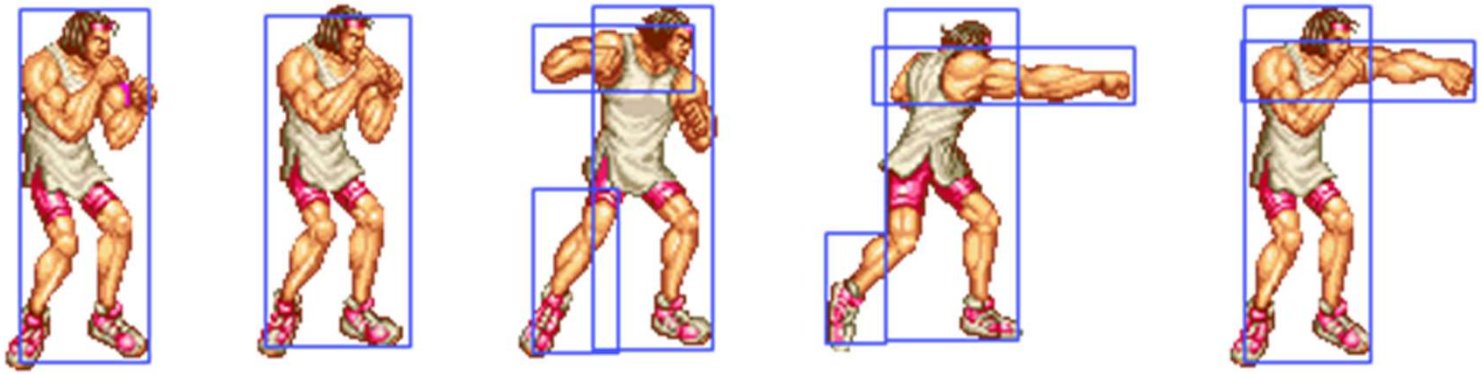




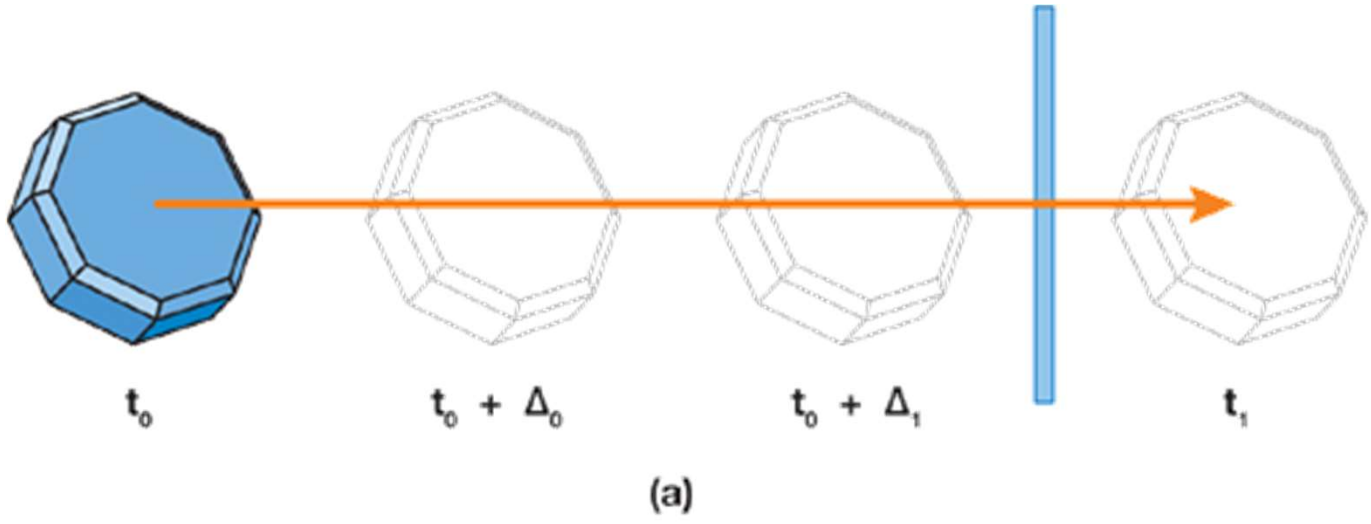
- Octree (3D)



Animated Objects



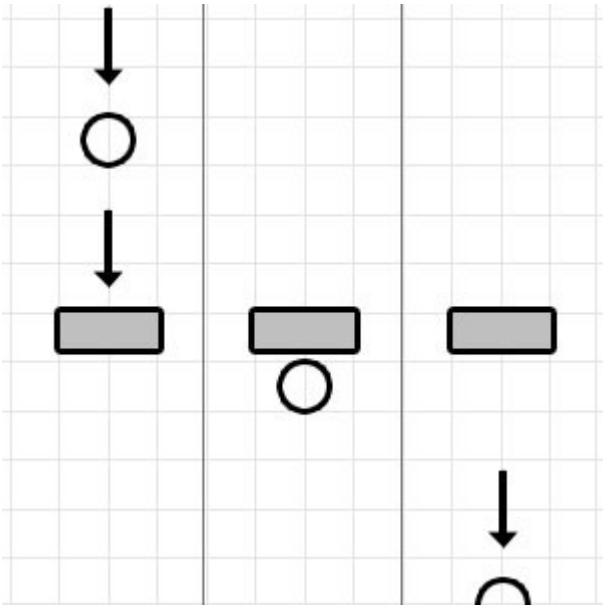
Trouble with Animated Objects

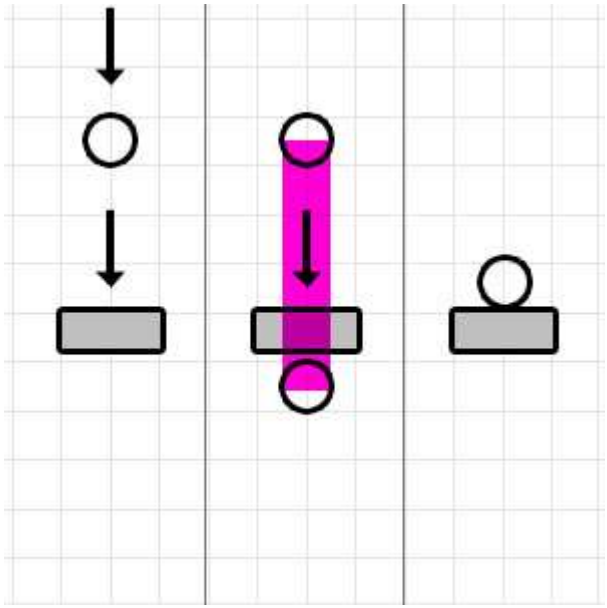


(a)

(b)

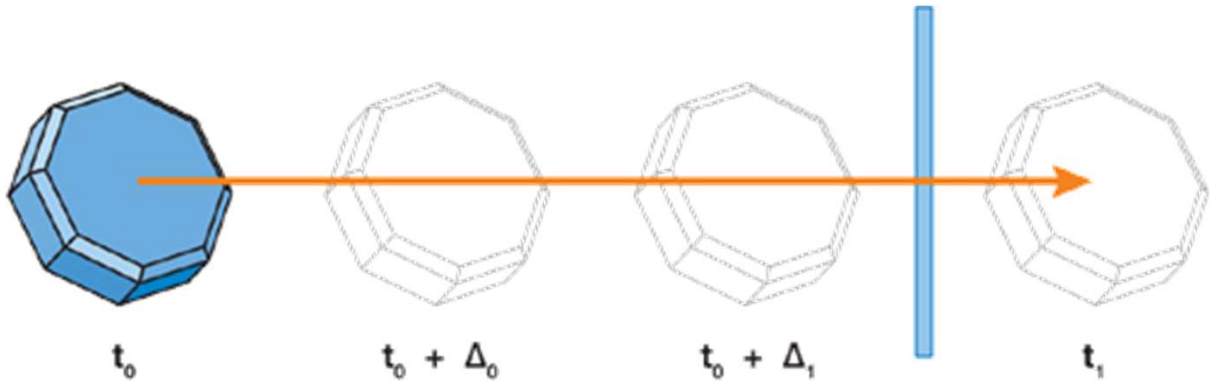
Trouble with Animated Objects





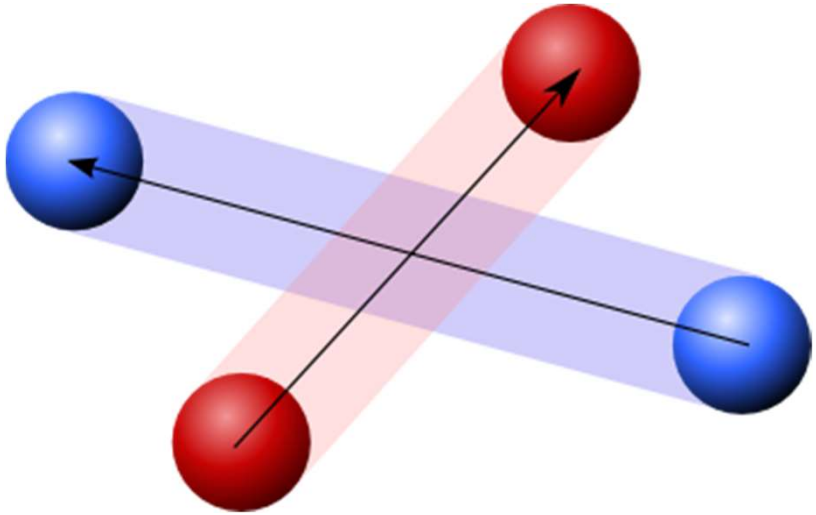
A posteriori (Discrete)

- Advance physics by time step then check for collision
- Simple
  - List of objects → return list of intersections
  - No time variable in calculations
  - Miss actual time of collision
- Need to "fix"



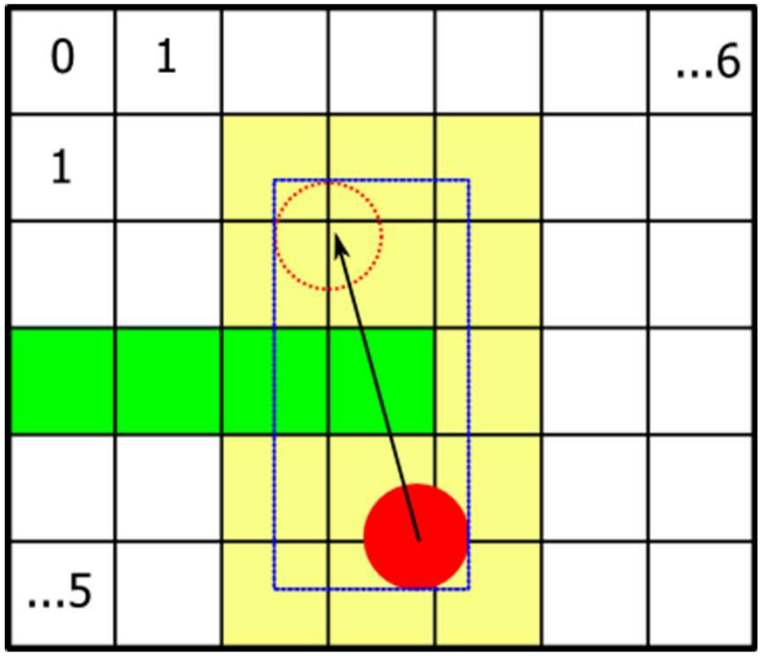
A priori (Continuous)

- A priori (continuous)
  - Predict future movement
    - Trajectories
  - Can be more precise
  - Can be more stable
  - More complex
    - Dimension of time
    - Often no closed form solution (numerical approach)
    - Aware of how objects move
      - Elastic objects (deforming)



Animated Objects - Practical Solutions

- Use extruded geometry
- Use overesized geometry
- …
- Cast ray(s)
- Evaluate often enough
  - Restrict speed
- Extensive testing
- Some cases will be missed



Remember Game Loop

```
while(not finished)
{
  input = getInput();
  UpdateGameState(input);
  DrawGameWorld();
  WaitForNextFrame();
}
```

Independent Render and Game Loop

- Do update in predefined intervals
  - Specify maximum speed in game
  - Specifiy minimal thickness of bounding geometry
- Independent from rendering loop
  - Slow rendering does not impact update cycle

```
Main() {
  OnUpdate += UpdateLoop;
  OnRender += RenderLoop;
  RunLoops();
}

UpdateLoop() {
 input = getInput();
 UpdateGameState(input);
}

RenderLoop() {
  DrawGameWorld();
  WaitForNextFrame();
}
```