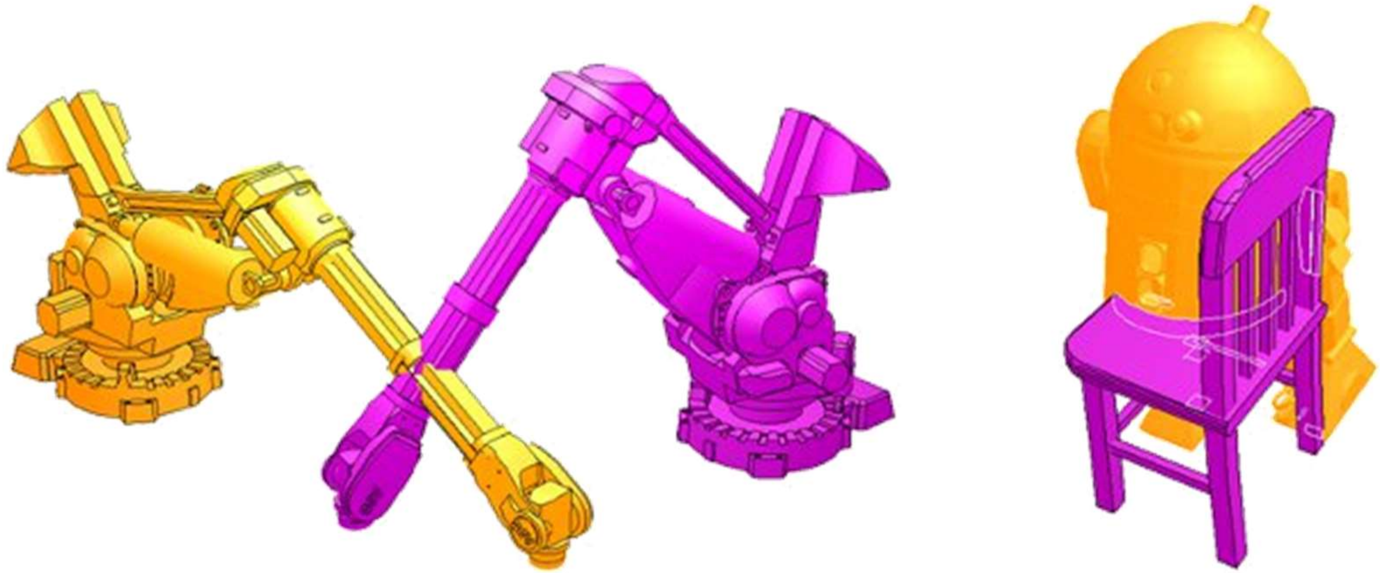
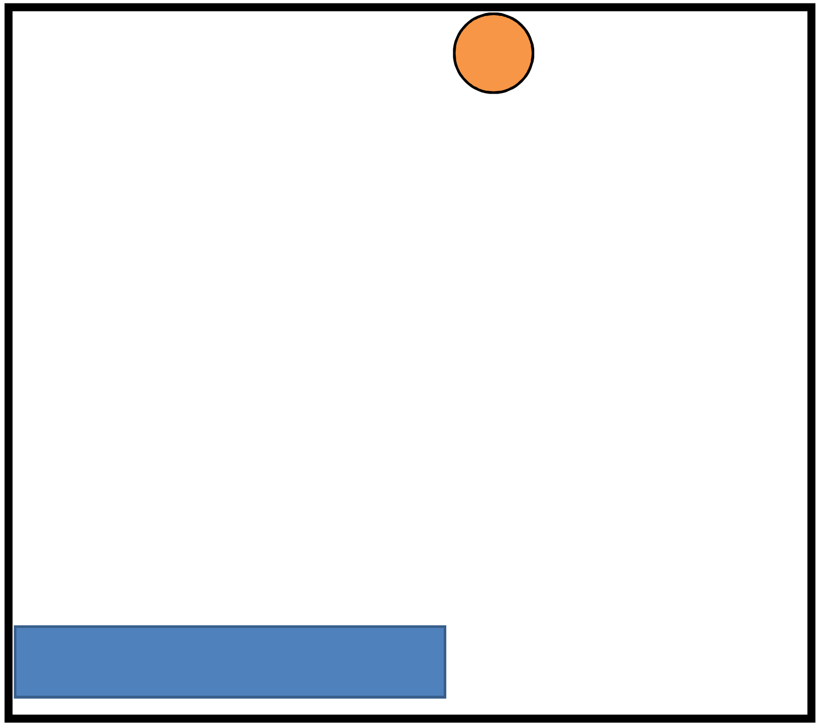
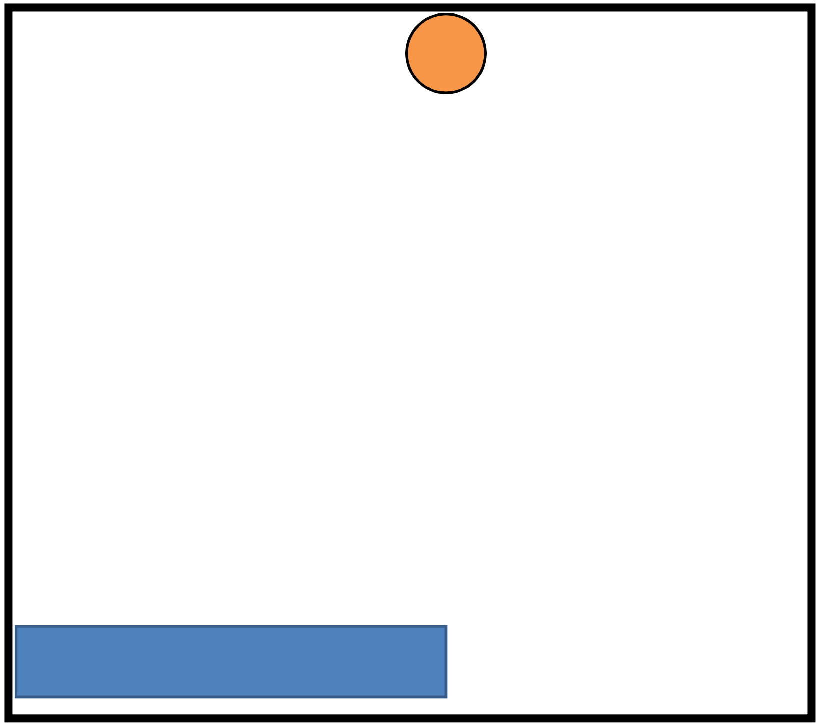
Handling Collisions



Why?

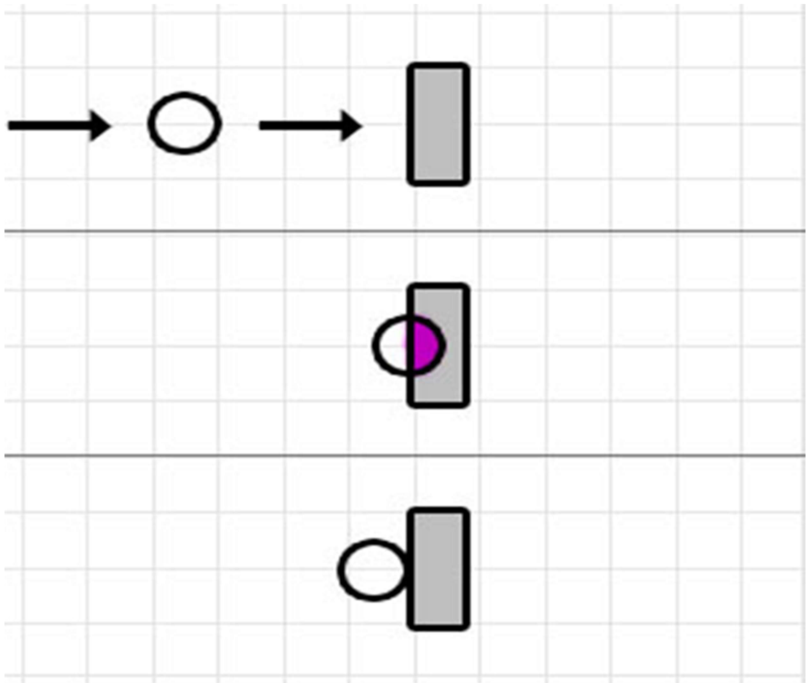
- Realisme / game play
  - Without objects pass through other objects





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?

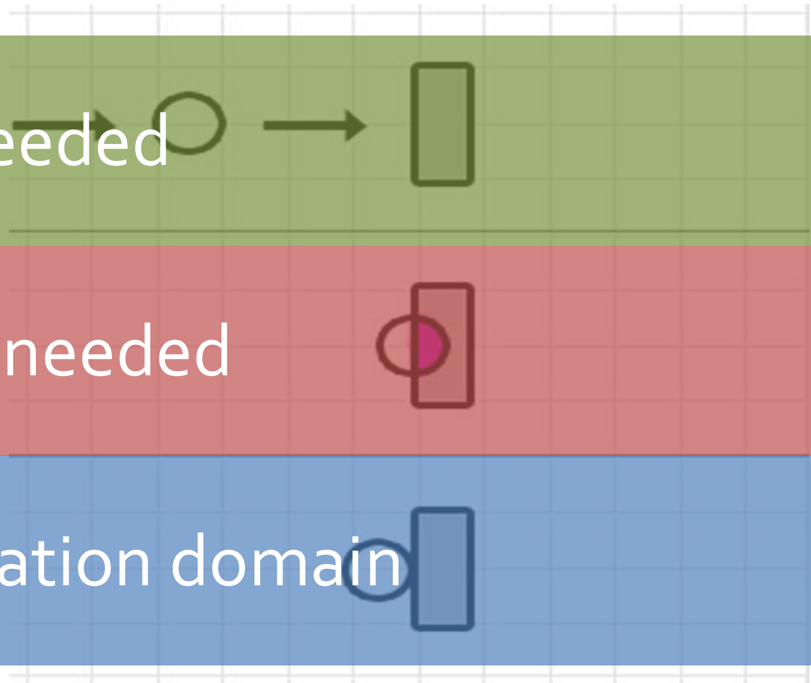
Always needed

- Collision determination
  - Where do they collide?

Not always needed

- Collision response
  - What happens now?

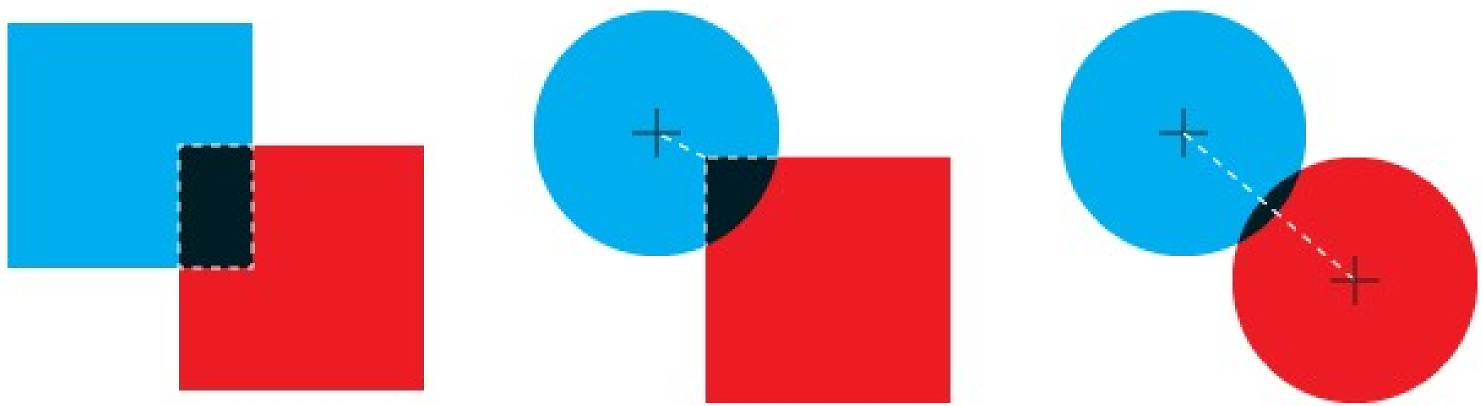
Specific to application domain



Collision Detection

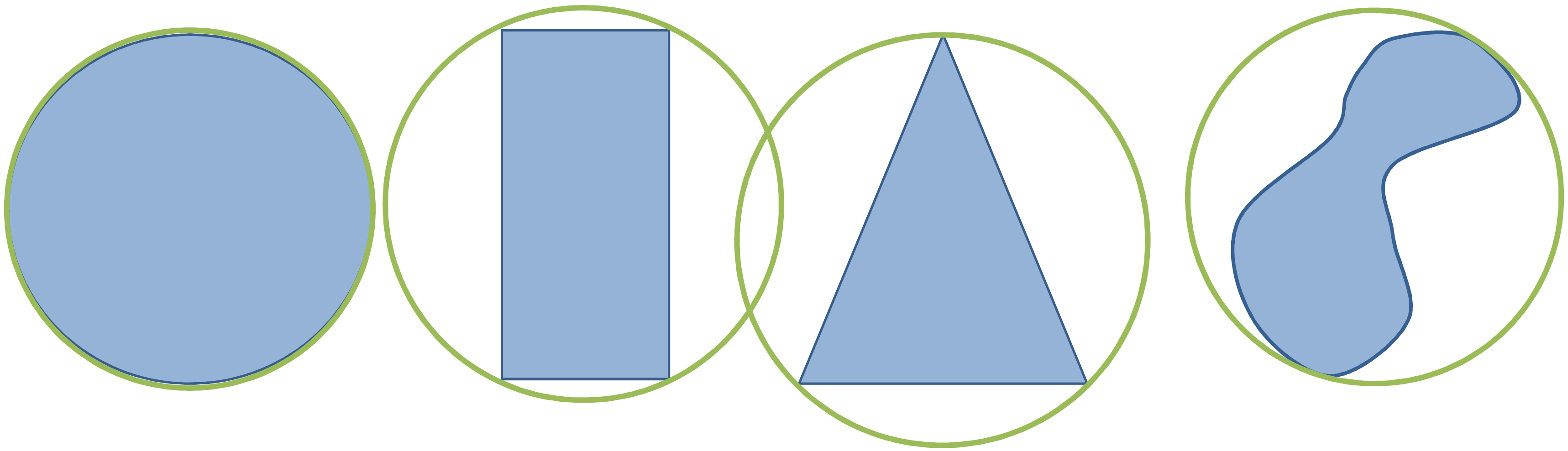
Collision Detection

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



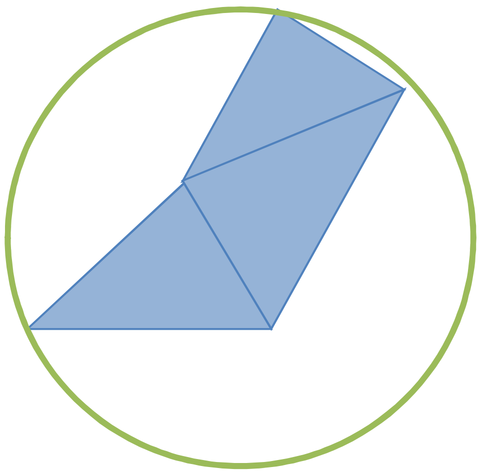
Practical Collision Detection

- Start with coarse approximation
  - Instead of complex objects bounding geometry is used
- Often exact result not needed
  - In games often one type of bounding geometry is used for all objects



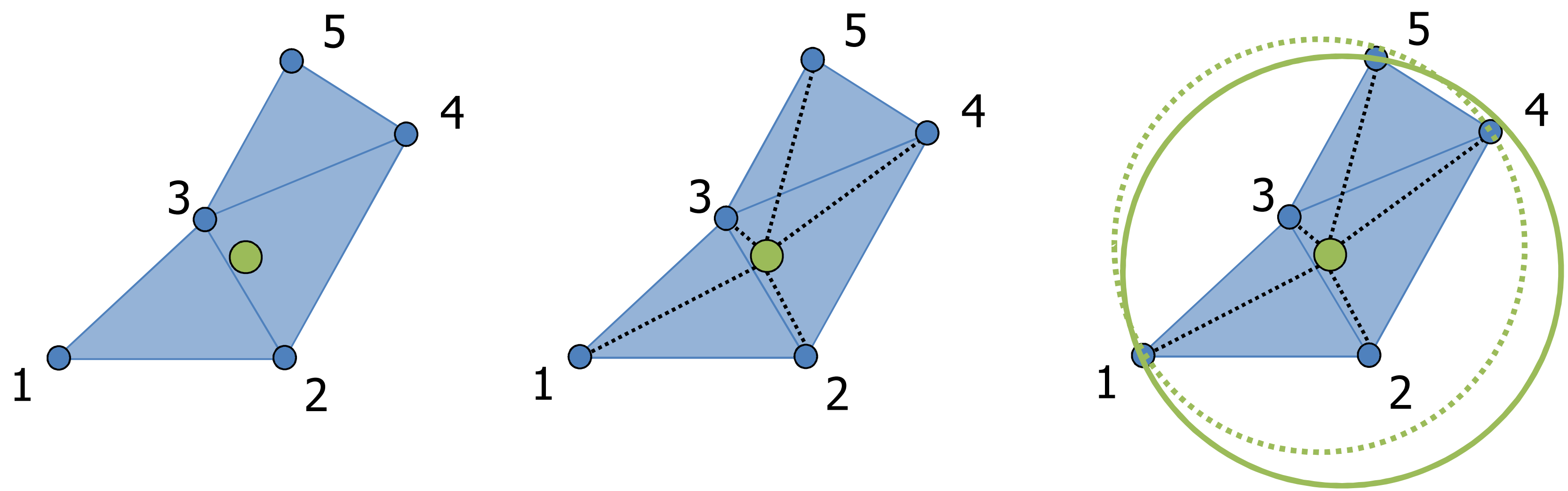
How do we find a bounding geometry?

- 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

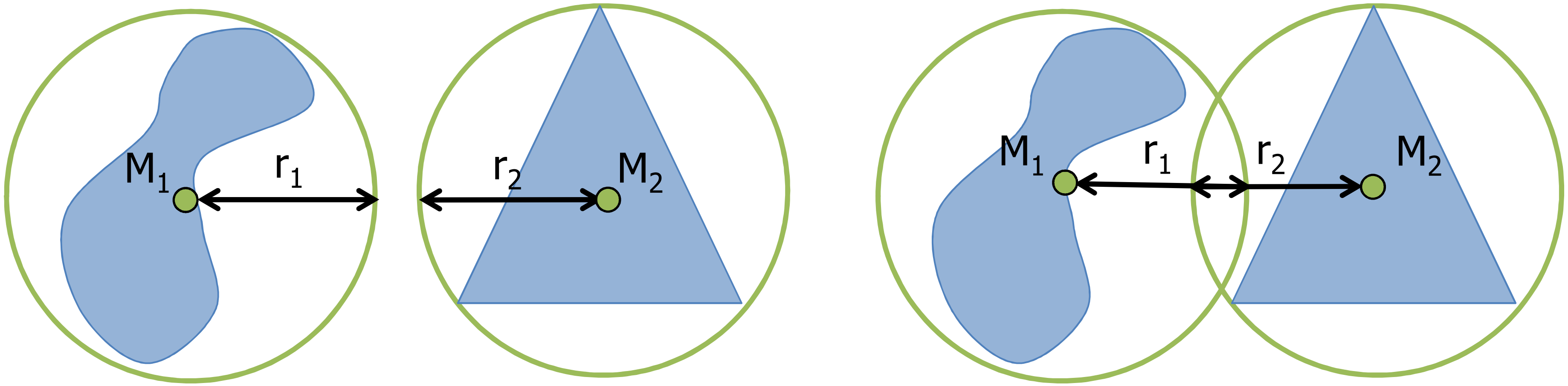


Bounding Sphere – Collision Detection

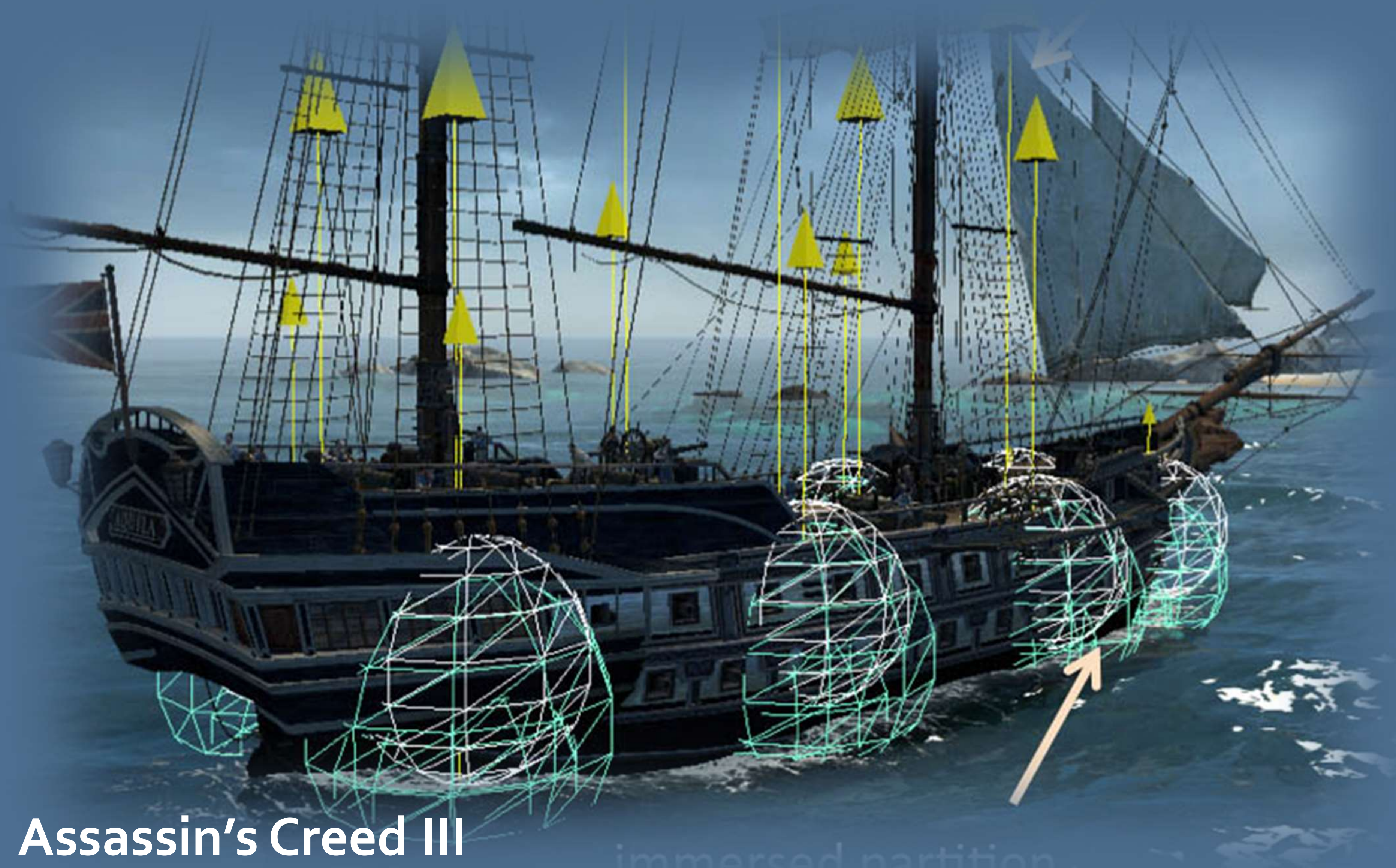
- Collision iff

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

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



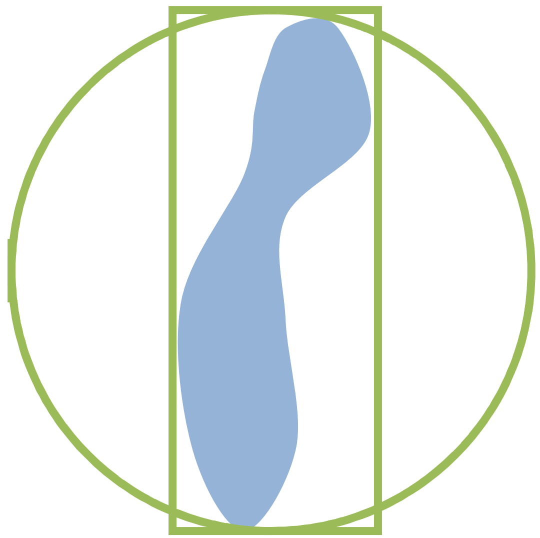
Creative Use of Bounding Spheres

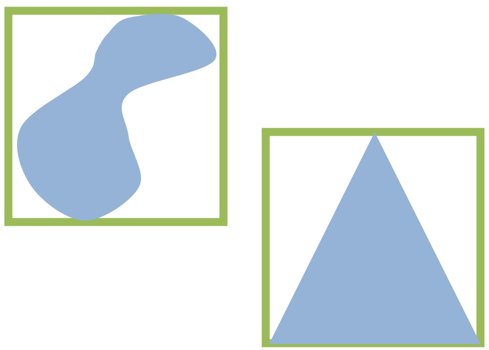


Assassin's Creed III

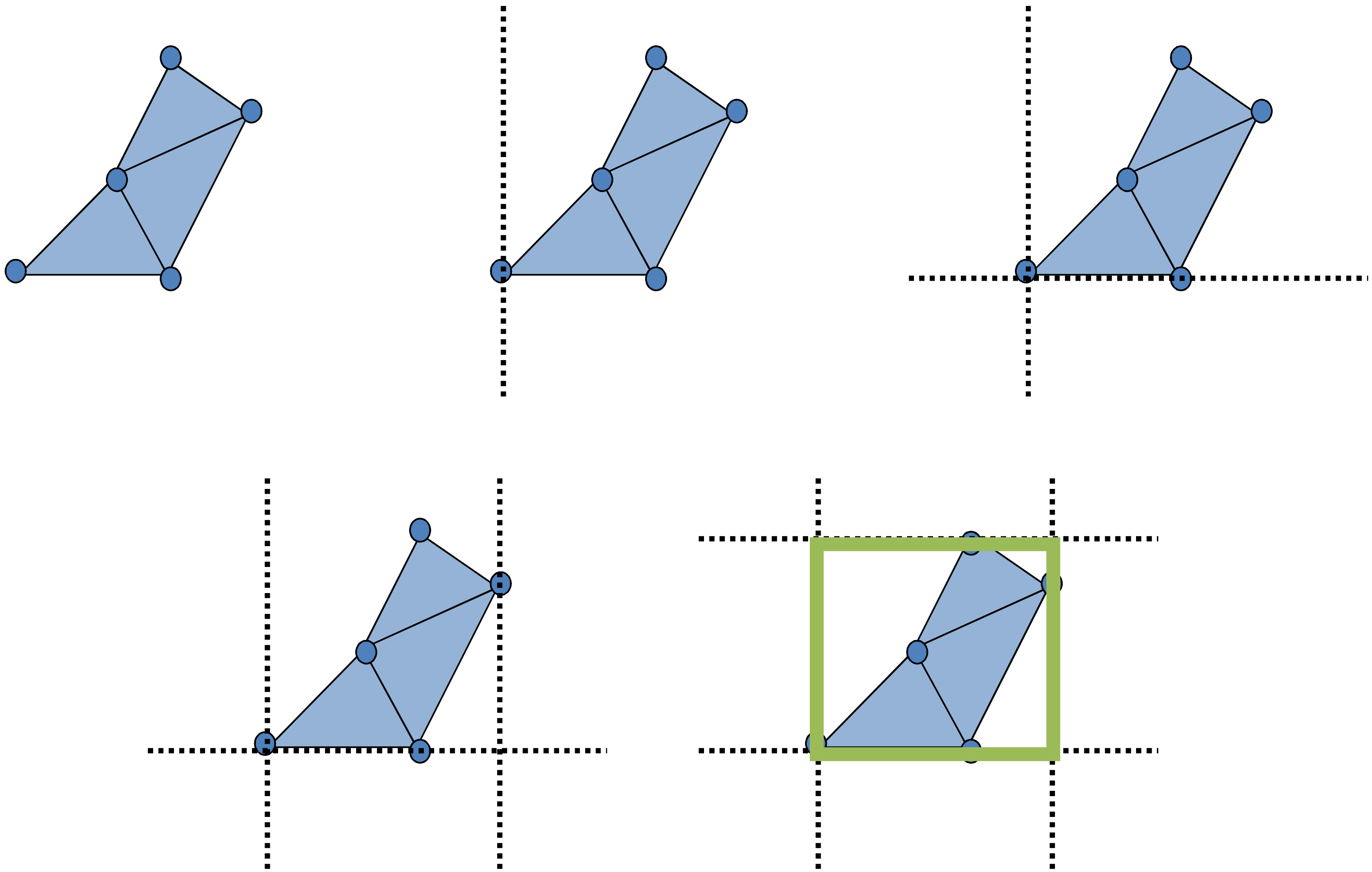
AABB-Algorithm

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



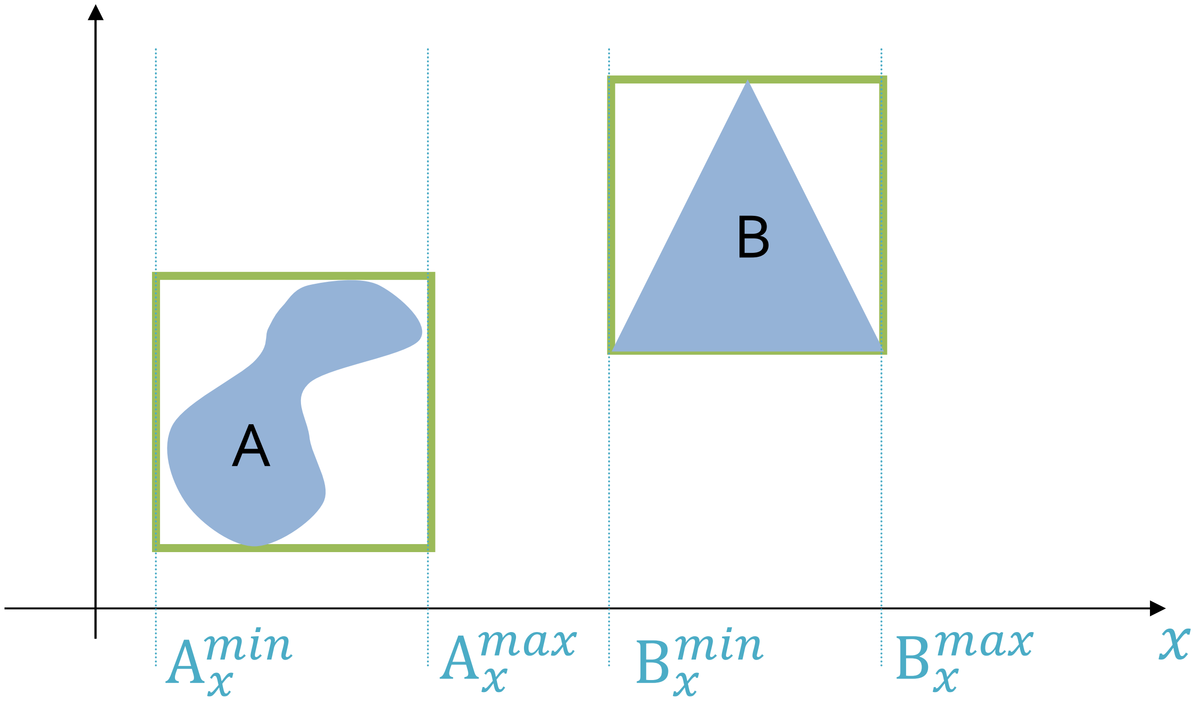


AABB – Calculation



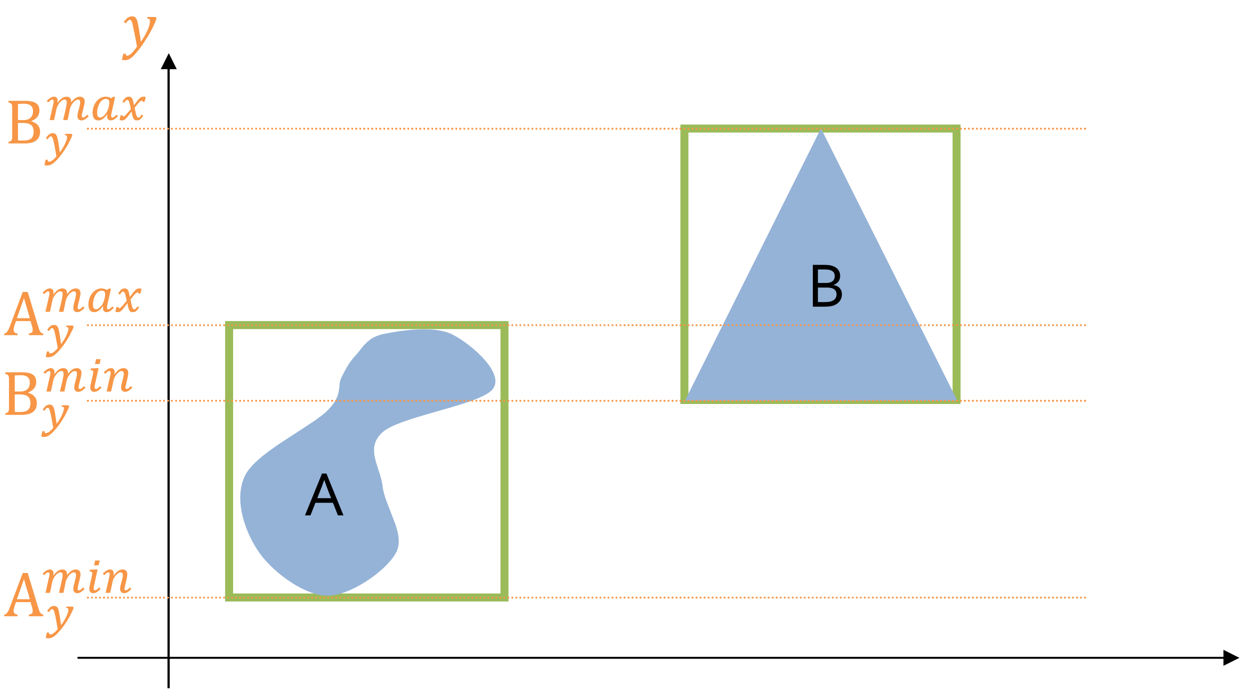
AABB-Algorithm

- No collision if
- $\left(A_x^{min} > B_x^{max}\right)\ or\ \left(B_x^{min} > A_x^{max}\right)$



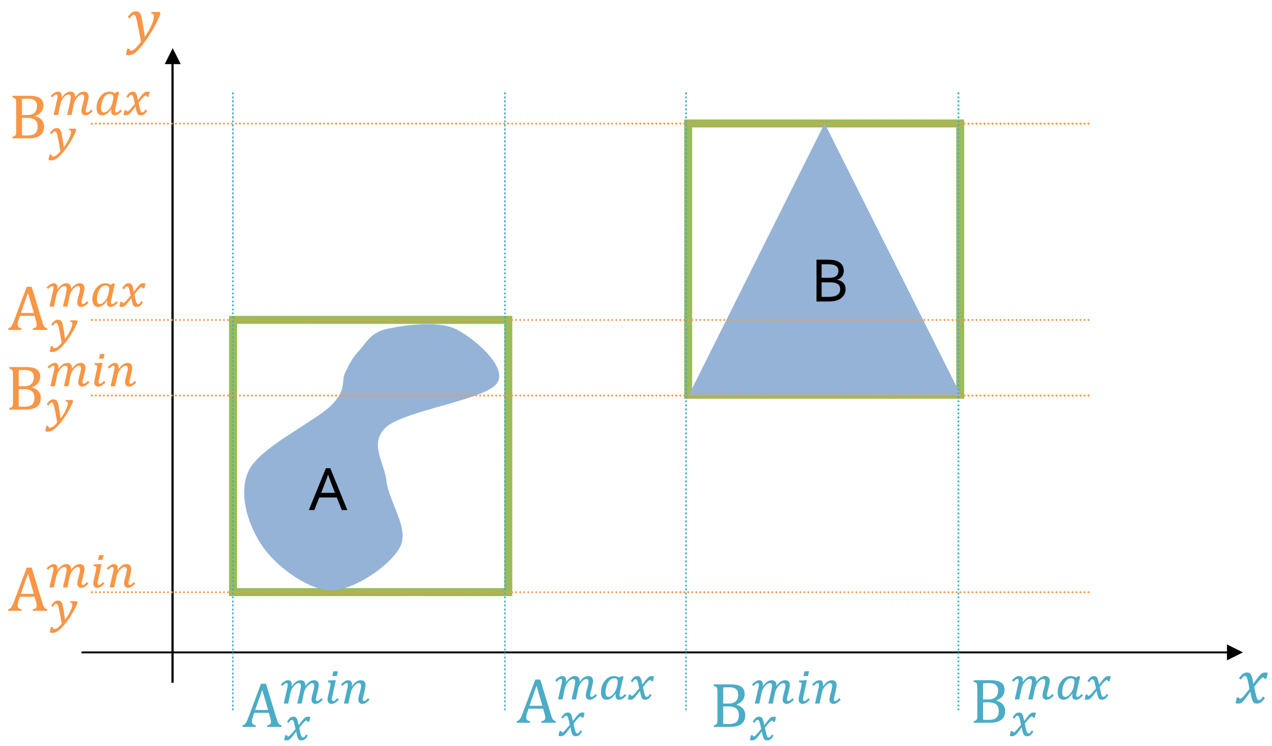
AABB-Algorithm

- No collision if
- $\left(A_y^{min} > B_y^{max}\right)\ or\ \left(B_y^{min} > A_y^{max}\right)$



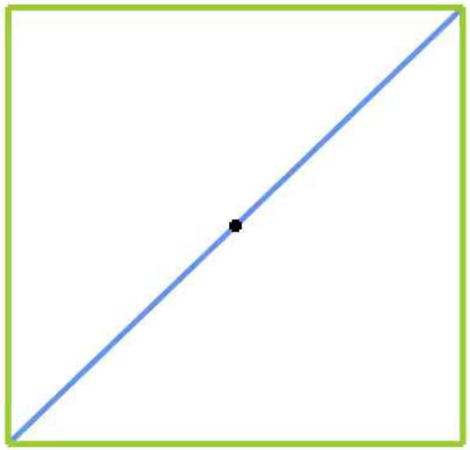
AABB-Algorithm

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

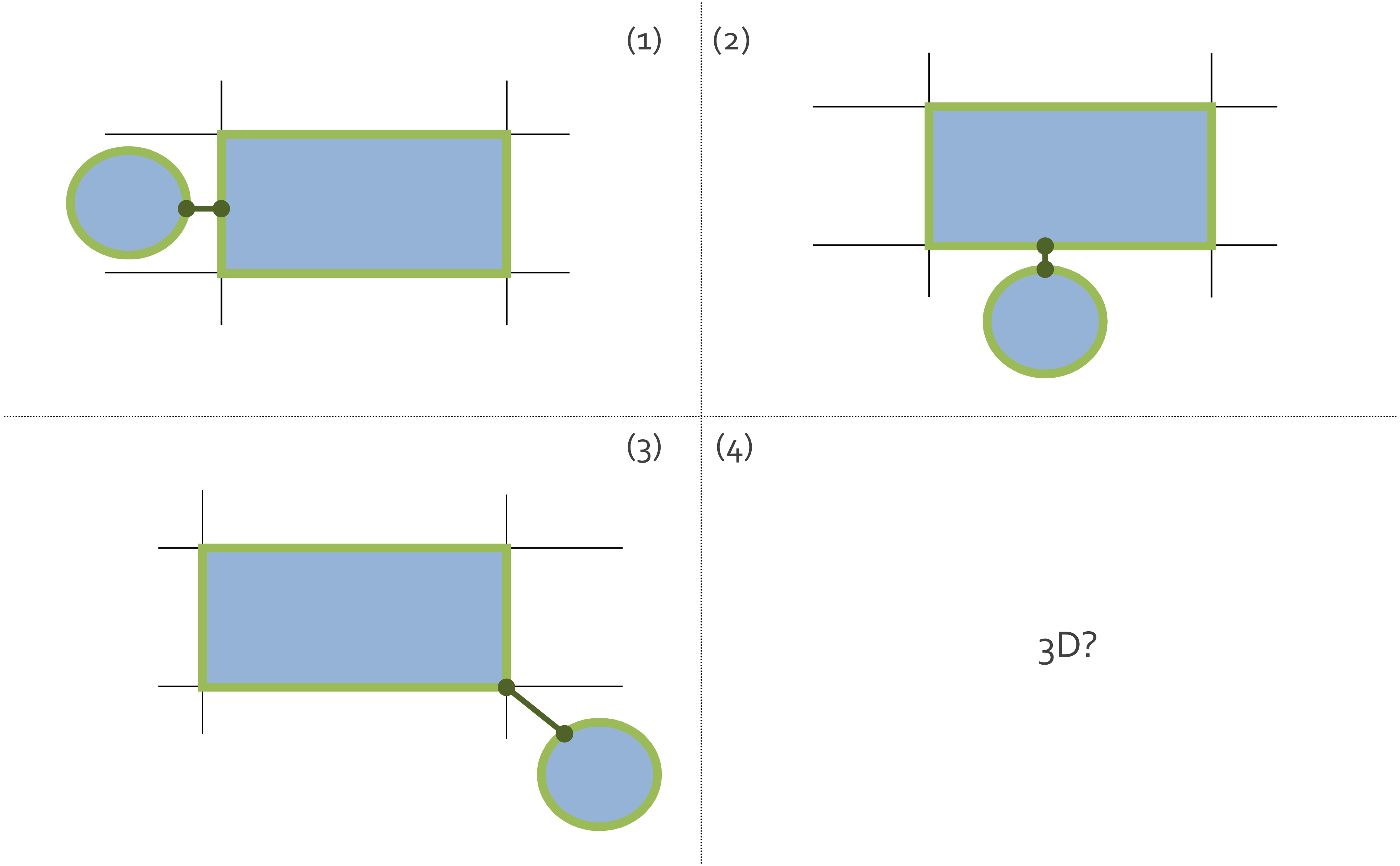


AABB - Problems

- While rotating an object, we have to recalculate the bounding box
- Avoid recalculation with Bounding sphere

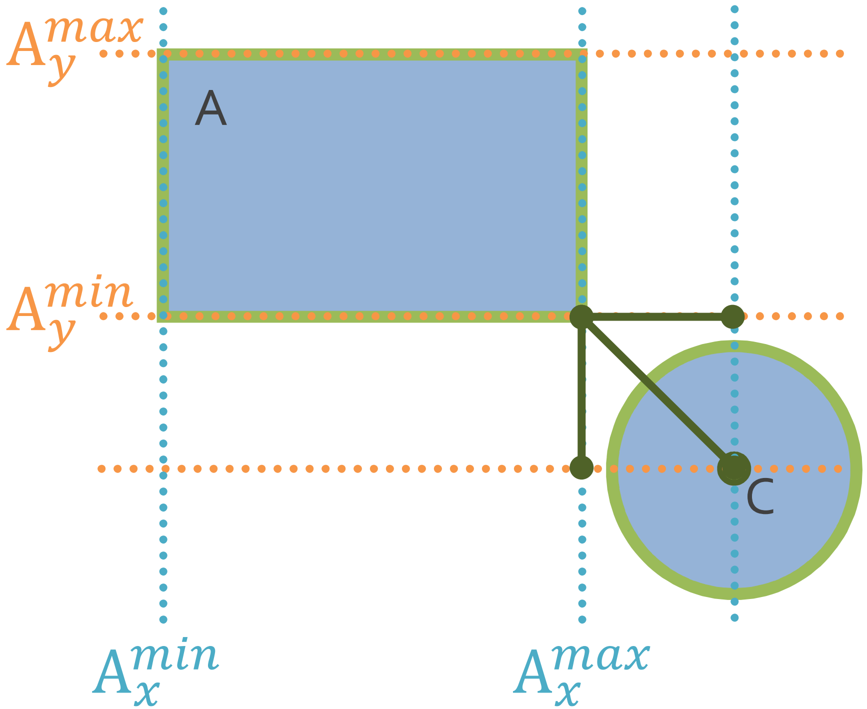


Sphere-Box Intersection



Sphere-Box Intersection

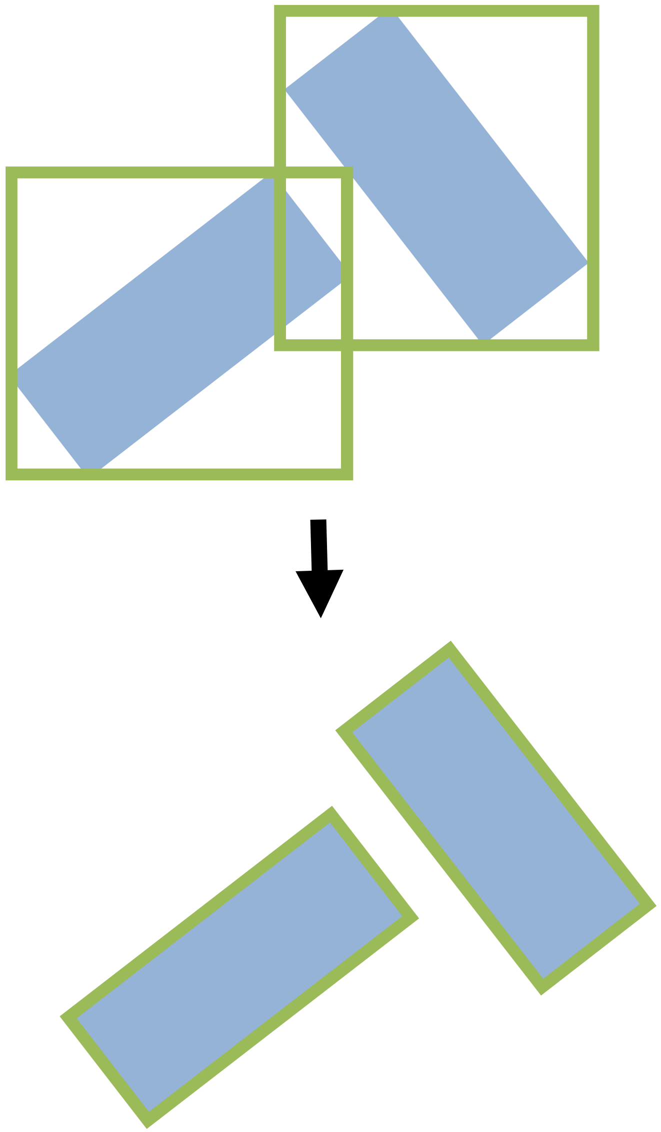
- Idea: Coordinate-wise Euclidean distance




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

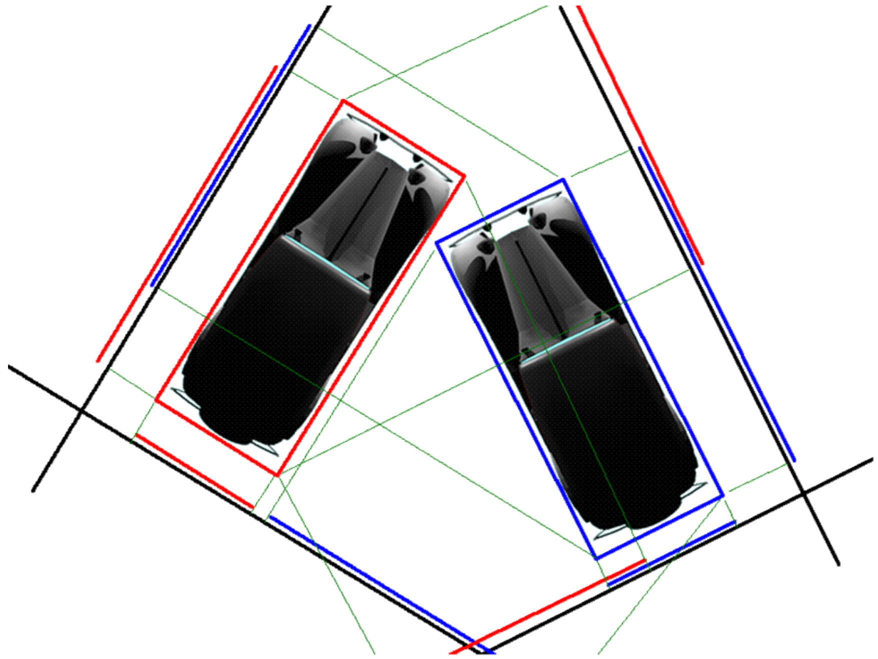
Oriented Bounding Box

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



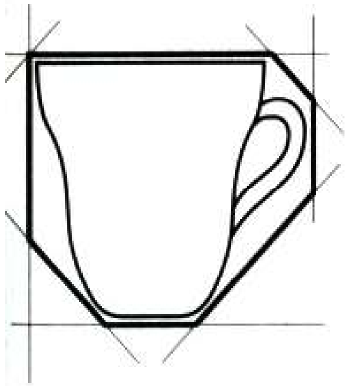
Oriented Bounding Box

- 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)

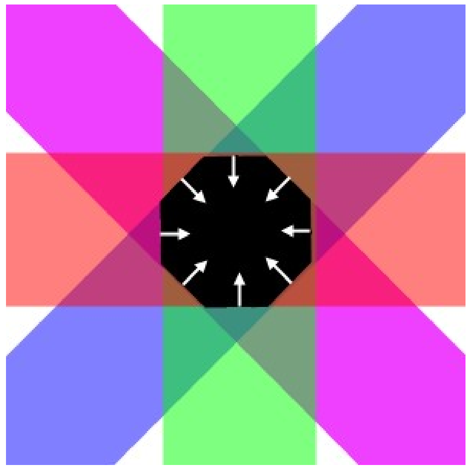


k-DOP

- k-Discrete Oriented Polytop
- OBB and AABB are 6-DOPs
- Optimal bounding boxes

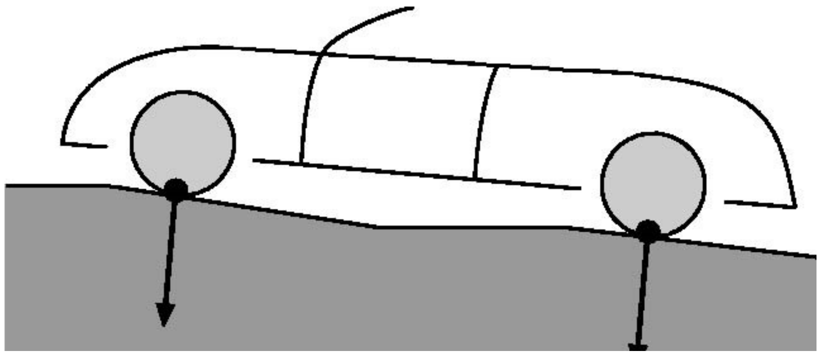


- If convex separating axis theorem applies

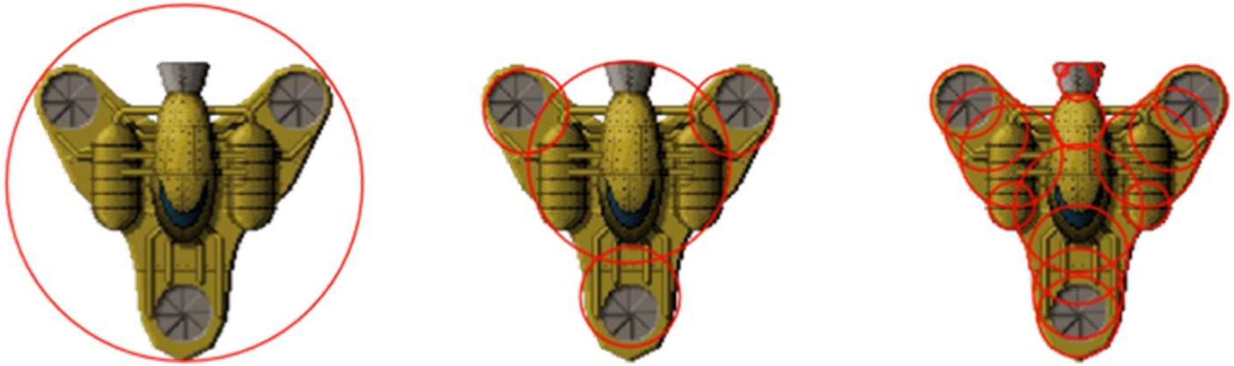


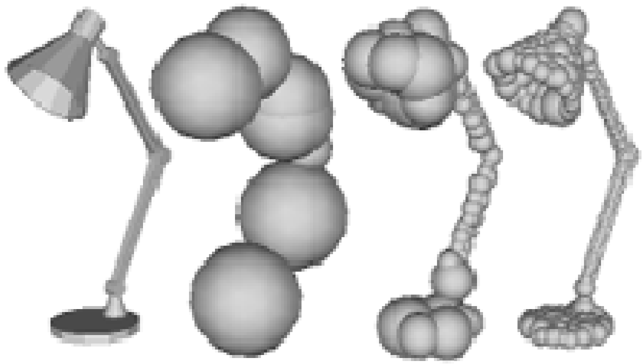
Collision Detection with Rays

- 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



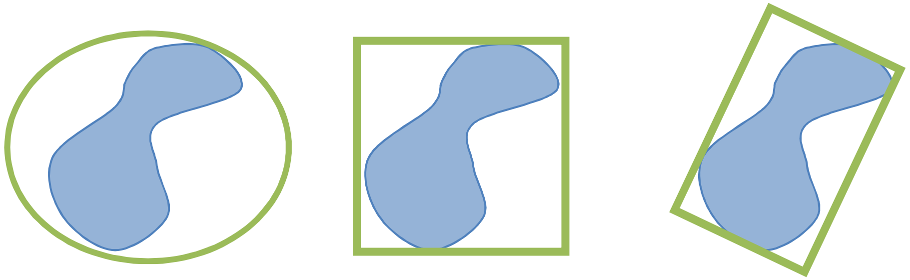
Hierarchy Trees





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
  - ...

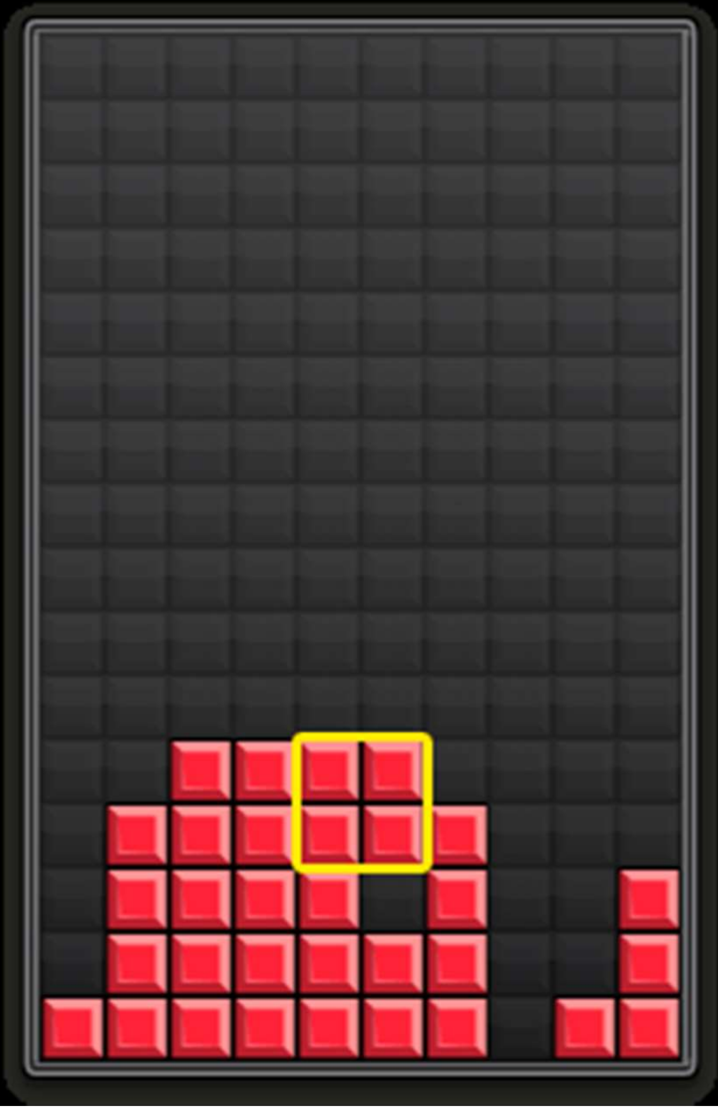


Broad Phase

Handling High Numbers of Objects

- Have to check each object with every other
  - $N \cdot (N1) \approx N^2$
- Hierarchical iregular subdivision
- Hierarchical Regular suvdivision
- Regular subdivion

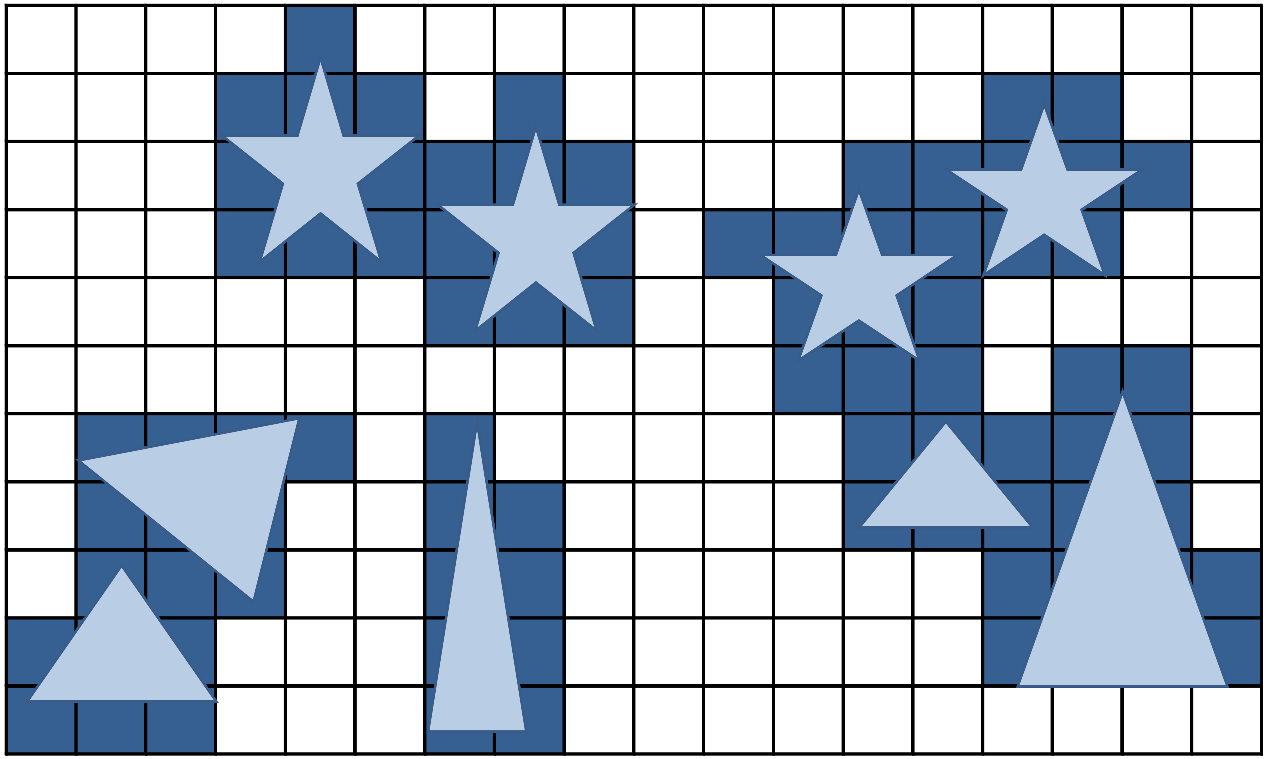
Regular Subdivision



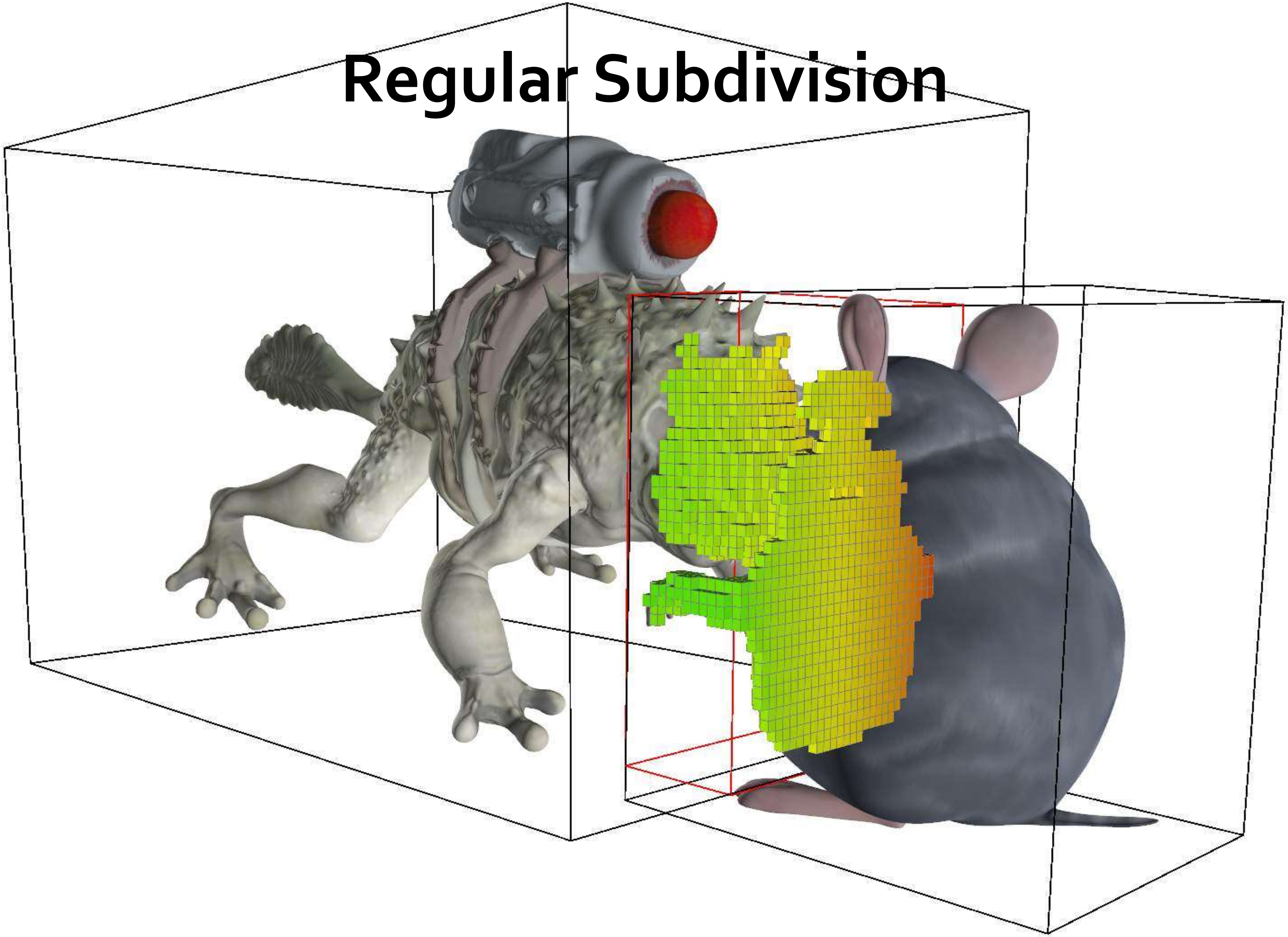
```
[[0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,1,1,0,0,0,0,0,0],
 [0,1,1,1,0,0,1,0,0,0],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

- Test with regular grid

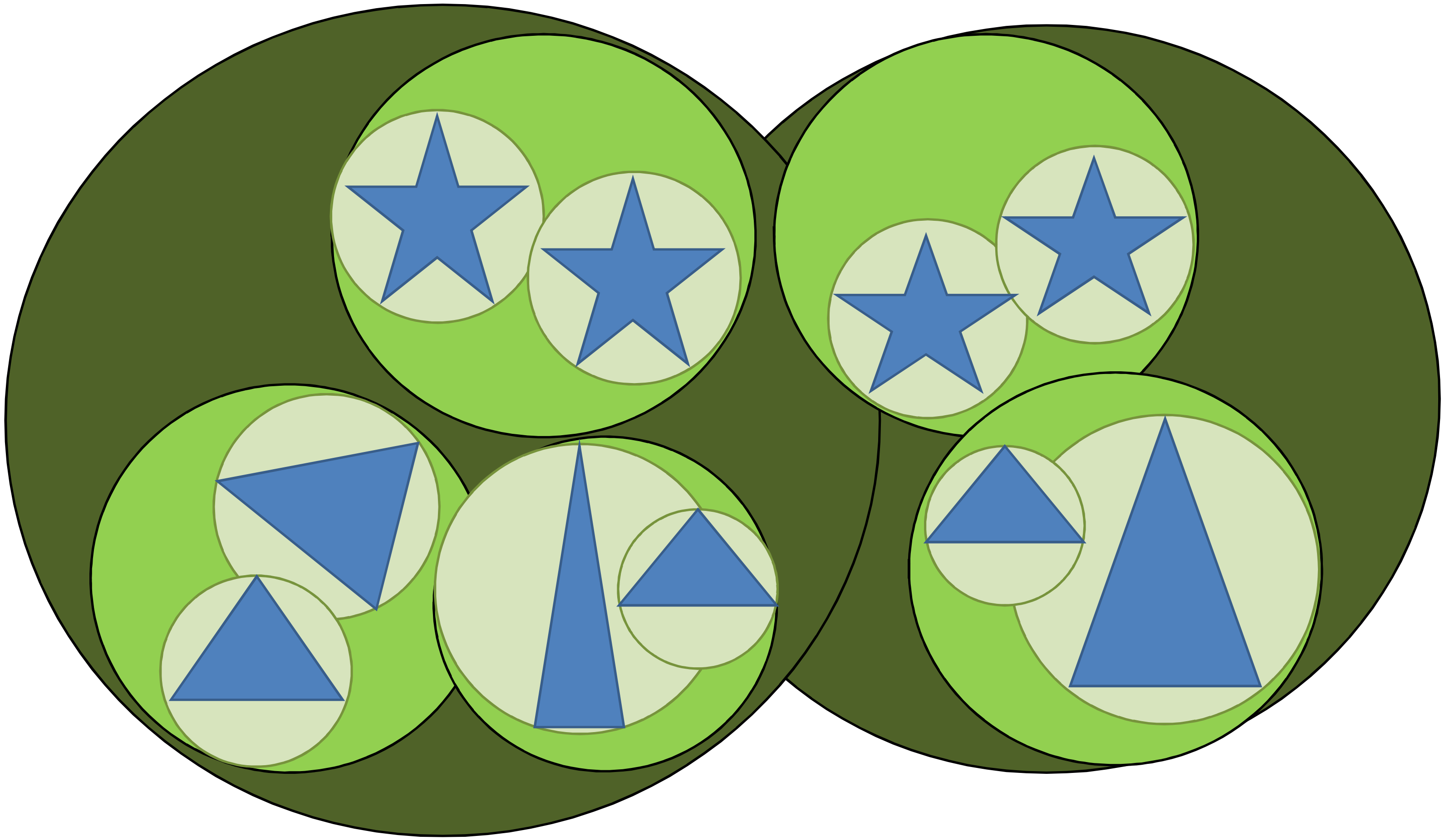


Regular Subdivision



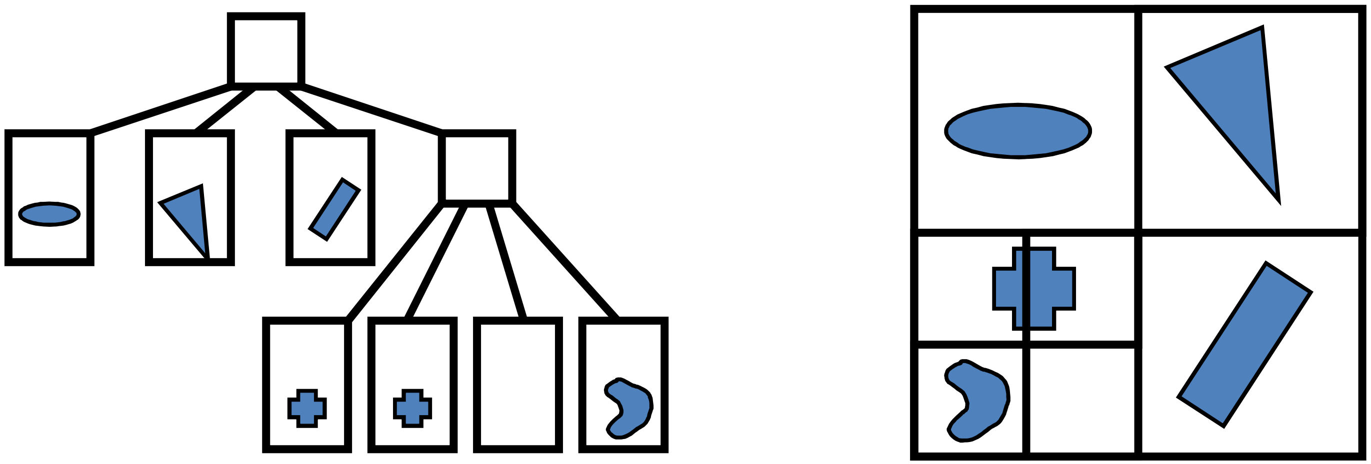
Hierarchy Trees

- Bounding Volume Hierarchy = BVH

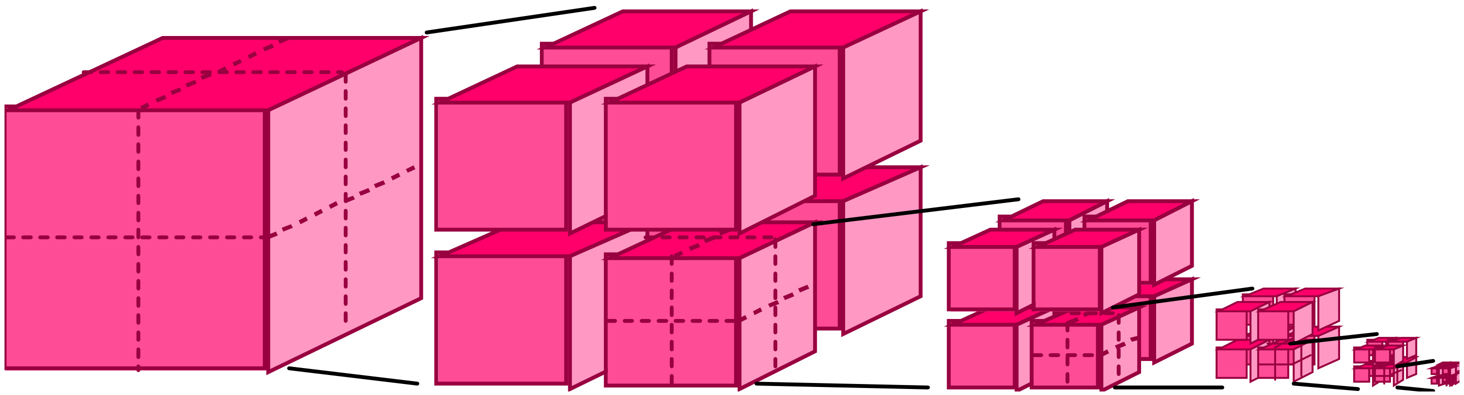


Quad/Octrees

- Quadtree (2D)

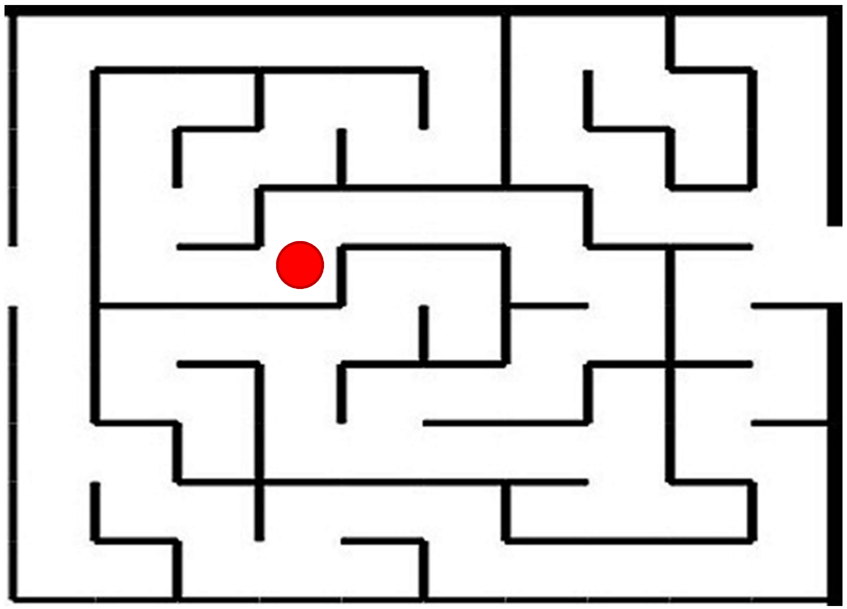


- Octree (3D)

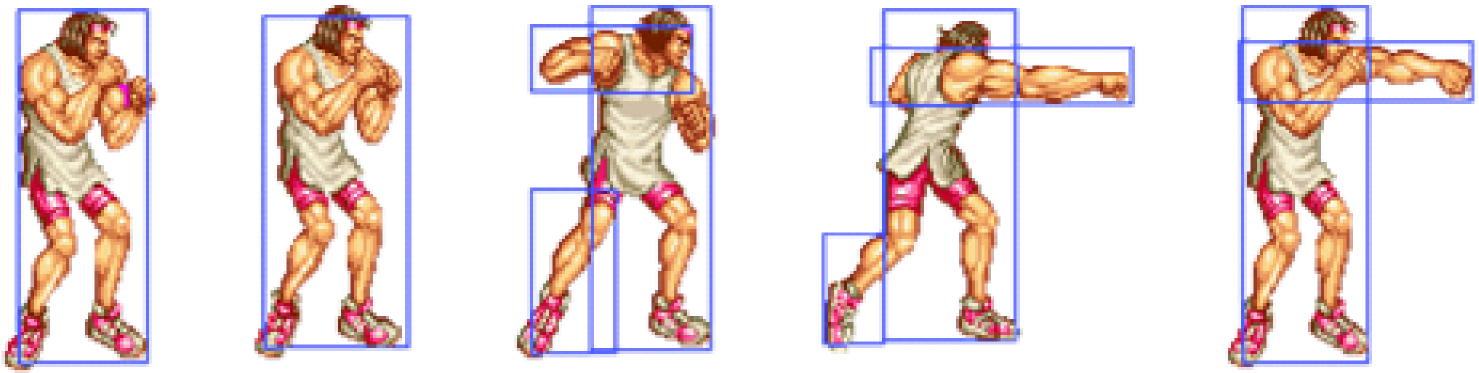


Another Simplification

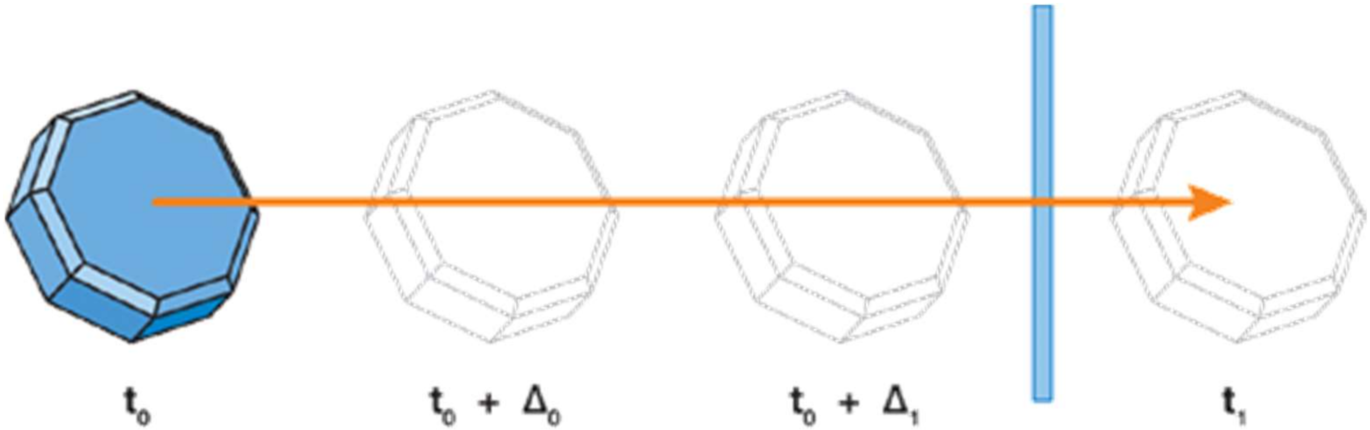
- Sometimes 3D can be turned into 2D operations
- Example: maze
- Approximate player by circle
- Test circle against lines of maze



Animated Objects

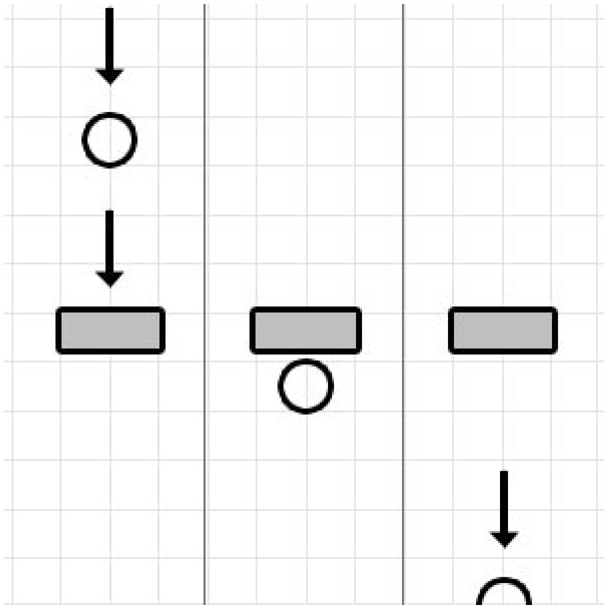


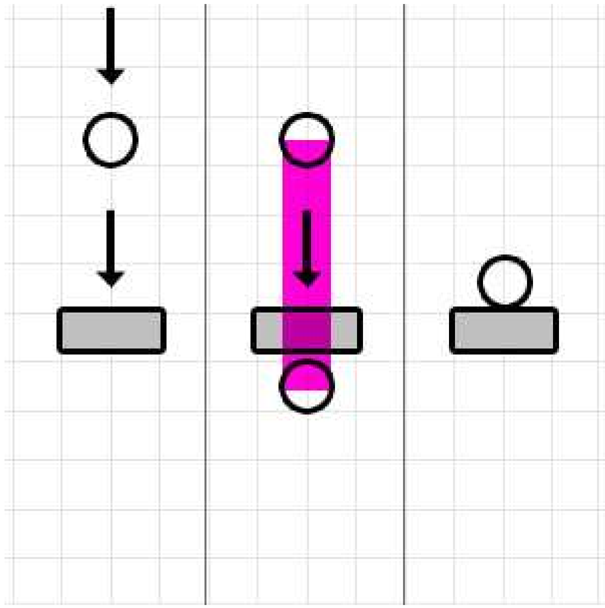
Trouble with Animated Objects



(a)

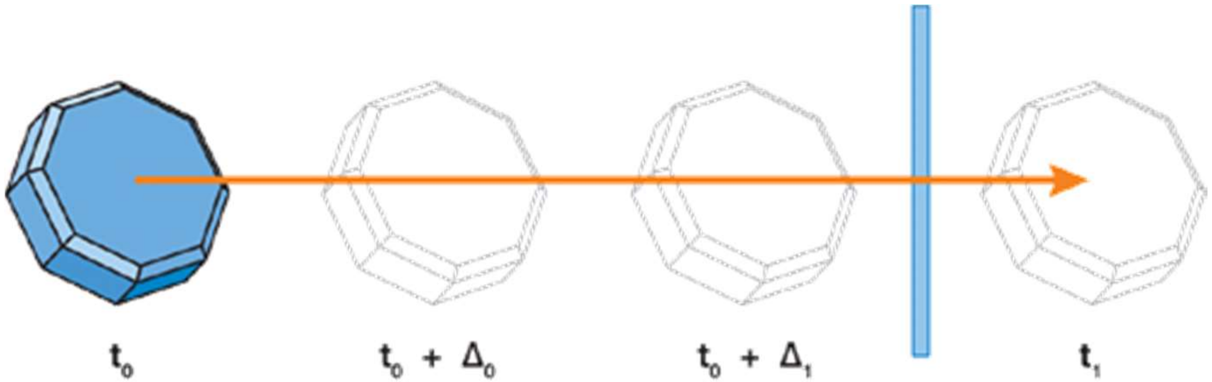
(b)





A posteriori (Discrete)

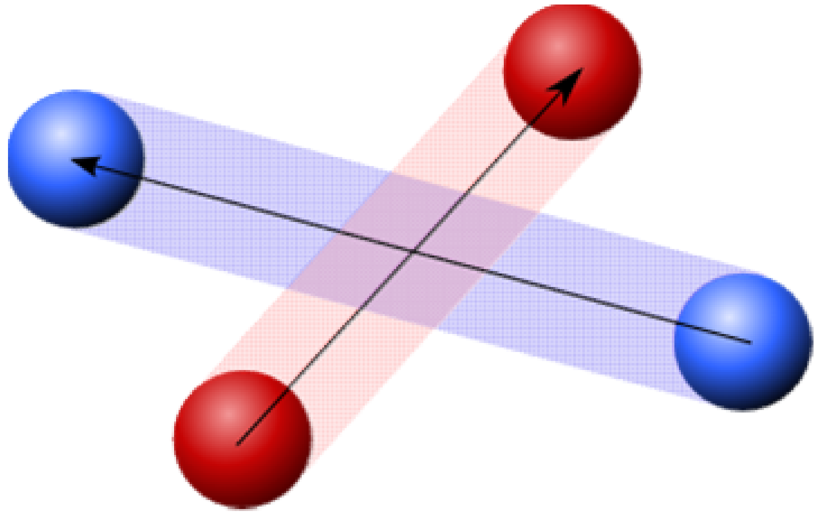
- Advance physics by time step then check for collision
- Simple
  - List of objects → return list of intersections
  - No time variable in calculations
  - Miss actual time of collision
- Need to “fix”



(a)

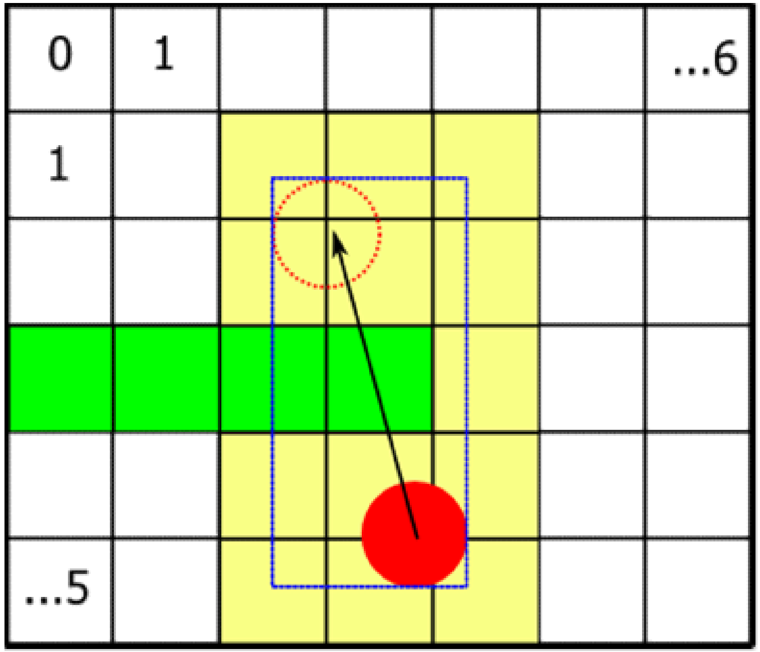
A priori (Continuous)

- 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



Independent Render and Game Loop

- Do update in predefined intervals
- Independent from rendering loop
  - Slow rendering does not impact update cycle