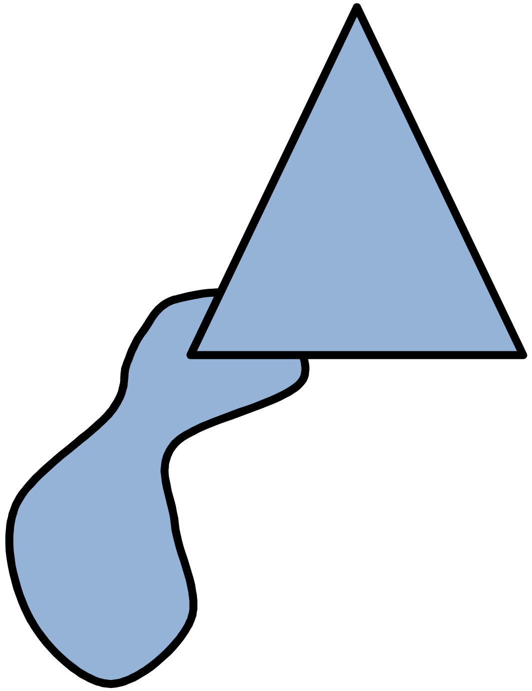
Collision Detection

Why?

- Realisme
    - Without ”quantum effects”
        - Objects pass through other objects
- Game play

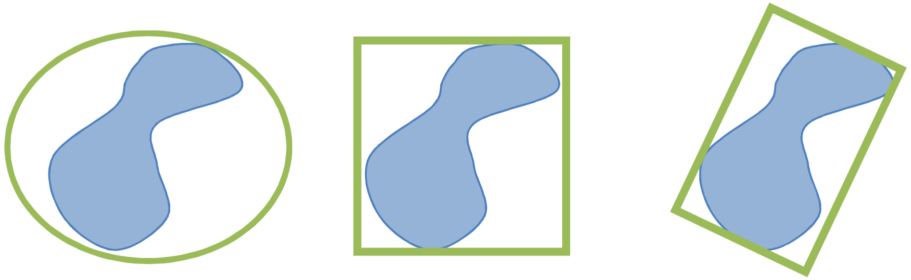
Three Major Parts

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



Phases

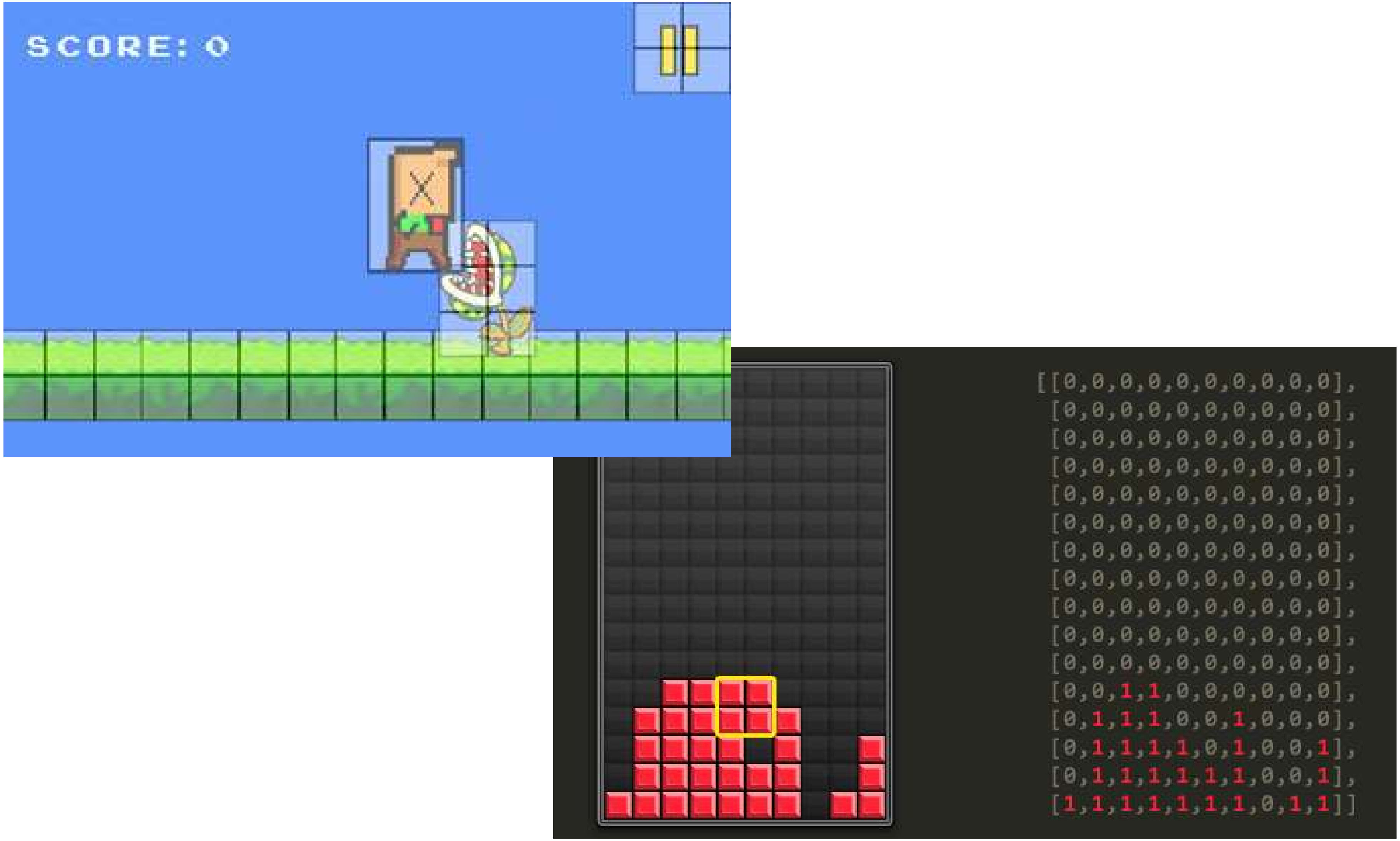
- Broad Phase (use placeholder geometry for speed)
  - Grids
  - Bounding Spheres
  - AABB-Algorithm
  - OBB-Algorithm
  - Subdivisions



- Narrow Phase (real object is intersected)
  - Point-Line
  - Point-Triangle
  - Triangle-Triangle
  - ...

Broad Phase

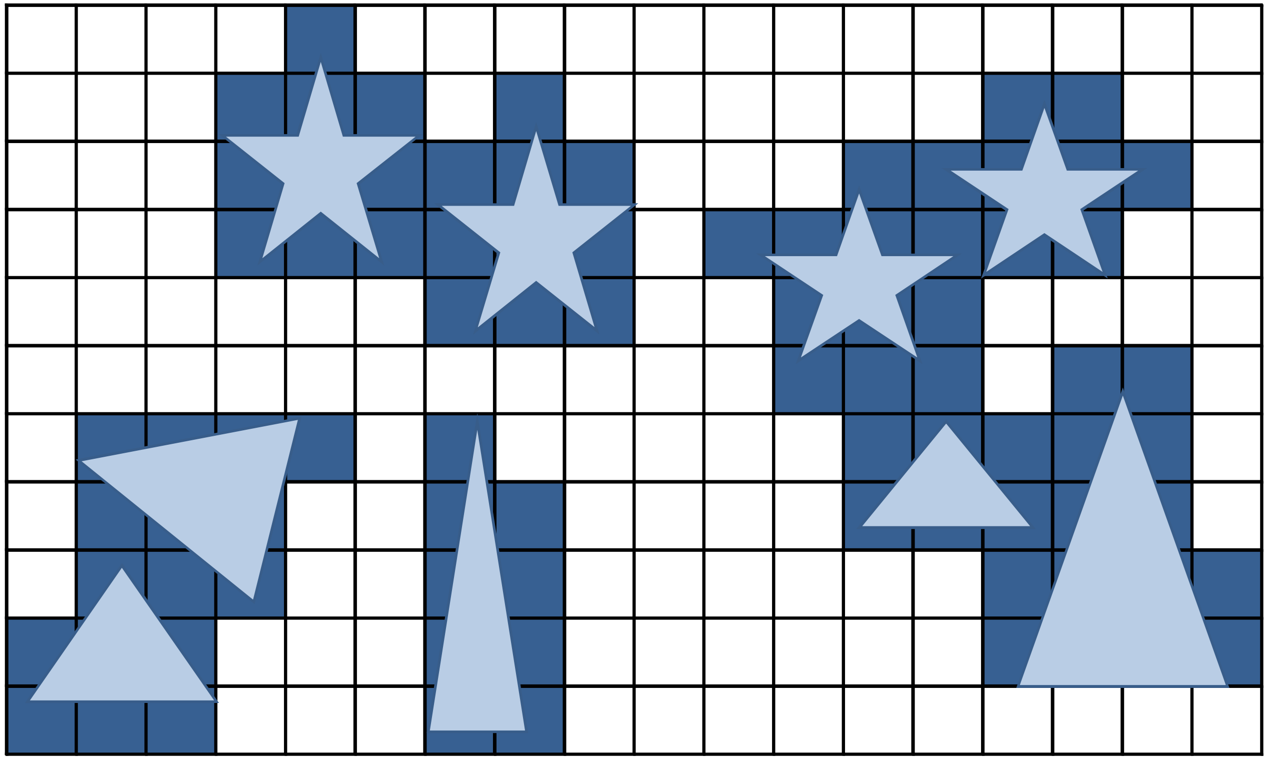
Regular Subdivision



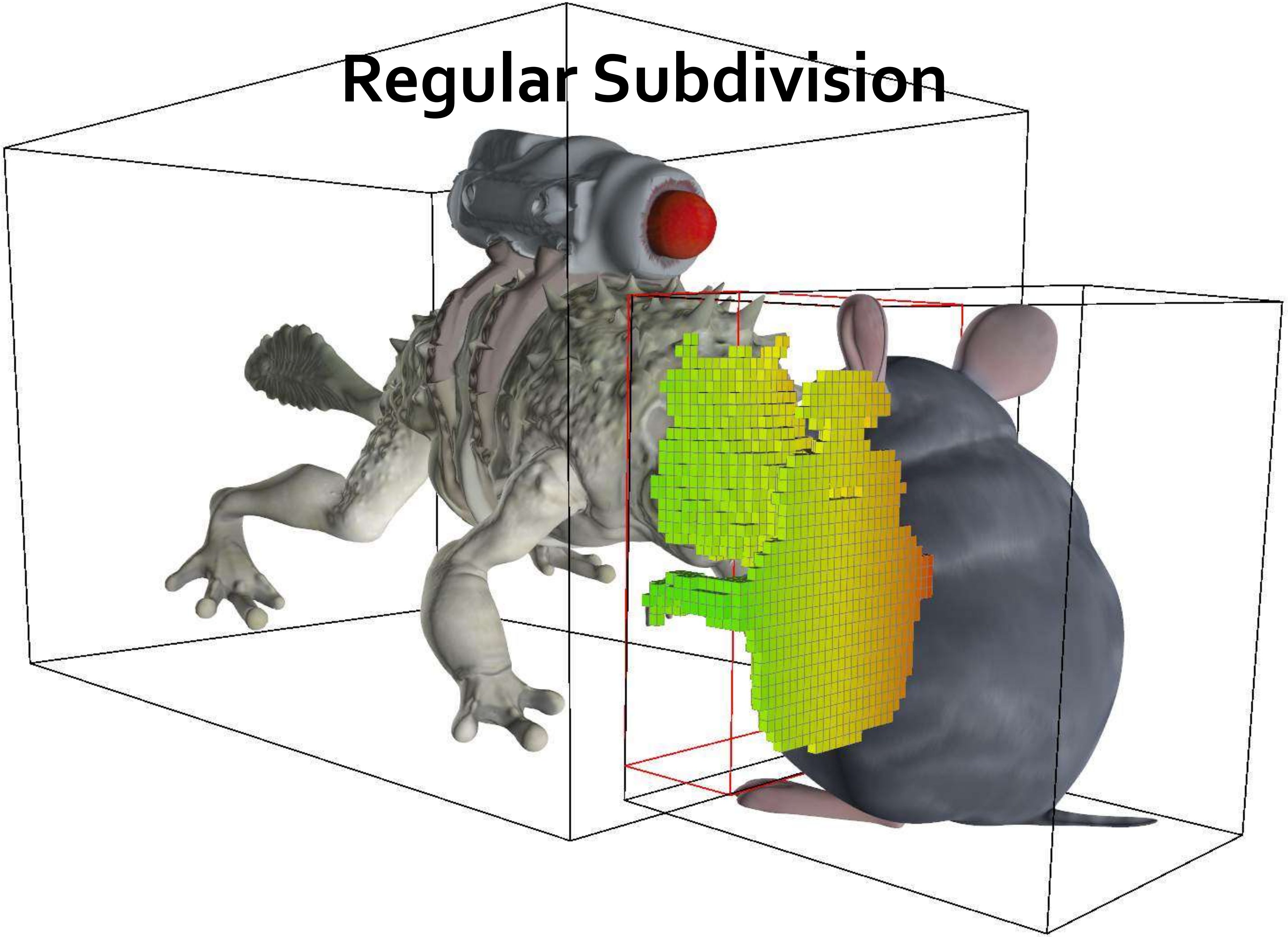
```
[[0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,1,1,0,0,0,0,0,0],
 [0,1,1,1,0,0,1,0,0,0],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

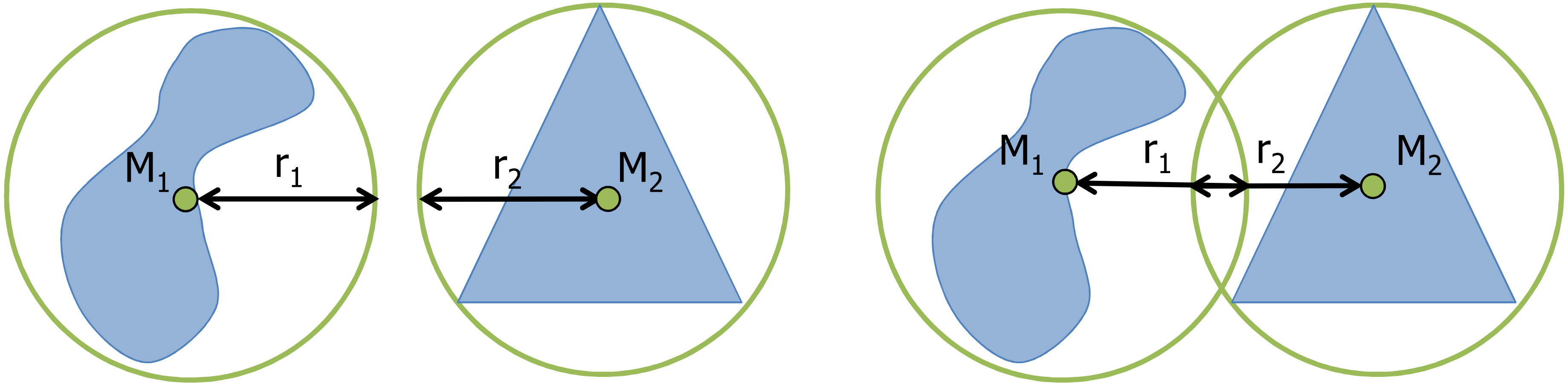


Regular Subdivision



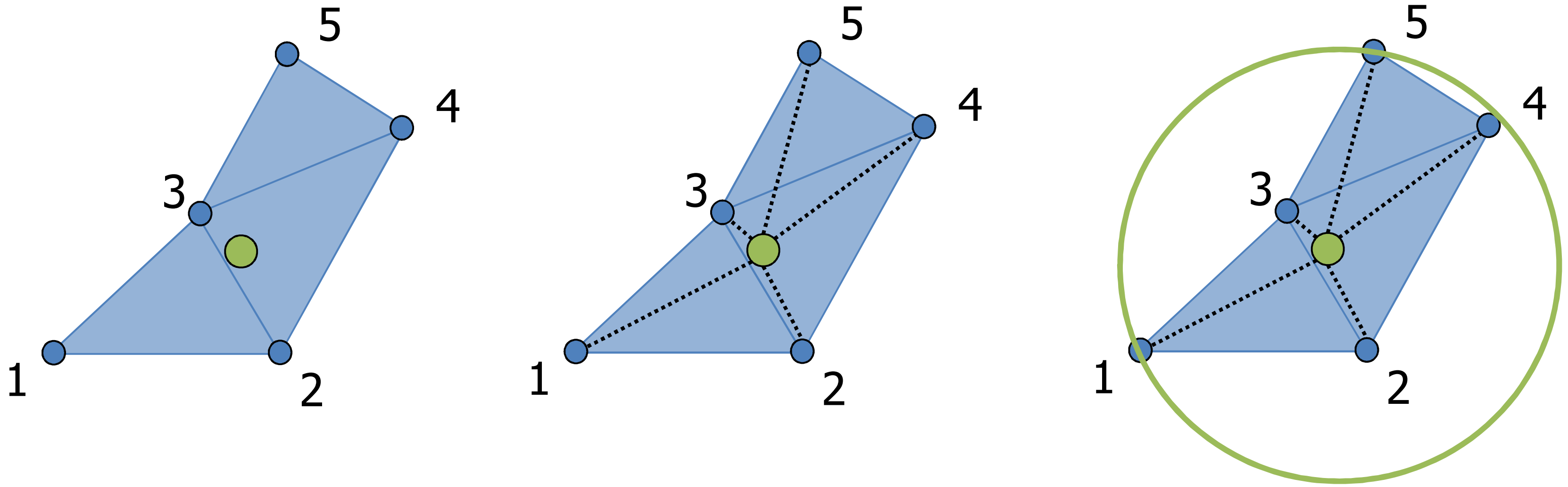
Bounding Spheres

- Collision if $distance(M_1, M_2)^2 < (r_1 + r_2)^2$

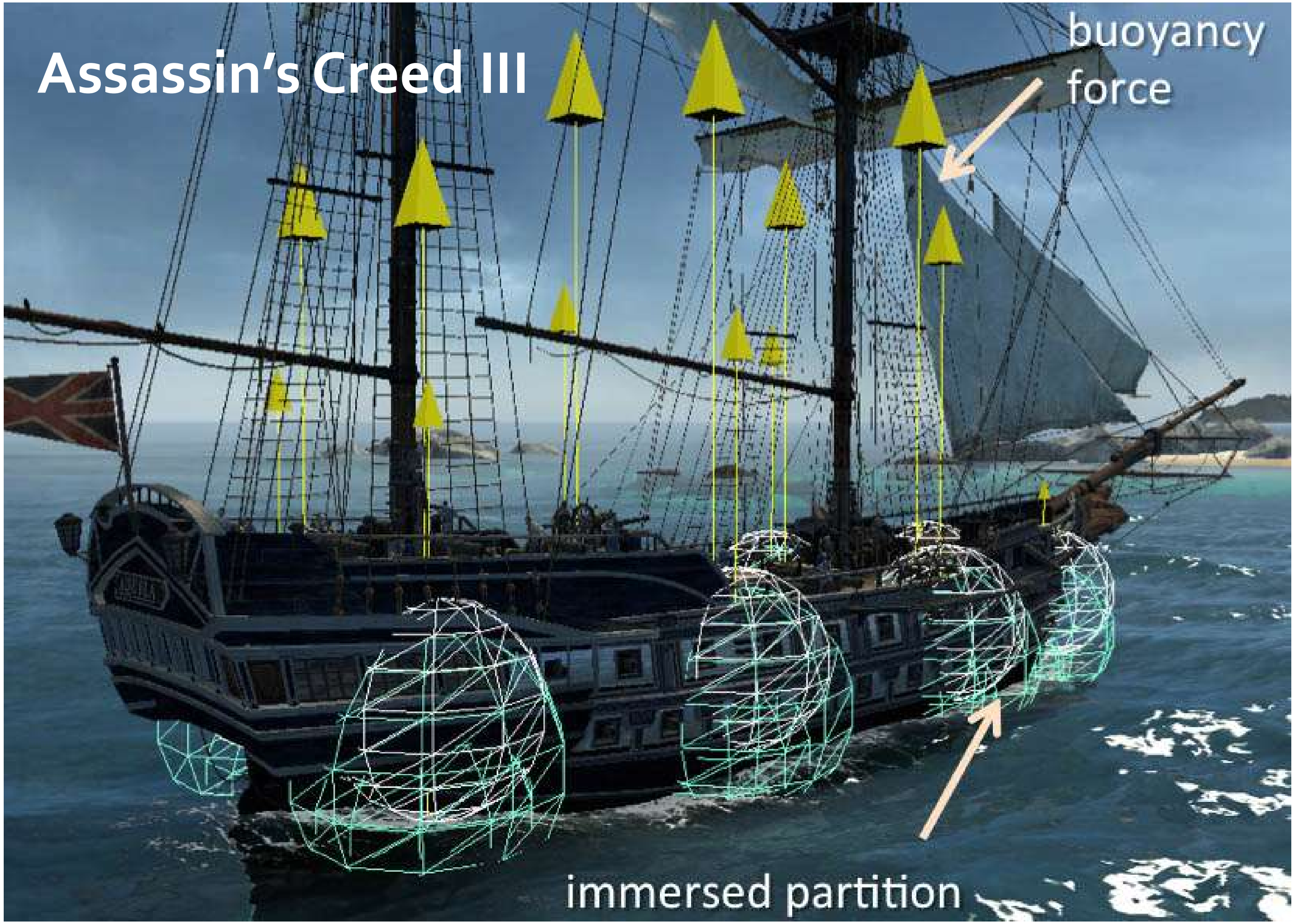


Calculating Bounding Spheres

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

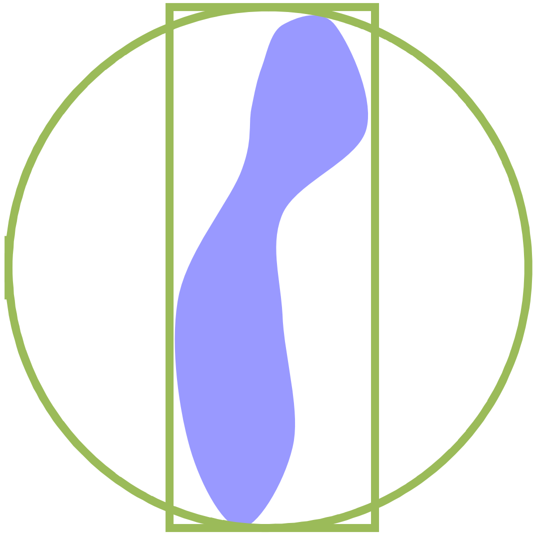


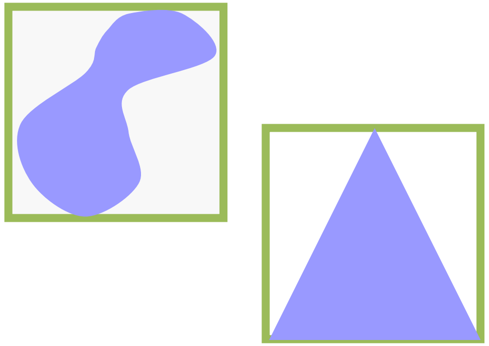
Creative Use of Bounding Spheres



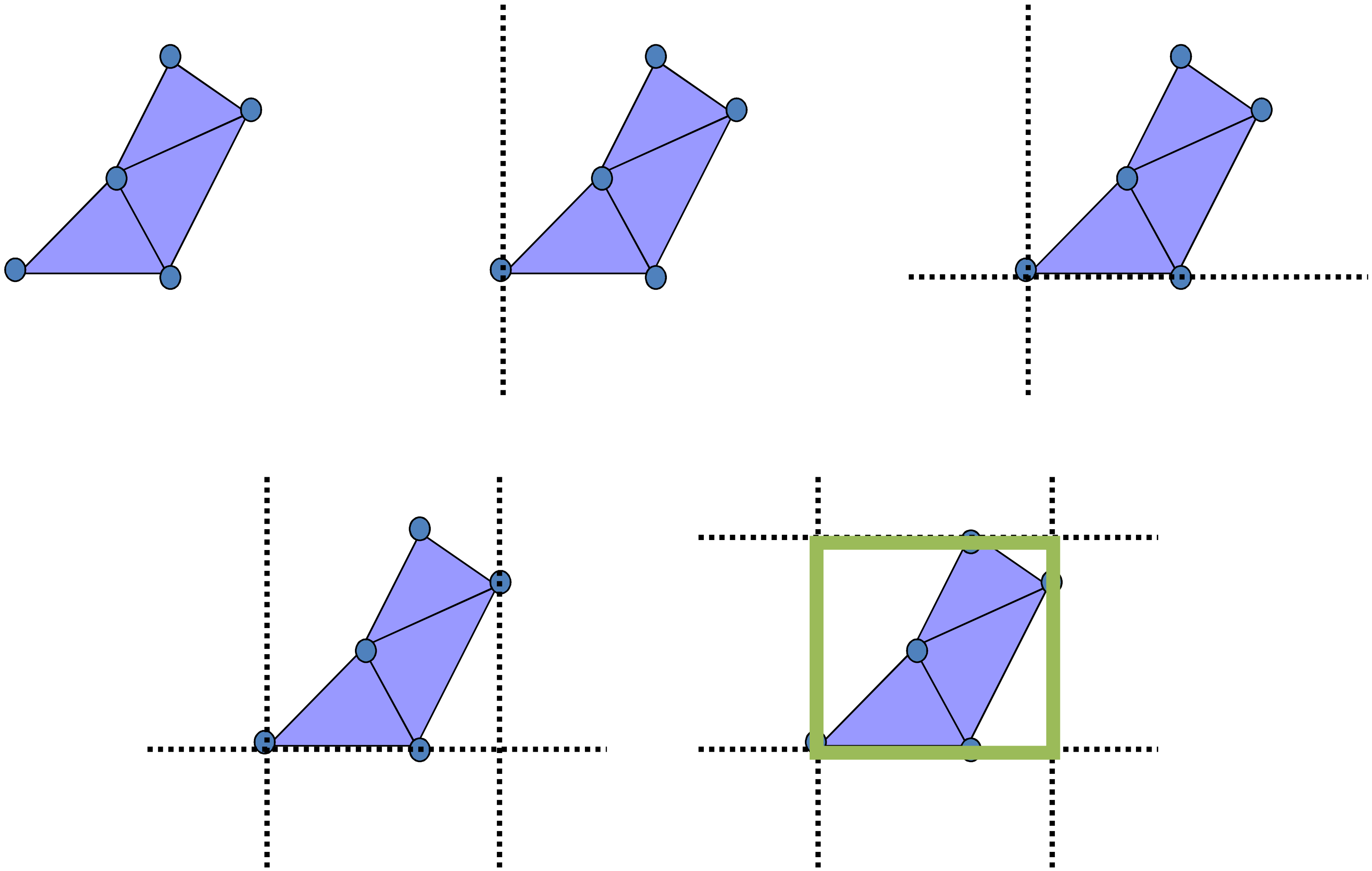
AABB-Algorithm

- Bounding-Spheres:
  - Efficient
  - Inaccurate
- Axis Aligned Bounding Boxes
  - Better Fit
  - Only slightly more complicated



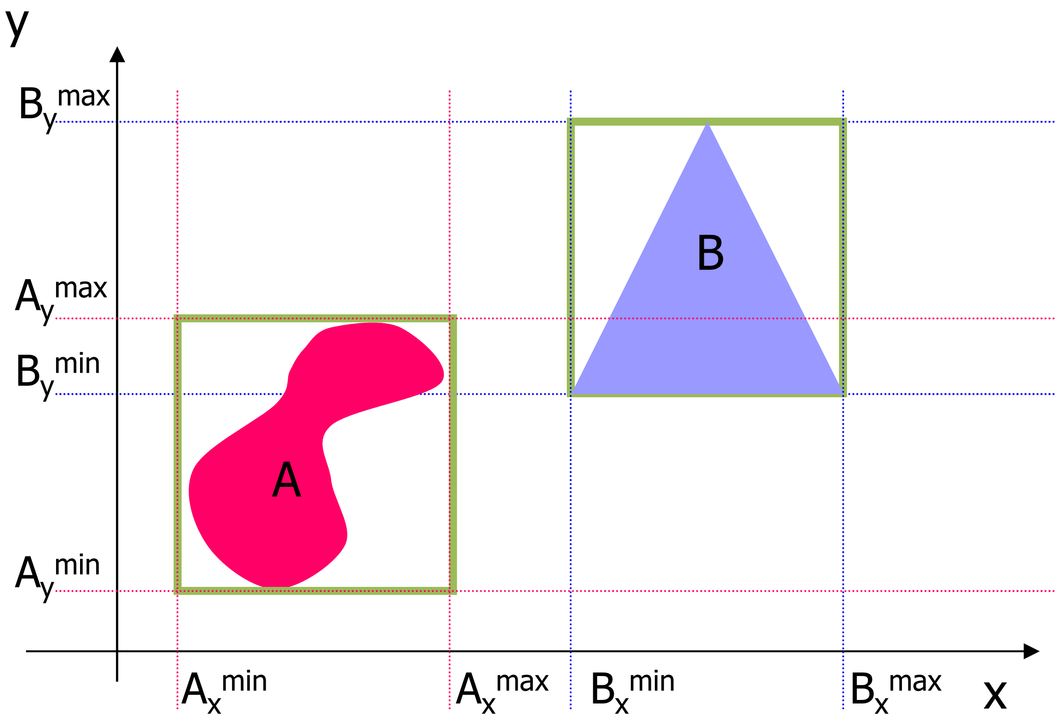


AABB-Algorithm



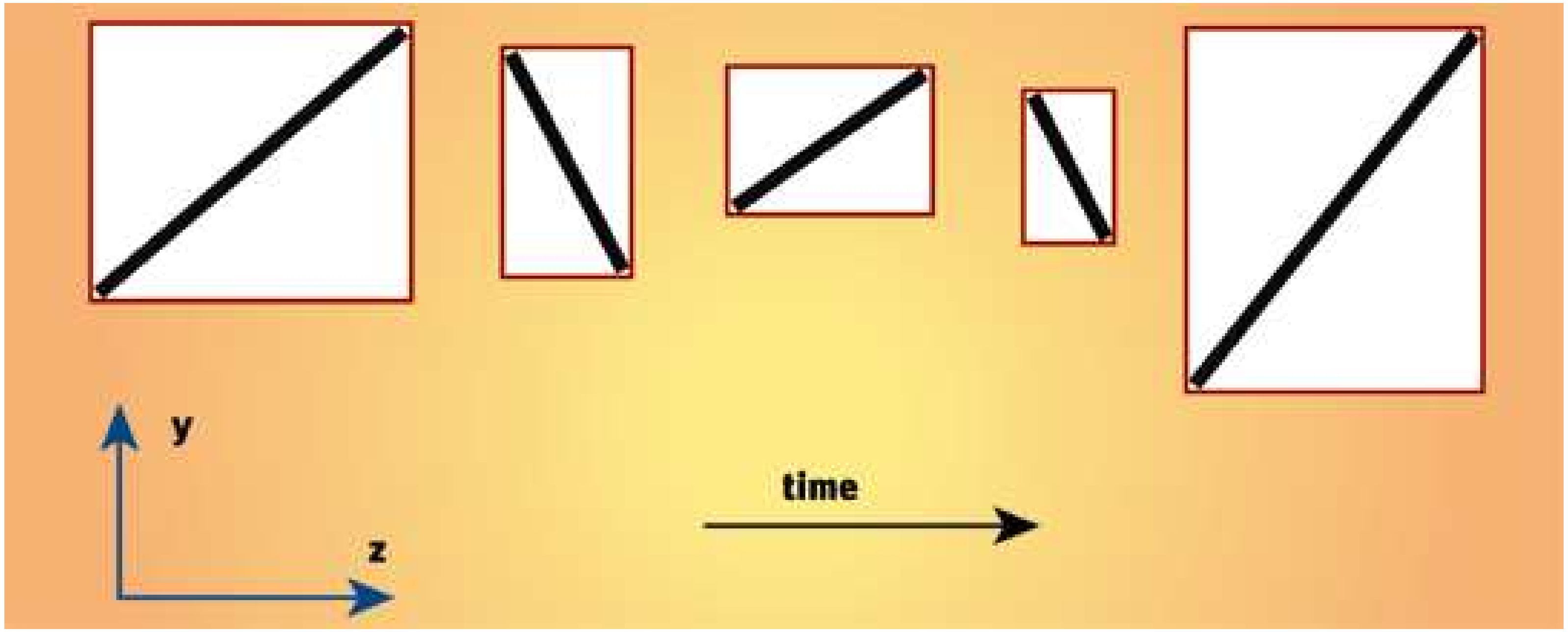
AABB-Algorithm

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



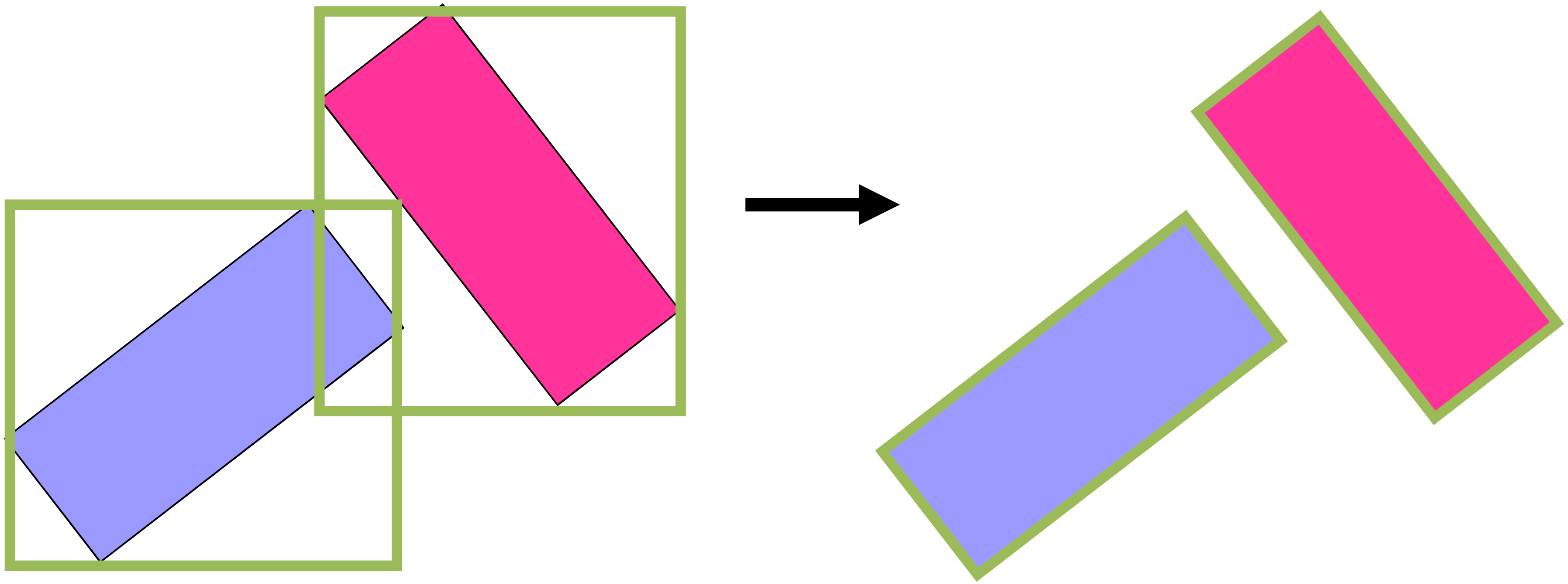
AABB - Problems

- While rotating an object, we have to recalculate the bounding box



Oriented Bounding Box

- Which problems do we have using the AABB approach?
  - *SIGGRAPH 1996, Gottschalk et al.*

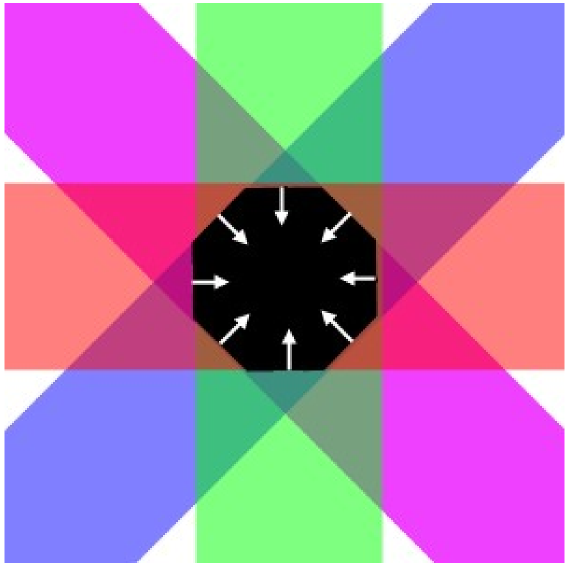


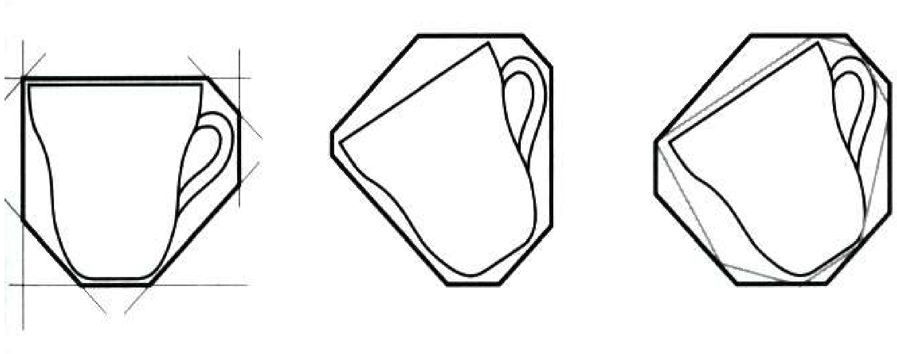
Oriented Bounding Box

- Rotation is no further a problem
- 95% of the situations are solved
- More complicated to calculate than AABB
- Separating axis theorem still works
- More math involved
- Find more information under
  - www.gamasutra.com
  - Game Programming Gems (I, II, III)

k-DOP

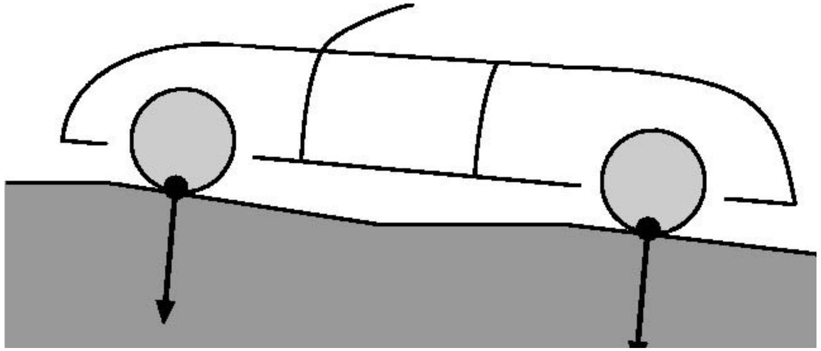
- k-Discrete Oriented Polytop
- OBB and AABB are 6-DOPs
- Optimal bounding boxes

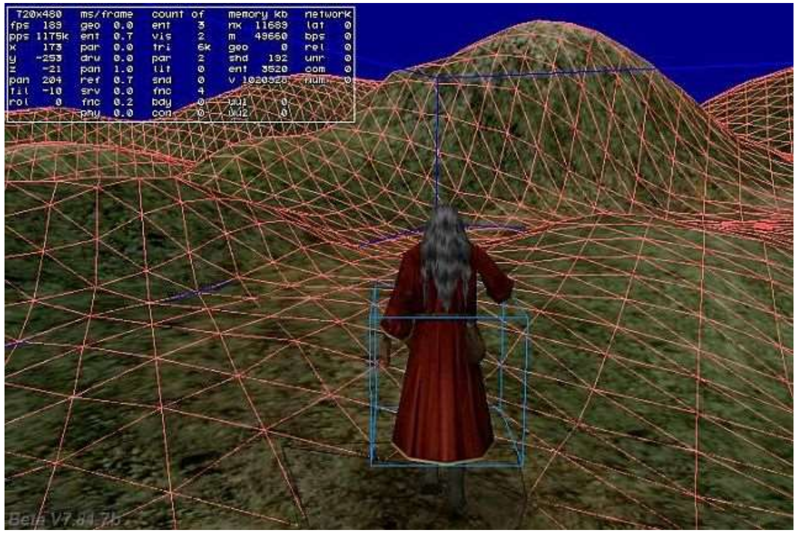




Collision Detection with Rays

- 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

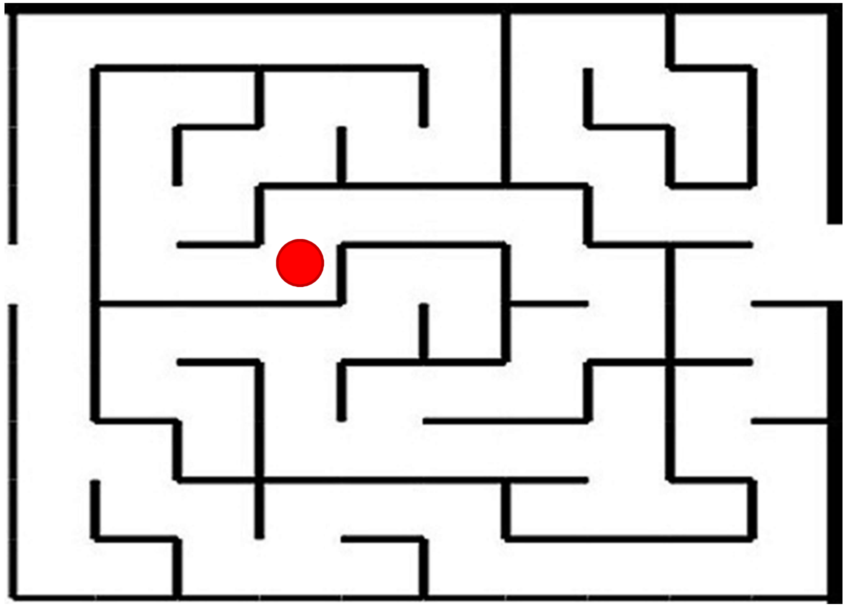




```
 720x480    ms/frame   count of    memory kb   network
fps   189   geo  0.0   ent    3    nx   11689  lat    0
pps 1175k   ent  0.7   vis    2    m    49660  bps    0
x     173   par  0.0   tri   6k    geo      0  rel    0
y    -253   dru  0.0   par    2    shd    192  unr    0
z     -21   pan  1.0   lit    0    ent   3520  com    0
pan   204   ref  0.7   snd    0    v  1020528  num    0
til   -10   srv  0.0   fnc    4
rol     0   fnc  0.2   bdy    0    uu1      0
            phy  0.0   con    0    uu2      0
```

Another Simplification

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



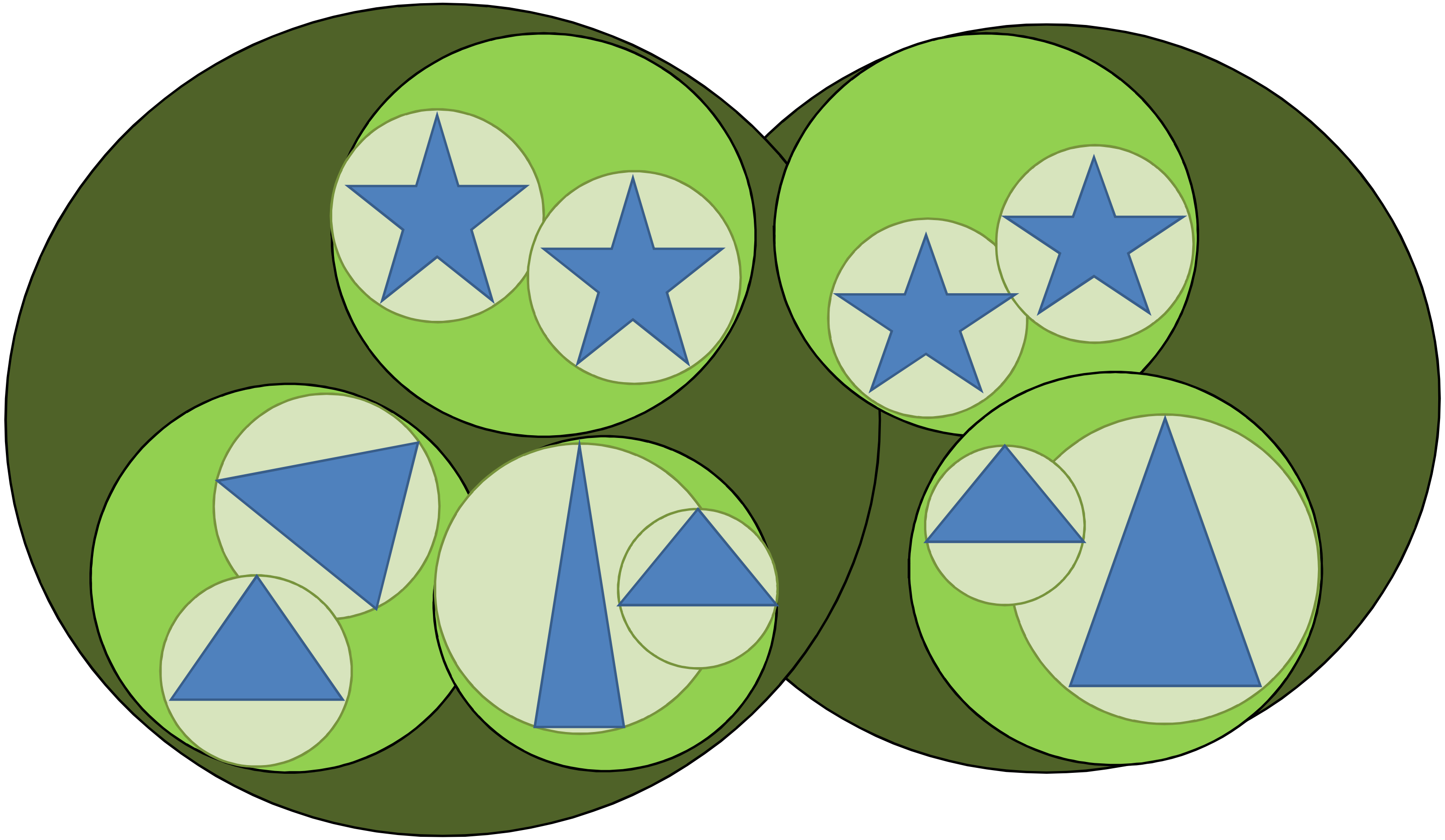
Acceleration

Handling High Numbers of Objects

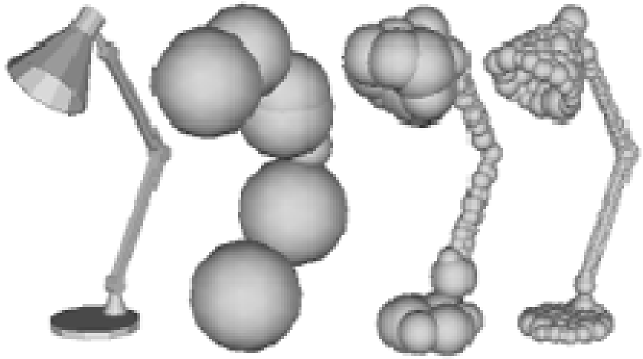
- Have to check each object with every other
  - $N \cdot (N1) \approx N^2$
- Hierarchical iregular subdivision
- Hierarchical Regular suvdivision
- Regular subdivion

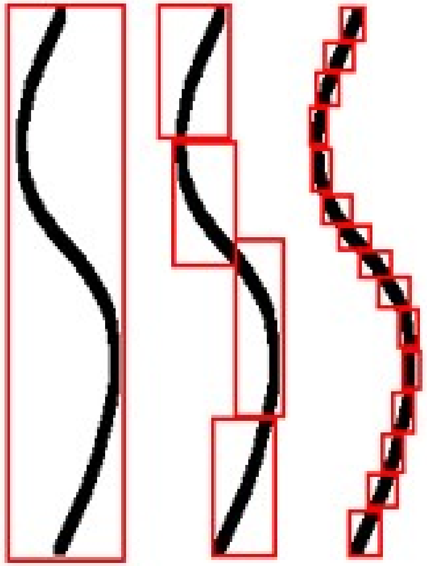
Hierarchy Trees

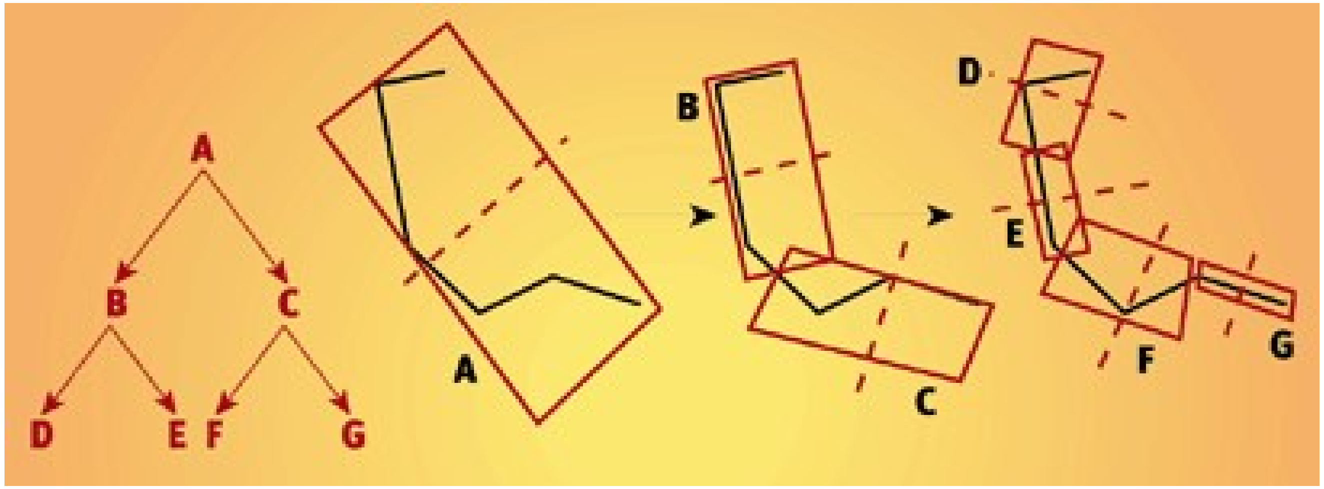
- Bounding Volume Hierarchy = BVH



Hierarchy Trees

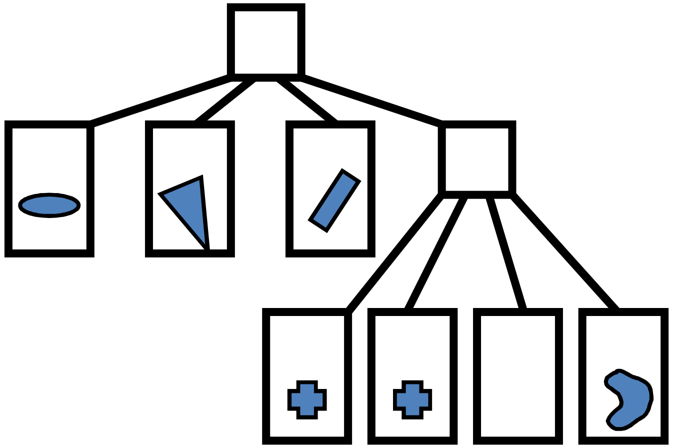


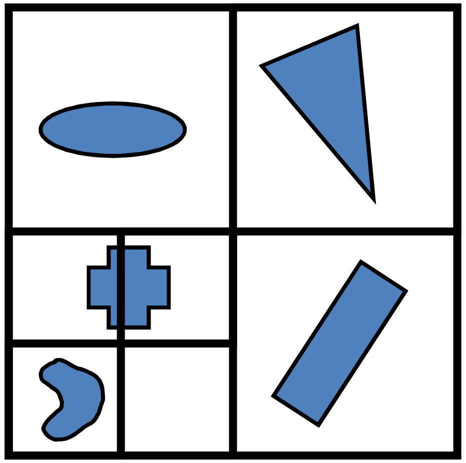




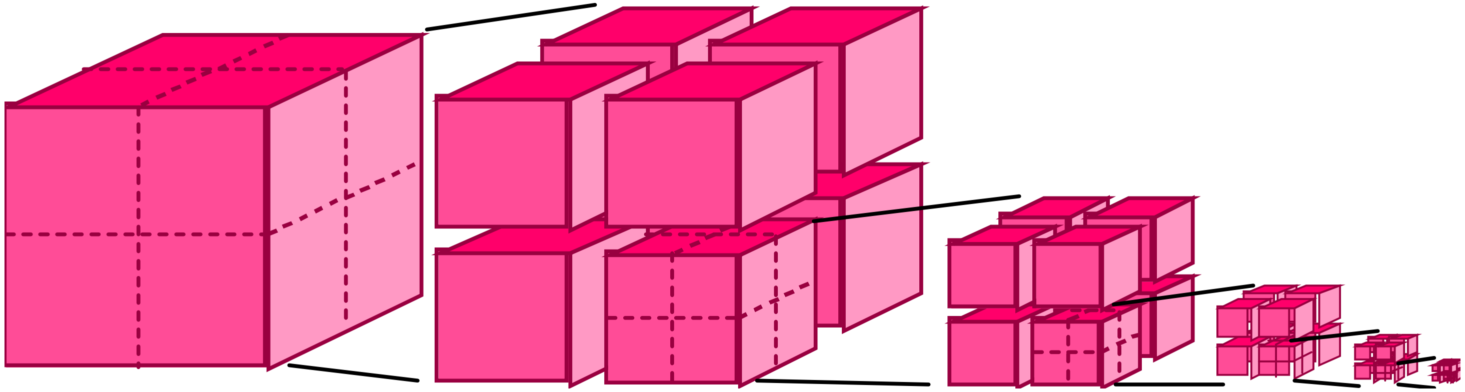
Quad/Octrees

- Quadtree (2D)



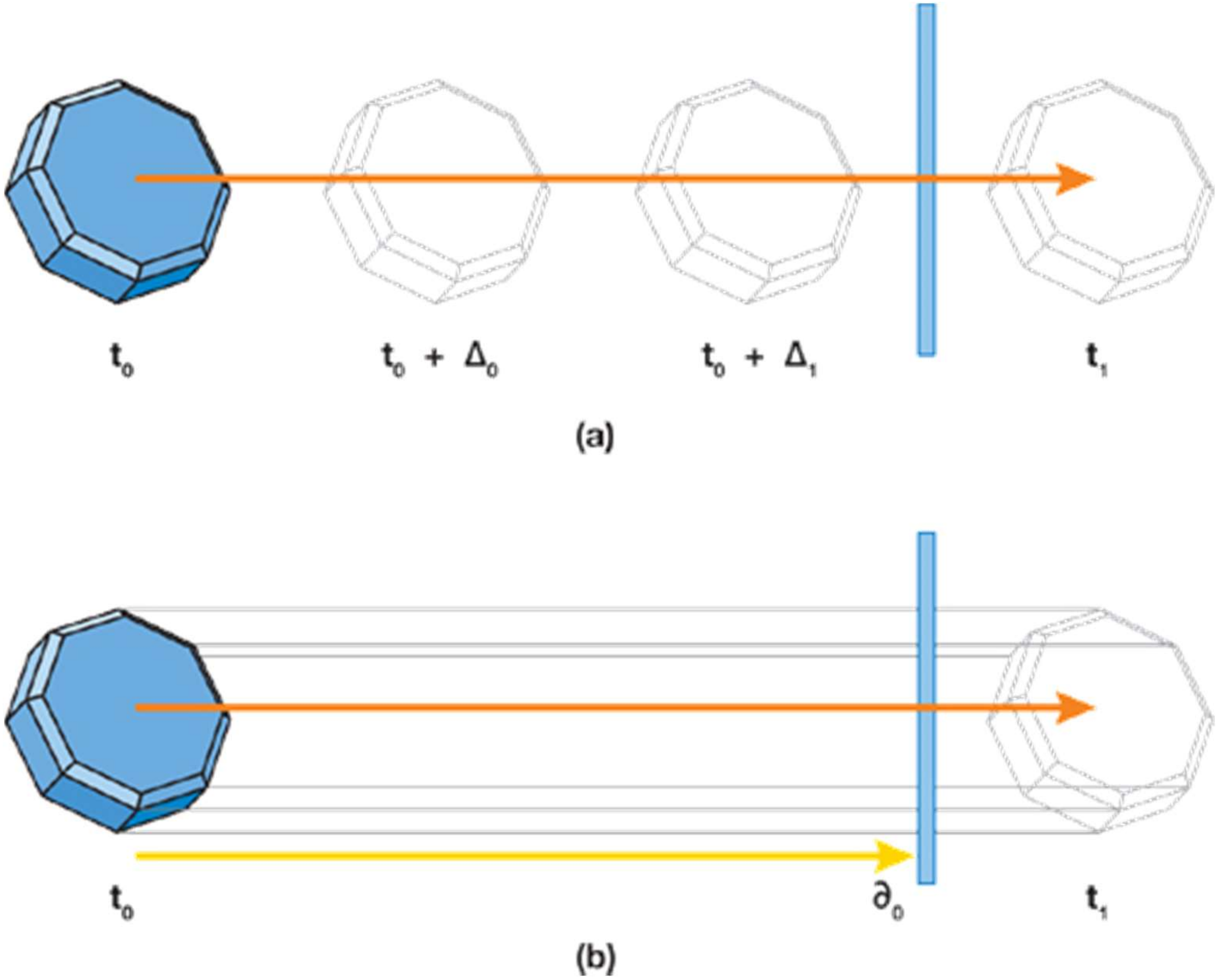


- Octree (3D)



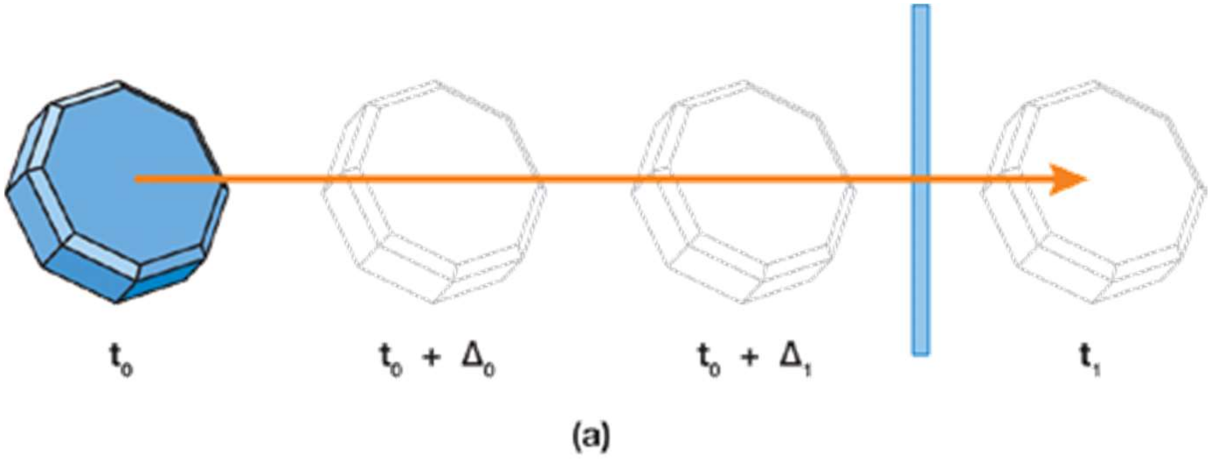
Animated Objects

Trouble with Animated Objects

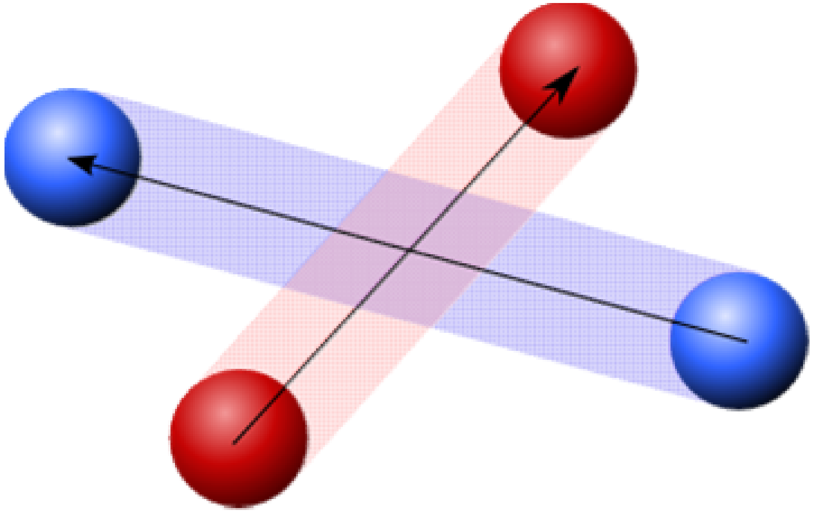


A posteriori (Discrete)

- 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”



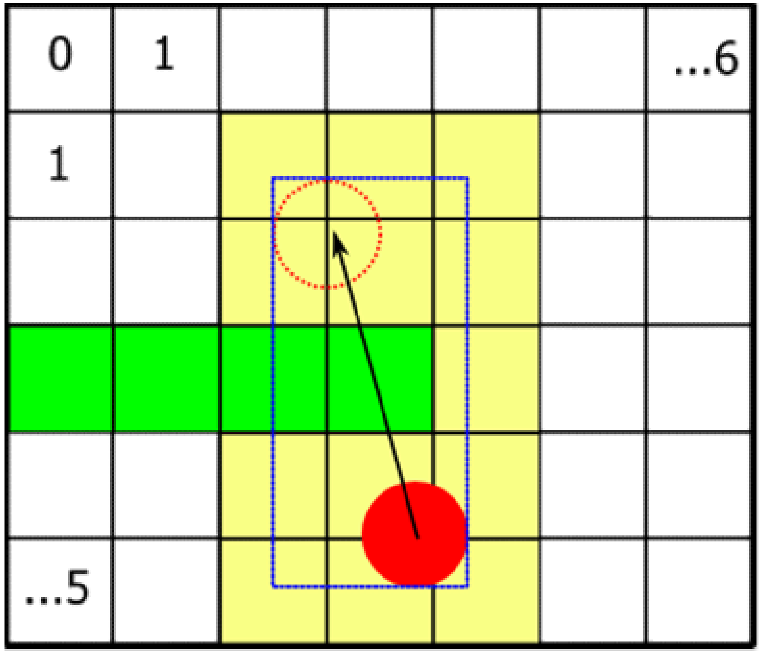
A priori (Continuous)



- 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



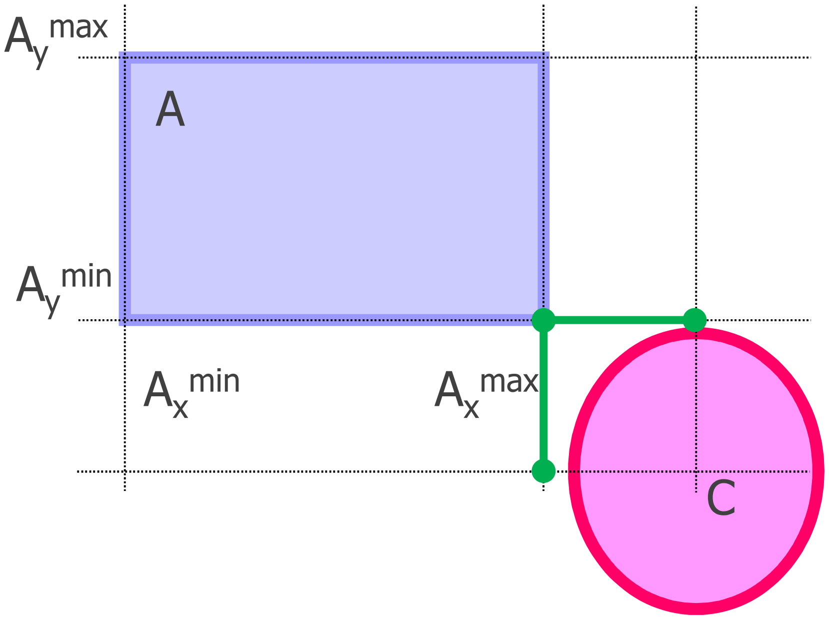
Narrow Phase

Narrow Phase

- Many specialized algorithms
  www.realtimerendering.com/intersections.html
- Often not needed
- Will talk about common cases

Sphere-Box Intersection

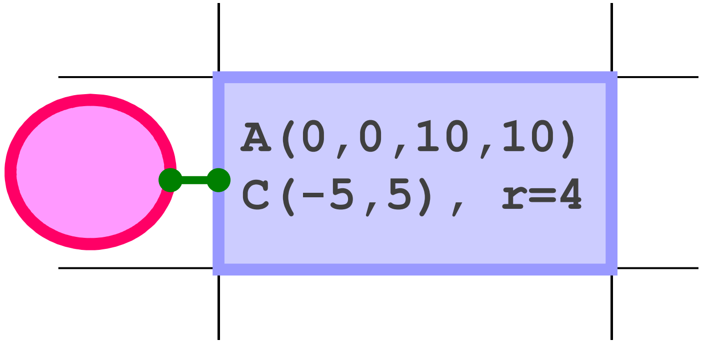
- Idea: Coordinate-wise Euclidean distance



```
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
```

Sphere-Box Intersection

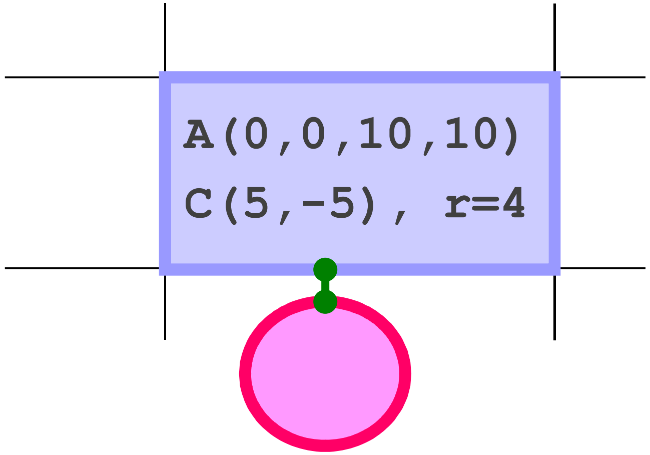
(1)



A(0,0,10,10)

C(-5,5), r=4

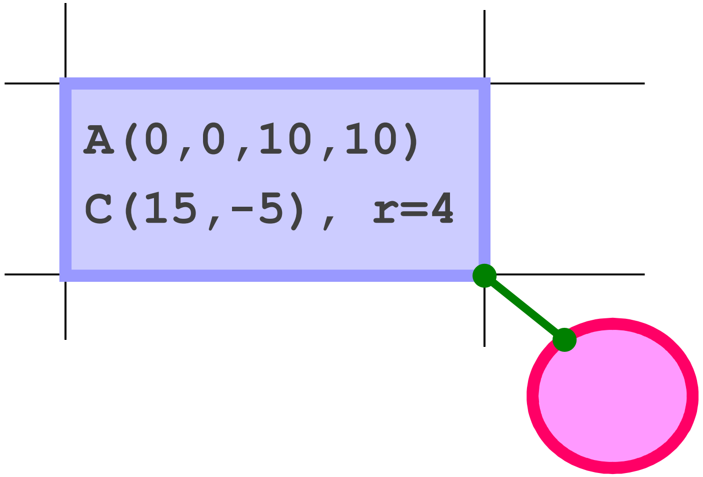
(2)



A(0,0,10,10)

C(5,-5), r=4

(3)



A(0,0,10,10)

C(15,-5), r=4

(4)

3D?