CS 5410

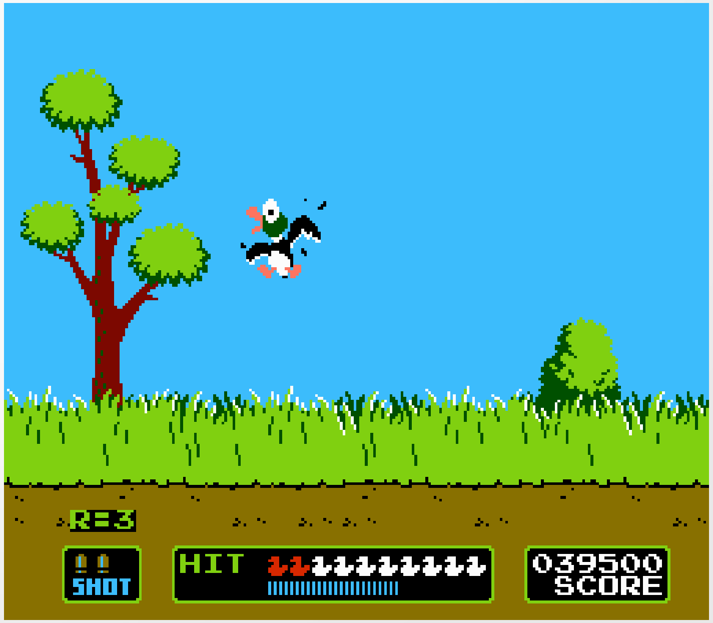
Intro to Collision Detection

Collision Detection

- Goal: Know when game entities have, well, collided
- Approaches
  - Test pixels in multiple, overlapping planes (drawing surfaces)
  - Polygon-by-Polygon basis
  - Bounding Box (BB)
  - Axis-Aligned Bounding Box (AABB)
  - Sphere/Circle
  - Subdivision methods
    - Grid
    - Hierarchical
  - Physics Engine

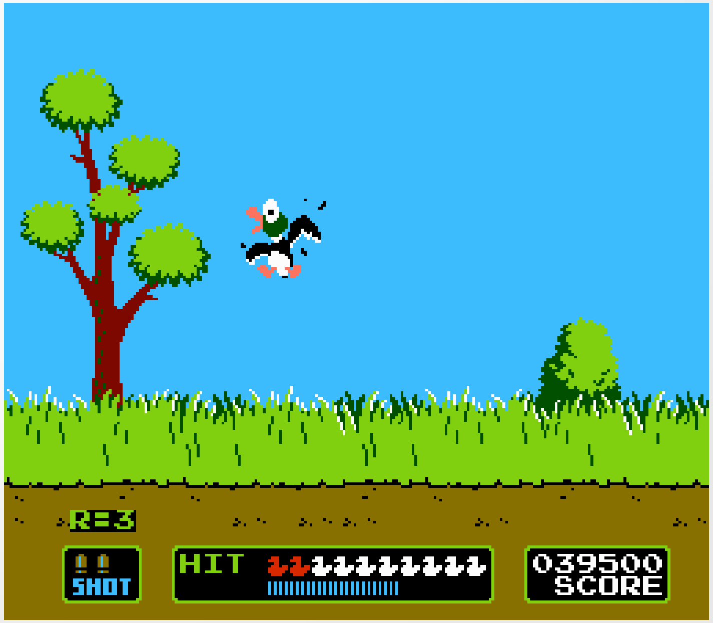
Pixels in Overlapping Planes

- Organize scene into multiple planes
  - Background imagery
  - Moving objects that can be hit
    - (alt 1)All objects than can be hit
    - One for each object
  - Moving objects that can’t be hit
  - Foreground imagery occluders
  - Game status, scoring, etc



Pixels in Overlapping Planes

- Upon weapon firing, compute weapon pattern
- Compute corresponding pixel values
- Sample plane with (hitable) moving objects
- Sample plane with occlusion planes
- Any pixels that have moving objects hit, but nothing in the occlusion planes represent hit objects

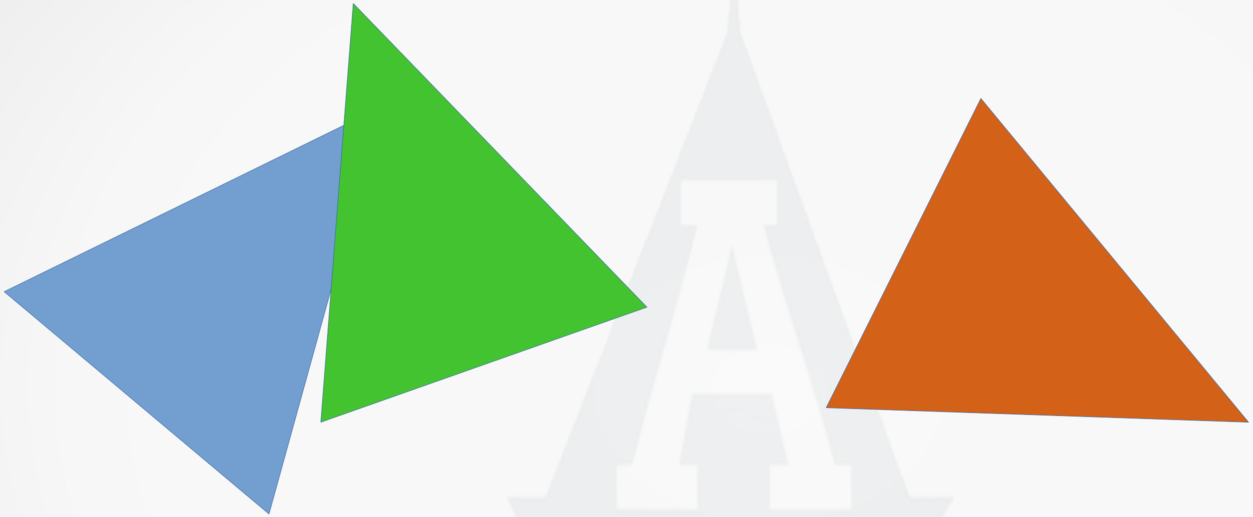


Polygon-by-Polygon

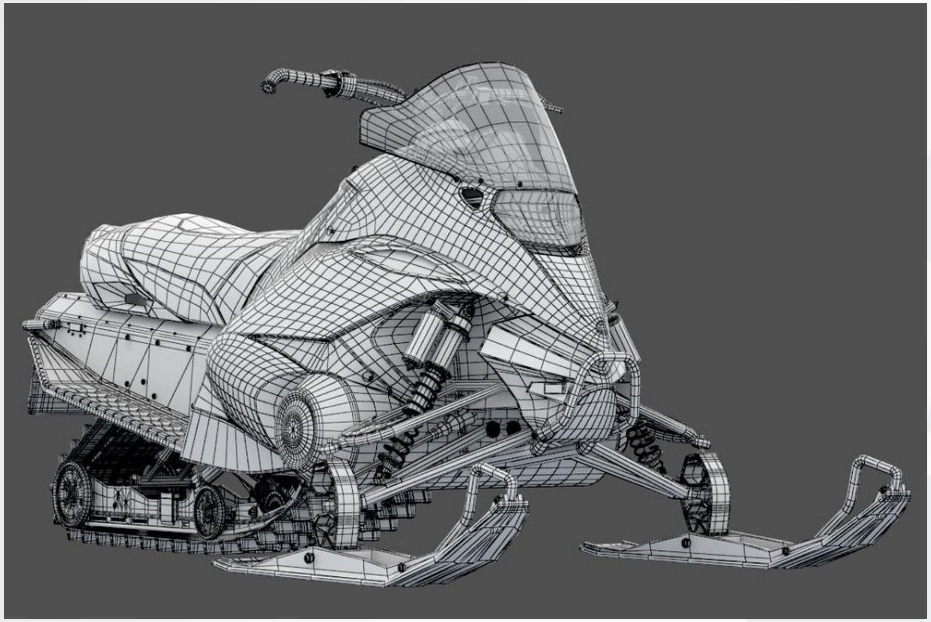
- https://web.stanford.edu/class/cs277/resources/papers/Moller1997b.pdf

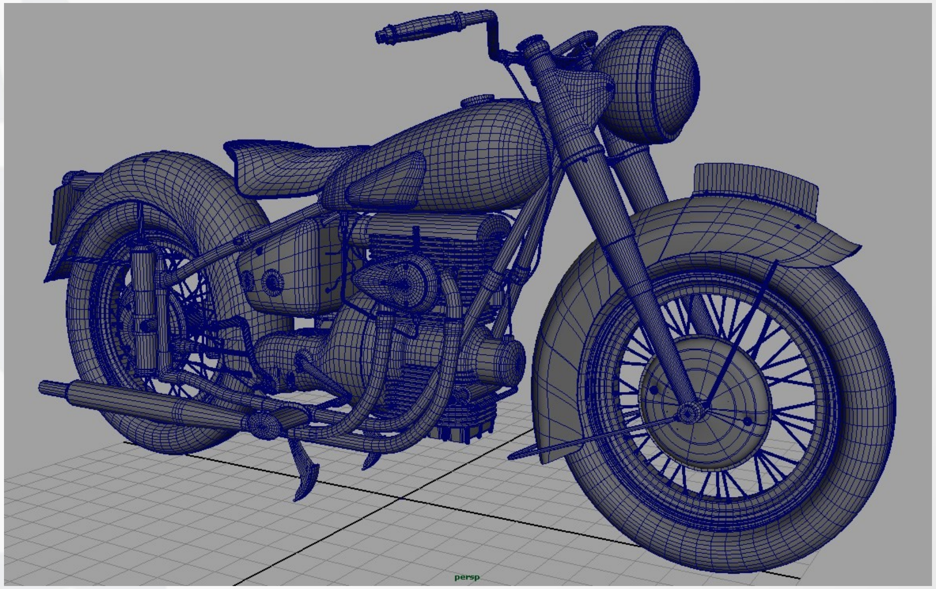
1. Compute plane equation of t_2
2. Reject as trivial if all points of t_1 are on same side
3. Compute plane equation of t_1
4. Reject as trivial if all points of t_2 are on same side
5. Compute intersection line and project onto largest axis
6. Compute the intervals for each triangle
7. Intersect the intervals

Polygon-by-Polygon



Polygon-by-Polygon

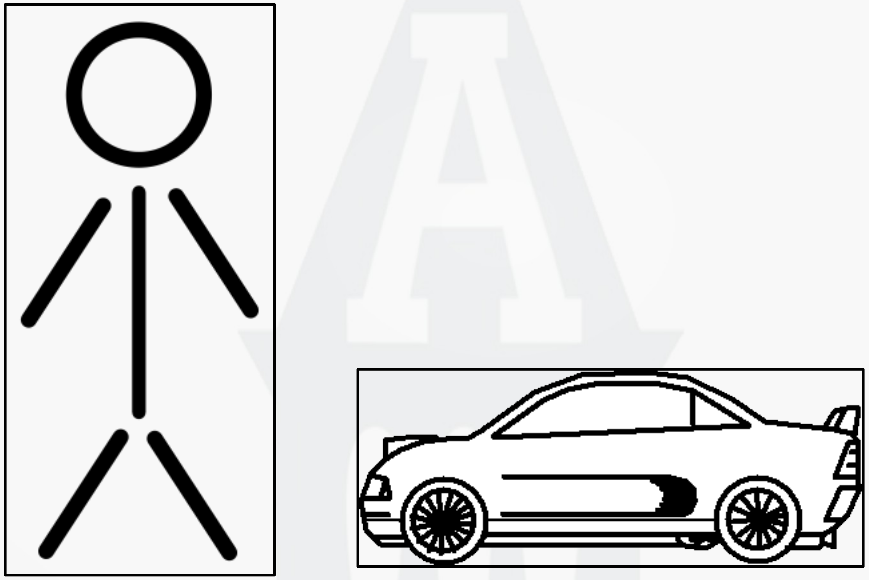




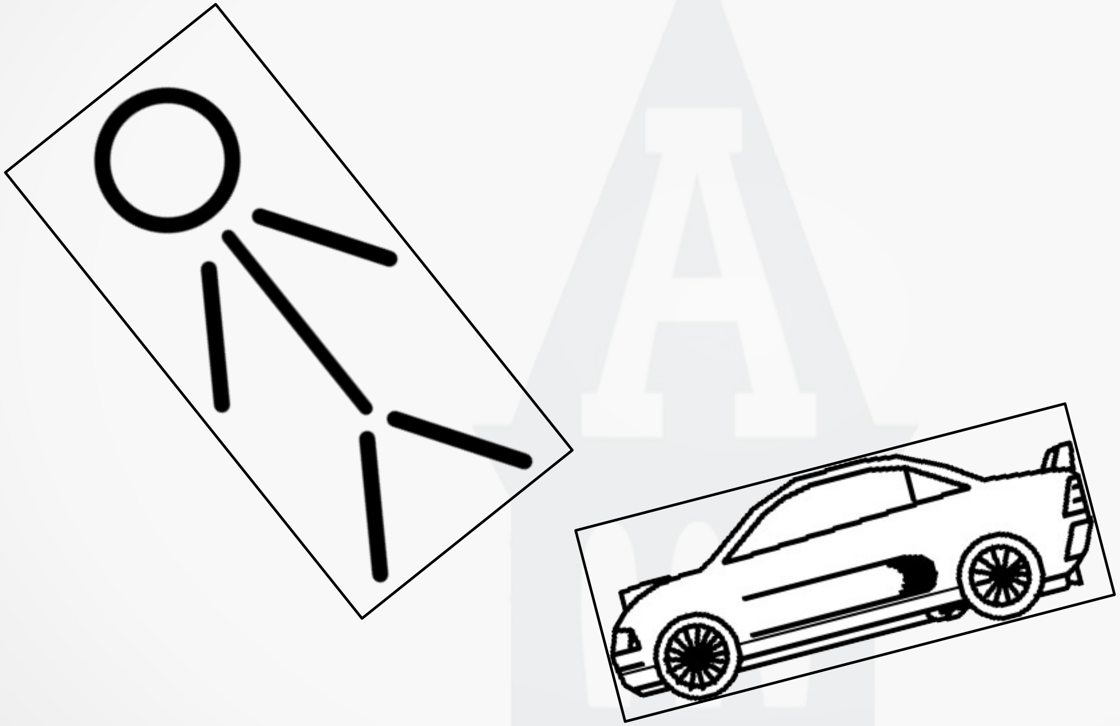
Bounding Box (BB)

- At initialization of object, find box that completely surrounds the object.
  - Initially aligned with the coordinate axes.
- As object moves/rotates, move/rotate the BB.
- Test BB with other BB for collision detection.
  - Separating Axis Theorem

Bounding Box - Initialization

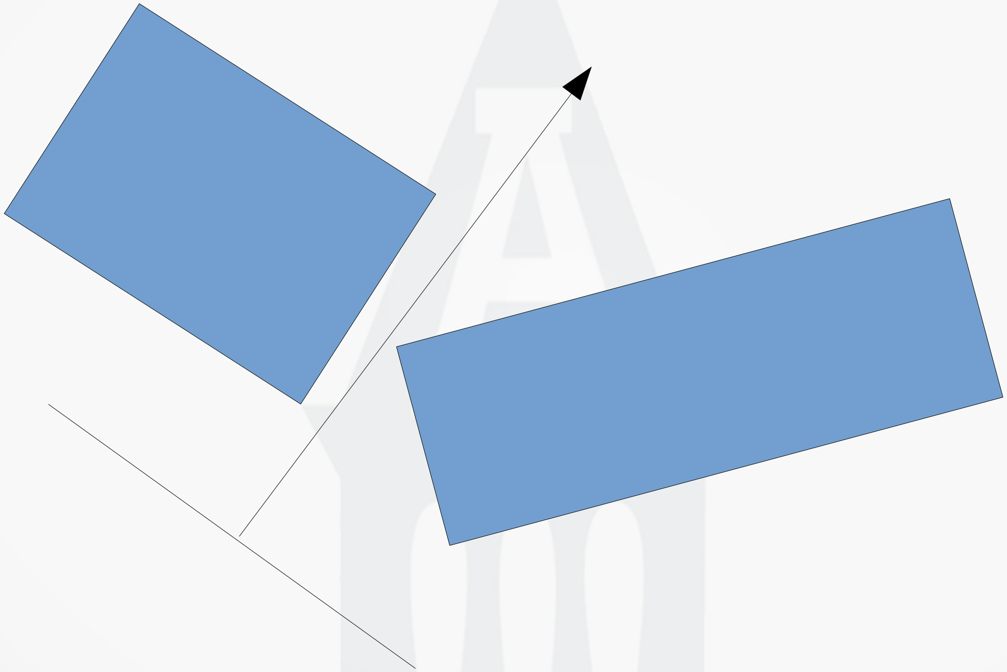


Bounding Box - Updated



Separating Axis Theorem

If you can find a line to separate the two polygons, they do not collide

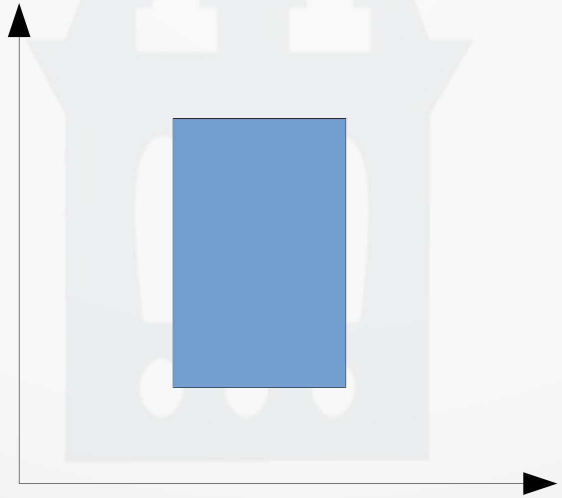


https://en.wikipedia.org/wiki/Hyperplane_separation_theorem

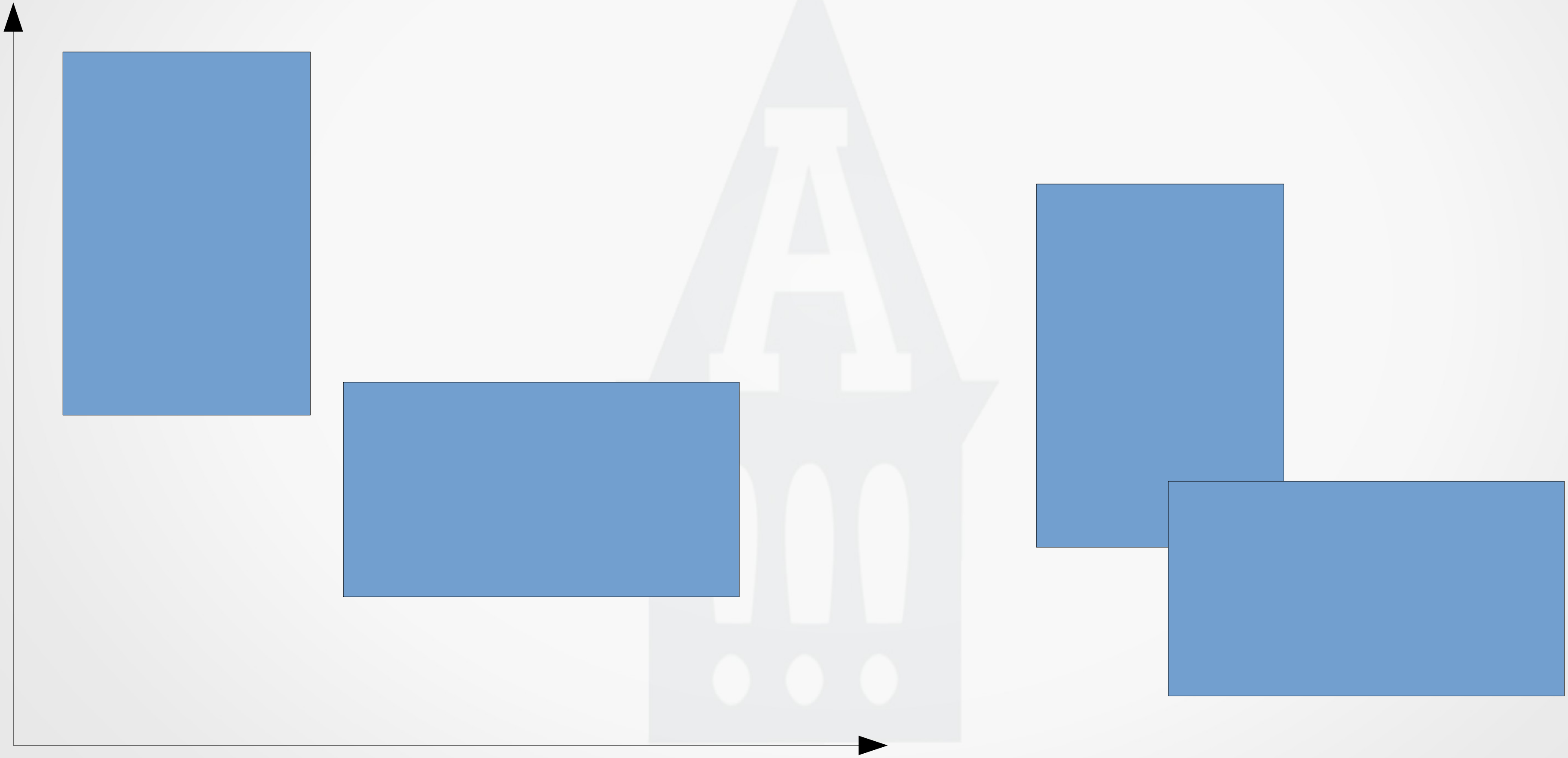
https://gamedevelopment.tutsplus.com/tutorials/collision-detection-using-the-separating-axis-theorem--gamedev-169

Axis-Aligned Bounding Box (AABB)

- At initialization of object, find box that completely surrounds the object.
- When object moves/rotates, move and adjust the AABB, but don't rotate.
- Test AABB with other AABB for collision detection.



AABB Collision Detection

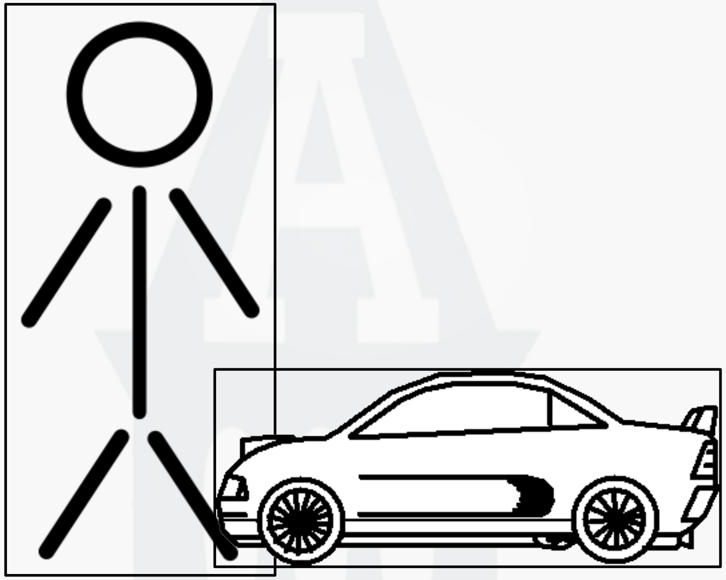


AABB – Intersection Test

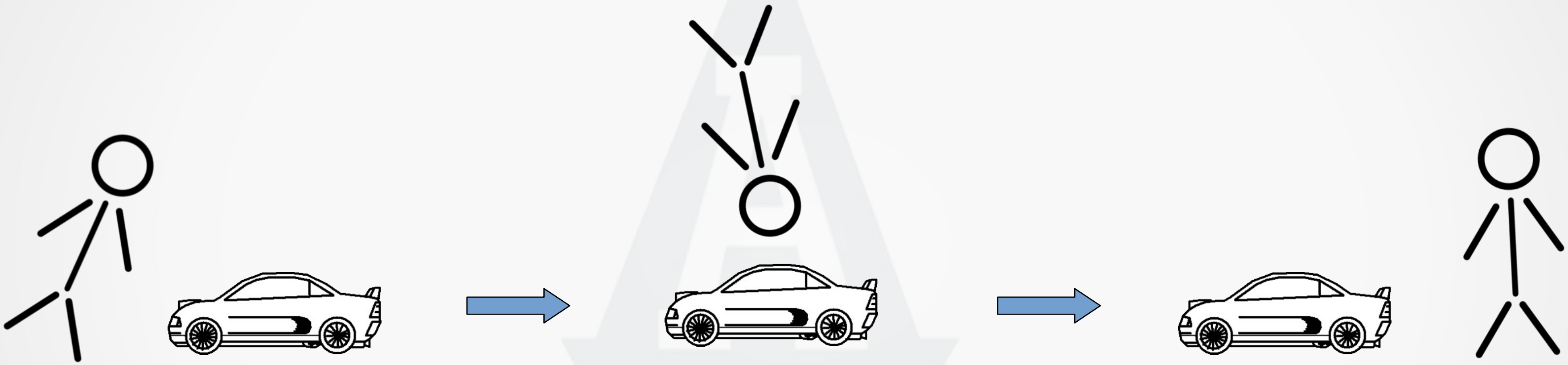
```
function intersect(r1, r2) {
    let theyDo = !(
        r2.left > r1.right ||
        r2.right < r1.left ||
        r2.top > r1.bottom ||
        r2.bottom < r1.top);

    return theyDo;
}
```

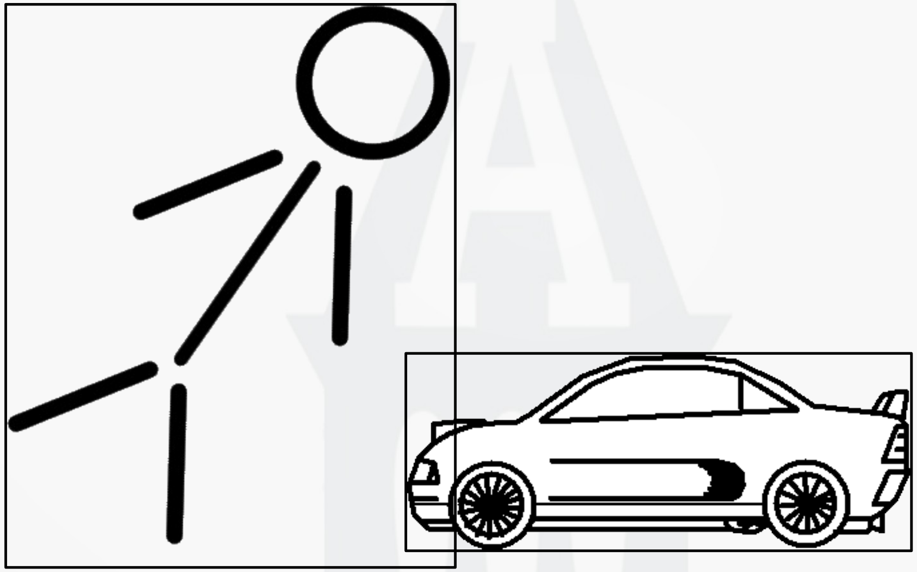
Axis-Aligned Bounding Box



Axis-Aligned Bounding Box



Axis-Aligned Bounding Box

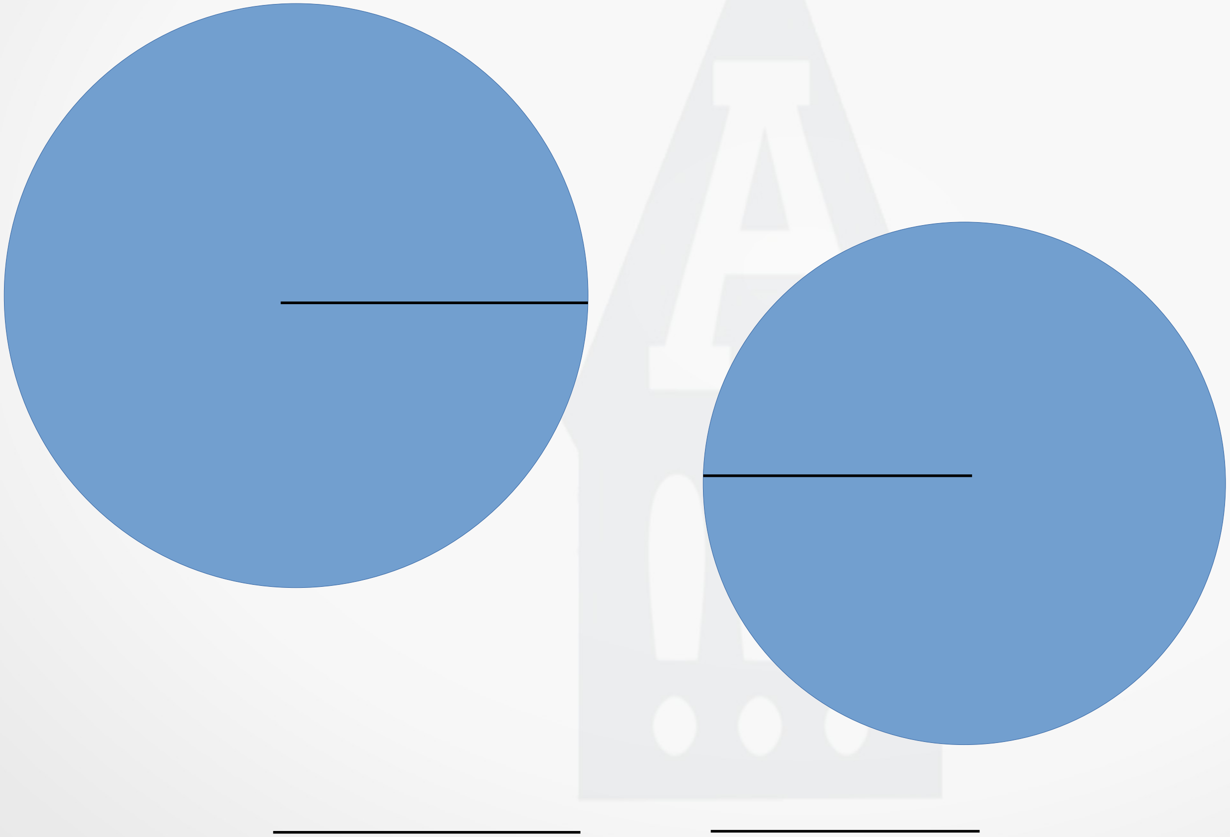


Sphere/Circle

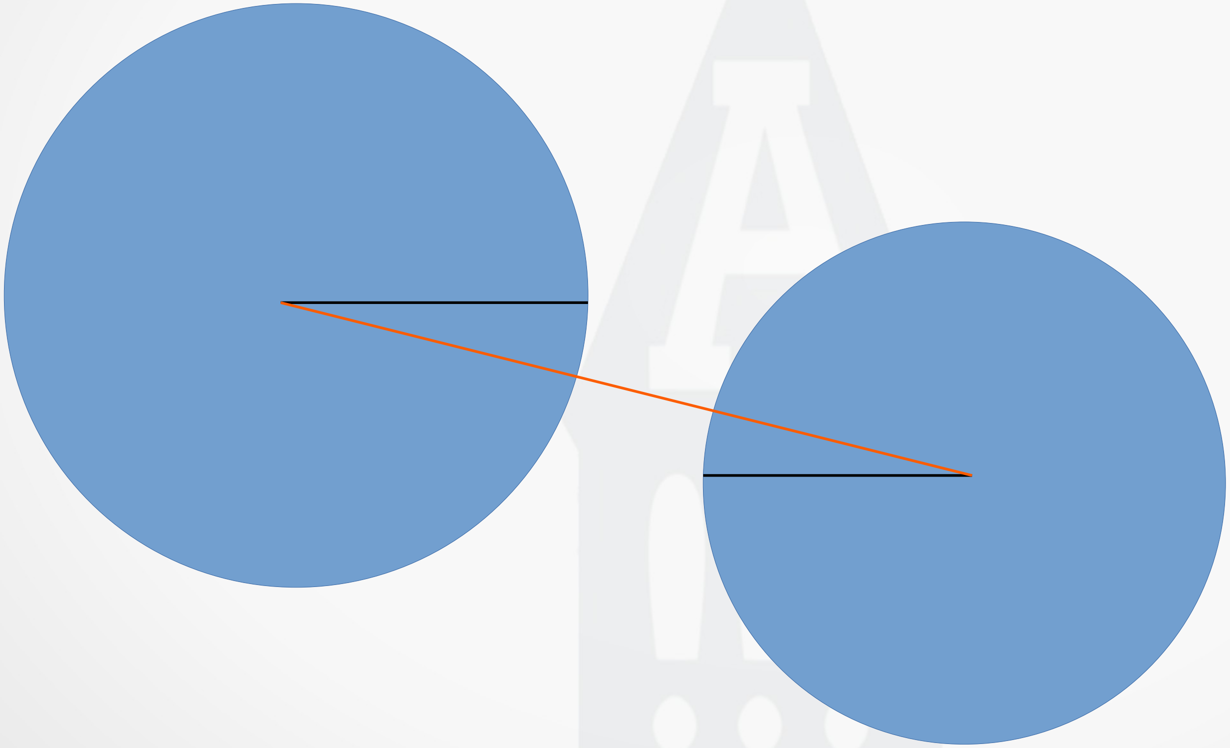
- At initialization, find sphere that surrounds the object.
- When the object rotates...nothing
- When the object moves, move the sphere center
- Collision detection
  - Compute sum of the two sphere radii
  - Compare that sum with distance between centers

Sphere/Circle

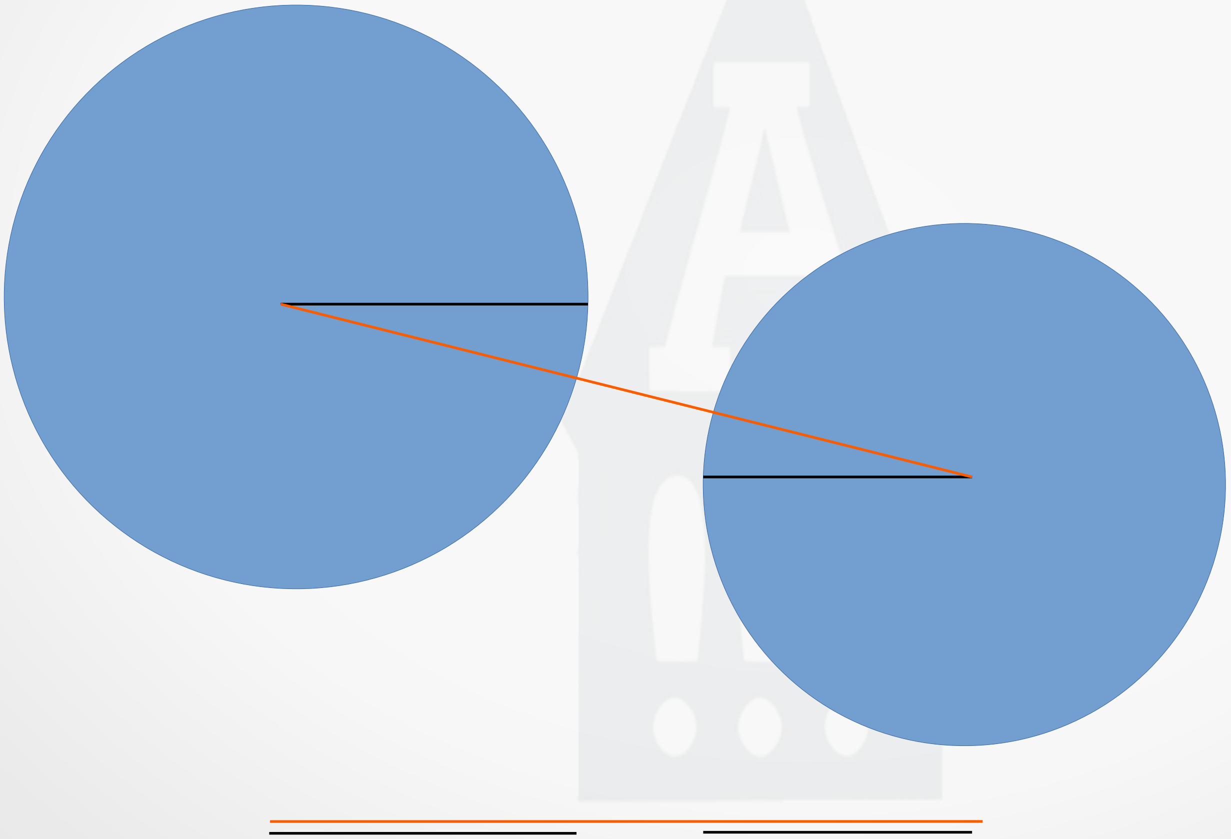
Sphere/Circle



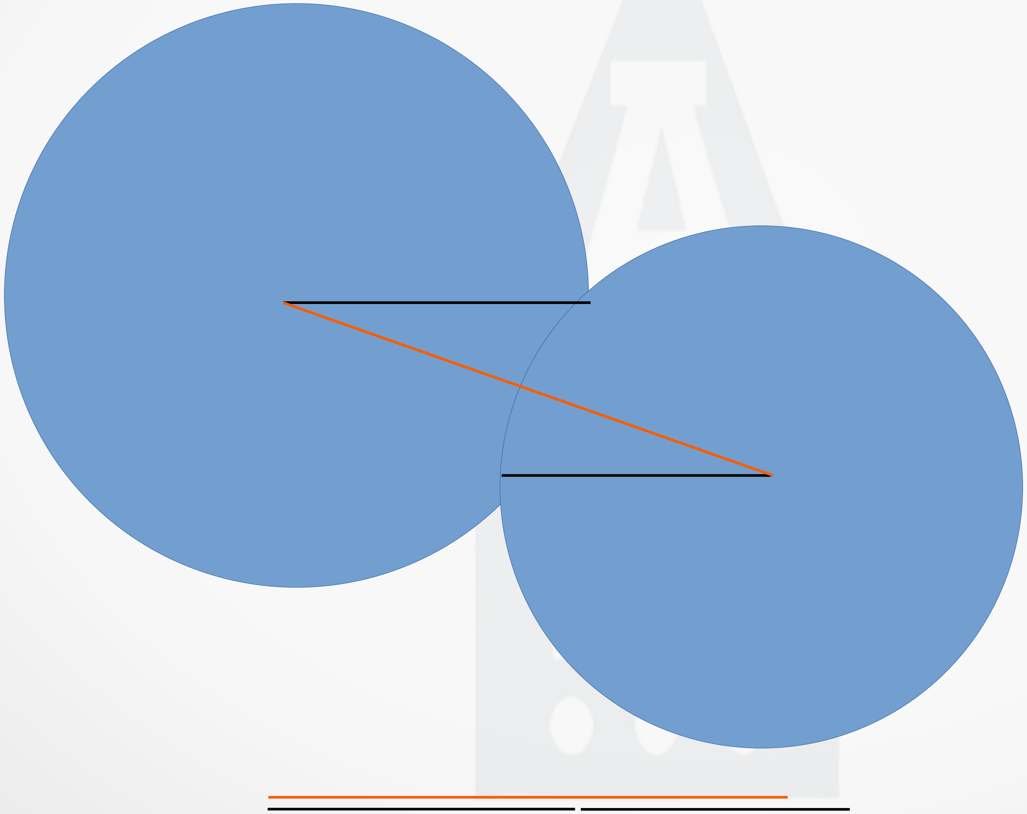
Sphere/Circle



Sphere/Circle



Sphere/Circle



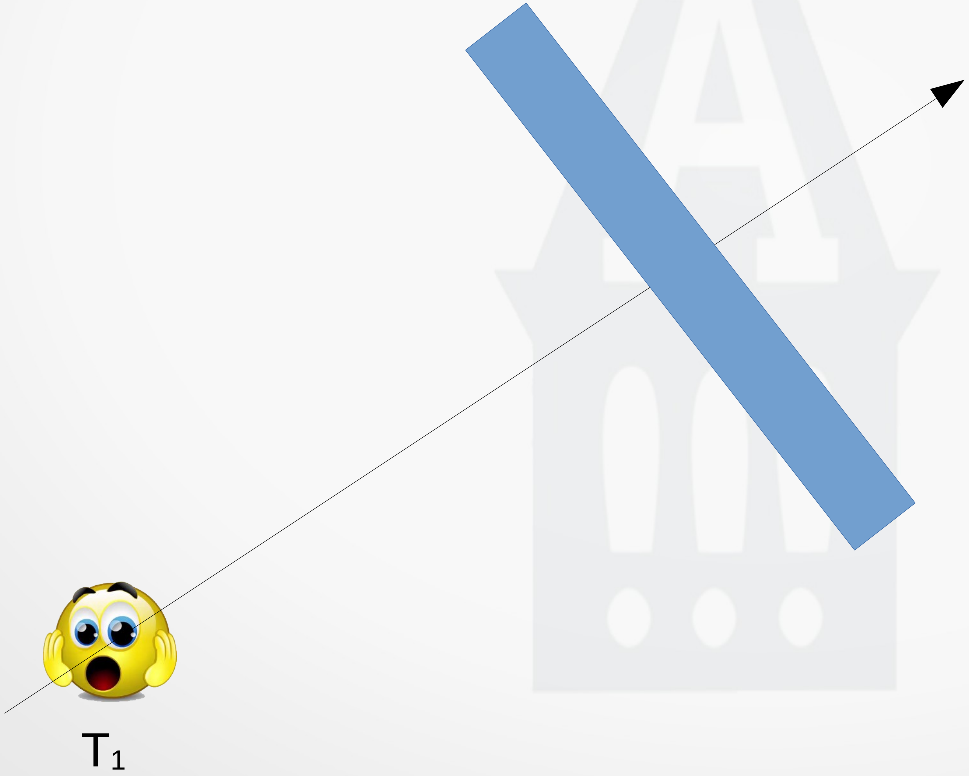
Comparing BB, AABB, Sphere

	Fast	Accurate	Good Enough
Bounding Box	3	1	1 ?
Axis-Aligned Bounding Box	1 ✓	3	3 ?
Sphere/Circle	2 ✓	2	2 ?

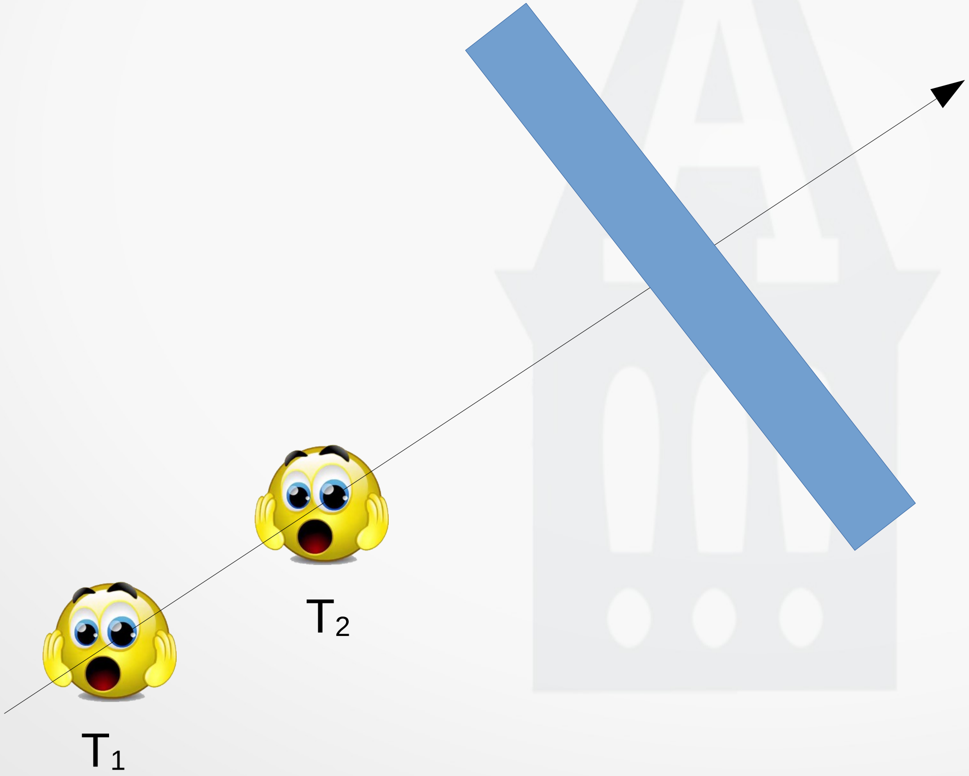
None handles complex shapes very well

Take a Side Trip - Important Note

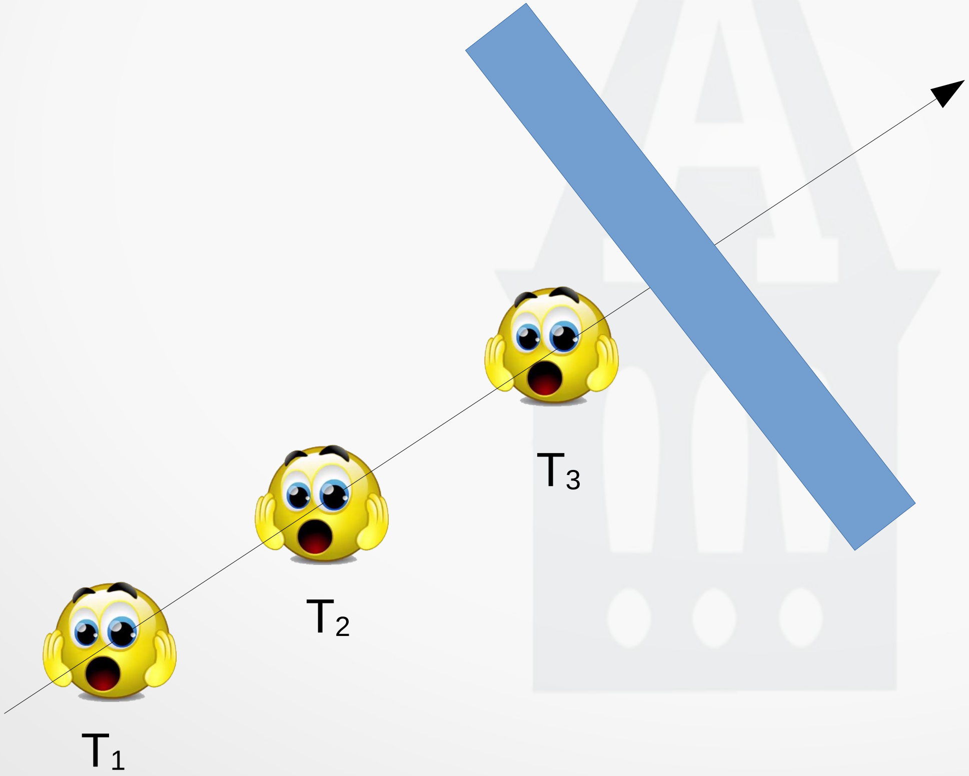
Simulation Rate vs Detection



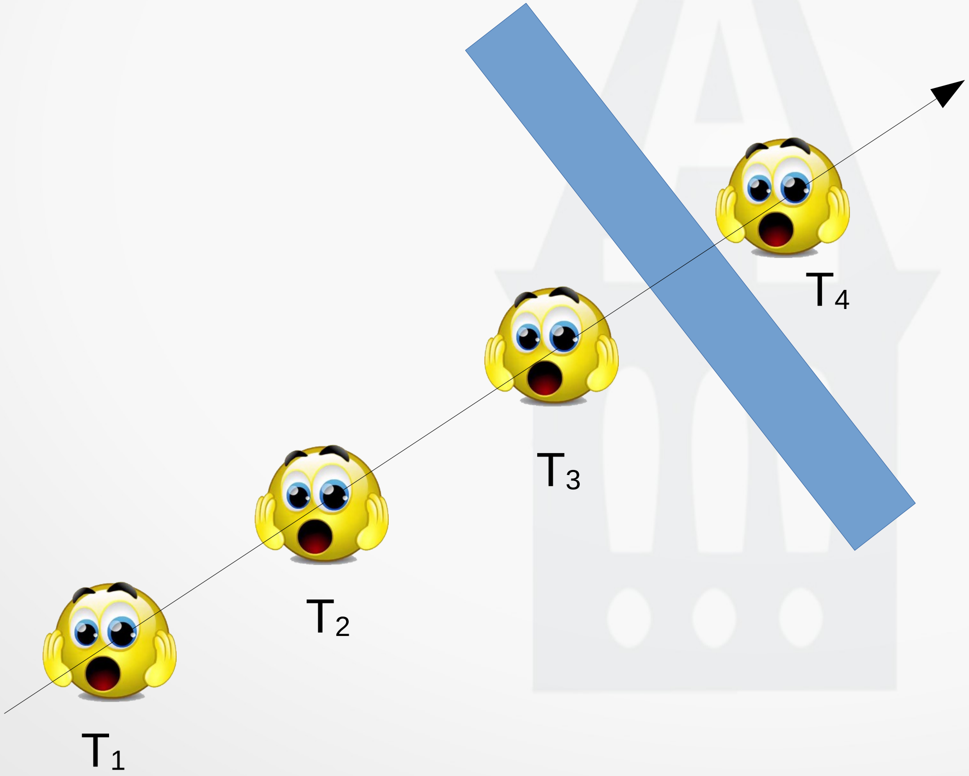
Simulation Rate vs Detection



Simulation Rate vs Detection

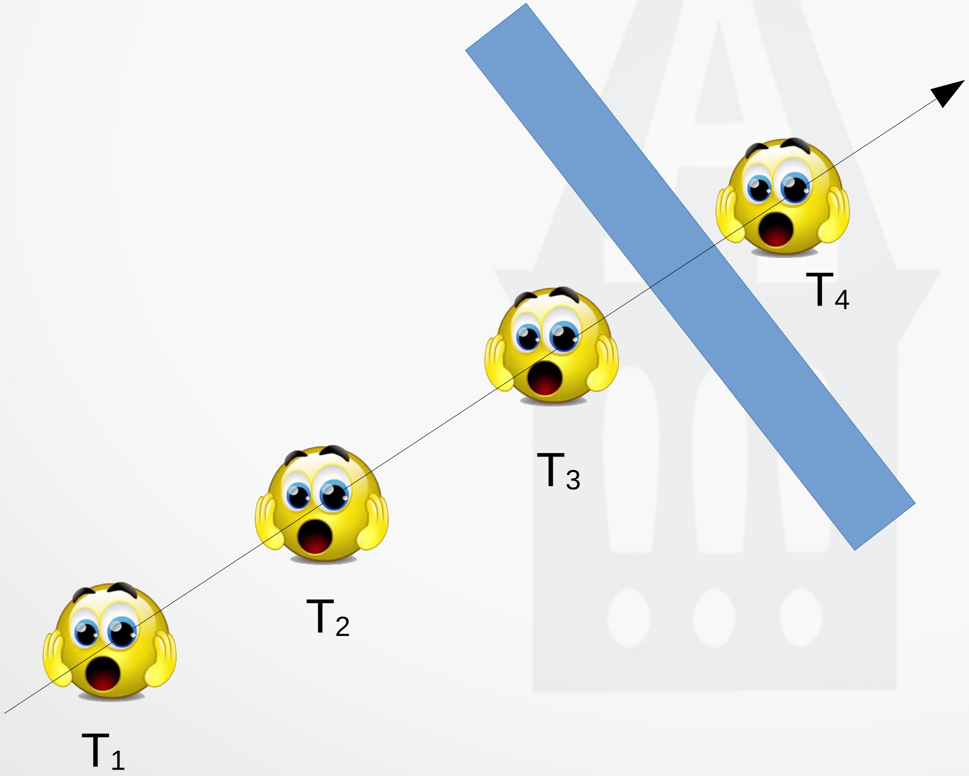


Simulation Rate vs Detection



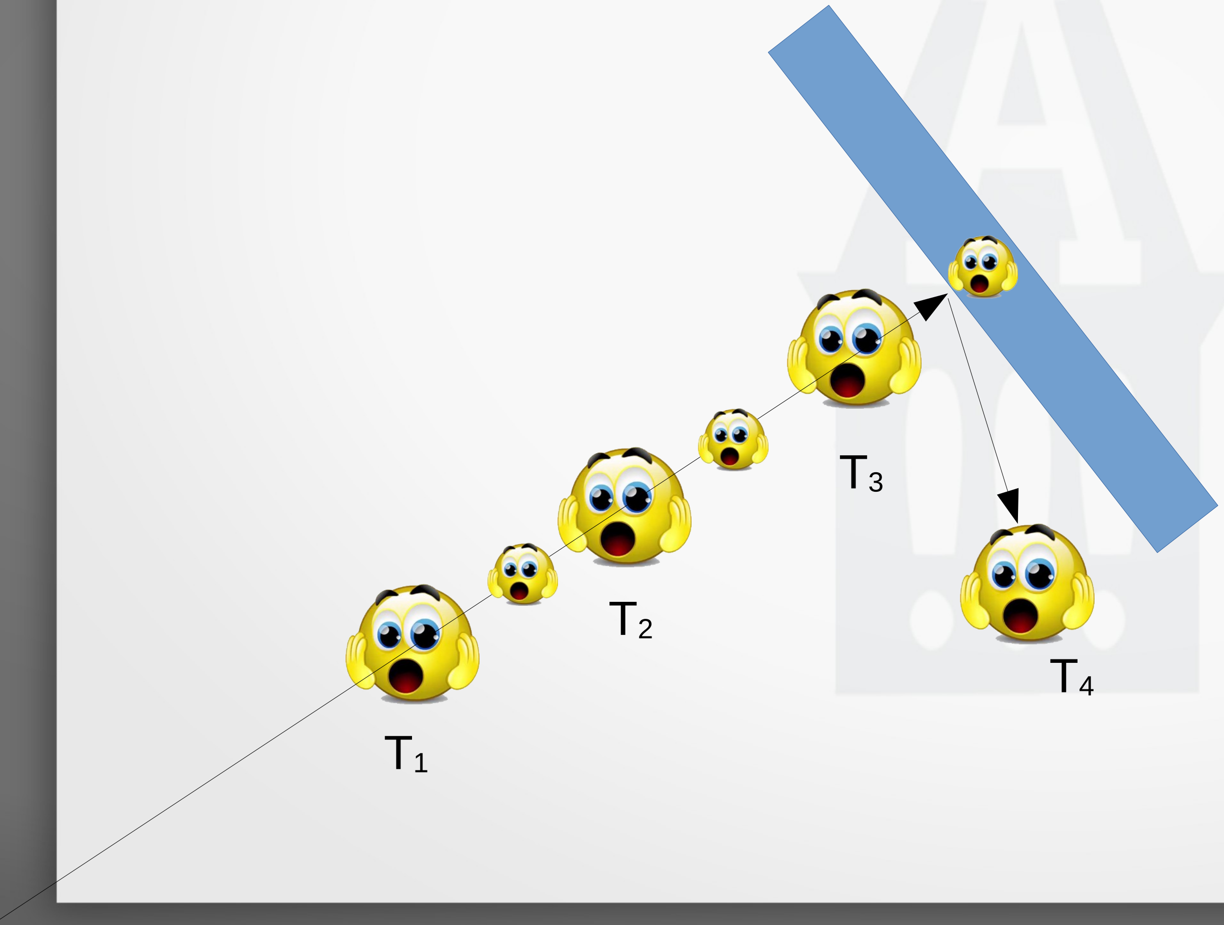
Simulation Rate vs Detection

- Game simulation (frame rate) may not be fast enough



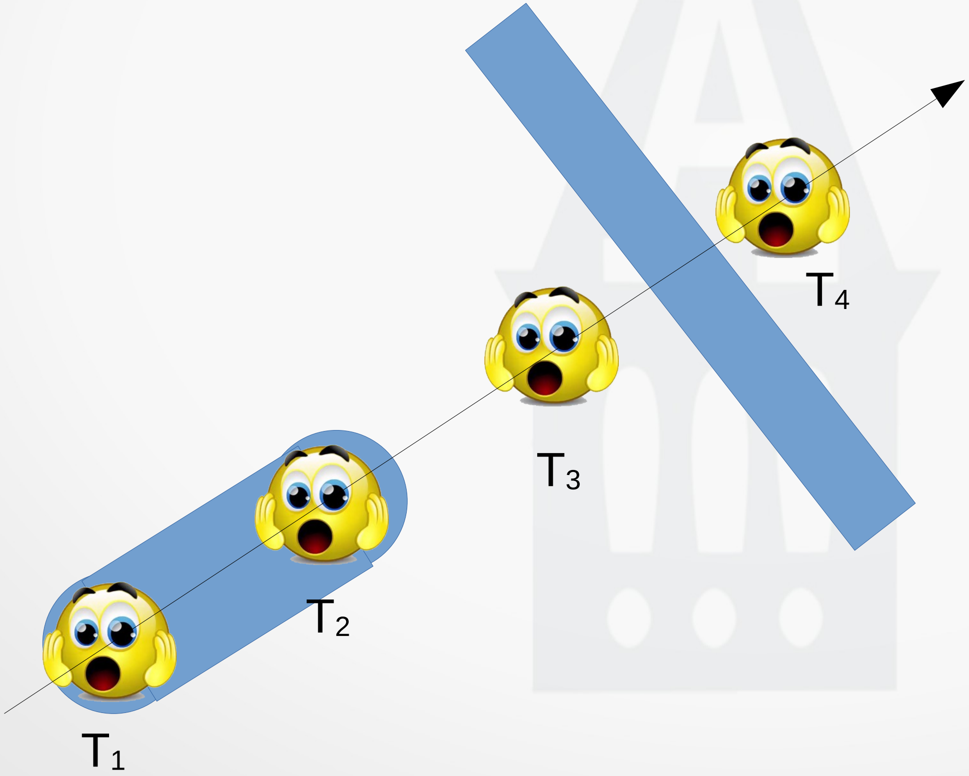
Simulation Rate vs Detection

- Increase simulation rate, but not rendering rate



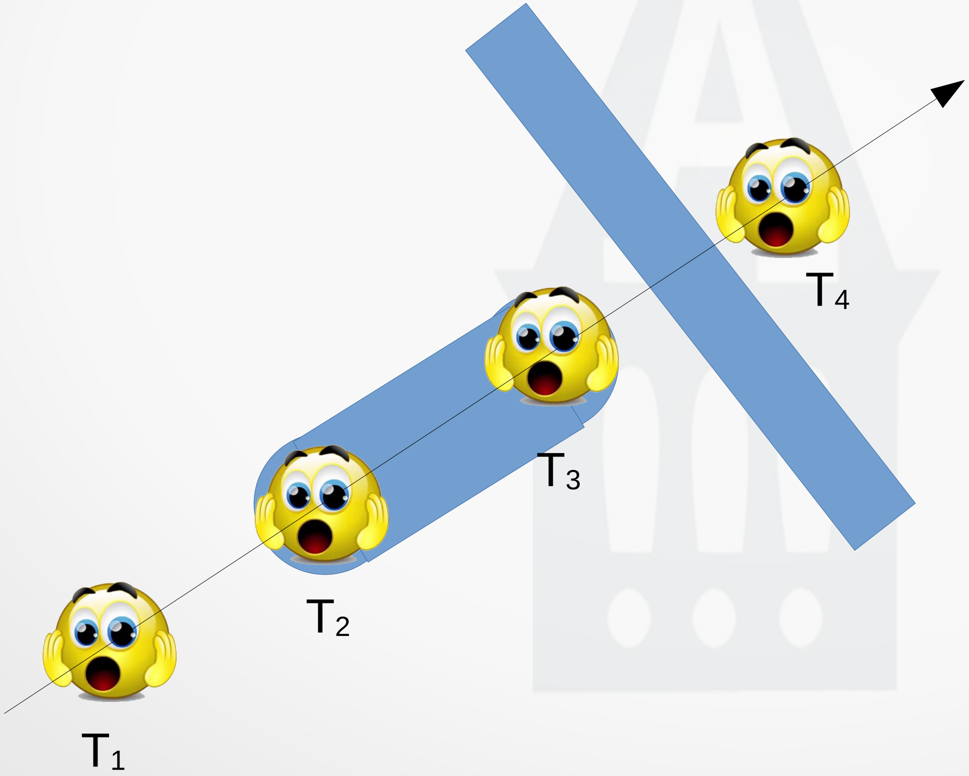
Simulation Rate vs Detection

- Use a swept shape, in this case…
  - Two spheres and a rectangle (a capsule)



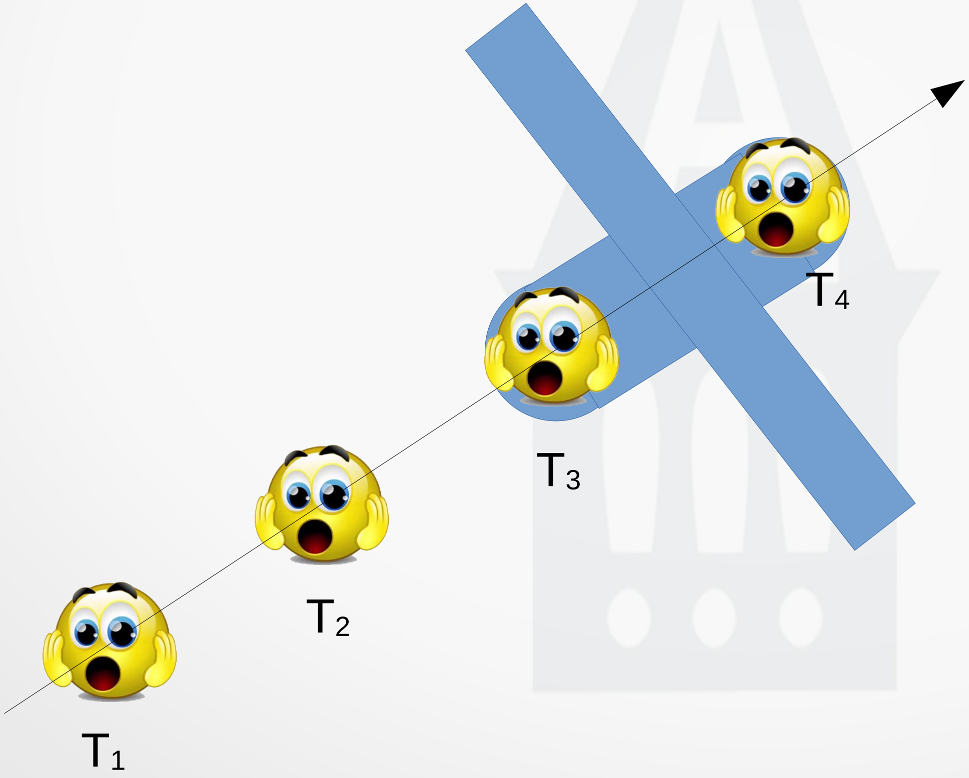
Simulation Rate vs Detection

- Use a swept shape, in this case…
  - Two spheres and a rectangle (a capsule)



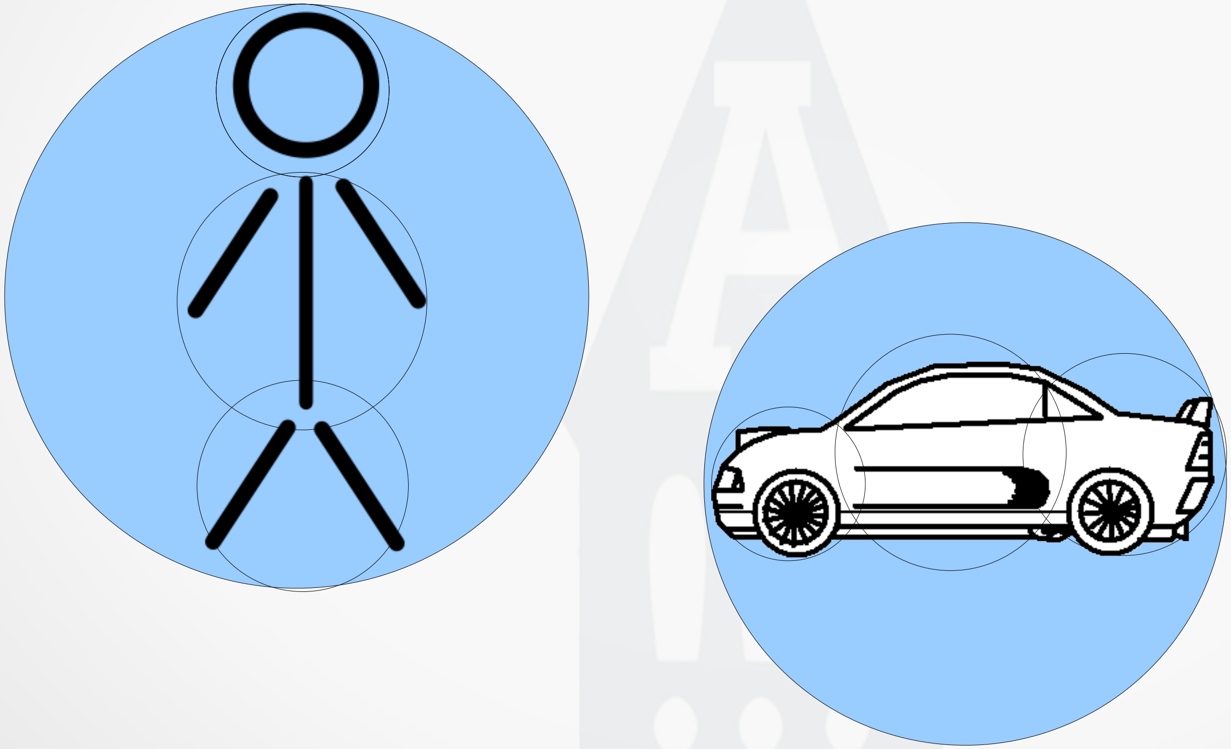
Simulation Rate vs Detection

- Use a swept shape, in this case…
  - Two spheres and a rectangle (a capsule)

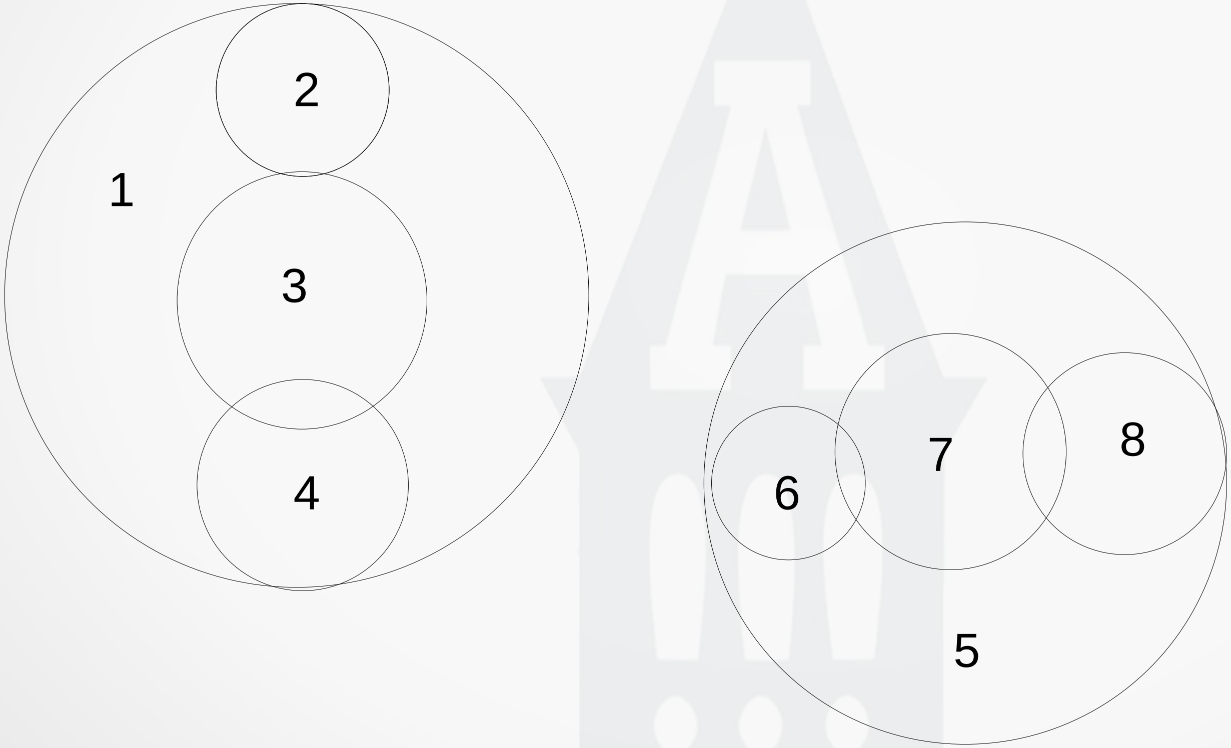


Hierarchical Subdivision Methods

Hierarchical Bounding Volumes

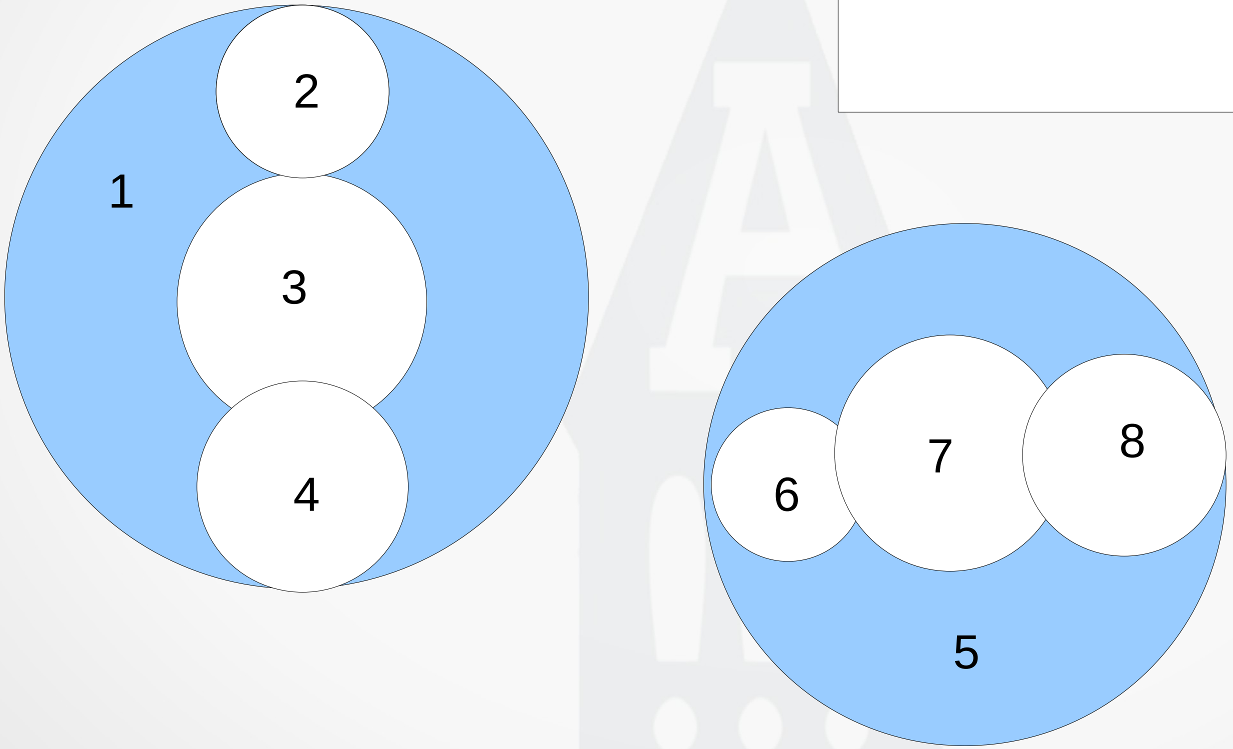


Hierarchical Bounding Volumes

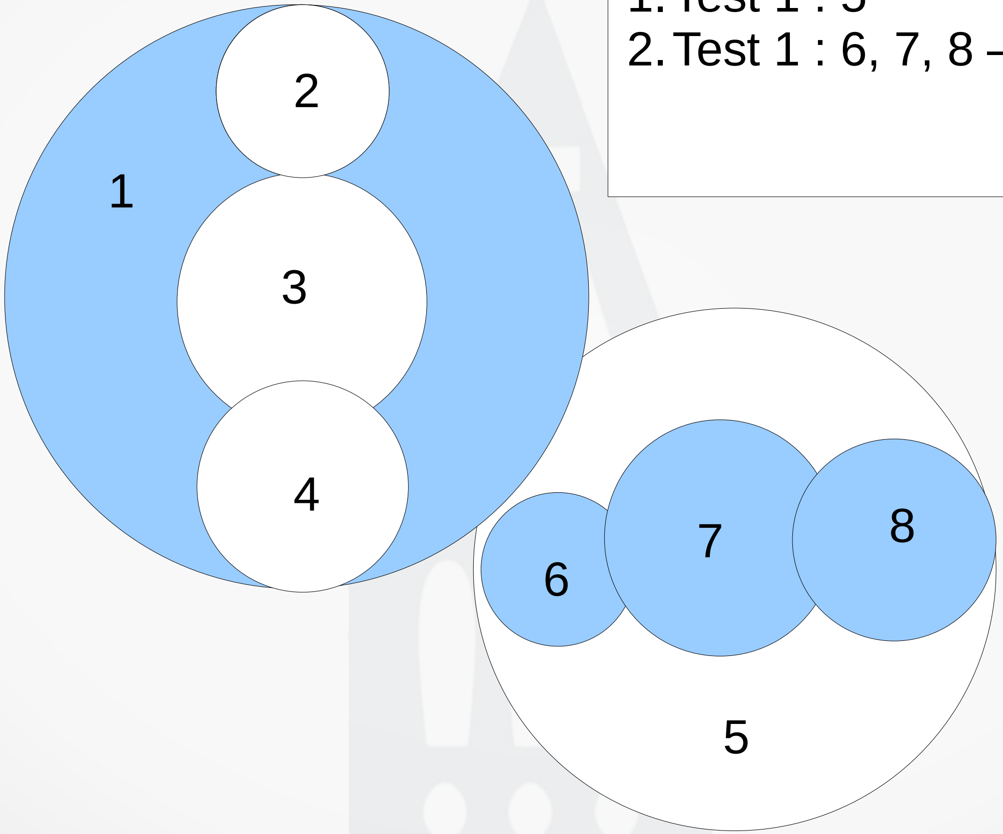


Hierarchical Bounding Volumes

1. Test 1 : 5 – early reject

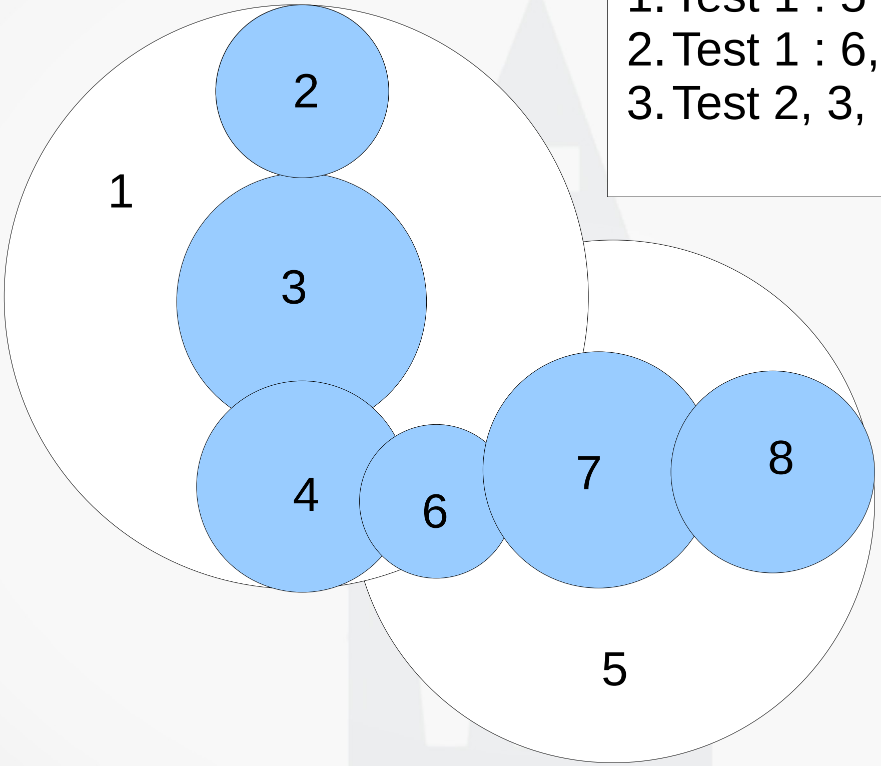


Hierarchical Bounding Volumes



1. Test 1 : 5
2. Test 1 : 6, 7, 8 – early reject

Hierarchical Bounding Volumes

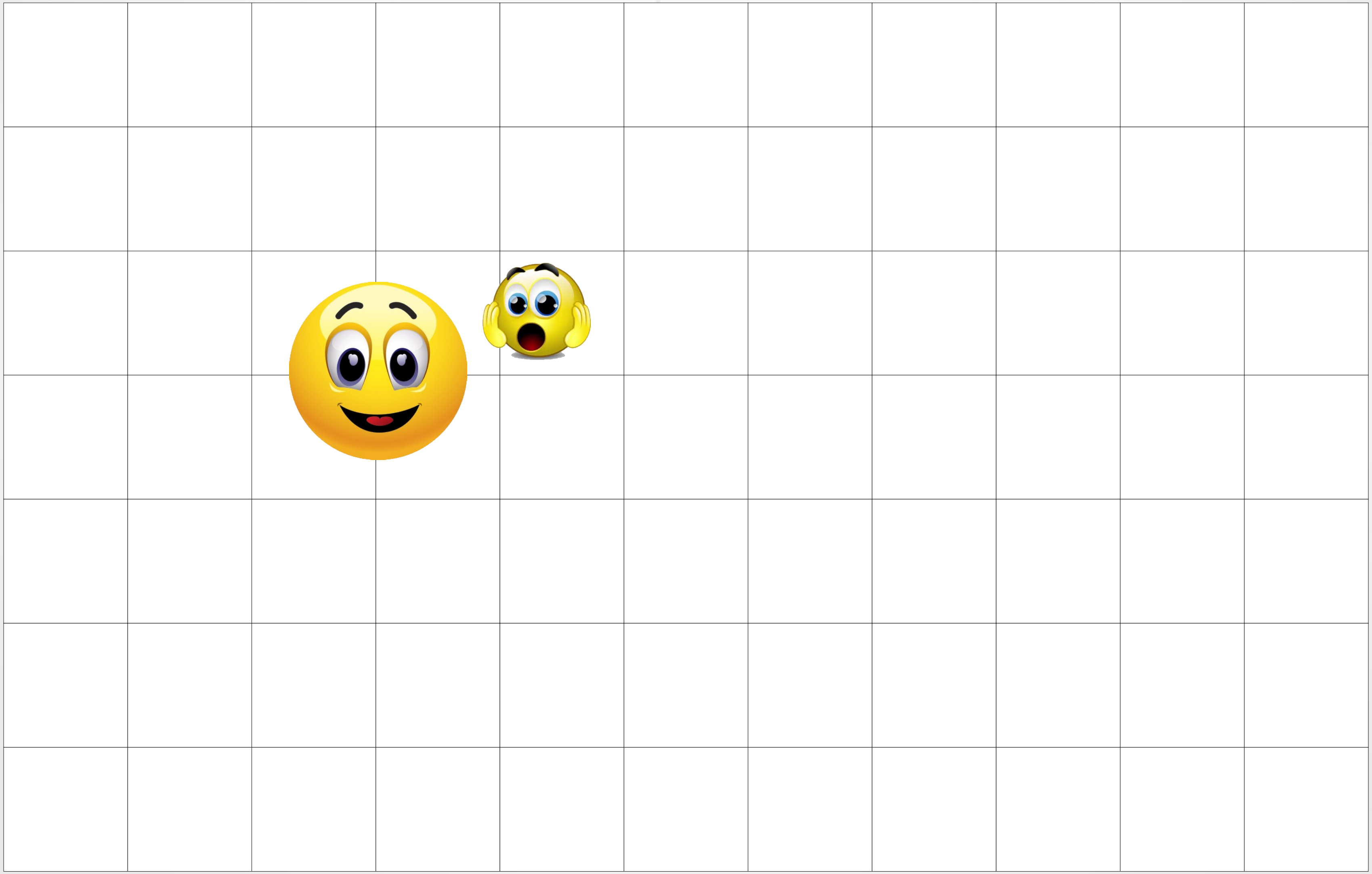


1. Test 1 : 5
2. Test 1 : 6, 7, 8
3. Test 2, 3, 4 : 6, 7, 8

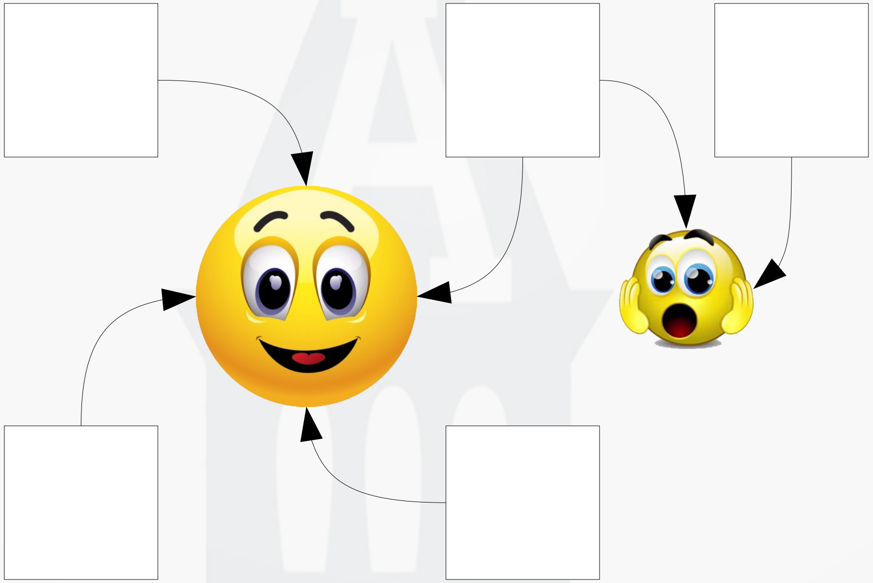
Grid Subdivison

- Subdivide the space into uniformly sized spaces
  - square for 2D
  - cube for 3D
- Initialize and maintain object ownership by space

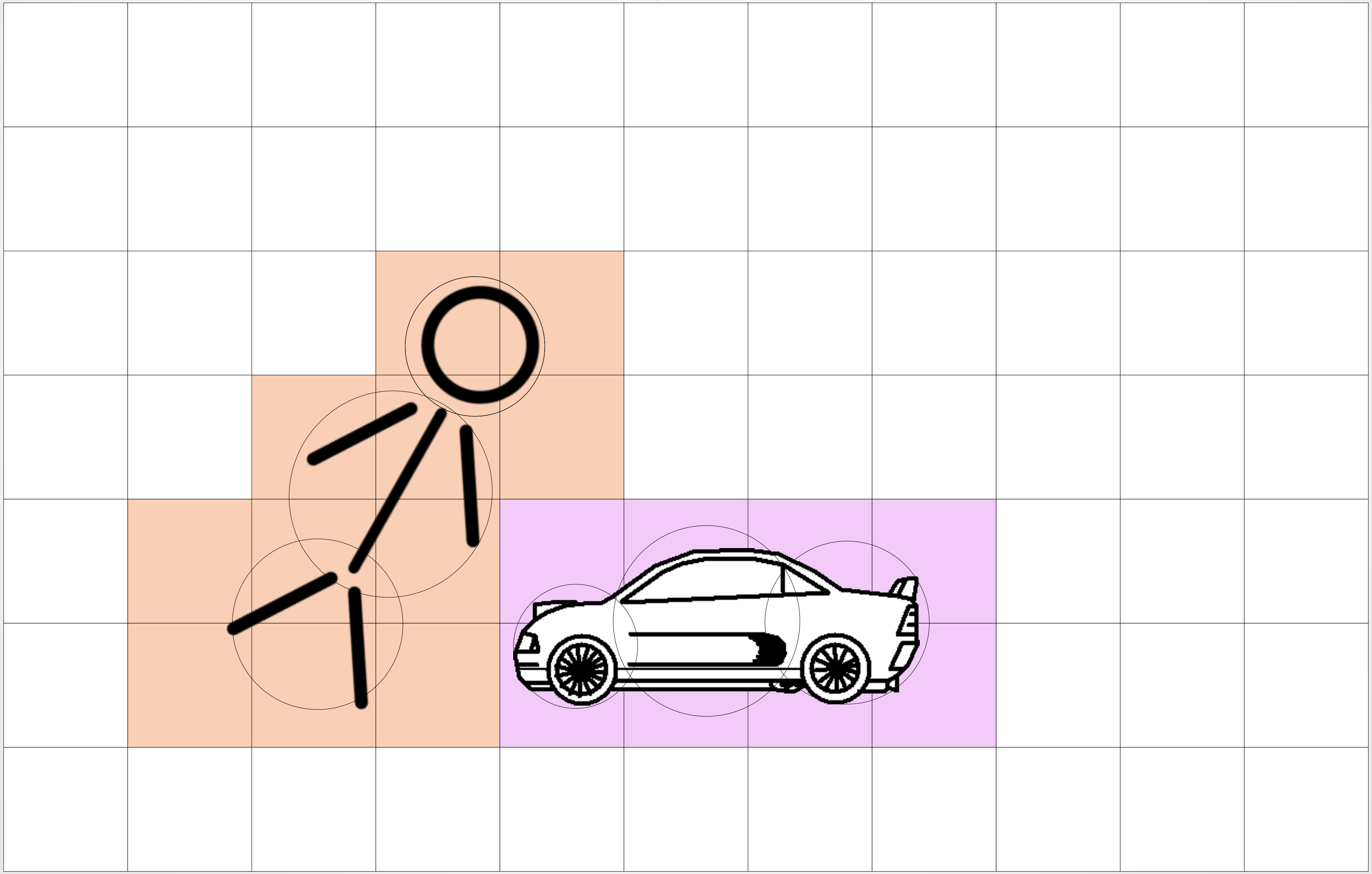
Grid Subdivison



Grid Subdivison

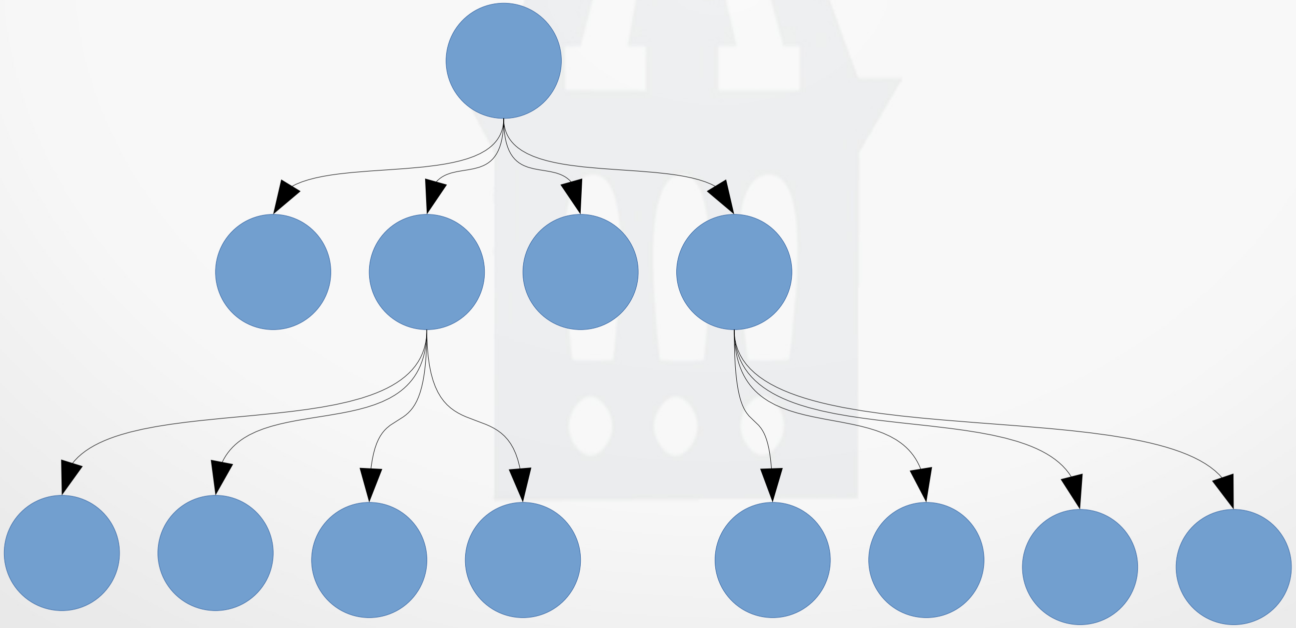


Grid Subdivison

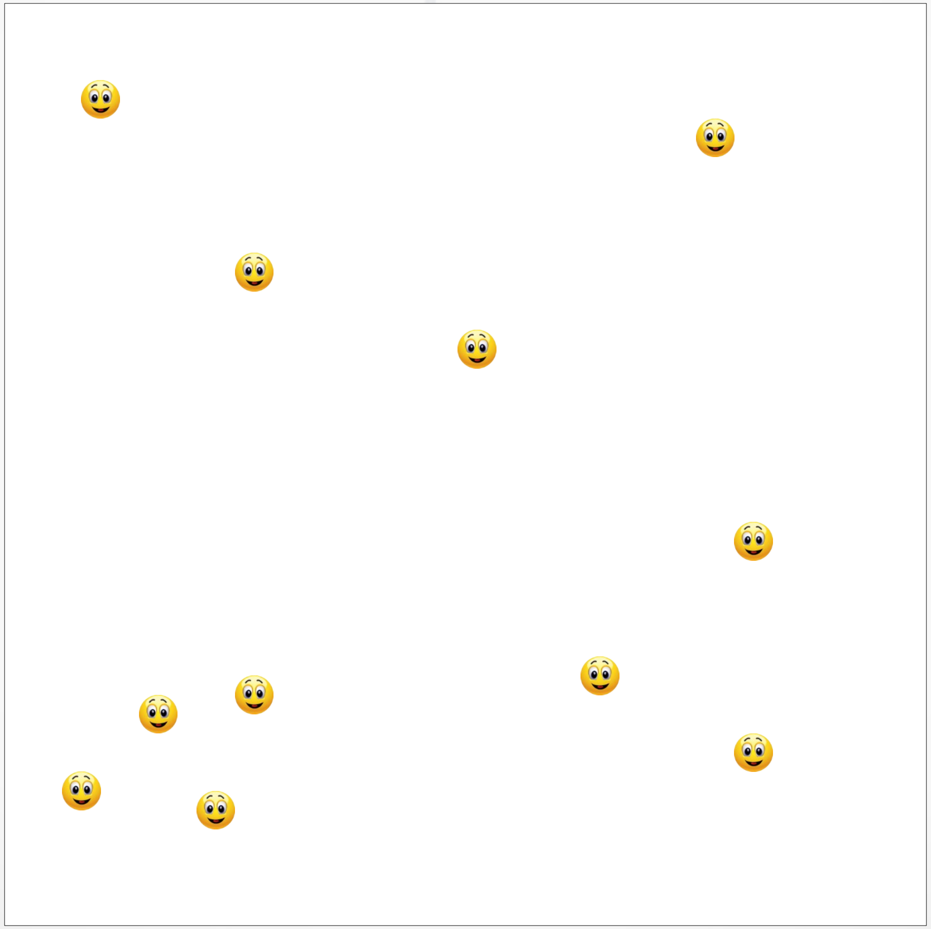


Quad-Tree Collision Detection

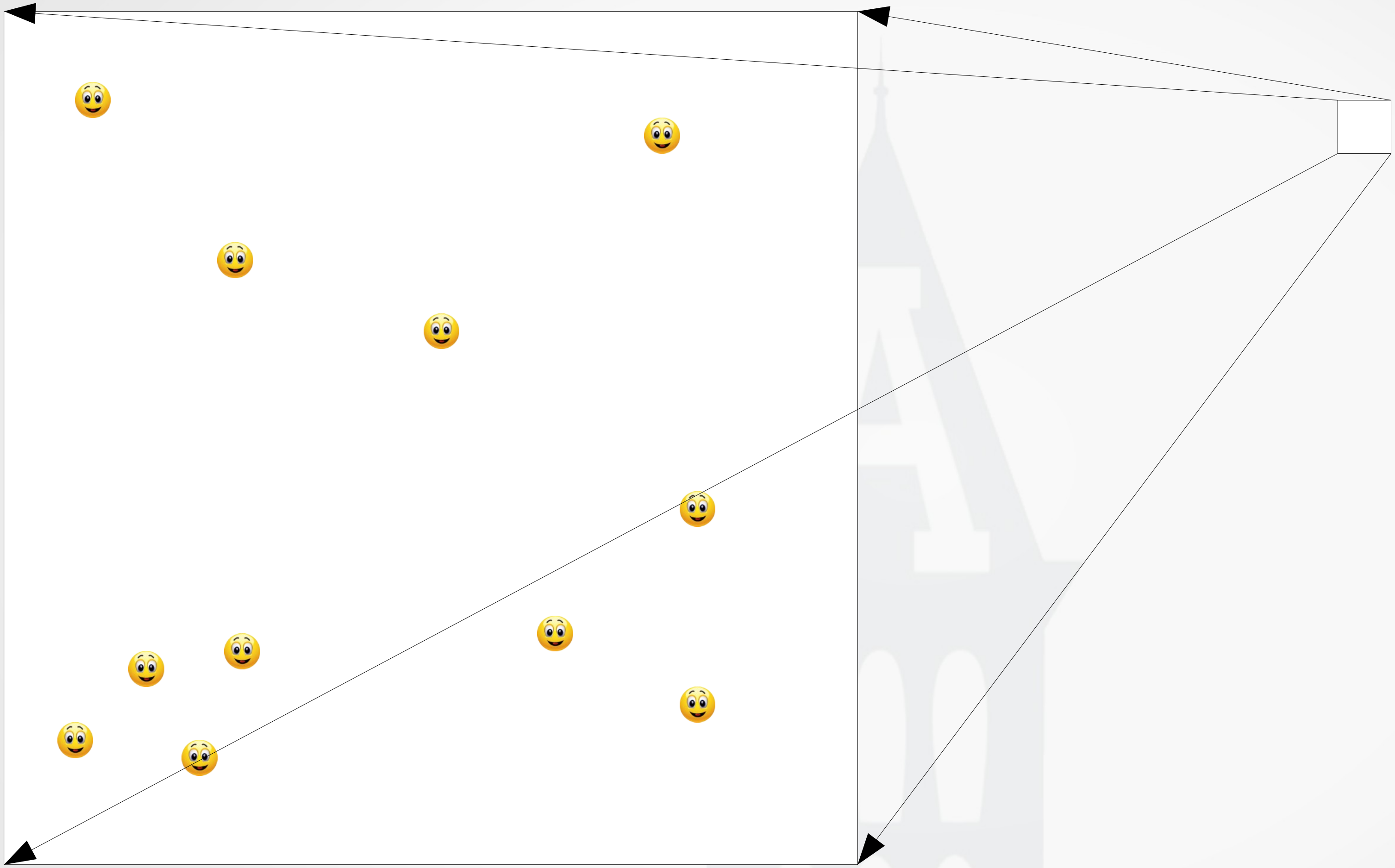
- A tree where each node has 4 children
  - Not up to 4 children, but always 4 children
- Subdivided regions can be rectangular or square
- Adaptively subdivide based on membership criteria



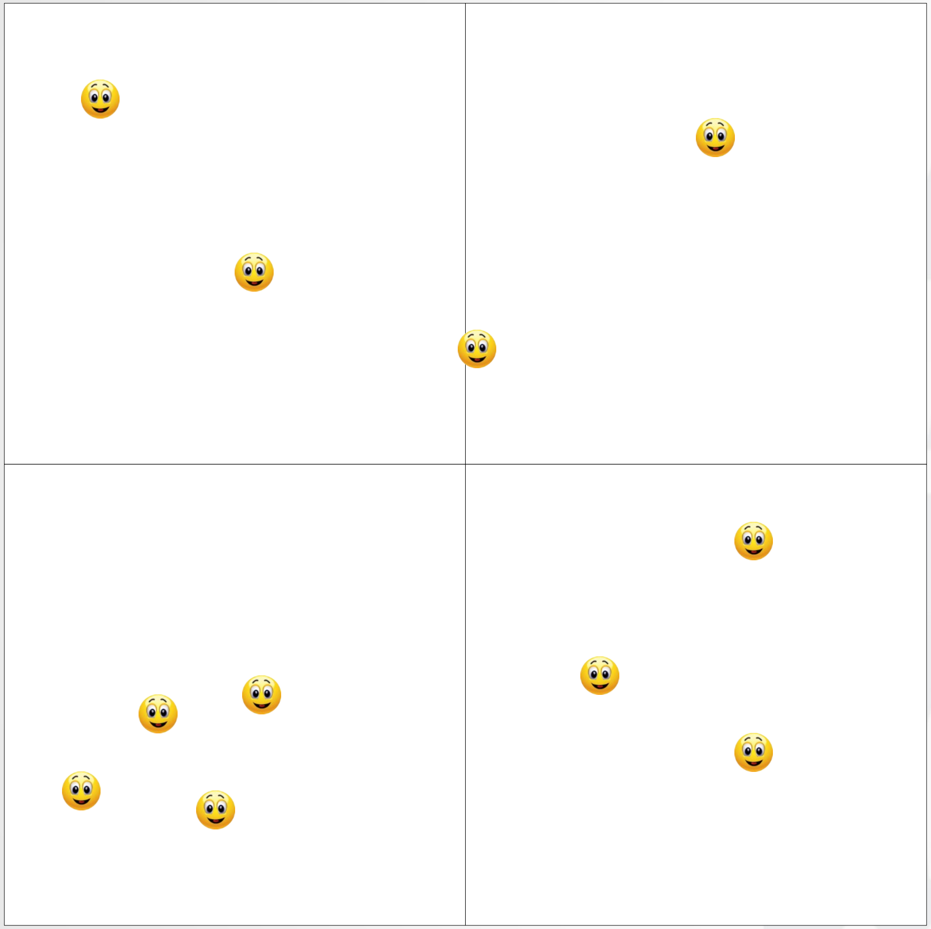
Quad-Tree Construction

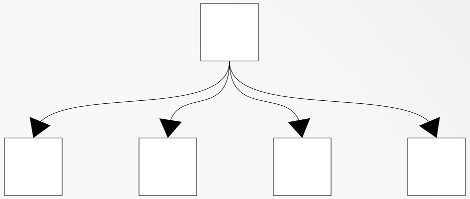


Quad-Tree Construction

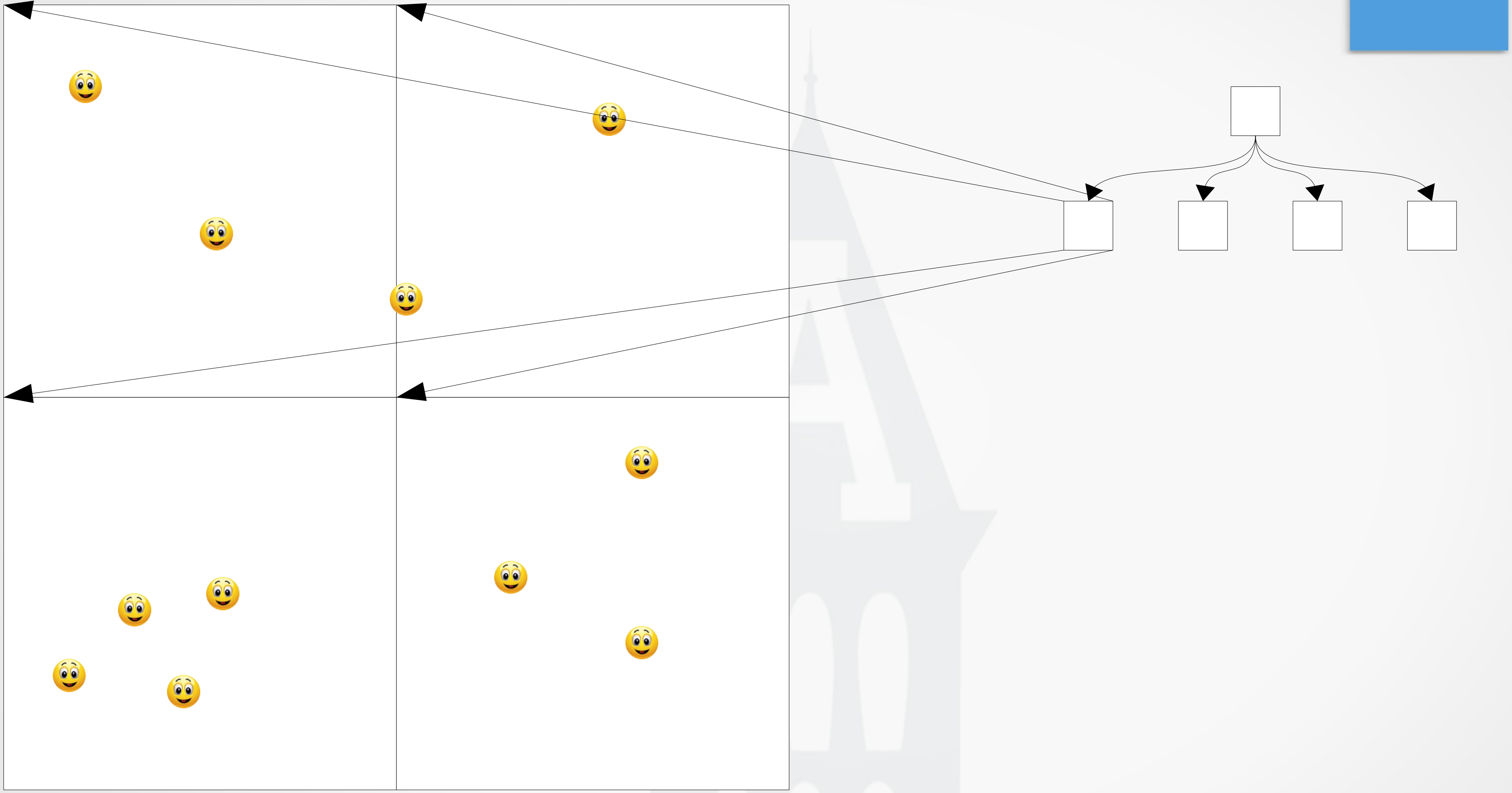


Quad-Tree Construction

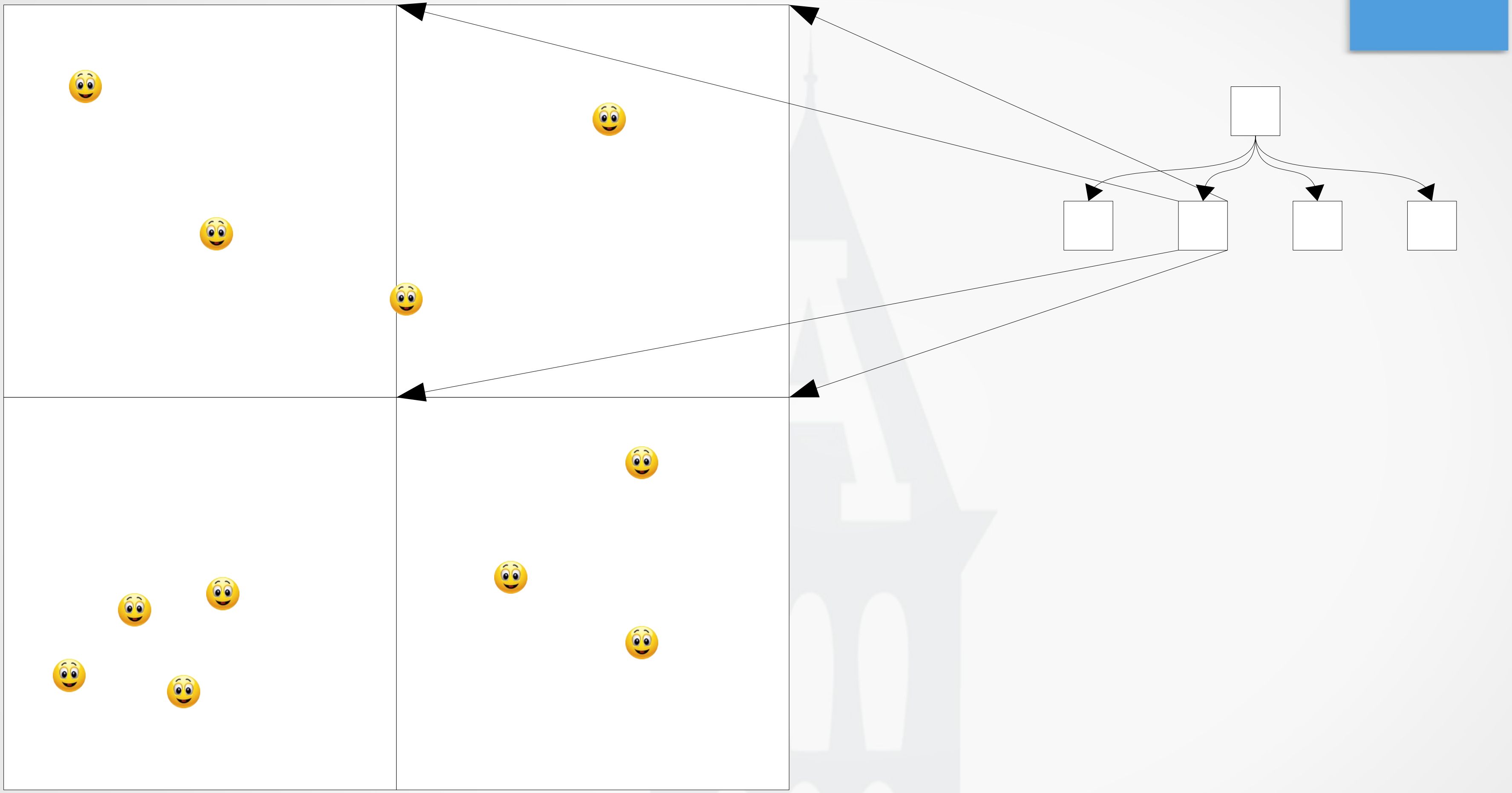




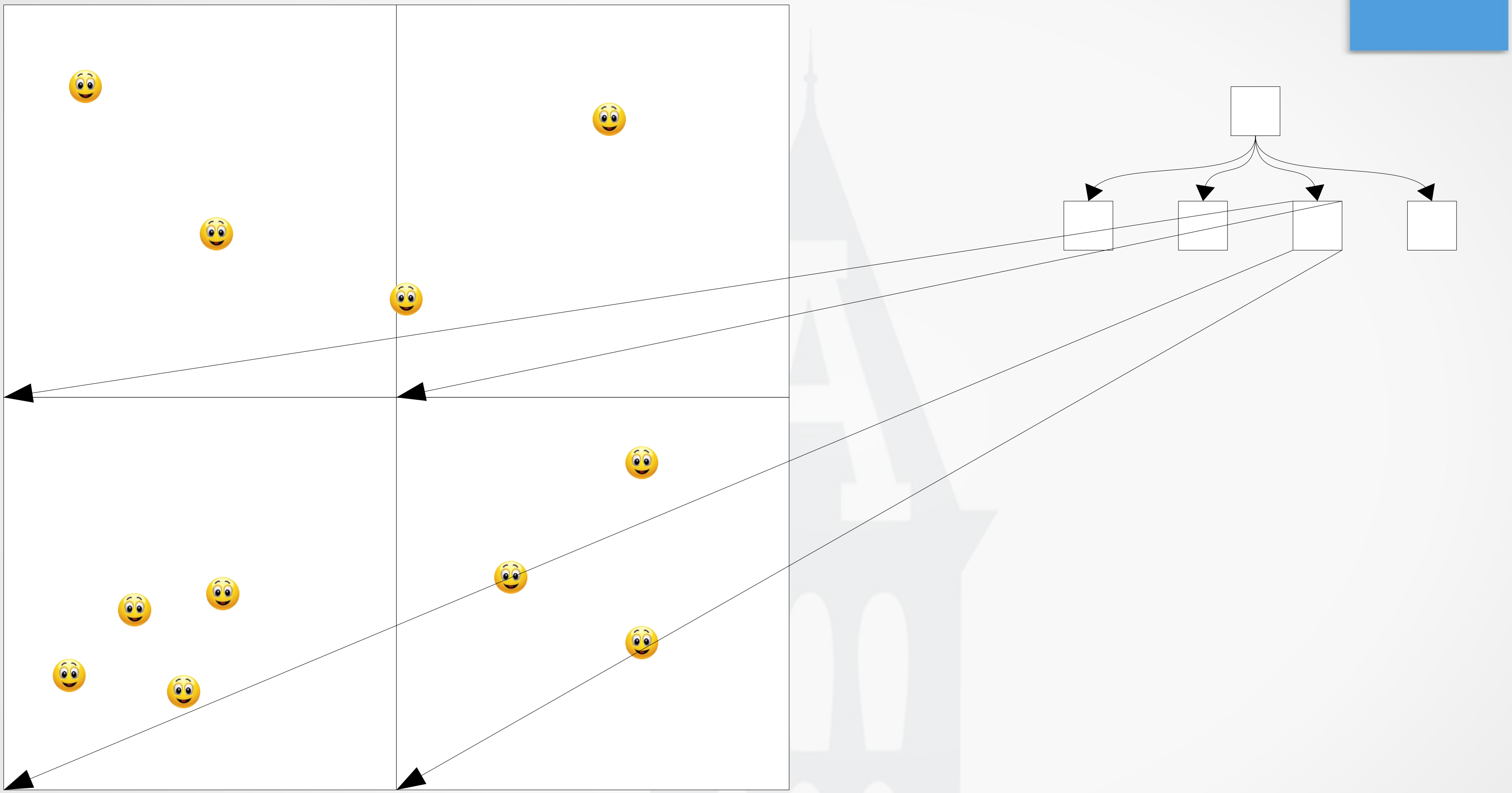
Quad-Tree Construction



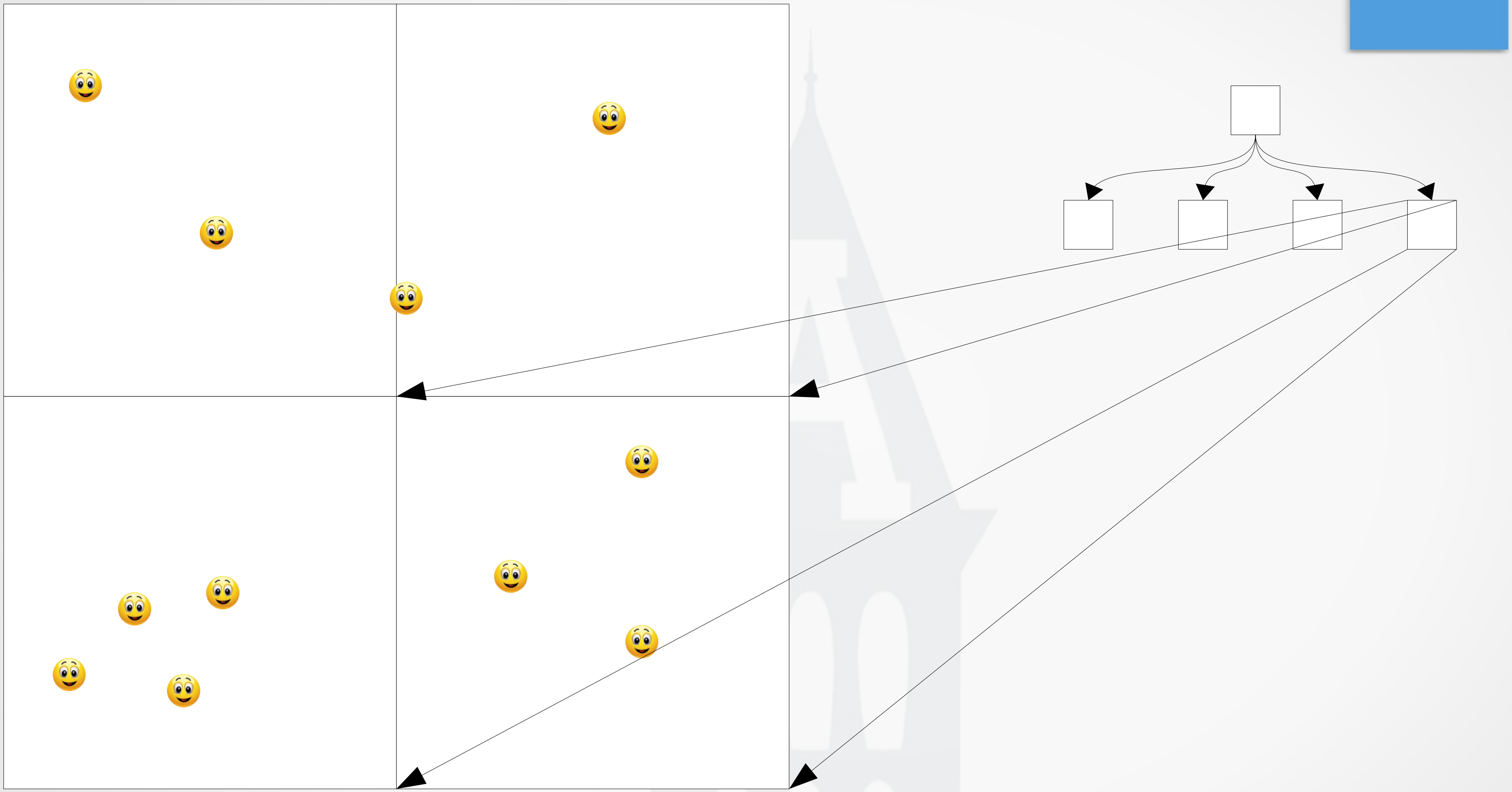
Quad-Tree Construction



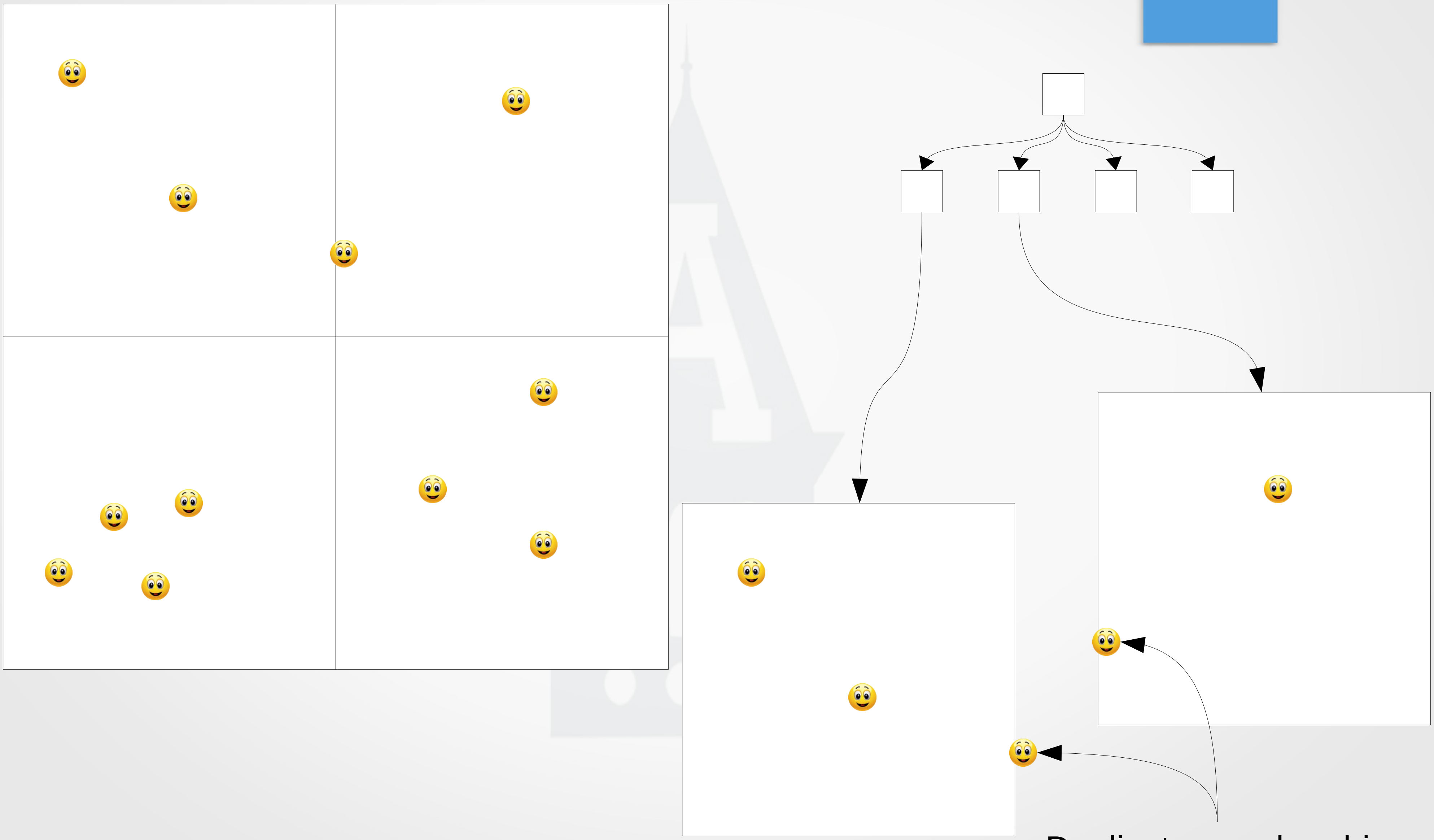
Quad-Tree Construction



Quad-Tree Construction

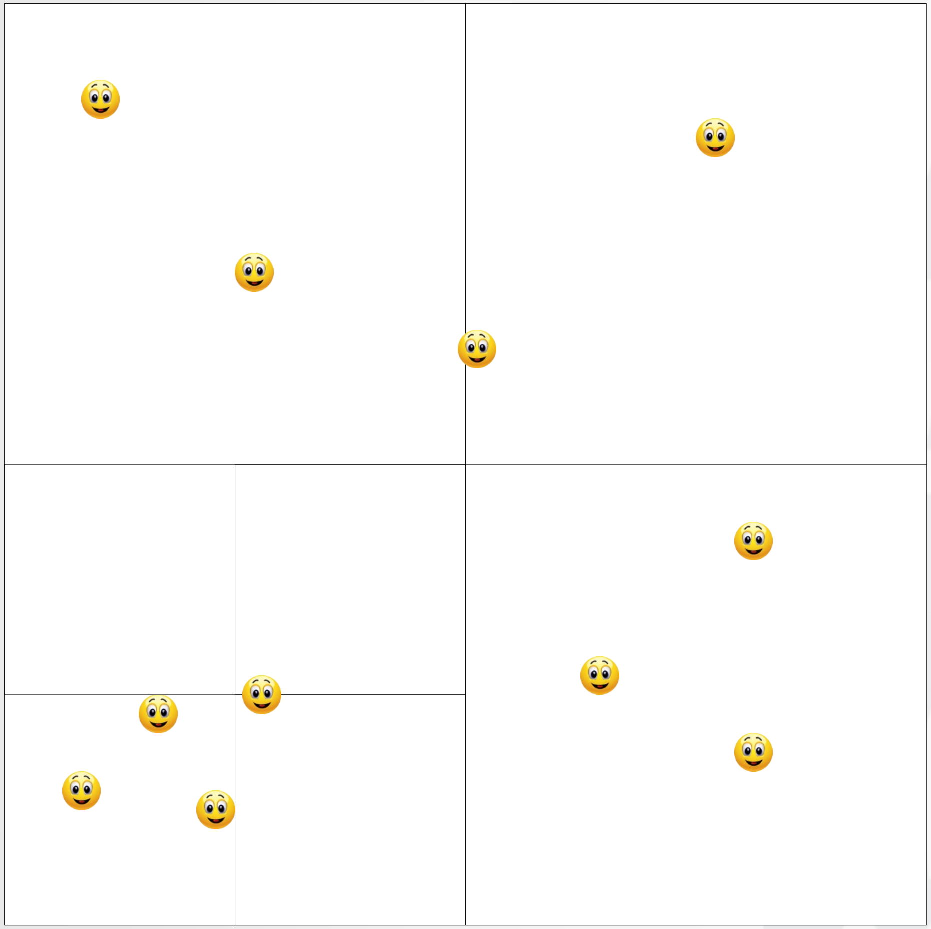


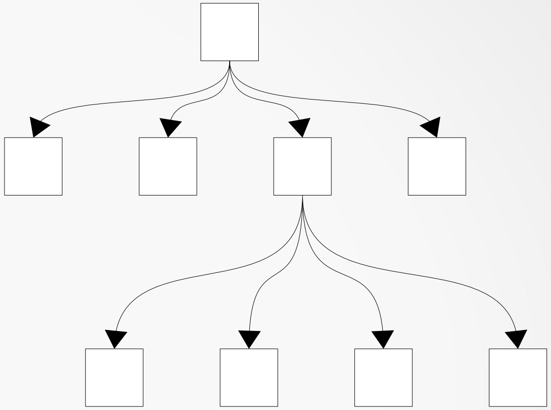
Quad-Tree Construction



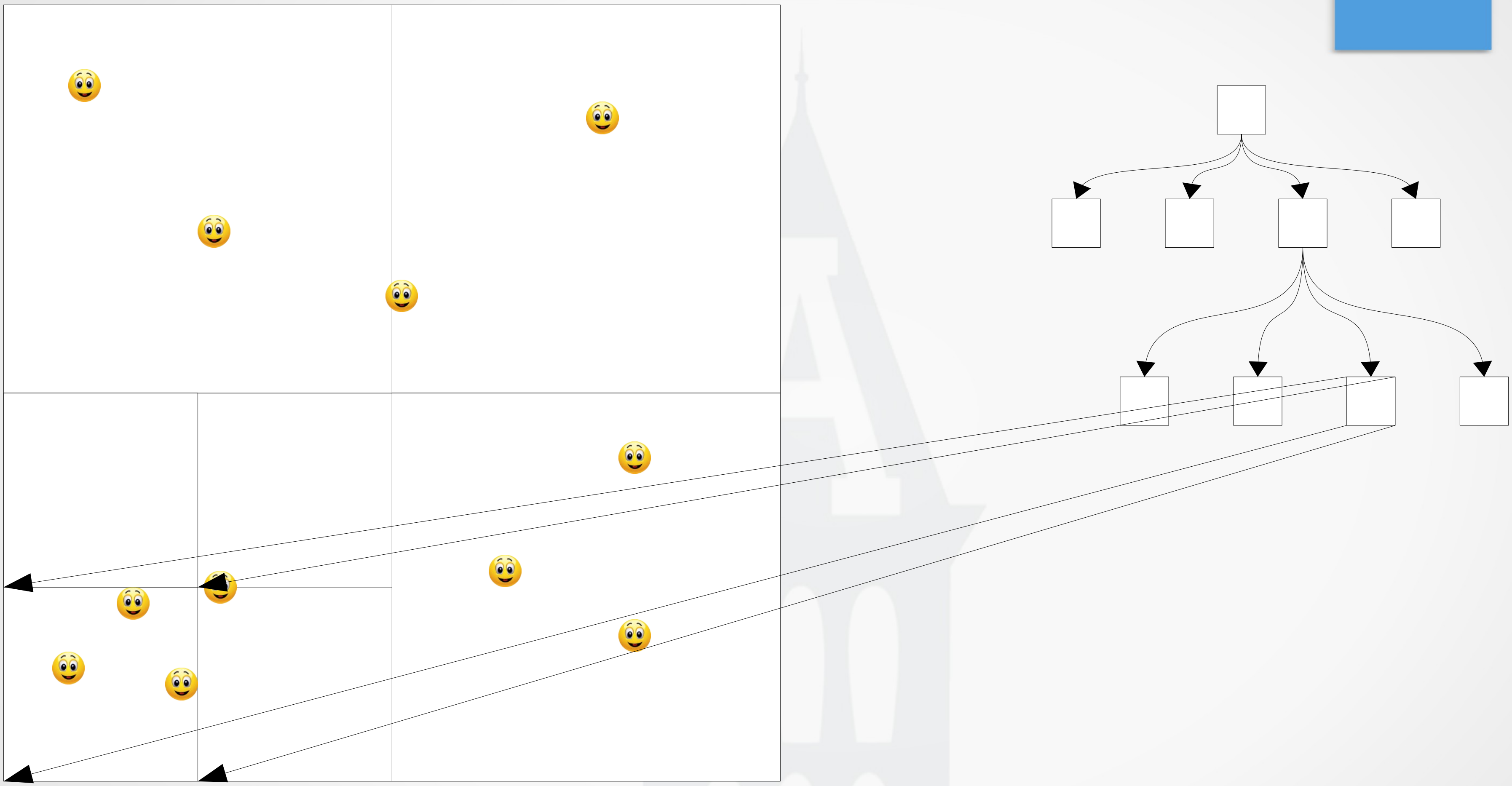
Duplicate membership

Quad-Tree Construction

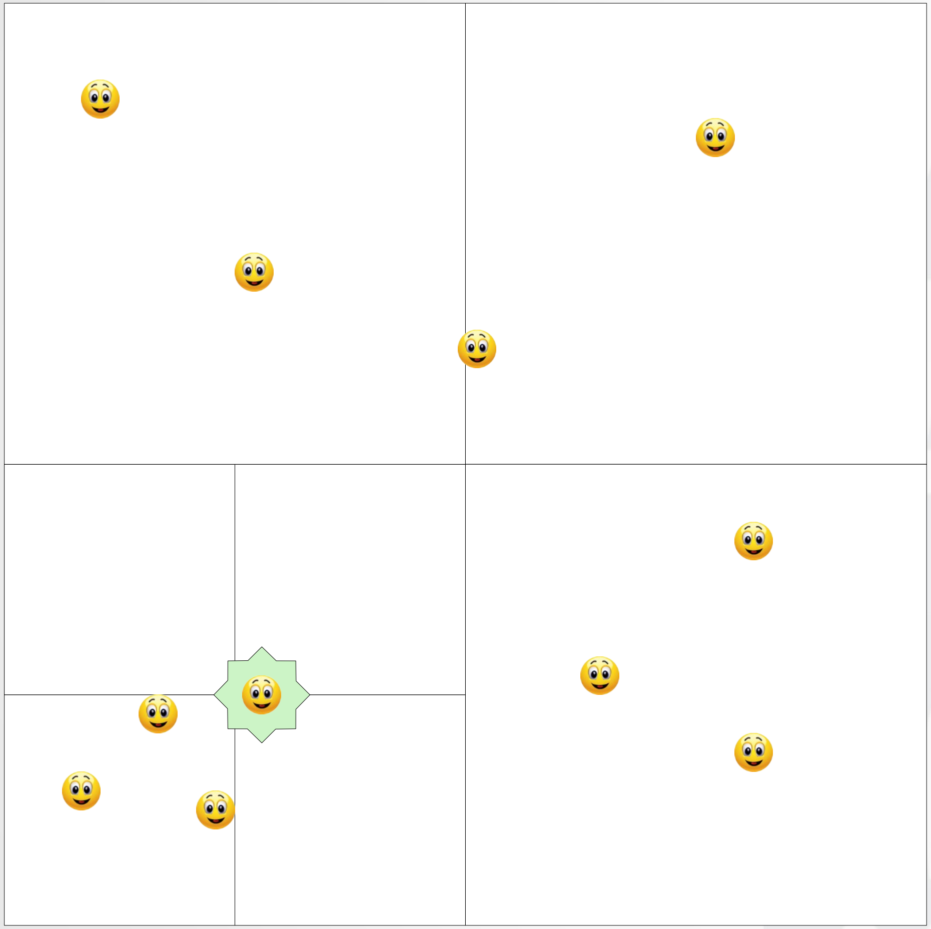


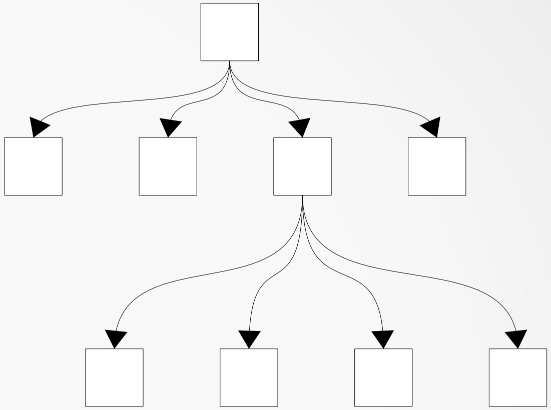


Quad-Tree Construction

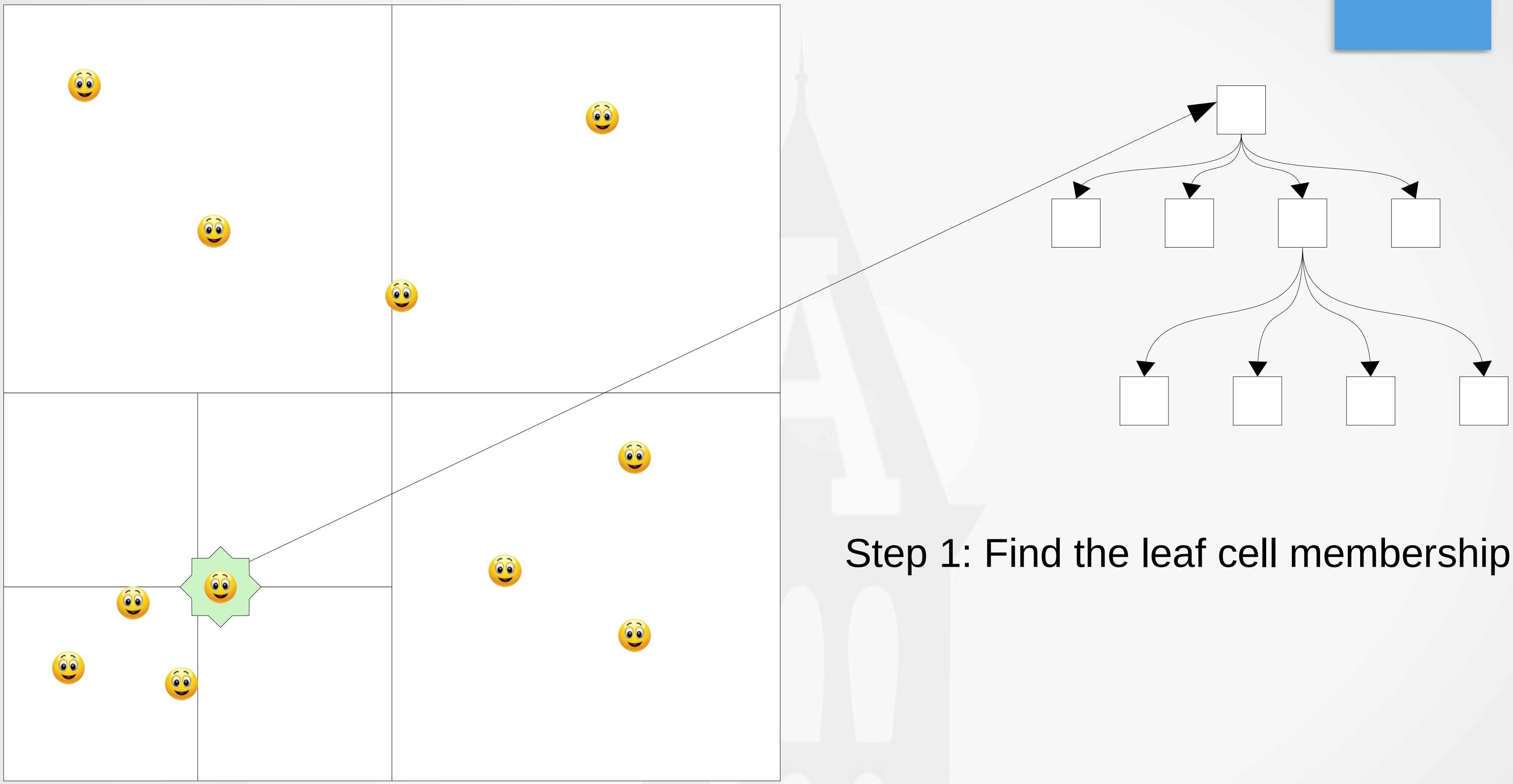


Quad-Tree Use



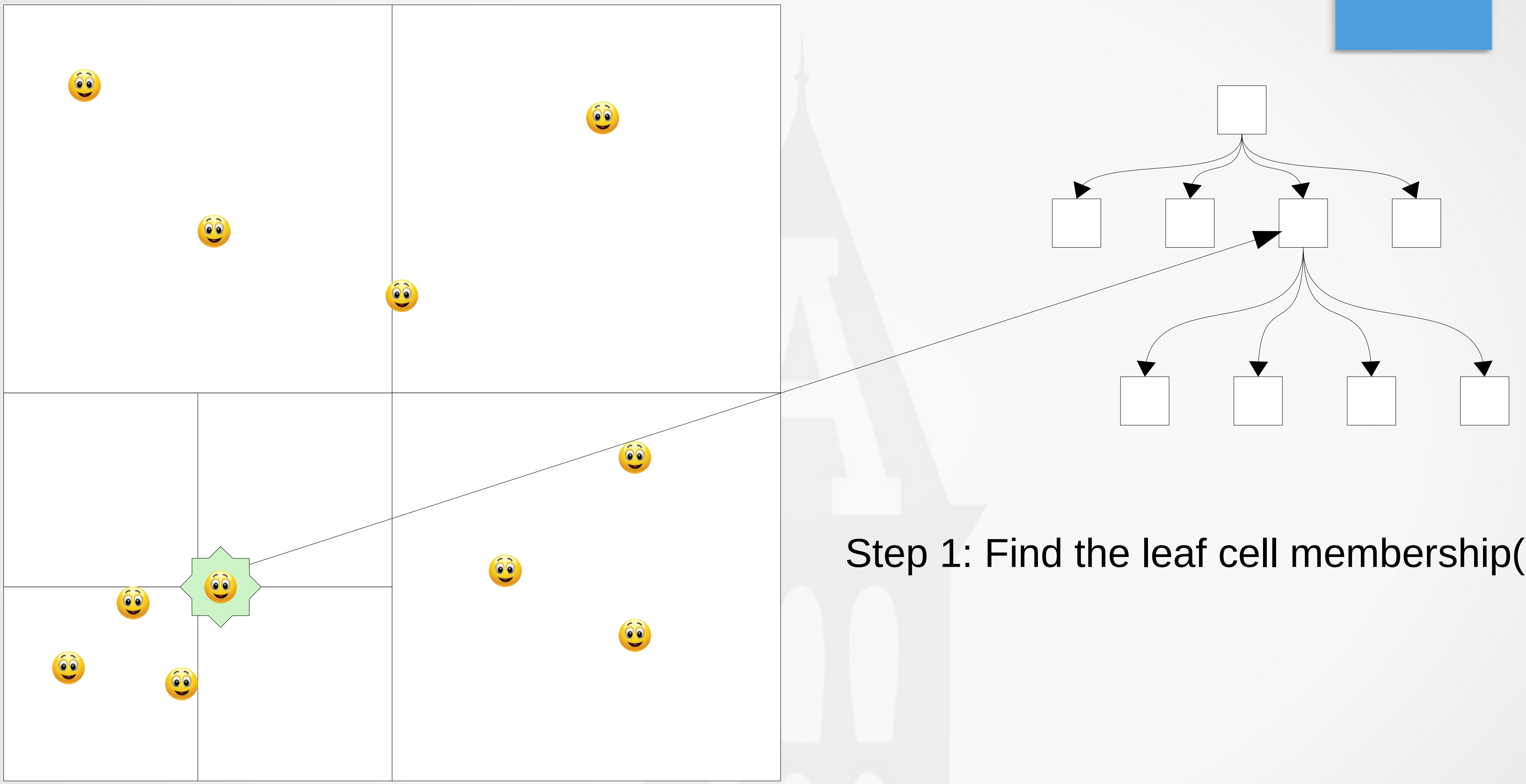


Quad-Tree Use



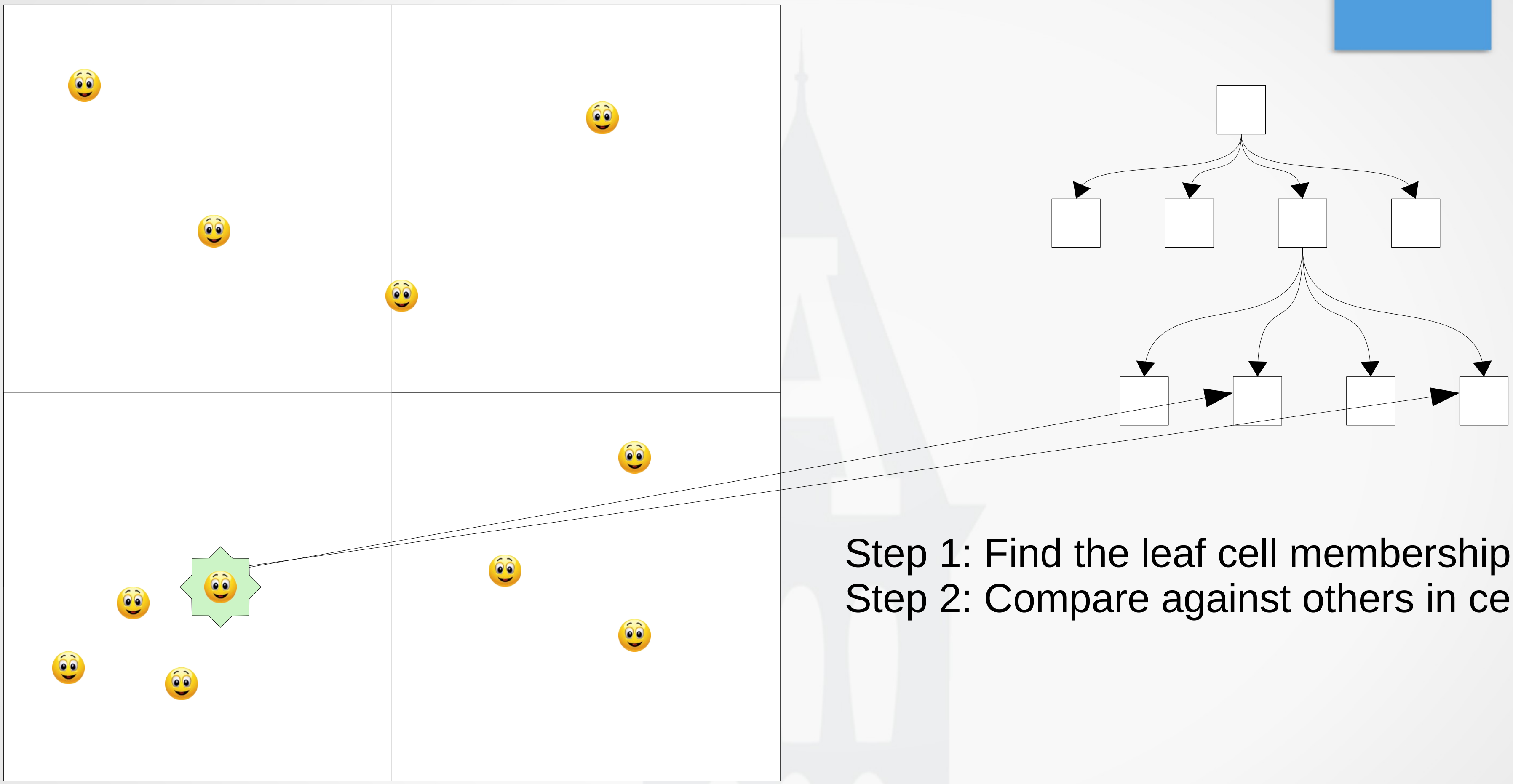
Step 1: Find the leaf cell membership(s)

Quad-Tree Use



Step 1: Find the leaf cell membership(s)

Quad-Tree Use



Step 1: Find the leaf cell membership(s)
Step 2: Compare against others in cell(s)

Quad-Tree Demo