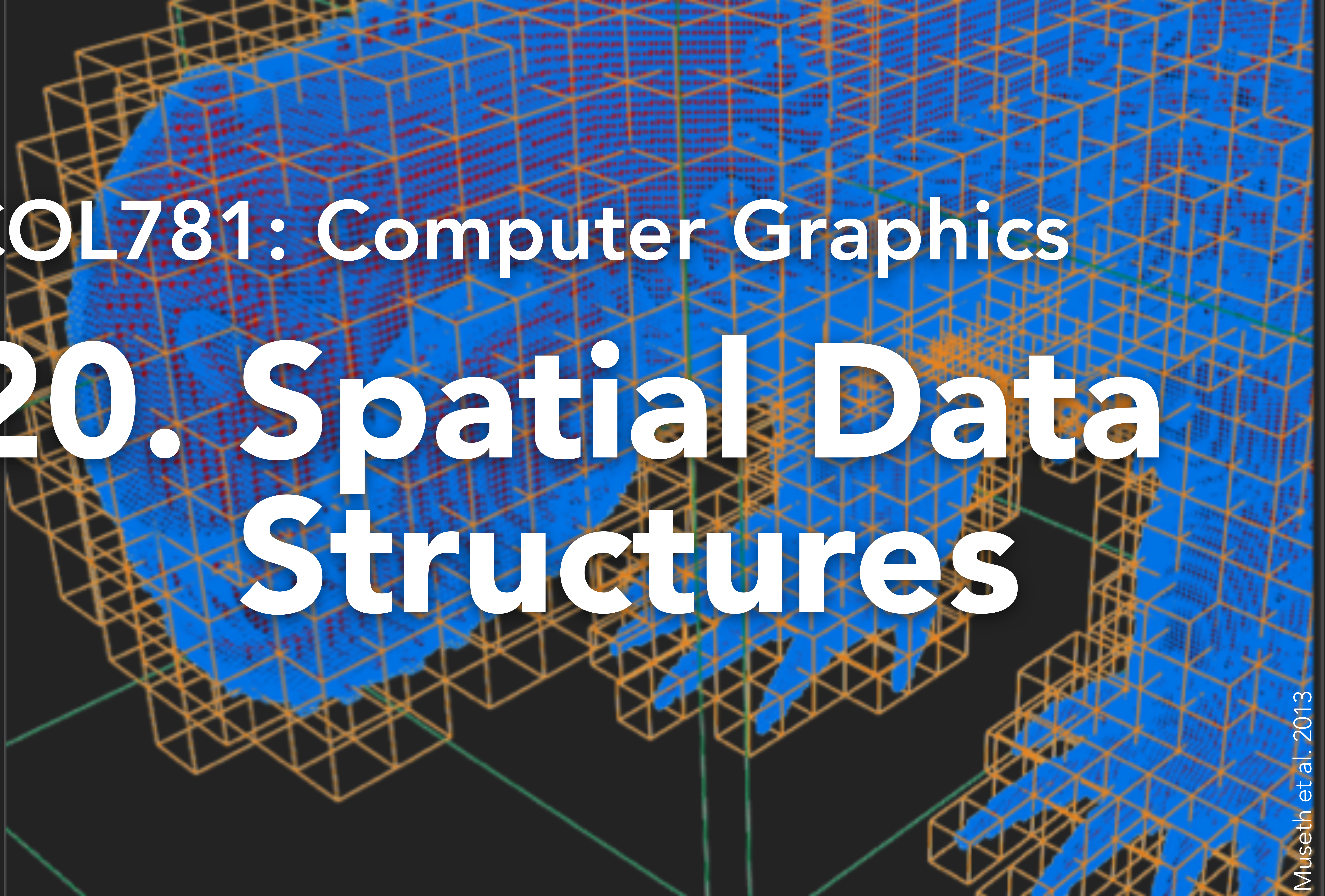
COL781: Computer Graphics

20. Spatial Data Structures



Museth et al. 2013

We want to render scenes containing millions of triangles.

- Rasterization cost = *O*(total #pixels covered by triangles)
- Ray tracing cost = *O*(#pixels × #triangles)?

We can do better!

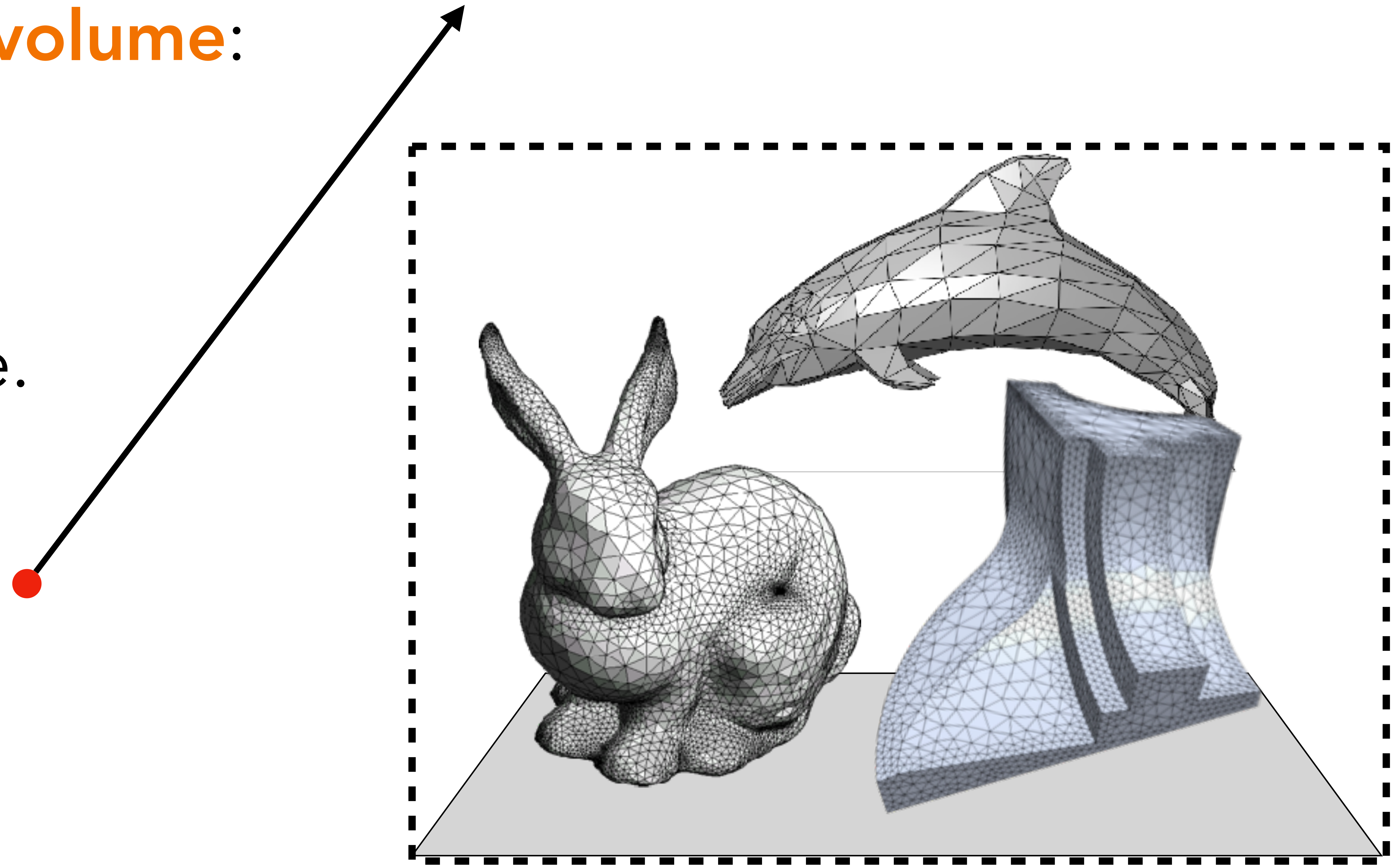
San Miguel scene, 10.7M triangles

Bounding volumes

How can we speed up ray intersection with a large, complex scene?

Construct a conservative **bounding volume**:
all scene geometry lies inside it

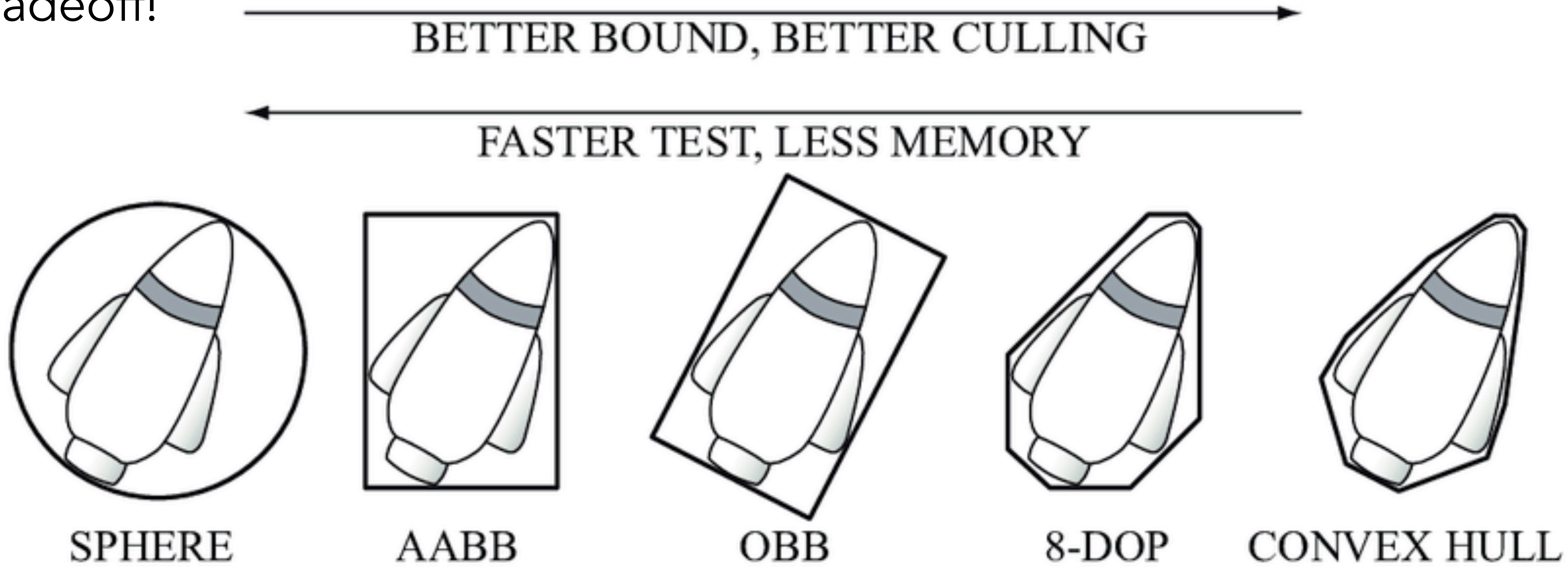
Super easy to reject rays that don't come close to intersecting the scene.



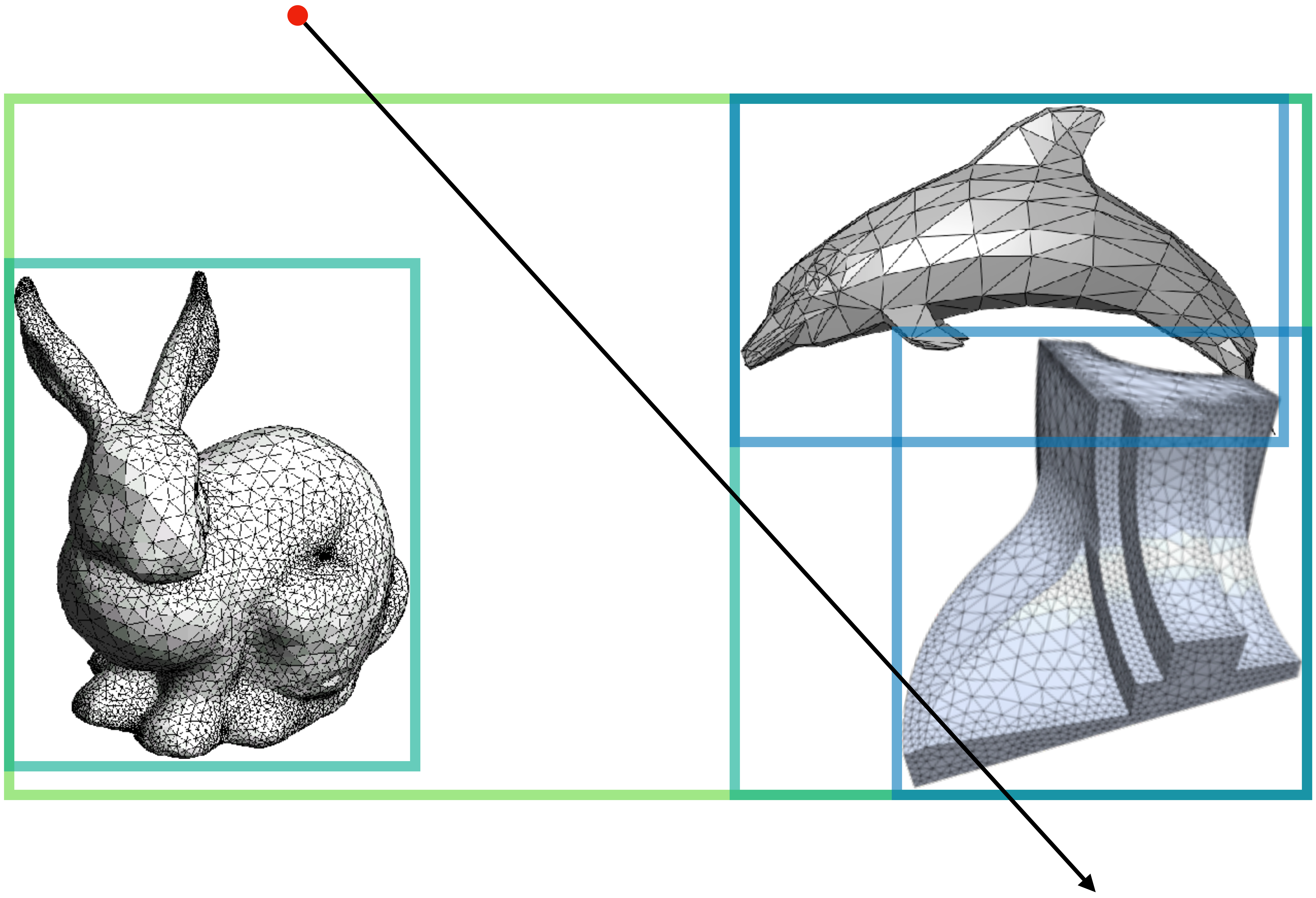
What do we want from a bounding volume?

- Tight (minimize # of false positives)
- Fast to intersect

This is a tradeoff!



Ericson, *Real-Time Collision Detection*

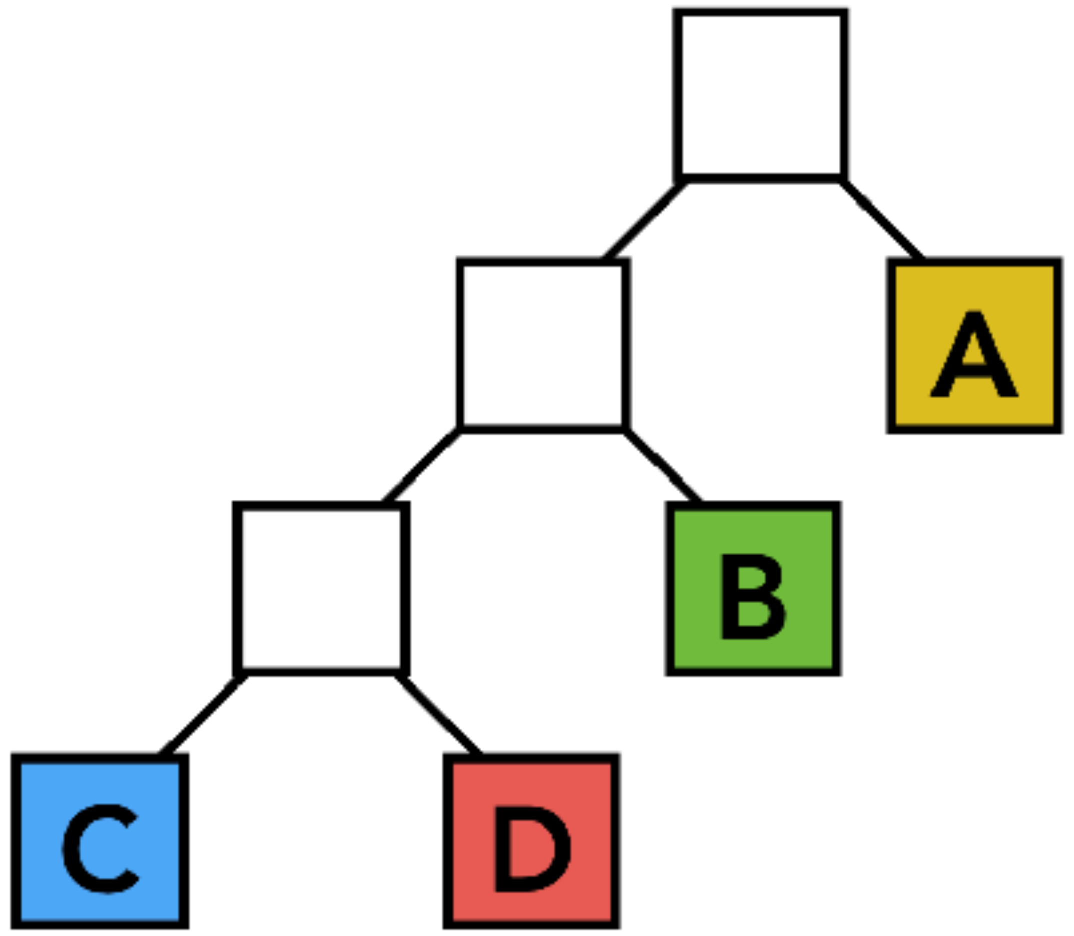


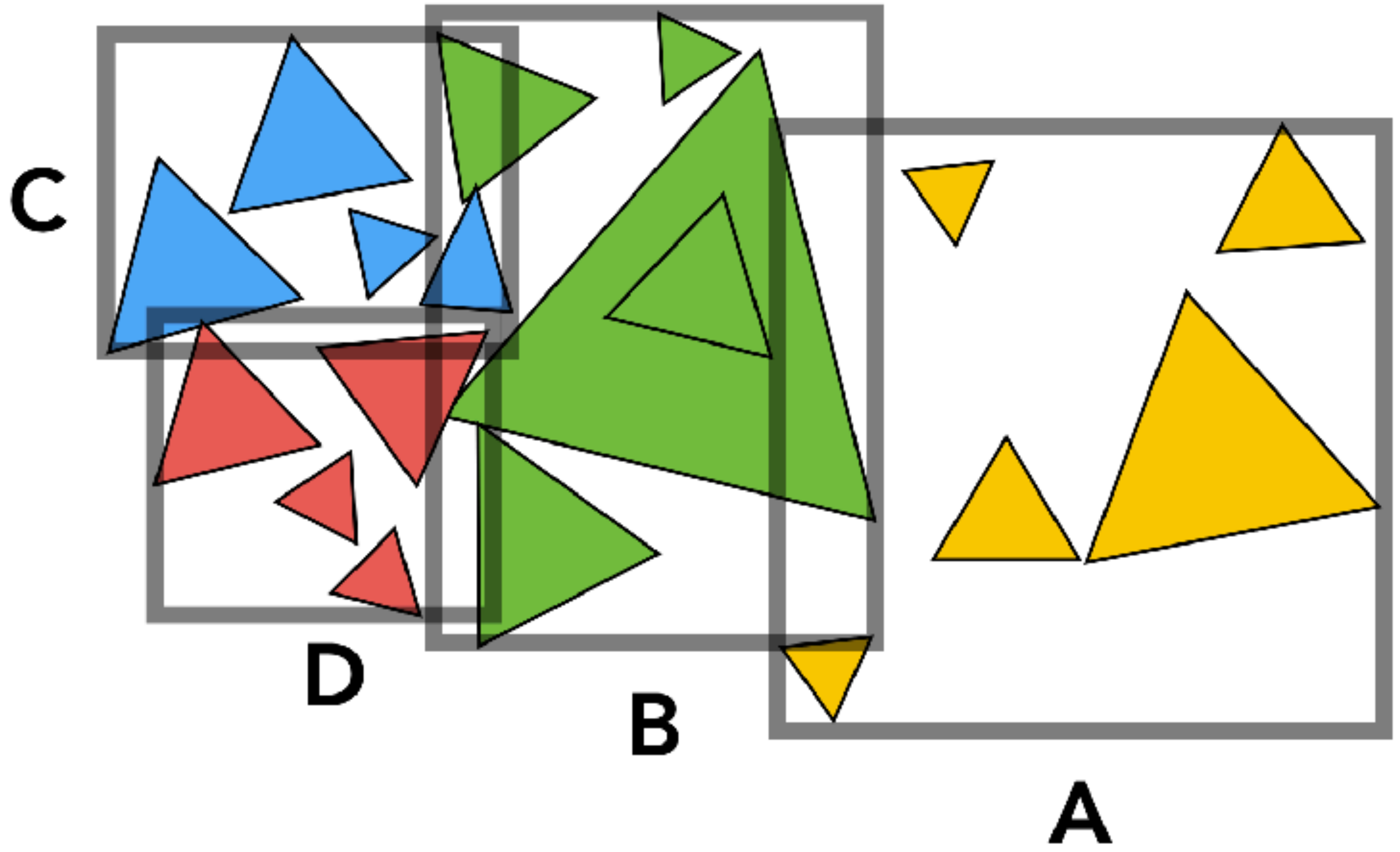
Bounding volume hierarchy (BVH)

Leaf nodes store a small set of objects and their bounding volume.

Internal nodes store the bounding volume of the union of their children.

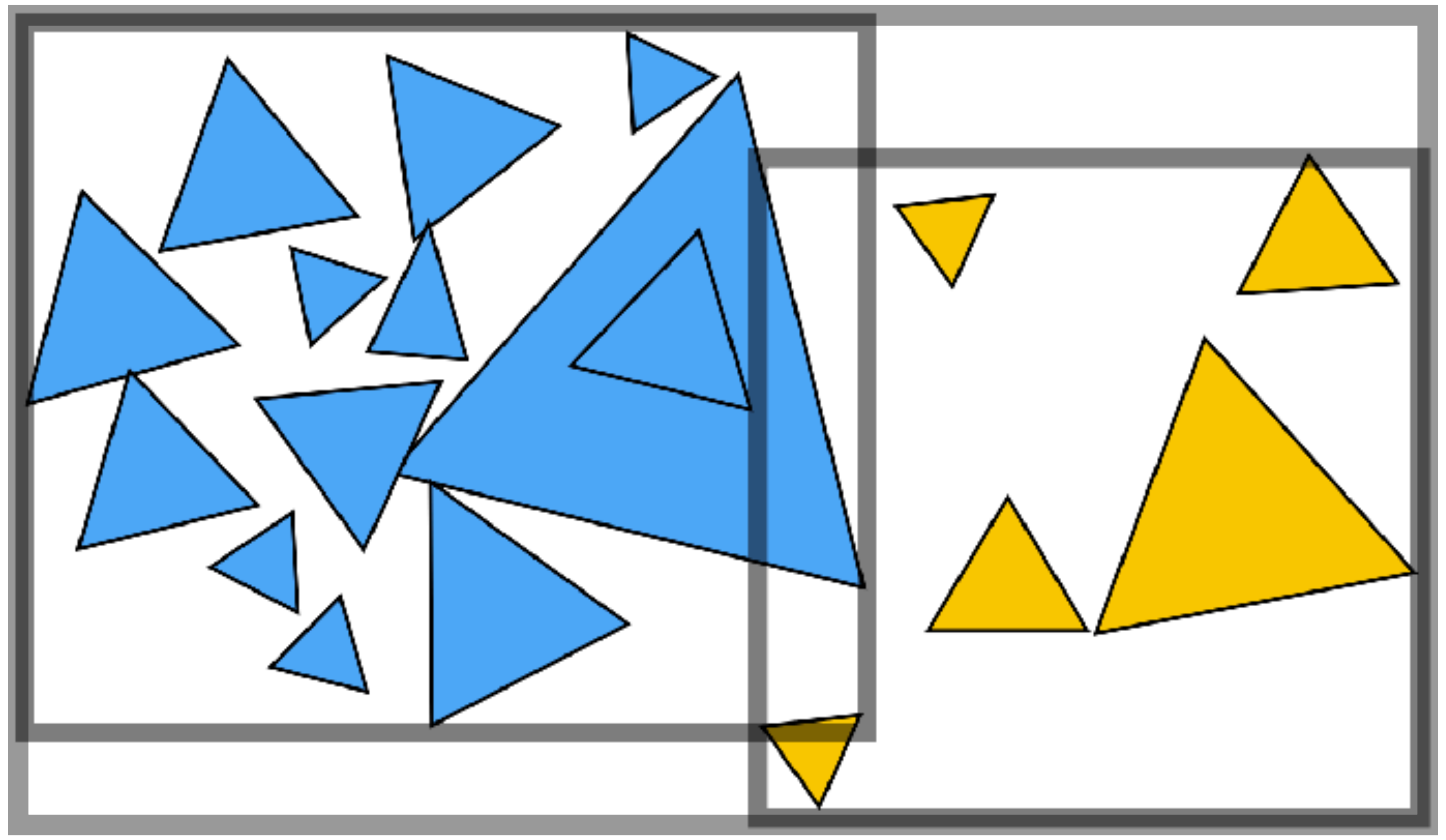
Note: Bounding volumes of siblings (or other unrelated nodes) can overlap!





BVH traversal

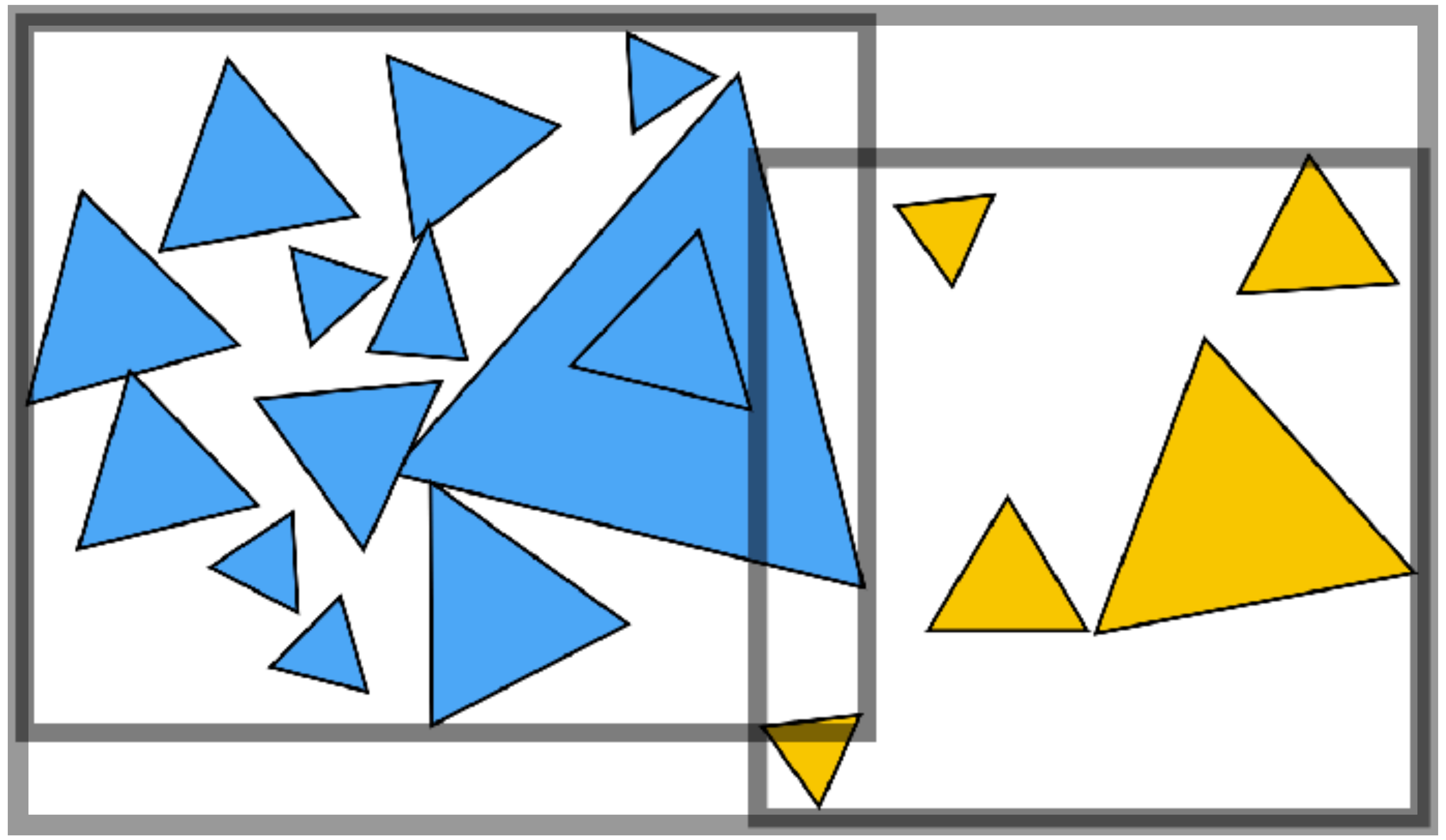
```
test ray against bounding volume
if hit:
    if leaf:
        intersect ray with objects
        return earliest hit
    else:
        intersect ray with child 1
        intersect ray with child 2
        return earliest hit
```



...

    ...

        intersect ray with child 1
        intersect ray with child 2
        return earliest hit



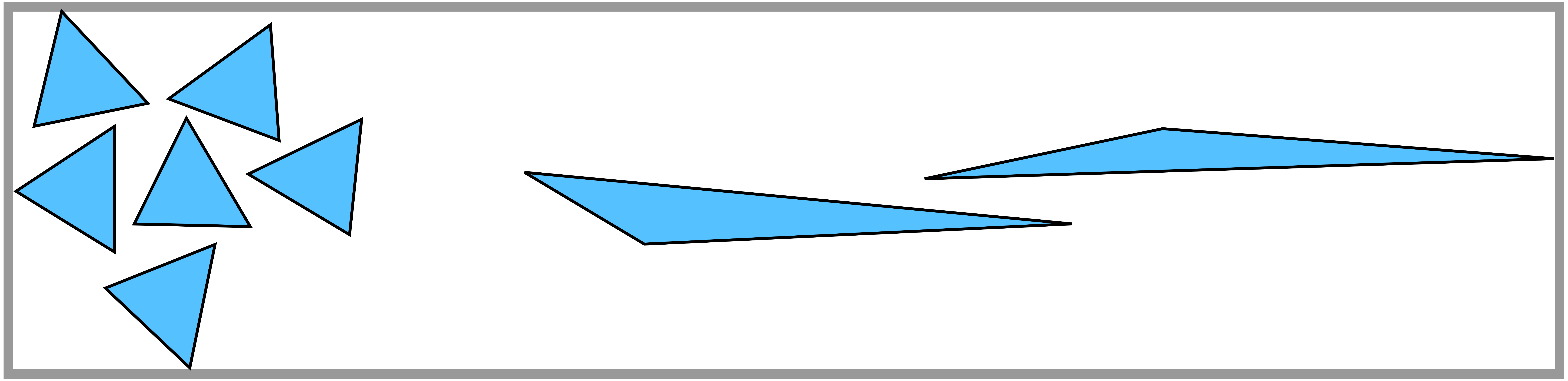
Smarter: find which child's BV is hit earlier, intersect ray with it first

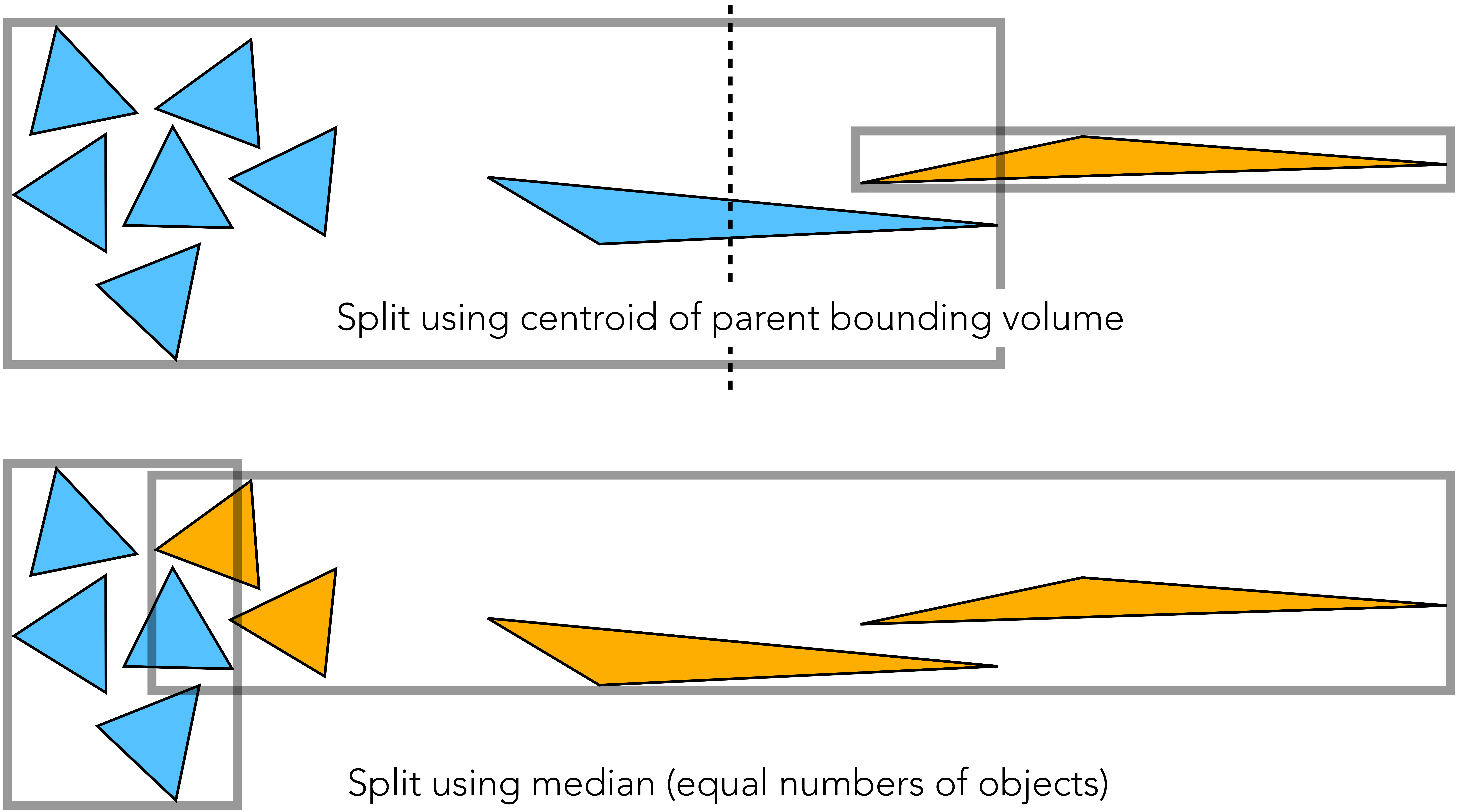
- If no object hit: intersect with other child
- If hit: can you skip recursing down the other child?
    - Only if the hit occurs before reaching the other child's BV!

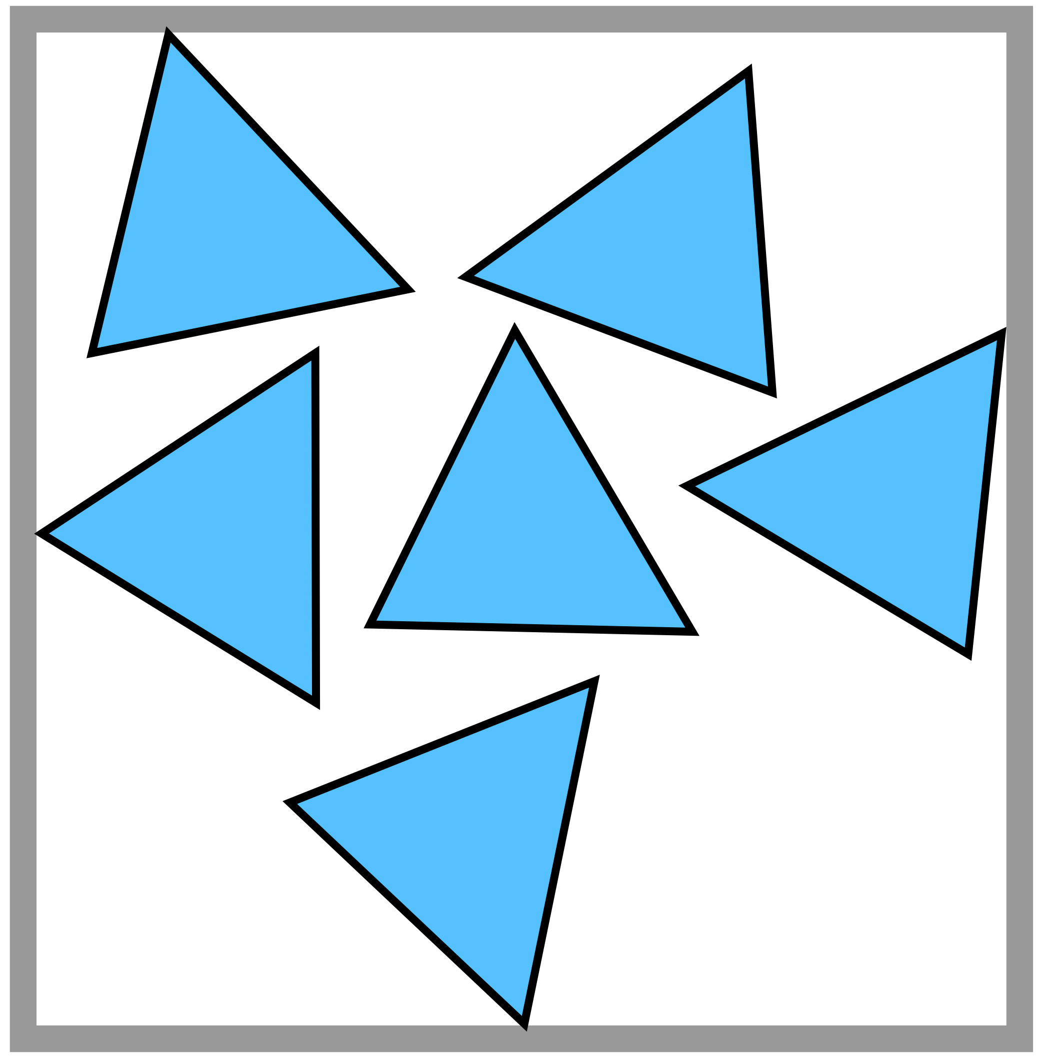
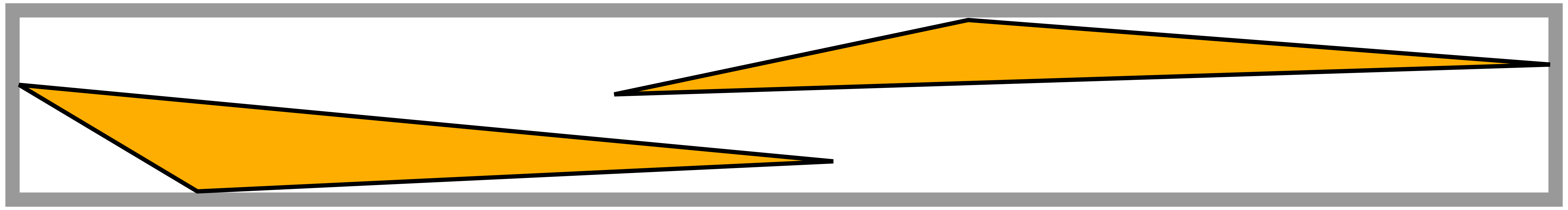
BVH construction

- Set root node = BV of all objects in scene
- Recursively create child nodes by splitting objects into two subsets

What’s a good way to split? And when should we stop?




Split using centroid of parent bounding volume
Split using median (equal numbers of objects)

Intuitively, this is the ideal partition:

- Minimal overlap between children
- Minimal empty space in bounding volumes

How to formalize this?

What we really want is to minimize the cost of intersecting a ray with the BVH.

- Cost of leaf = $N\ C_{\text{isect}}$
  N = number of objects
  C_{isect} = cost of intersecting an object

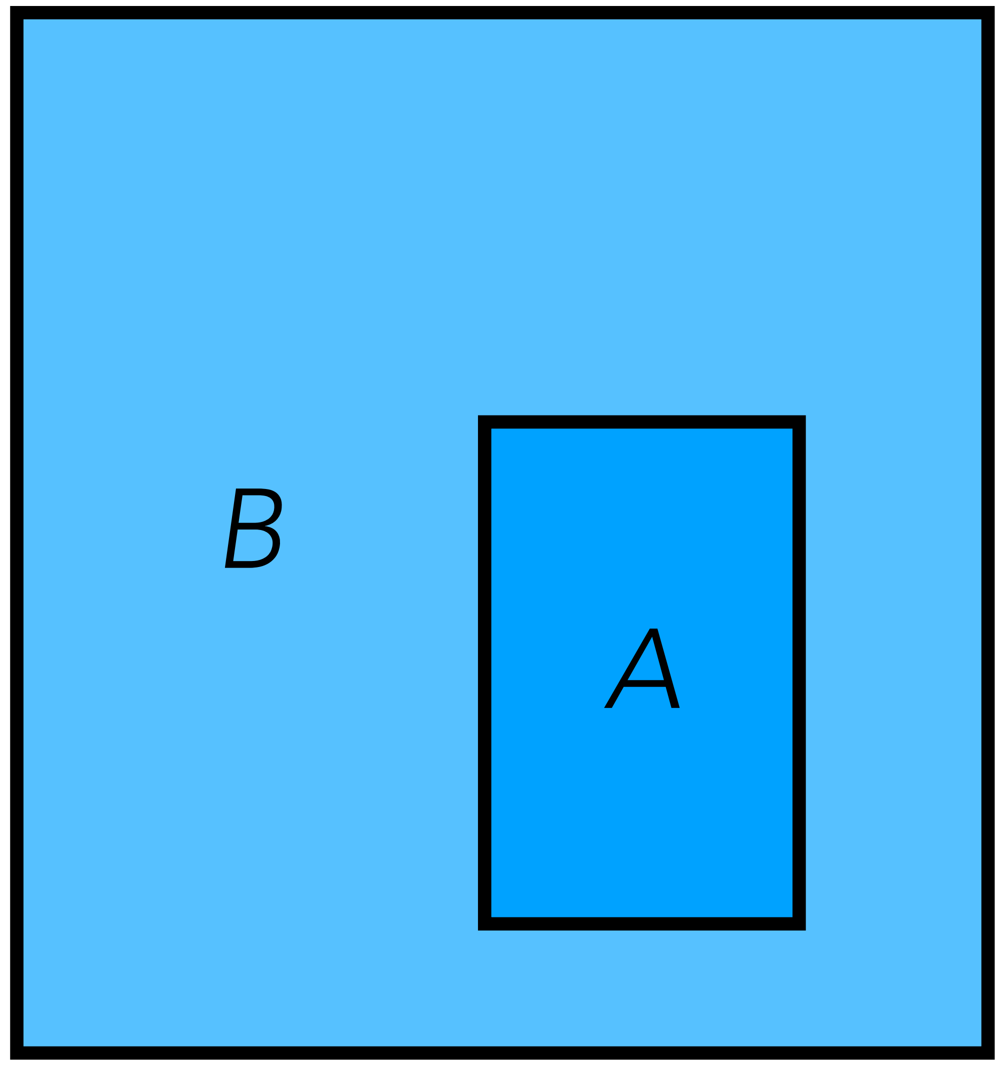
- Cost of internal node = $C_{\text{trav}} + p_L C_L + p_R C_R$
  C_{trav} = cost of traversing an internal node (e.g. ray-BV intersection)
  p_L, p_R = probability of hitting child BVs
  C_L, C_R = cost of intersecting children

Assume $C_L, C_R \approx N_L, N_R$: number of objects in subtree.

How to estimate p_L, p_R?

Surface area heuristic

Fact: For two convex shapes $A \subseteq B$, the probability that a random ray which hits B also hits A is equal to the ratio of their surface areas.



$$p(\text{hit } A \mid \text{hit } B) = \frac{S_A}{S_B}$$

So, cost of internal node $\approx C_{\text{trav}} + \frac{S_L}{S_P} N_L C_{\text{isect}} + \frac{S_R}{S_P} N_R C_{\text{isect}}$

To split a node:

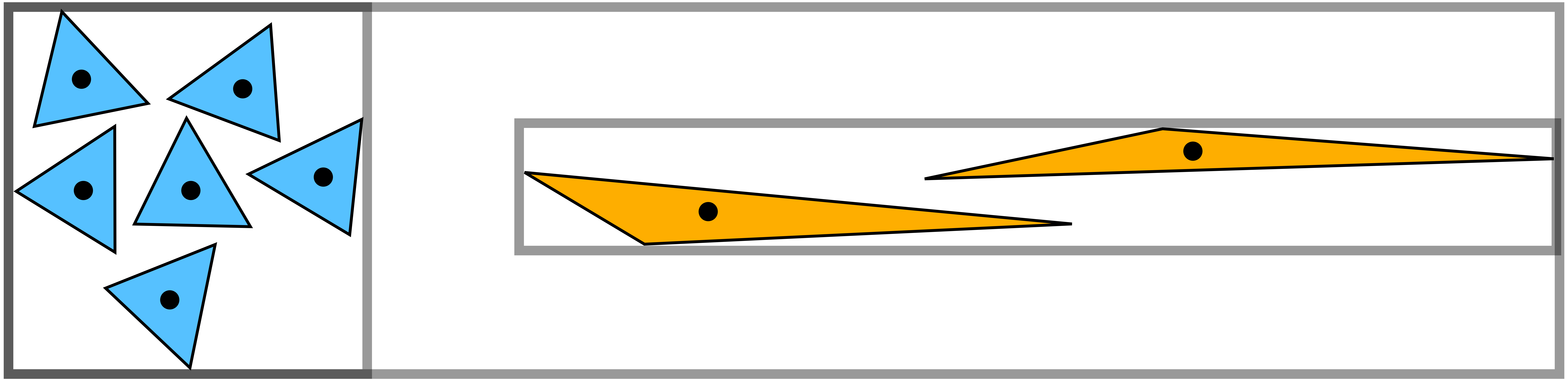
For each axis *x*, *y*, *z*:

    Sort objects by centroid   (Faster: just collect into *B* buckets)

    For various choices of partition:

        Evaluate SAH cost

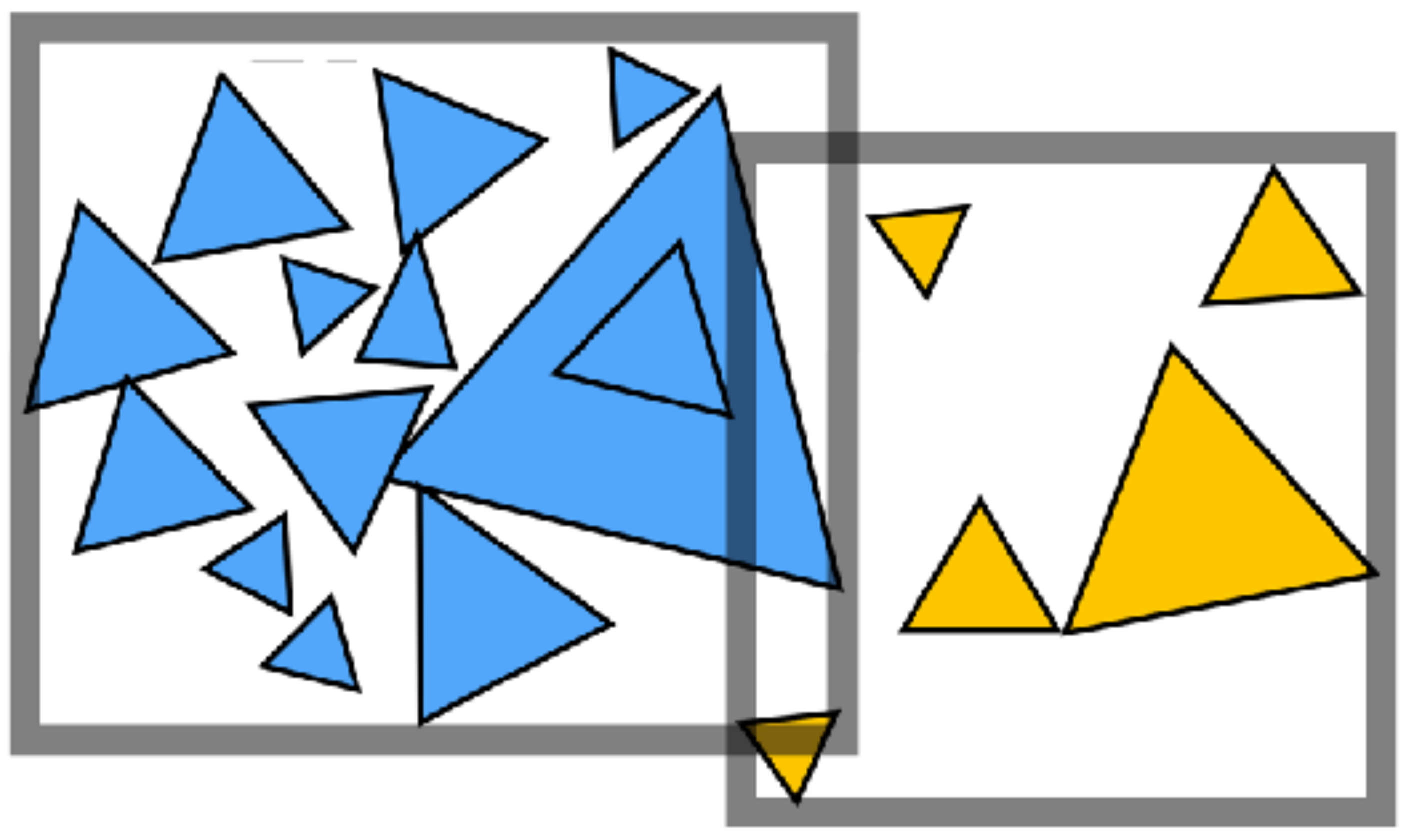
Split using partition with lowest cost



Object partitioning vs. space partitioning

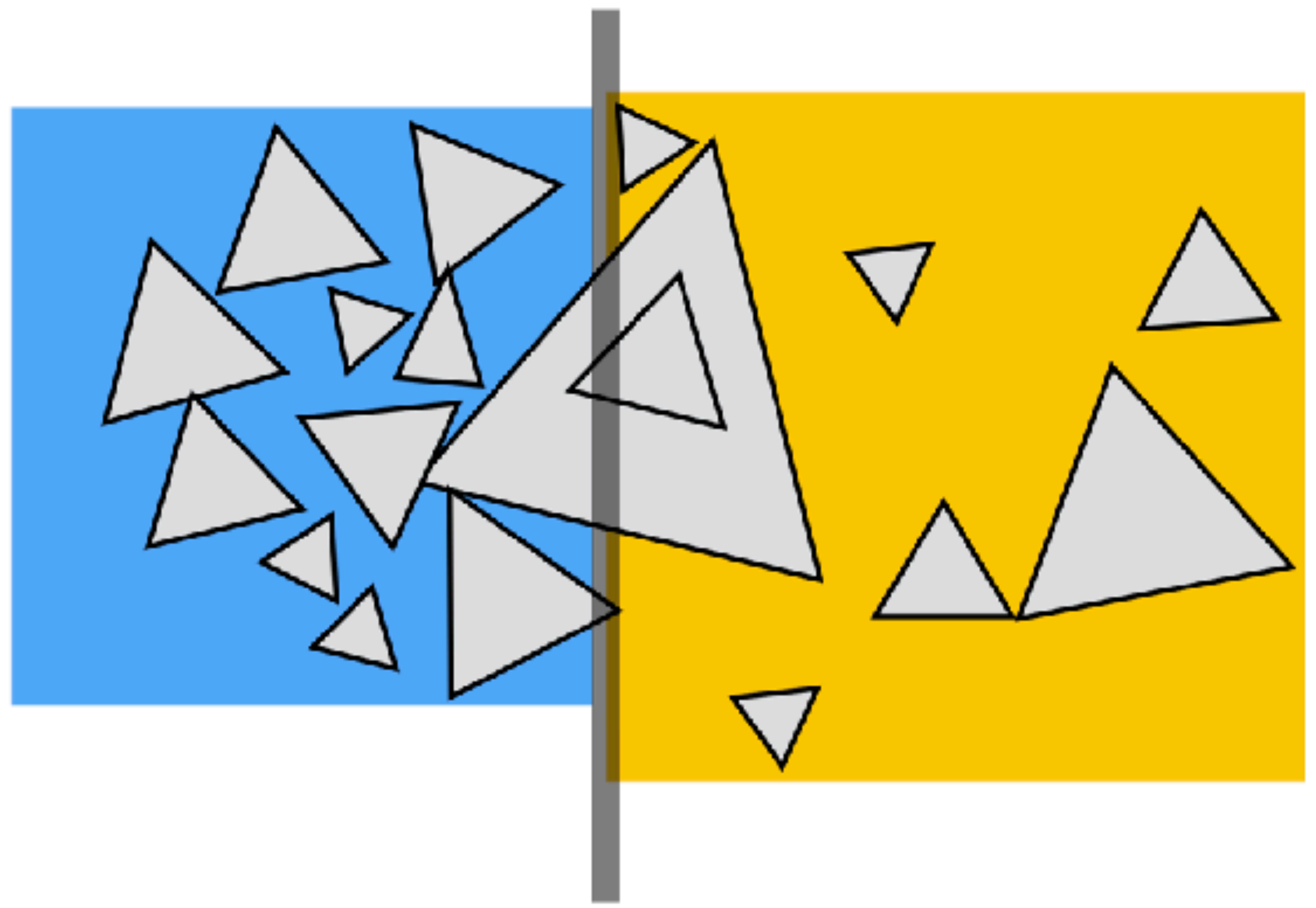
BVH is an **object partitioning** scheme: split objects into disjoint groups

- Bounding volumes may overlap
- Each object lies in one leaf node



Space partitioning: split space into regions (k-d trees, grids, octrees, …)

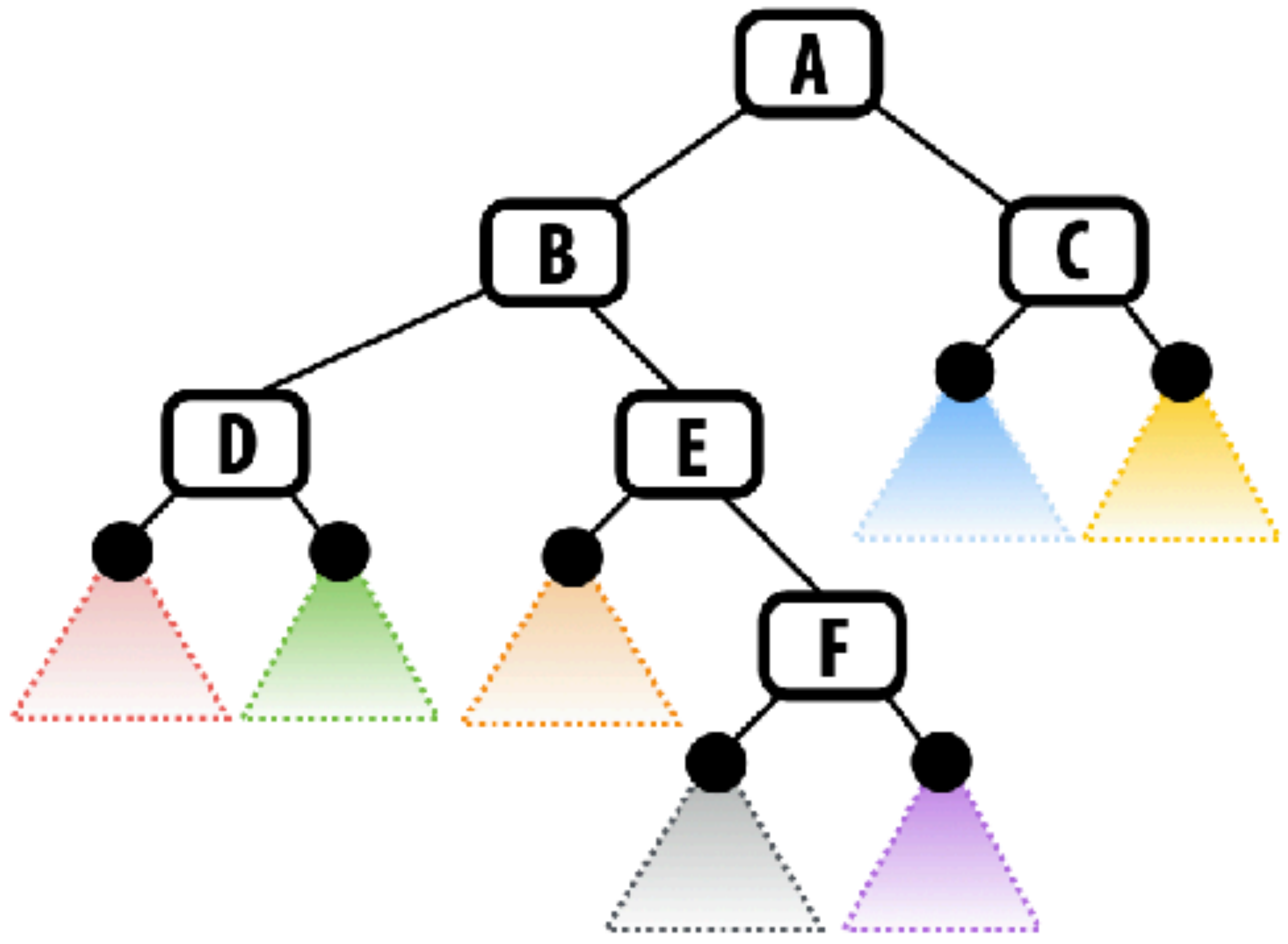
- Regions are non-overlapping
- Objects may lie in multiple regions

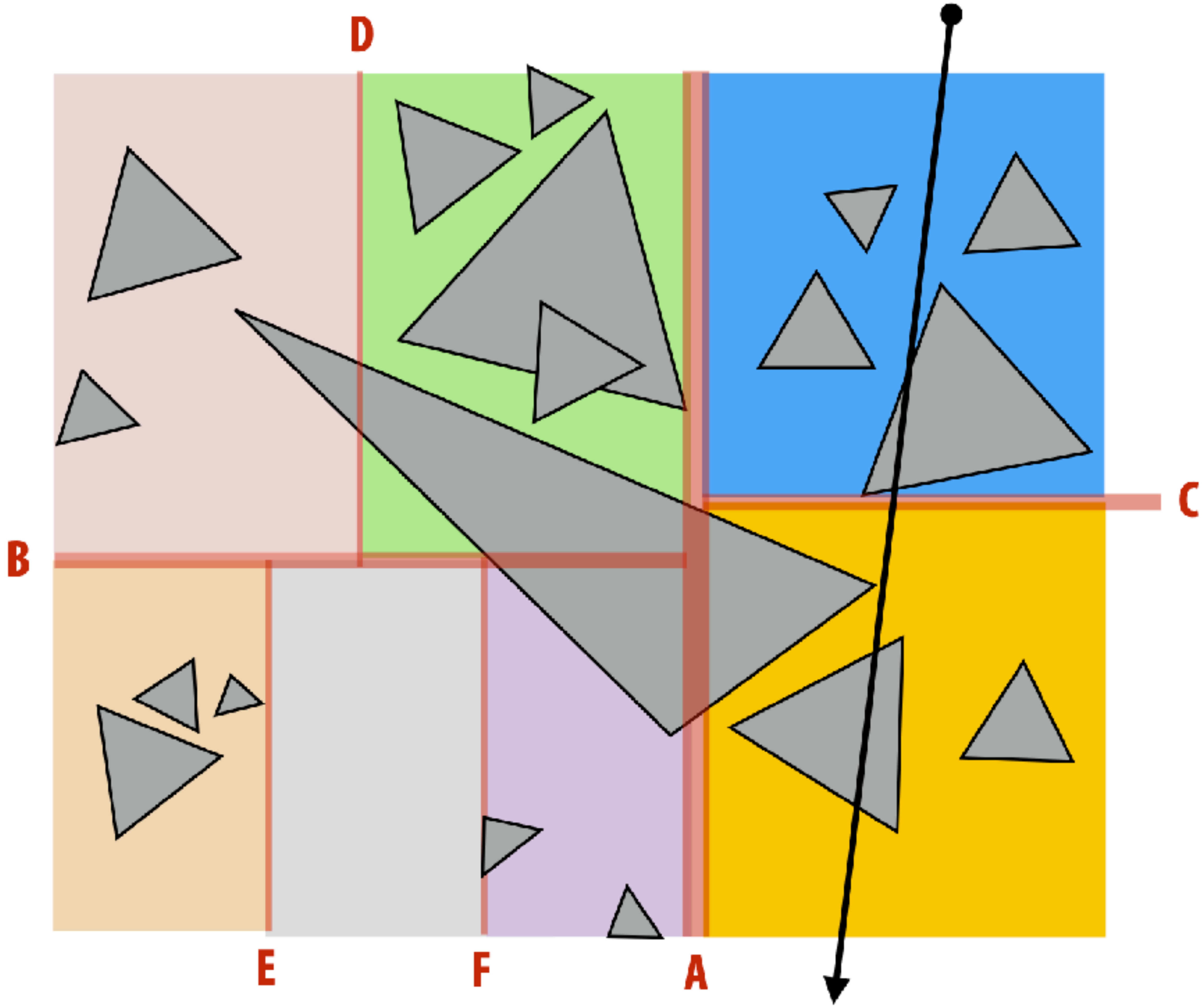


K-d trees

Divide space via axis-aligned split planes

Leaf nodes store list of objects.
Internal nodes store only the split plane.





K-d tree traversal

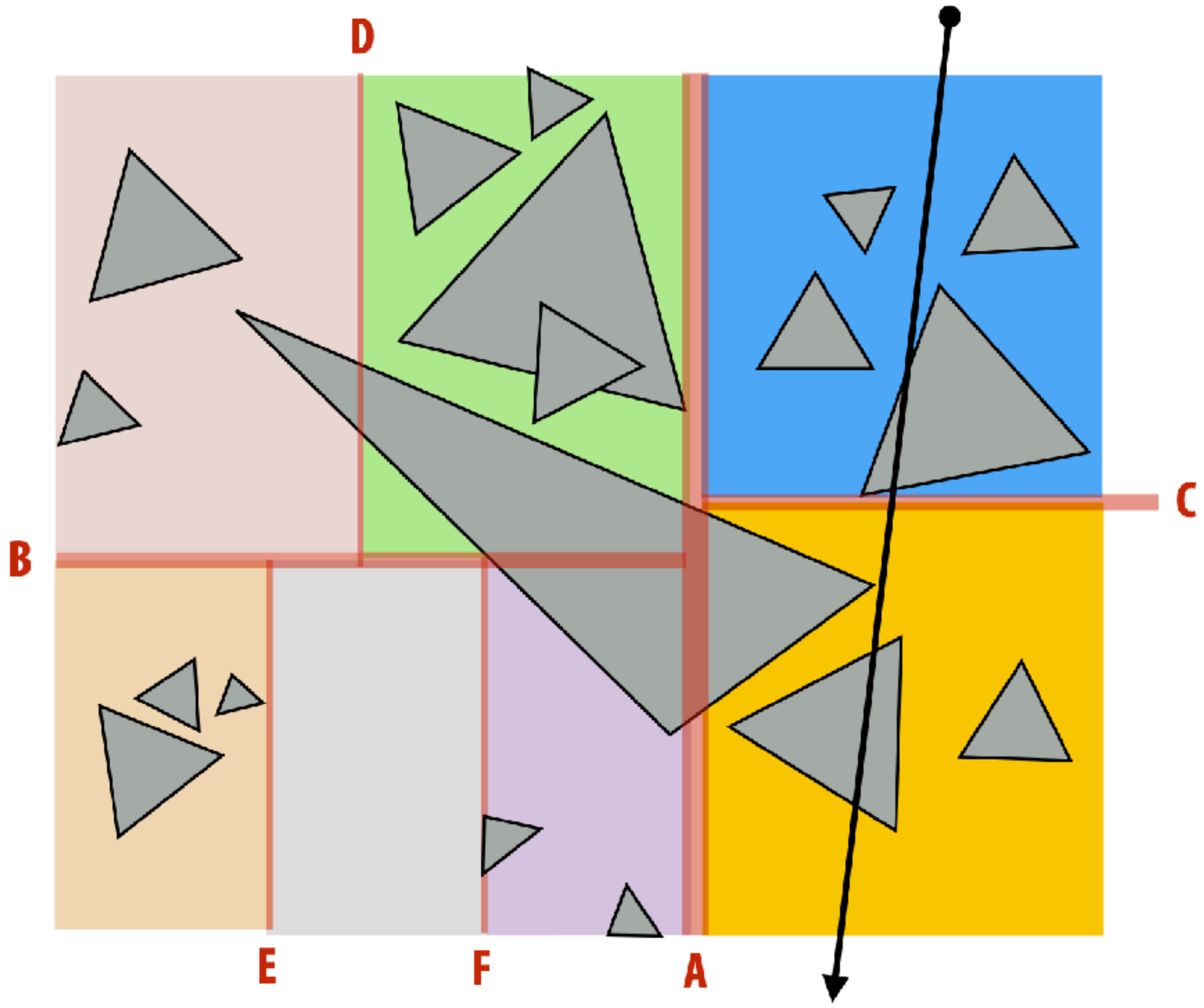
Keep track of interval [t_{min}, t_{max}] covered by current node.

At internal node:

intersect ray with front child in interval [t_{min}, t_{split}]

if not hit:

    intersect ray with back child in interval [t_{split}, t_{max}]



At leaf:

intersect ray with objects

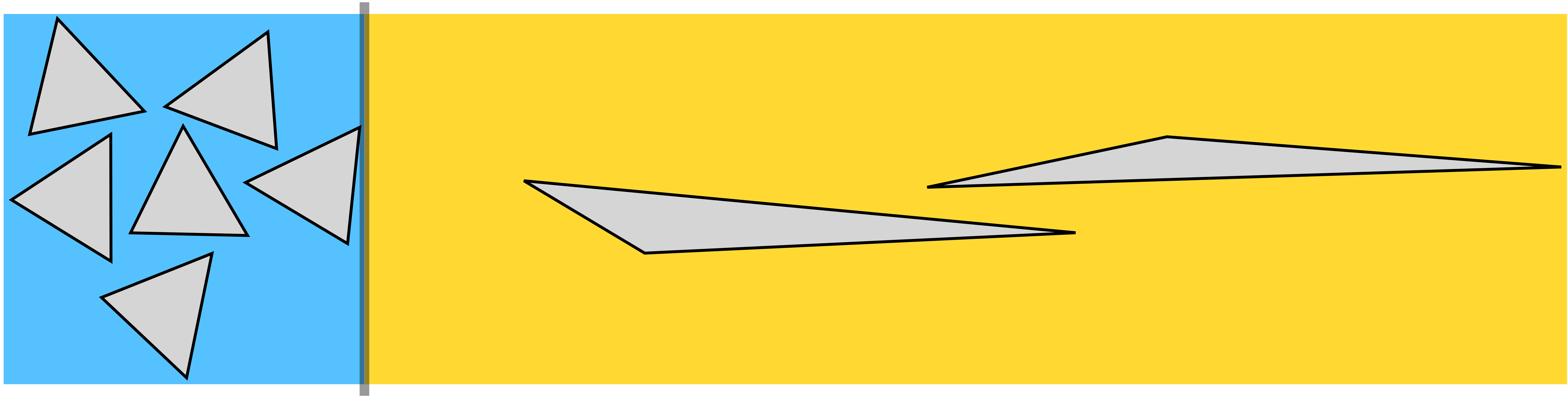
return earliest hit in [t_{min}, t_{max}]   Why?

Unlike BVH, we can always return the hit as soon as we find it!

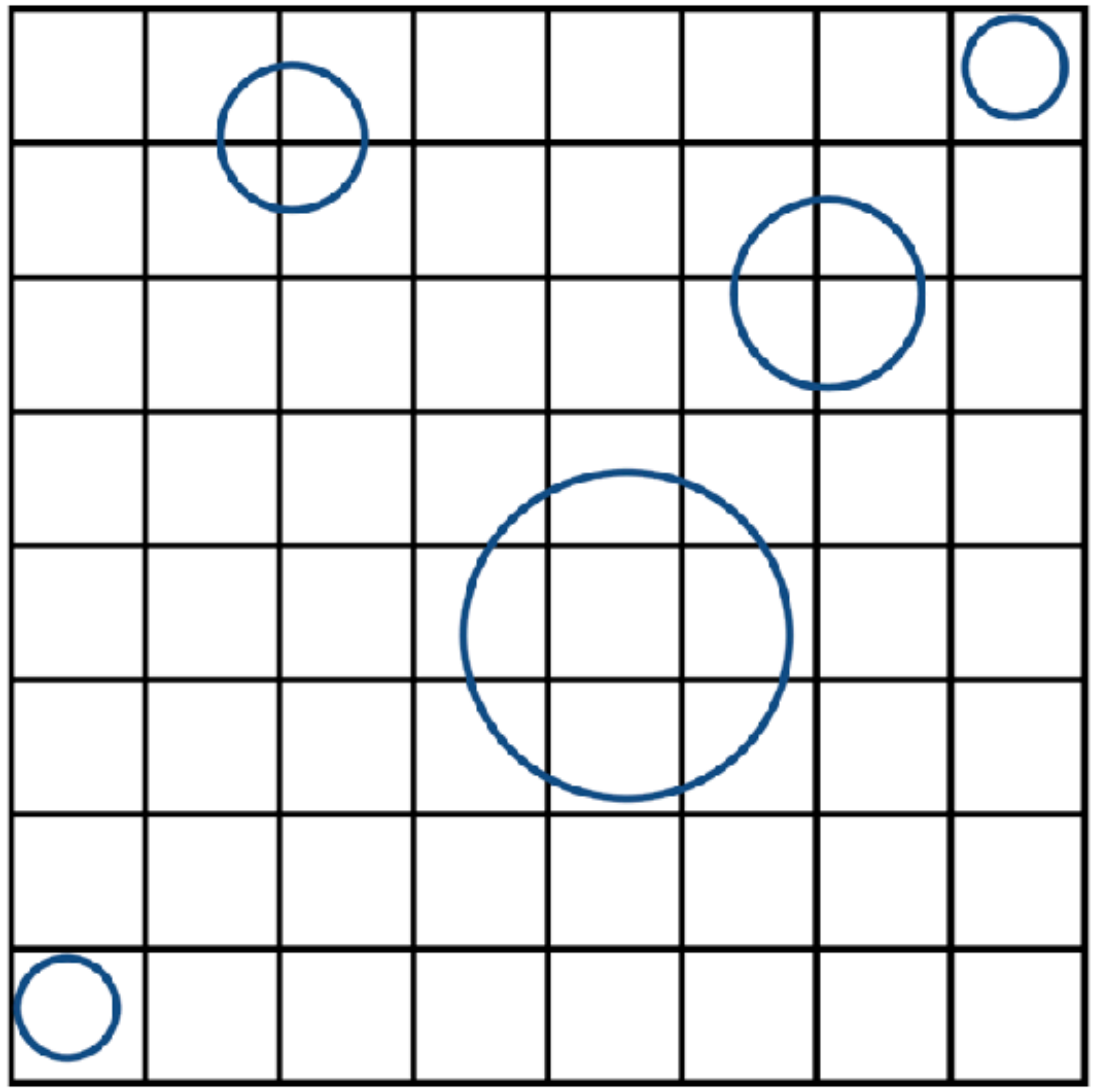
K-d tree construction

- Set root node = bounding box of all objects in scene
- Recursively create child nodes by choosing split planes

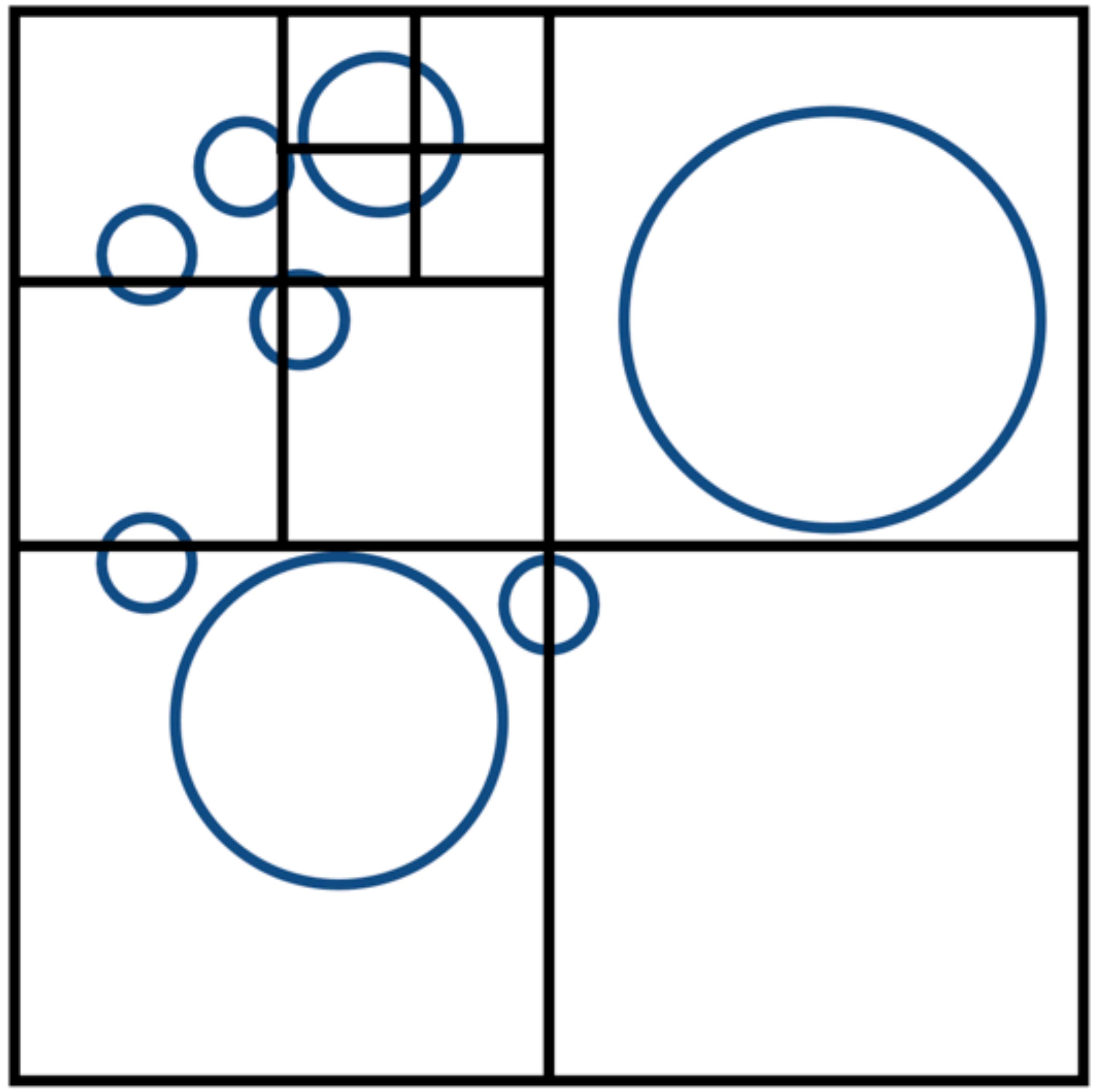
How to choose a split plane? Can reuse same surface area heuristic as in BVH



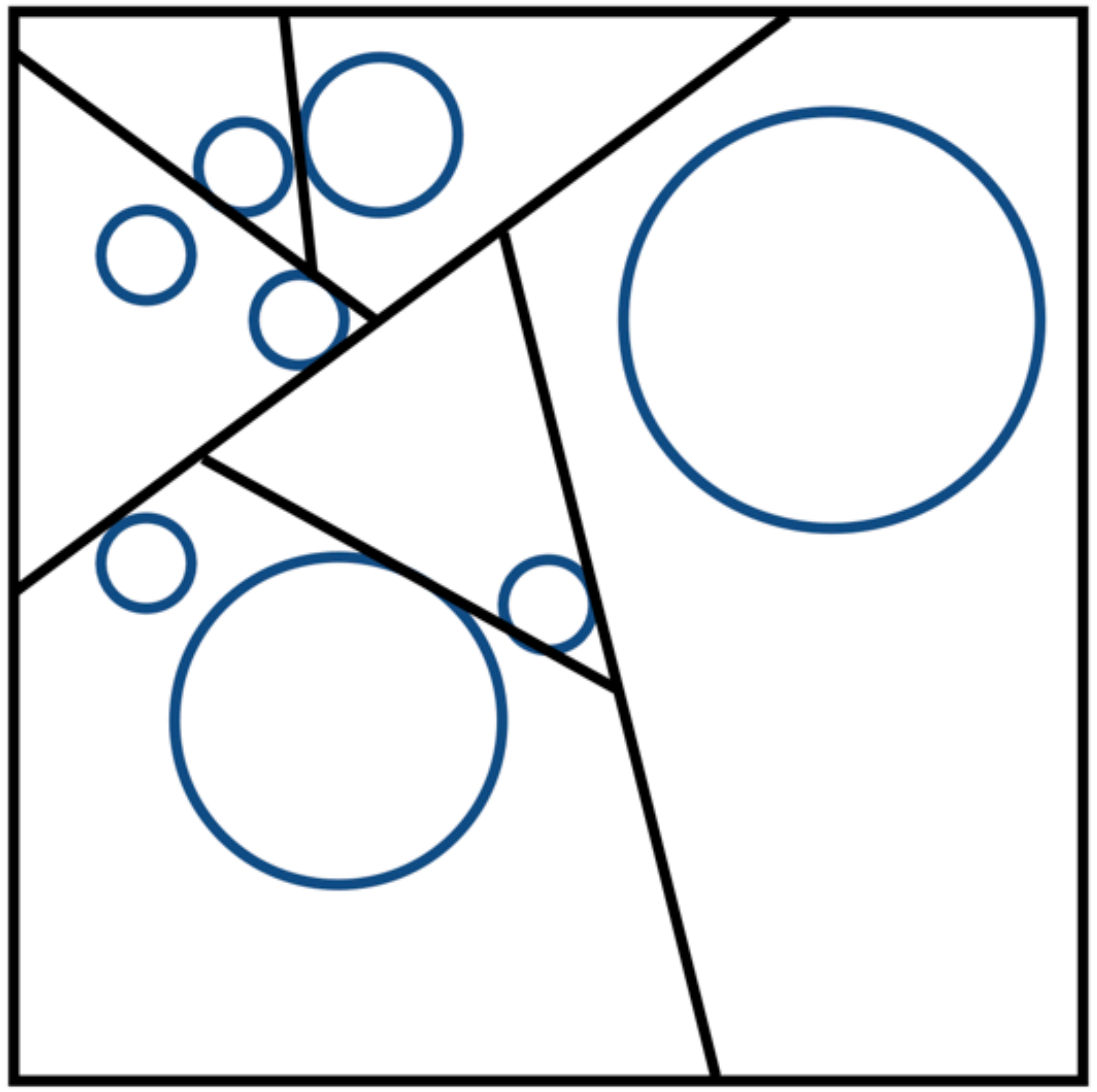
Other space partitioning techniques



Uniform grids



Quadtrees / octrees



Binary space partitioning
(BSP trees)

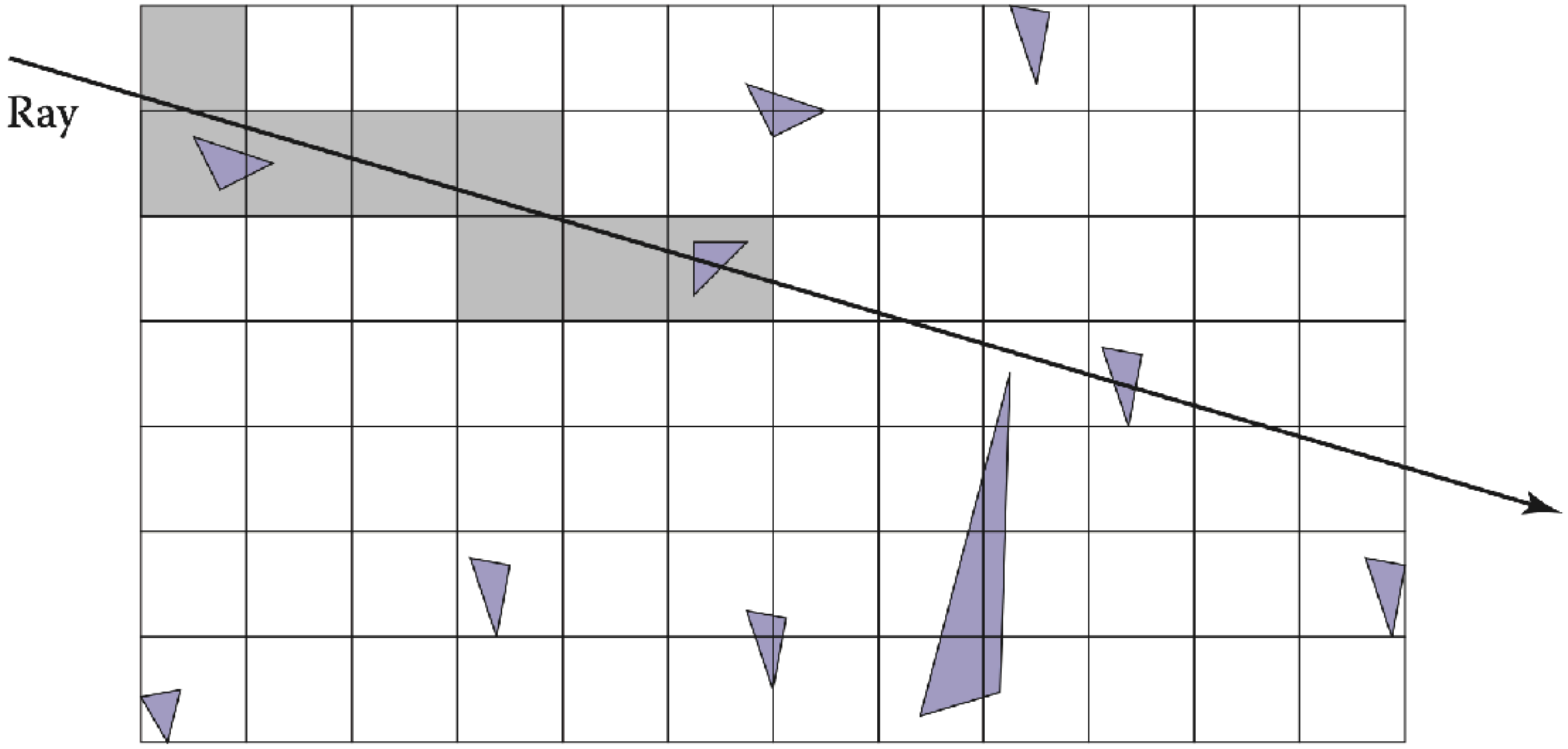
Uniform grids

Even simpler strategy: divide bounding box into equal sized cells.
Each cell stores list of overlapping objects.

Usually pick resolution so
#cells = *O*(#objects)

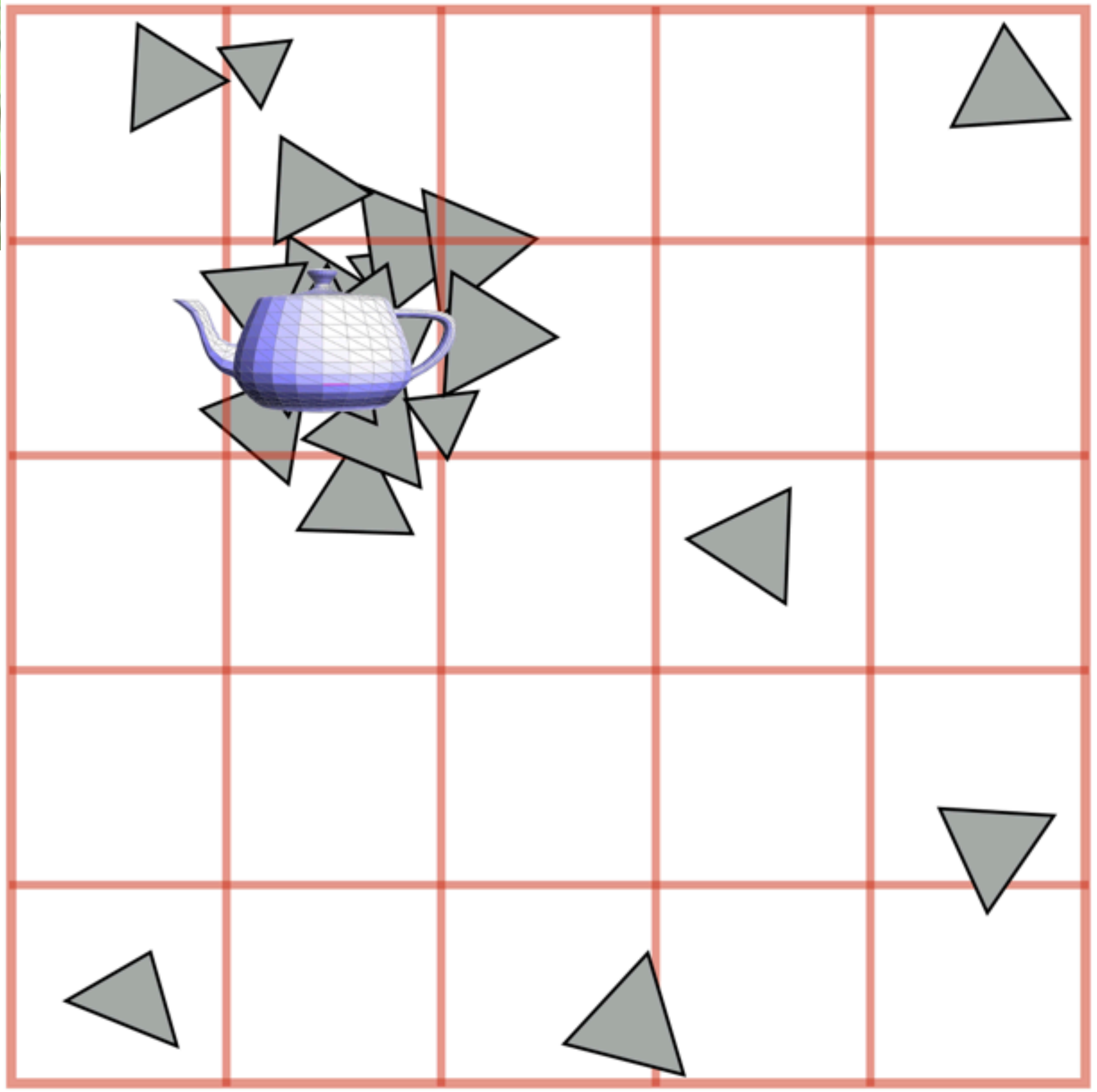
- Very easy to traverse, no recursion

- Does not adapt well to nonuniform distribution of object locations or sizes



Good for grids:

Bad for grids:



"Teapot in a stadium" problem

- Assignment 2 updated
- No class tomorrow

All the best for the minor exam!!

Post any doubts on Moodle soon, I will not reply after I go on holiday :)