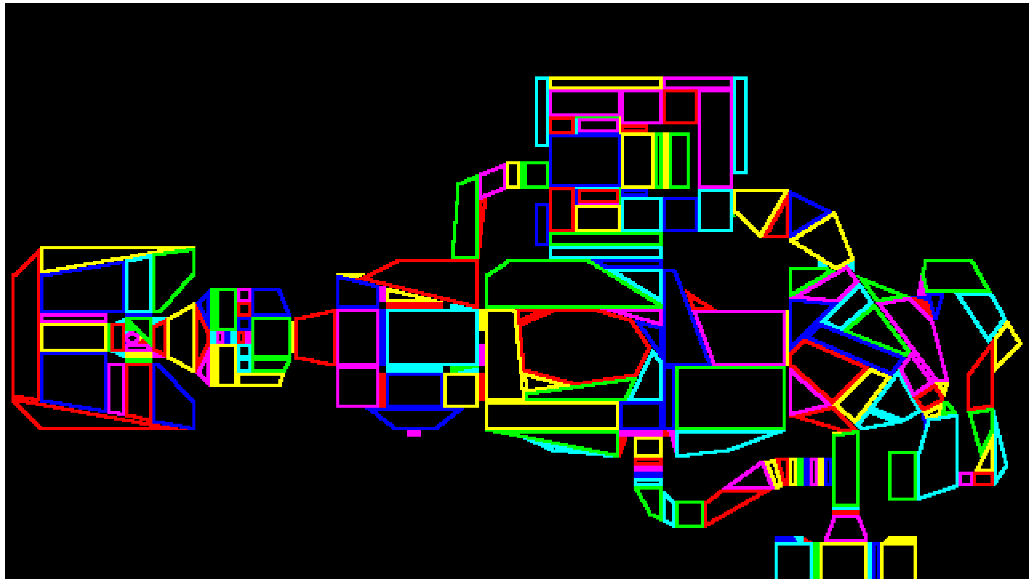
Bounding Volume Hierarchies



Some Slides/Images adapted from Marschner and Shirley and David Levin

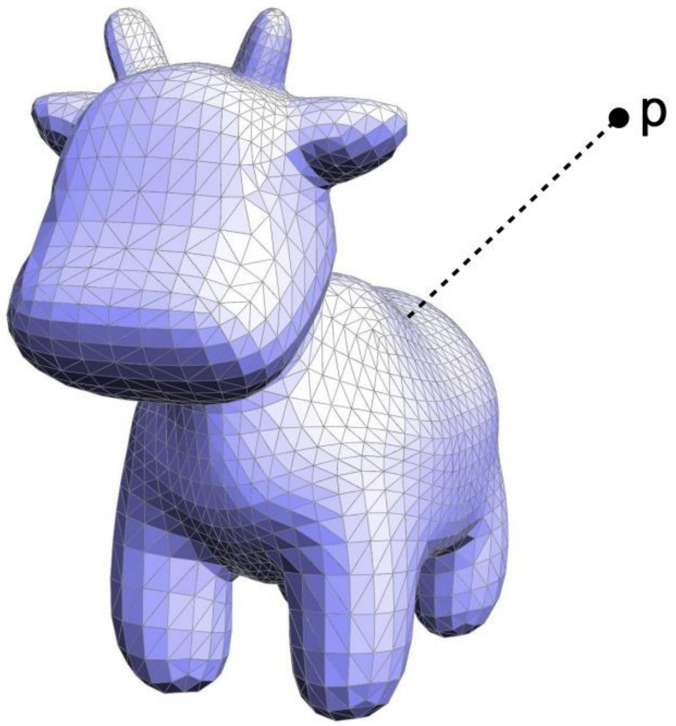
Agenda

- Motivation: Common Geometric Queries in Graphics
- Bounding Volumes
  - Spheres
  - Boxes (AABB, OOBB)
- Constructing Object-Partitioning Hierarchies
  - Sphere Trees
  - AABB Trees
- Space-Partitioning Hierarchies
  - Uniform Spatial Subdivision
  - Axis-Aligned Spatial Subdivision

Geometric modeling and geometric queries

Closest point on a triangle?
Project p to p’ in plane of triangle.
Calculate barycentric coordinates of p’.
Clamp coordinates to [0,1].
Point with clamped coordiantes is closest to p.

Loop over all triangles and keep the closest.
What is the complexity?
What if the object has a billion triangles?

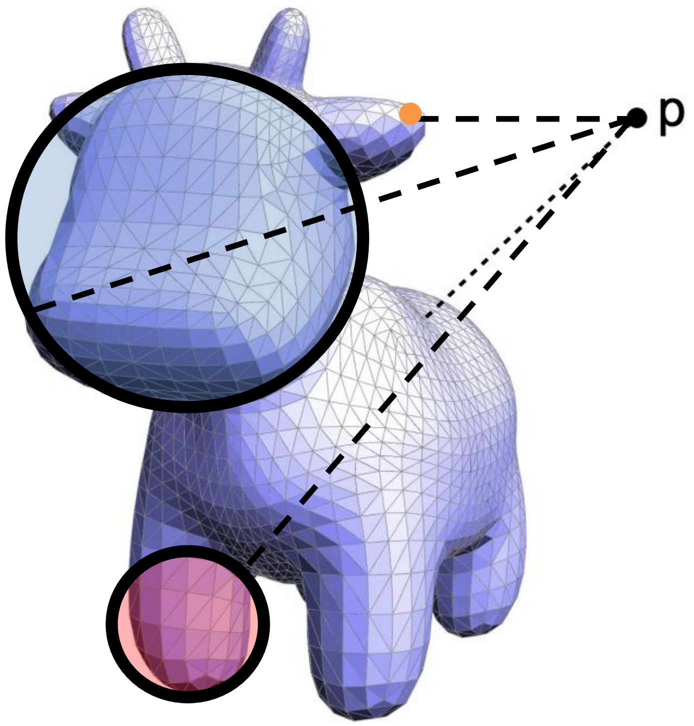


What point on the mesh is closest to **p**?

http://15462.courses.cs.cmu.edu/fall2016content/lectures/12_accelstructures/12_accelstructures_slides.pdf

Geometric modeling and geometric queries

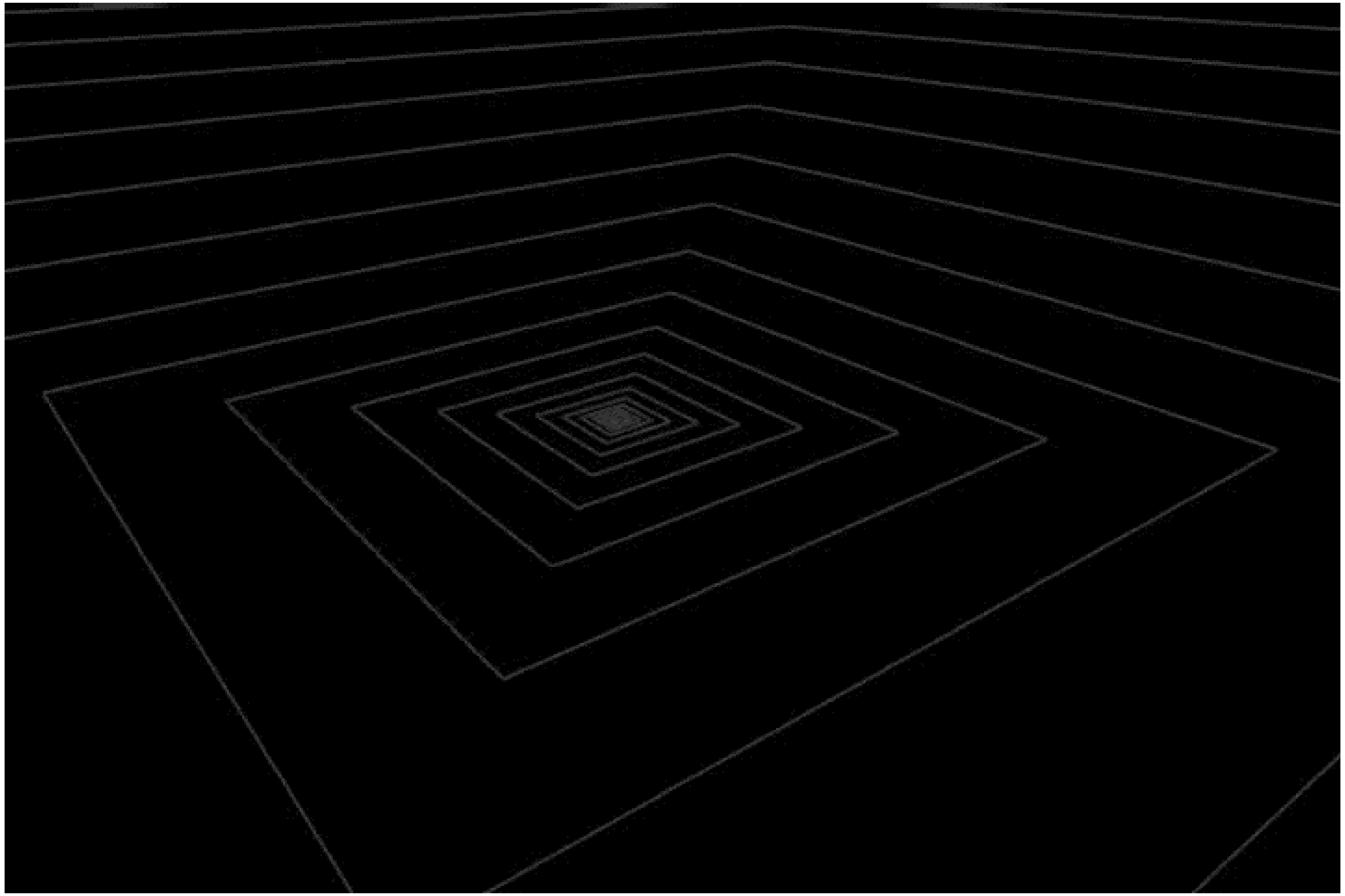
- Closest point to a sphere with center c and radius r is:
  $c \pm r$ * normalized (p-c)
- Closest point on red sphere is further than furthest point on blue sphere.
- Closest point on blue sphere is further than the orange point on a triangle.



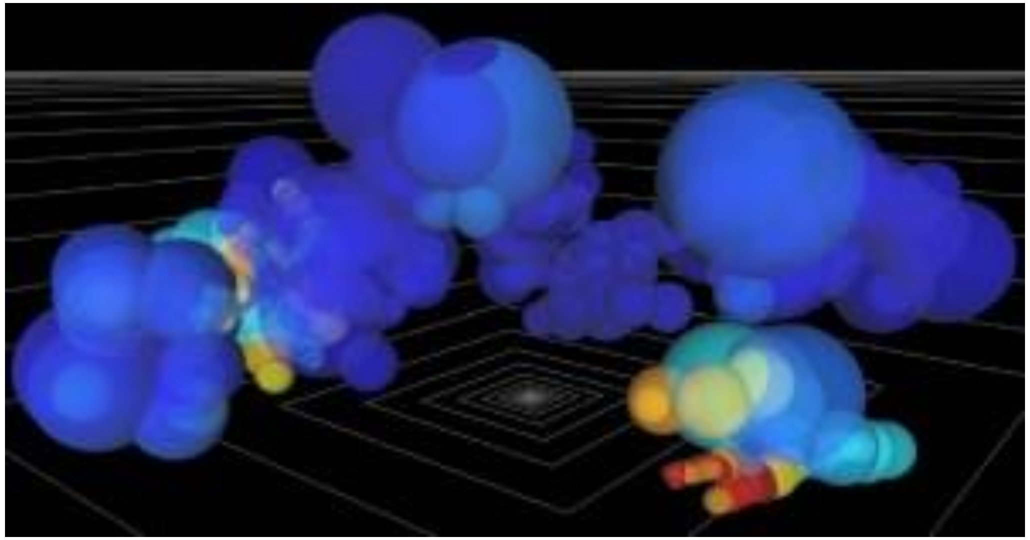
What point on the mesh is closest to **p**?

http://15462.courses.cs.cmu.edu/fall2016content/lectures/12_accelstructures/12_accelstructures_slides.pdf

Collision Intersection

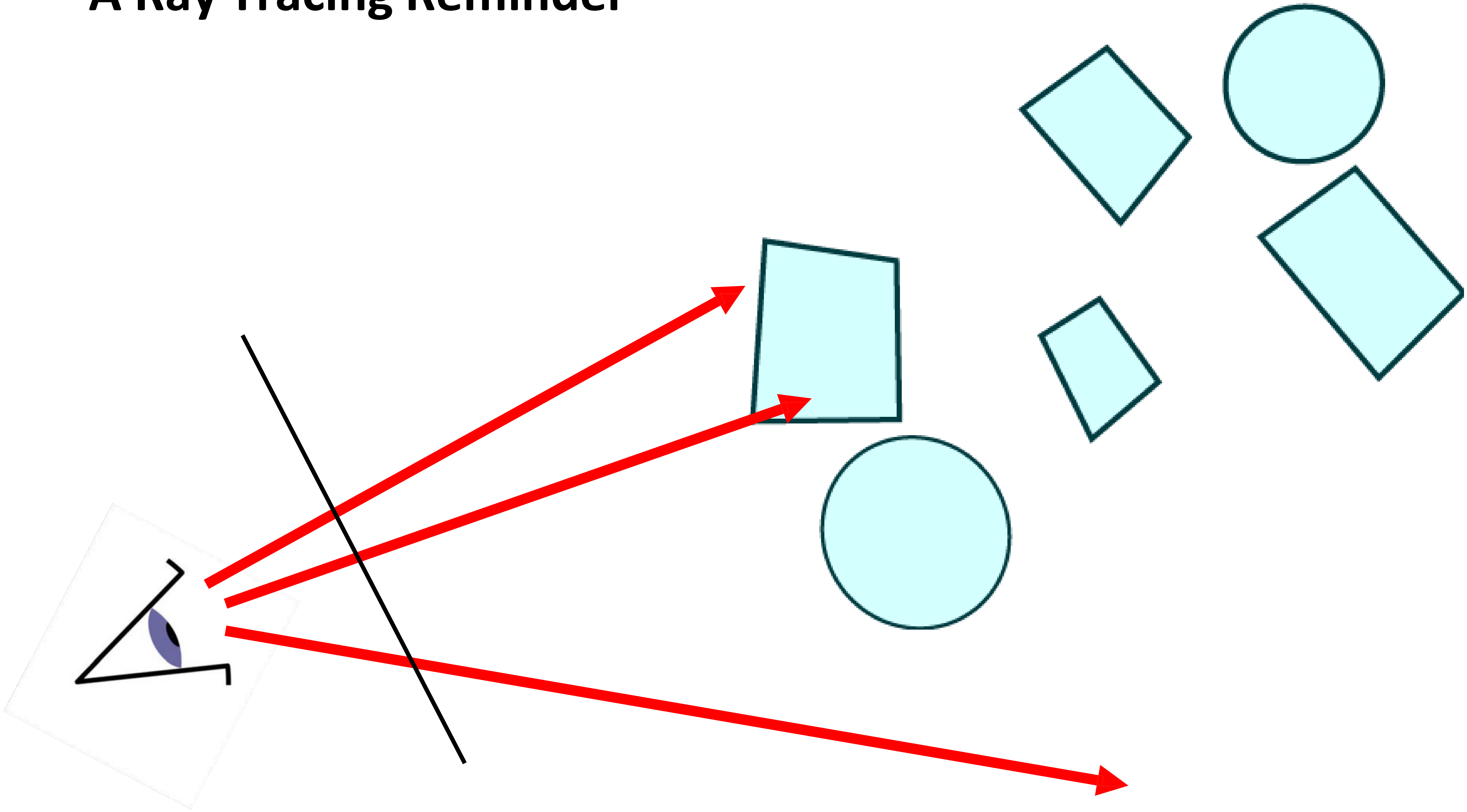


How can you test if two spheres intersect?

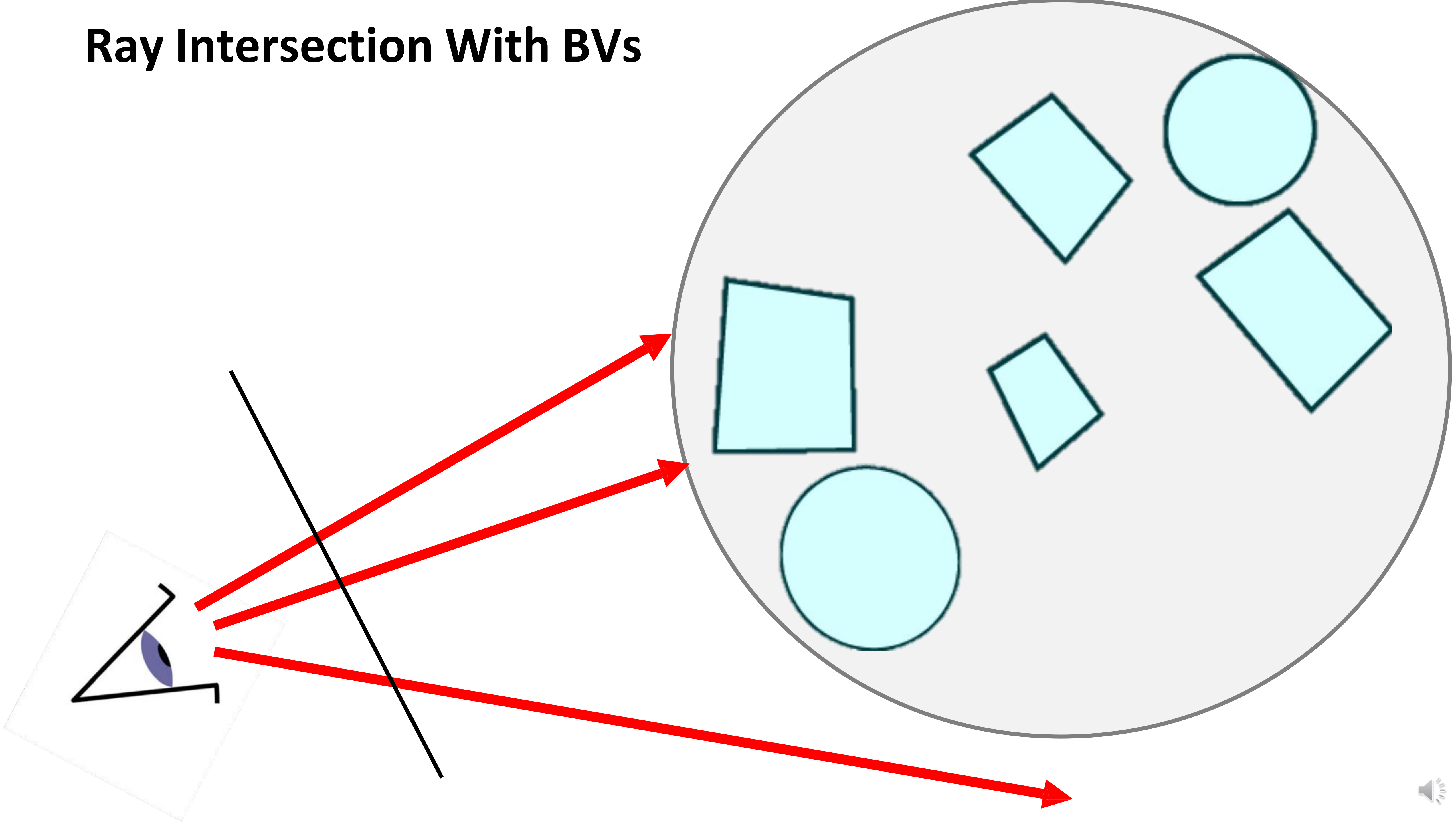


http://graphics.cs.cmu.edu/projects/bdtree/ Bounded Deformation-Tree [James and Pai]

A Ray Tracing Reminder



Ray Intersection With BVs



Bounding volumes

Quick way to avoid expensive testing intersections and collisions:

bound object with a simple volume **bvol** (volume encloses object)

If a ray/object doesn't hit/intersect **bvol**,

it doesn't hit/intersect the object

else

test object for hit/intersect

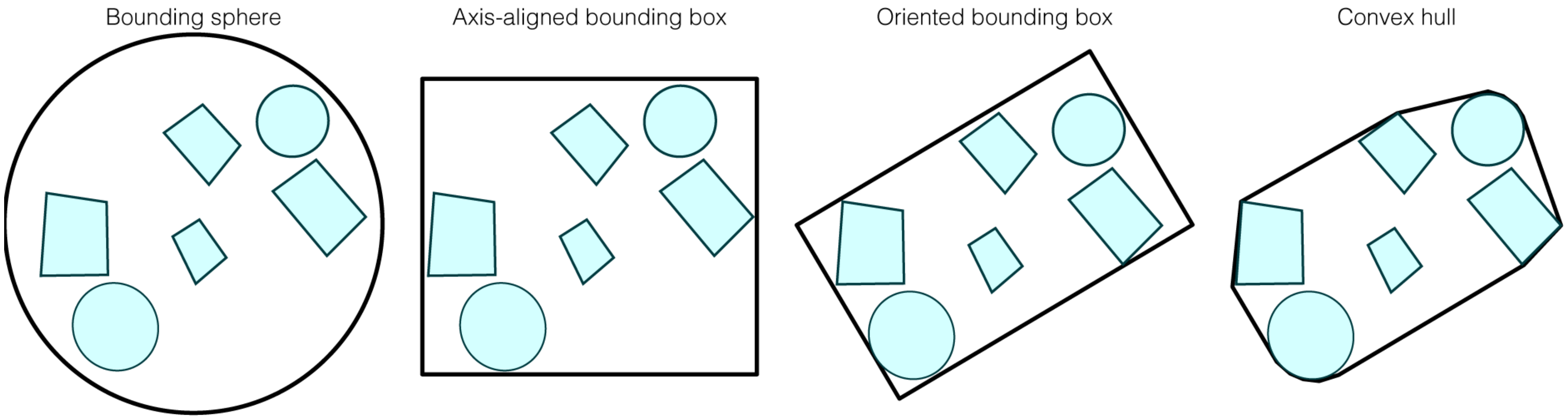
Cost: more for hits and near misses, less for far misses

Worth doing? Yes if:

- Cost of **bvol** intersection test is small: simple shapes (spheres, boxes, …)
- Cost of object intersect test is large: **bvol** most useful for complex objects
- Tightness of fit is good:
  - Loose fit leads to extra object intersections
  - Tradeoff between tightness and **bvol** intersection cost

Choice of bounding volumes

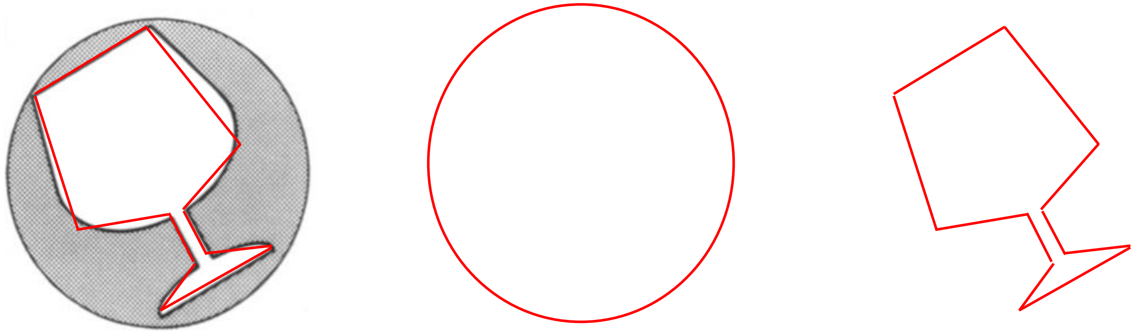
Spheres: easy to intersect, not always tight.
Axis-aligned Bounding Boxes (AABBs): easy to intersect, tighter for axis-aligned objects.
Oriented bounding boxes (OBBs): easy to intersect (transformation cost), tighter than AABBs.
Convex Hull: not as easy to intersect as the above, tighter than the above.



Proxy Geometry vs. Bounding volumes

Another concept often used in CG is proxy geometry or Level-Of-Detail LOD.
A proxy is a simplified representation of the object, that can be used as the object when rendering and processing speed is more important than visual accuracy.

Note, that proxy geometry is an approximation to the object and typically not a bounding volume. And a bounding volume itself is typically not a good visual proxy for an object.



[Glassner 89, Fig 4.5]

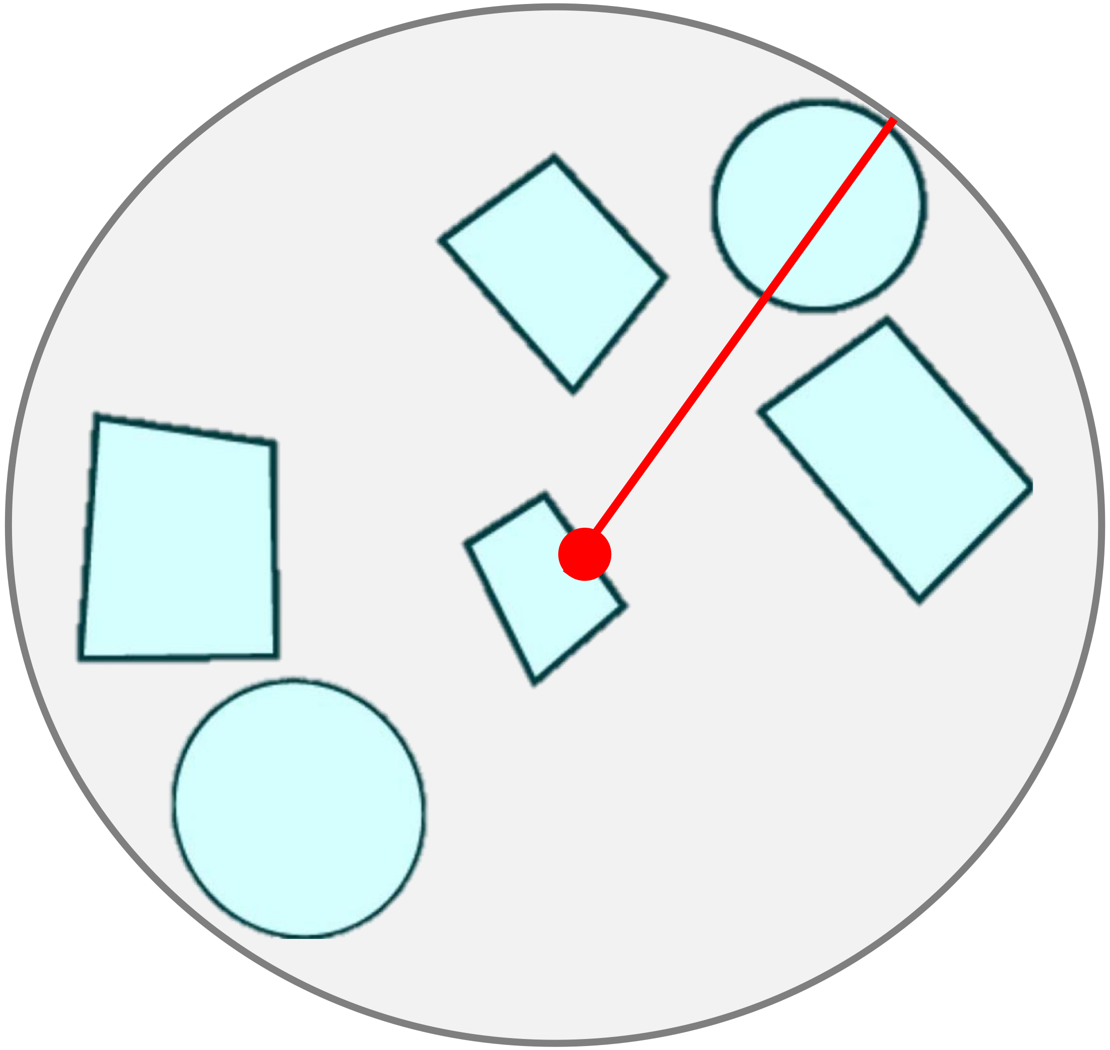
Building a Bounding Sphere

Parameters of a Sphere:

1. Center = $$\mathbf{c} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{v}^i$$

2. Radius = $$r = \max(\mathbf{v}^i - \mathbf{c})$$

$$\mathbf{v}^i \in \text{Vertices}$$



Ray-Sphere Intersection

Substitute ray equation into implicit equation for sphere

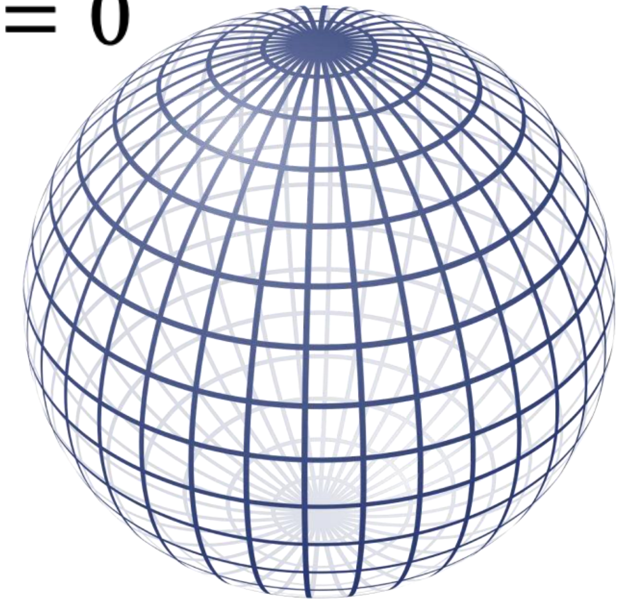
$$(\mathbf{e} + t\vec{\mathbf{d}} - \mathbf{c}) \cdot (\mathbf{e} + t\vec{\mathbf{d}} - \mathbf{c}) - r^2 = 0$$

Rearrange

$$(\vec{\mathbf{d}} \cdot \vec{\mathbf{d}})t^2 + 2\vec{\mathbf{d}} \cdot (\mathbf{e} - \mathbf{c})t + (\mathbf{e} - \mathbf{c}) \cdot (\mathbf{e} - \mathbf{c}) - r^2 = 0$$

Looks familiar…

$$At^2 + Bt + C = 0$$



It’s a quadratic! (can use the quadratic equation)

Axis aligned bounding boxes

Probably easiest to implement

Computing for primitives

- Cube: duh!
- Sphere, cylinder, etc.: pretty obvious
- Groups or meshes: min/max of component parts

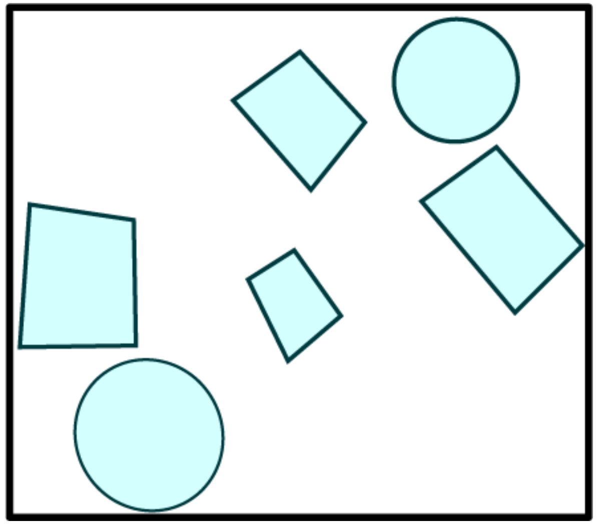
$$x_{\text{min}} = \min(v_x^i)$$

$$x_{\text{max}} = \max(v_x^i)$$

$$y_{\text{min}} = \min(v_y^i)$$

$$y_{\text{max}} = \max(v_y^i)$$

$$\mathbf{v}^i \in \text{Vertices}$$



Ray-AABB Intersection

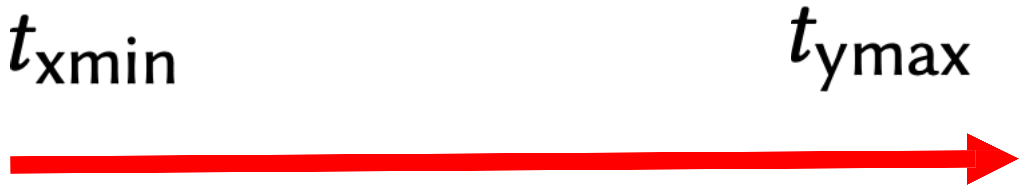
How to intersect an AABB with a ray ***p(t)=p$_e$ + p$_d$ *t***

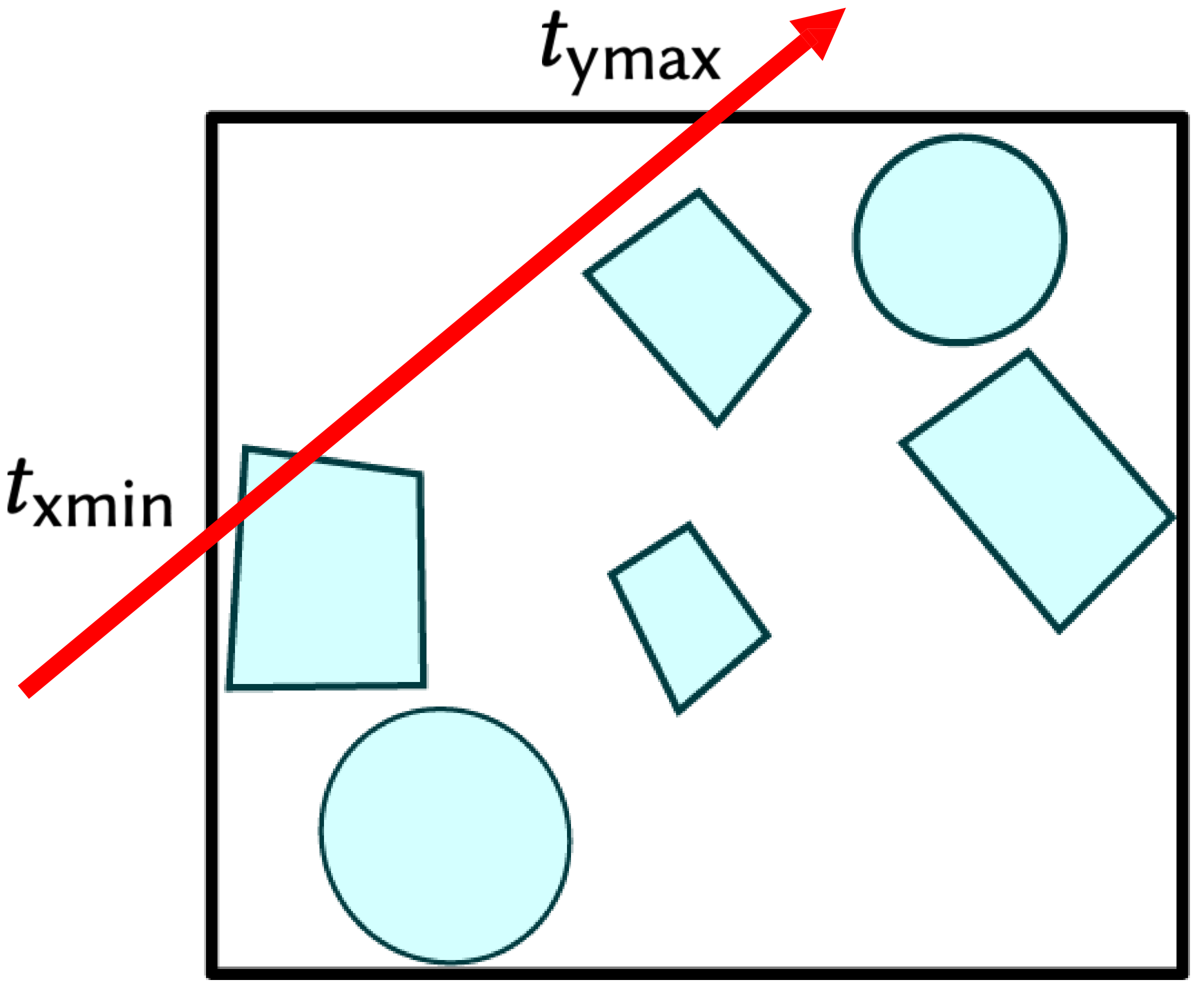
Treat them as an intersection of slabs (see book 12.3.1)

$$t_{\text{xmin}} = (x_{\text{min}} - x_e)/x_d$$
$$t_{\text{xmax}} = (x_{\text{max}} - x_e)/x_d$$
$$t_{\text{ymin}} = (y_{\text{min}} - y_e)/y_d$$
$$t_{\text{ymax}} = (y_{\text{max}} - y_e)/y_d$$

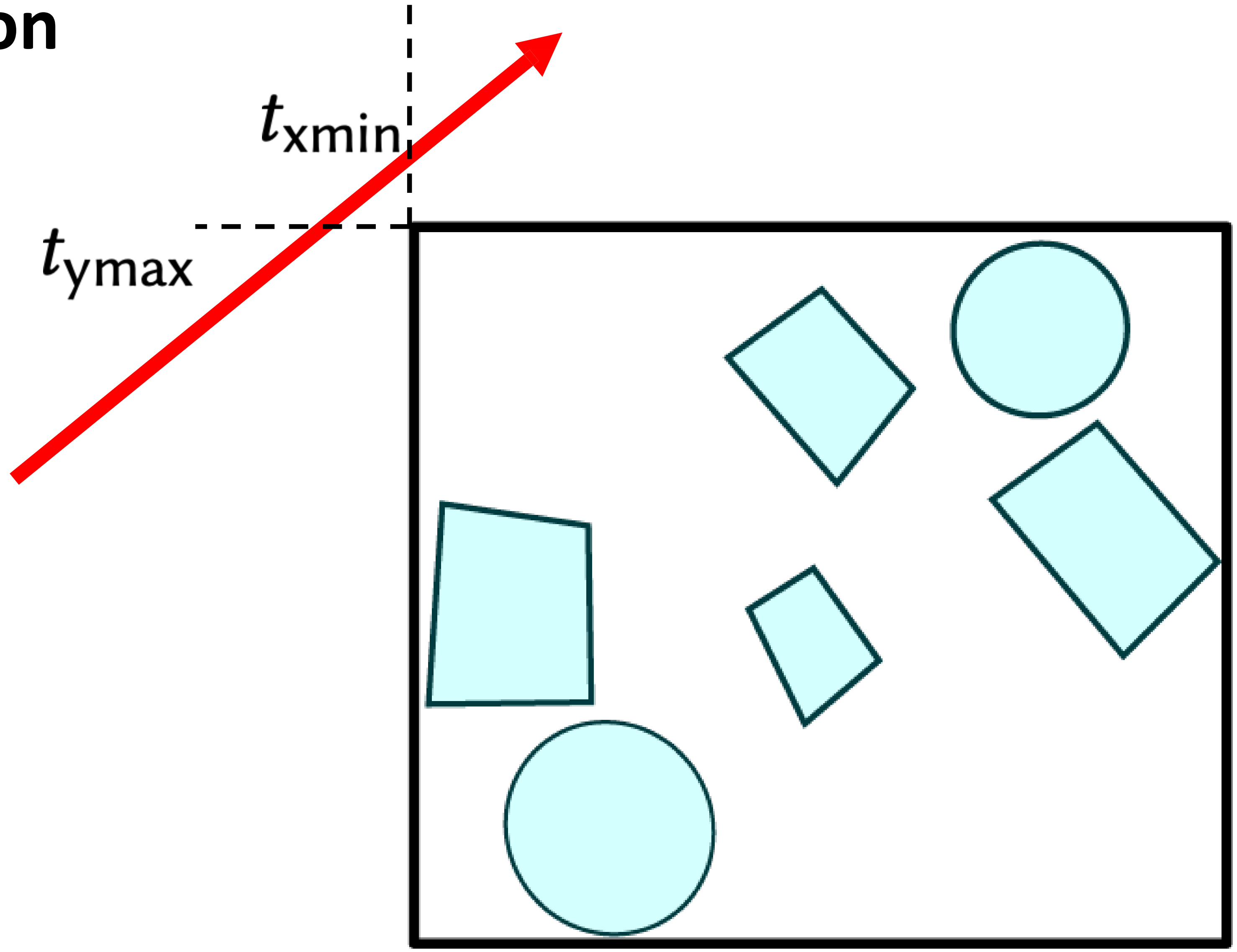
Ray-AABB Intersection

$$t_{\text{xmin}} < t_{\text{ymax}}$$

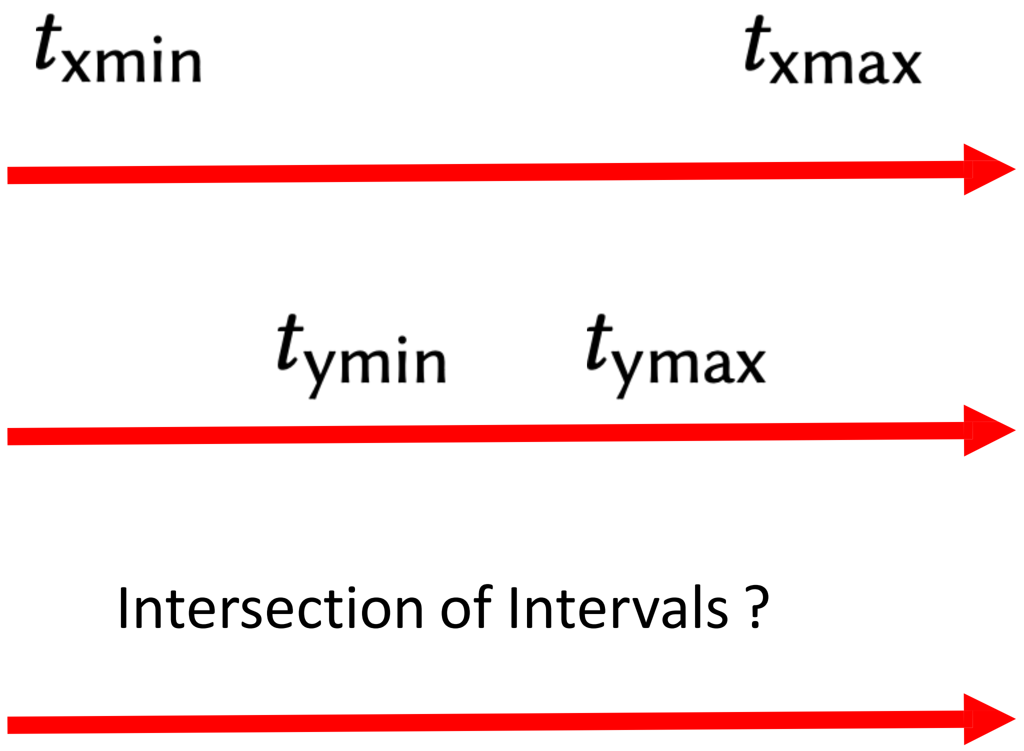




Ray-AABB Intersection

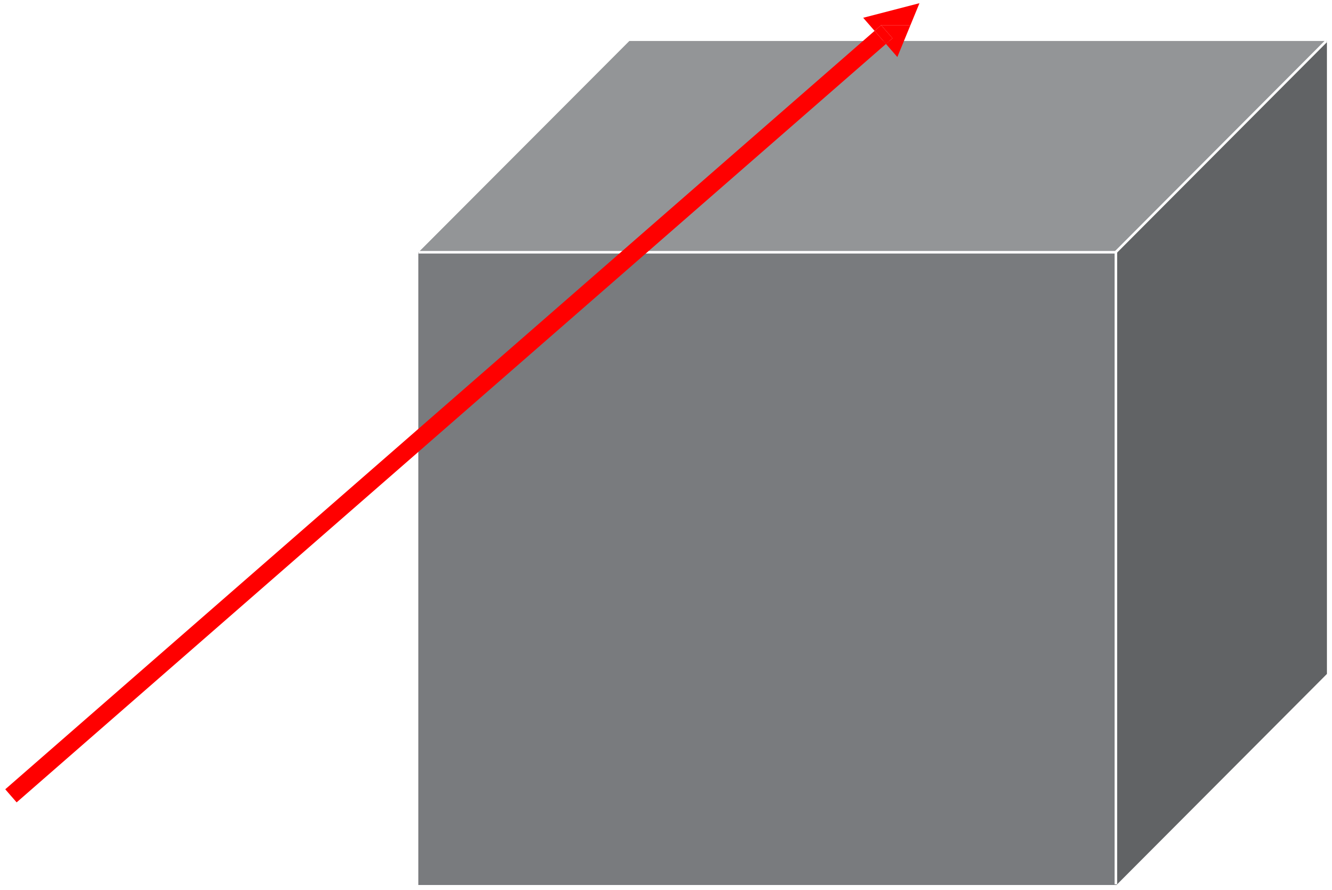


Ray-AABB Intersection

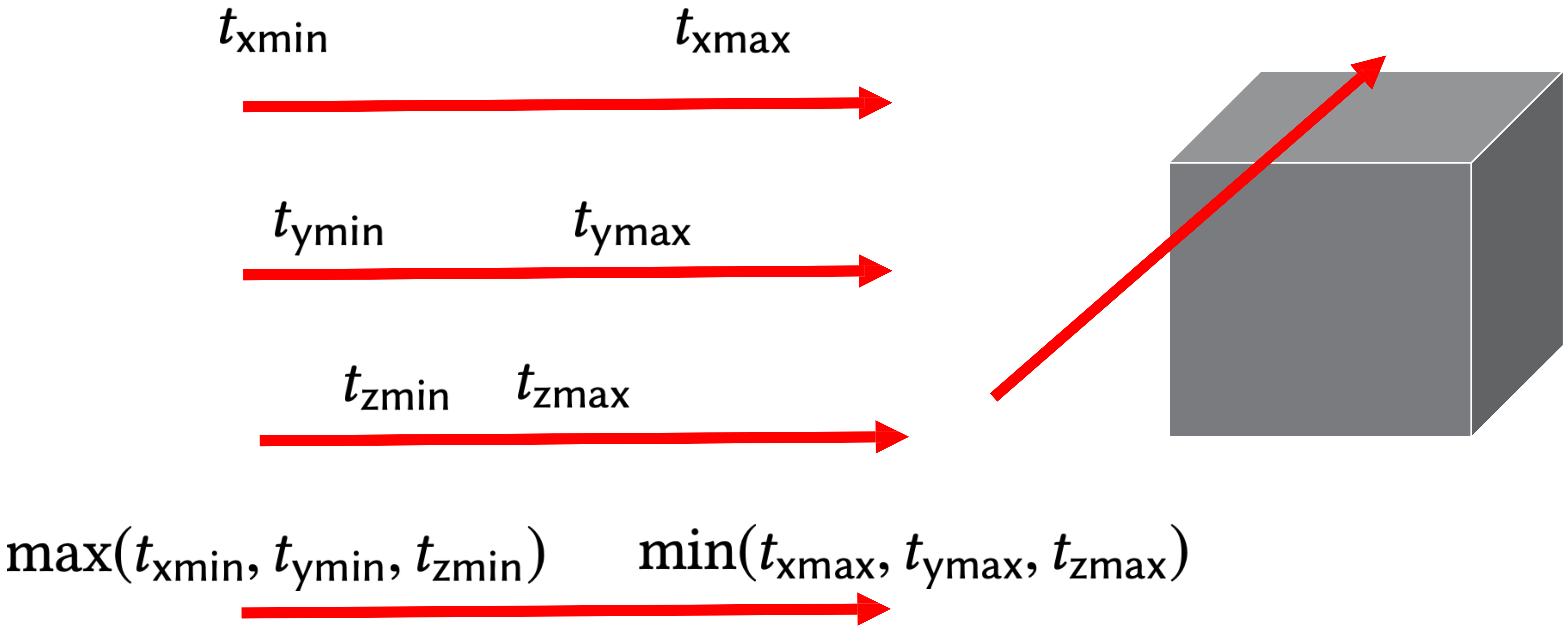


$$\max(t_{xmin}, t_{ymin}) \quad < \quad \min(t_{xmax}, t_{ymax})$$

What happens in 3D ?

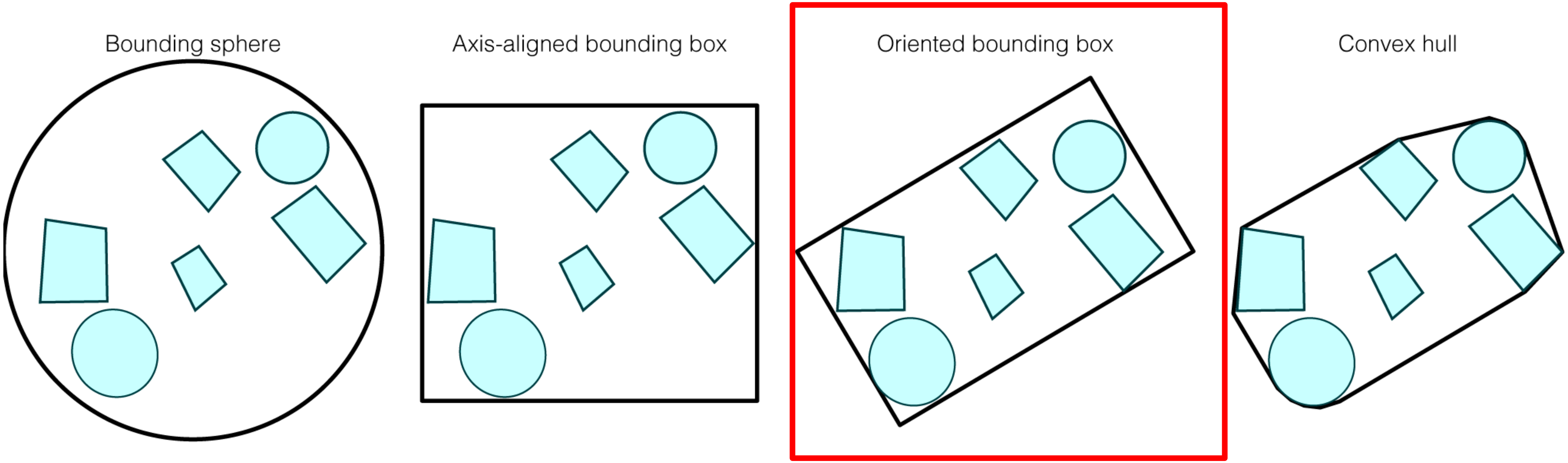


Ray-AABB Intersection



Bounding Volumes (BVs)

“Simple” geometry that fully encloses a **collection** of other geometry



Sphere
(a lot)

AABB
(a lot)

OOBB
(a little)

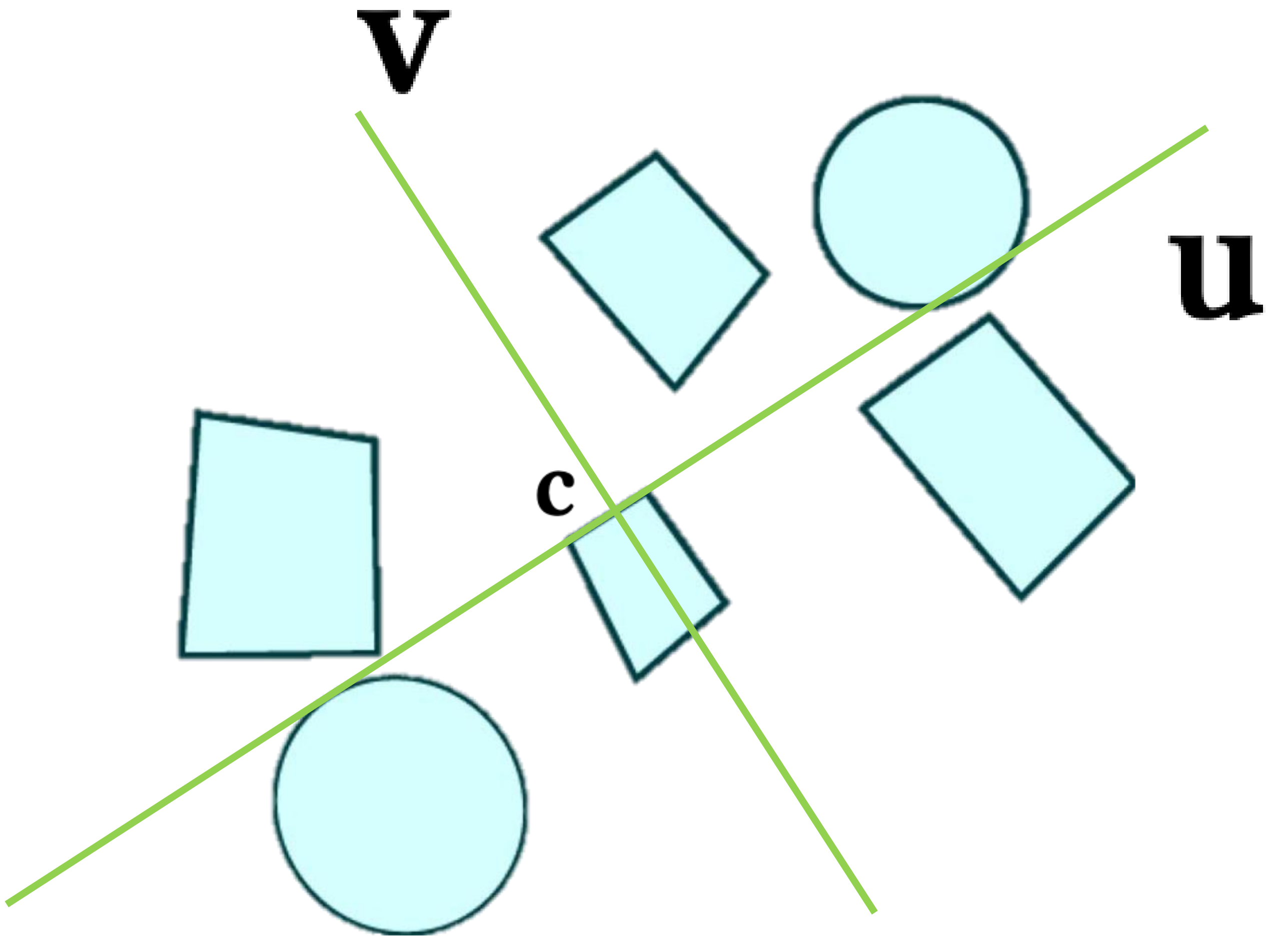
Convex Hull
(nope!)

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

Building an Object-Oriented Bounding Box (OOBB)

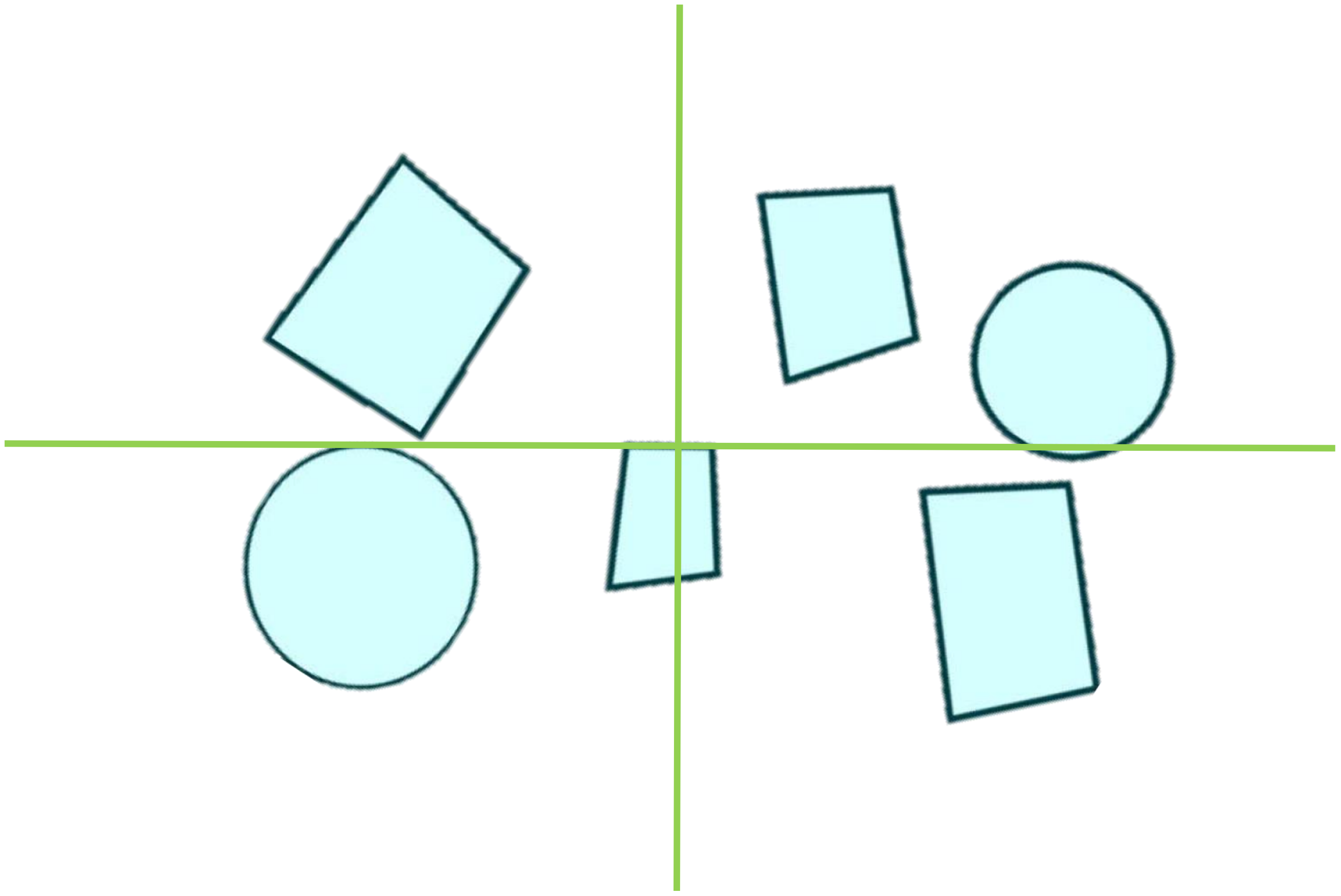
$$\mathbf{c} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{v}^i$$

$$\begin{bmatrix}\mathbf{u} & \mathbf{v}\end{bmatrix}$$



Find directions of maximum and minimum variance

Building an Object-Oriented Bounding Box (OOBB)



Build Rotation Matrix

Implementing a bounding volume

Add new Surface subclass, *BoundedSurface*

Contains a *bvol* and a reference to a *surface*

Intersection method:

```
if (!bvol.intersect(ray,t))
        return false;
else
        return surface.intersect(ray,t);
```

This change is transparent to the renderer (only it might run faster).

Implementing a bounding volume hierarchy

A *BoundedSurface* can contain a *surface* list.

Any *surface* in this list might also be a *BoundedSurface*

=> A bounding volume hierarchy

Spatial Data Structures

Basic Idea – asymptotic improvement in spatial queries by subdividing

Two types of subdivisions – ***object-based*** and *spatial Our*

object-based data structures will be boundary volume hierarchies or BVHs.

BVHs are hierarchies of BVs represented by trees

AABB Tree Construction

Make AABB for whole scene/object, then split into two parts

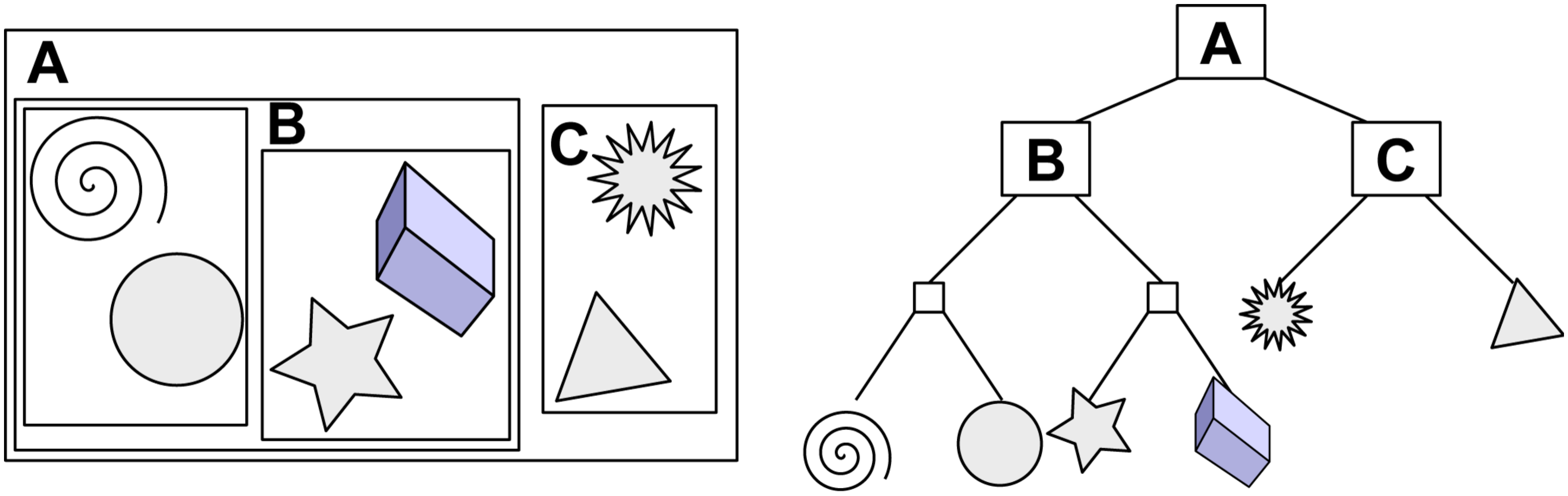
- Recurse on parts.
- Stop when there is one (or a few) object/triangle in your box.

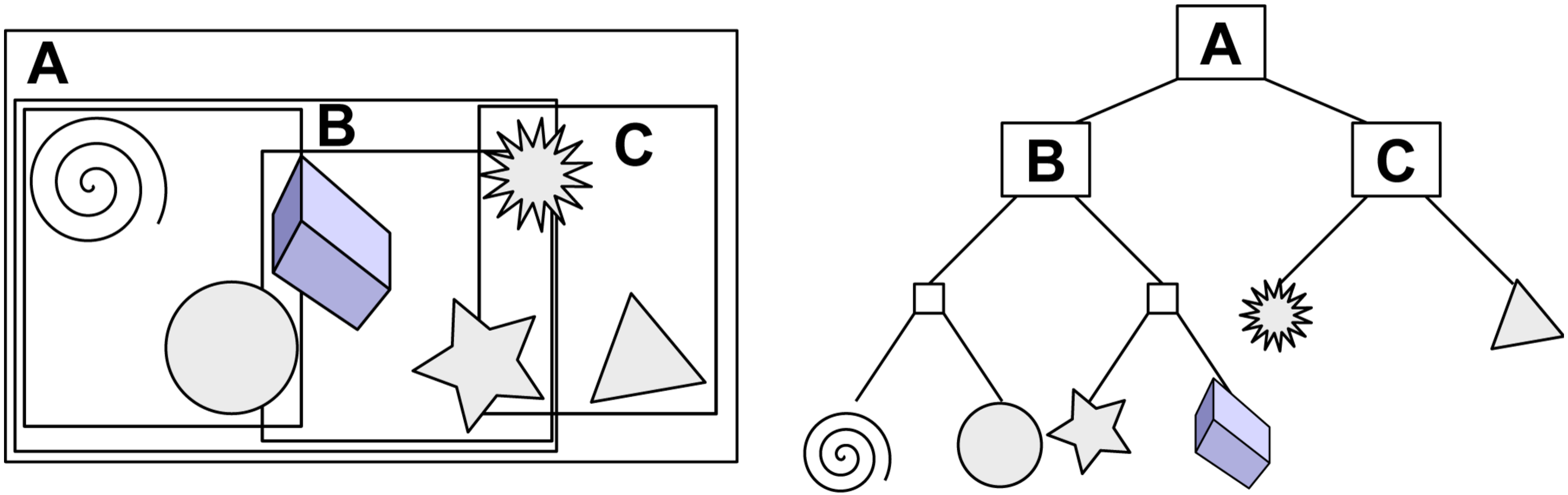
How to split into parts?

Space based: partition objects based on value relative to the center of longest dimension.

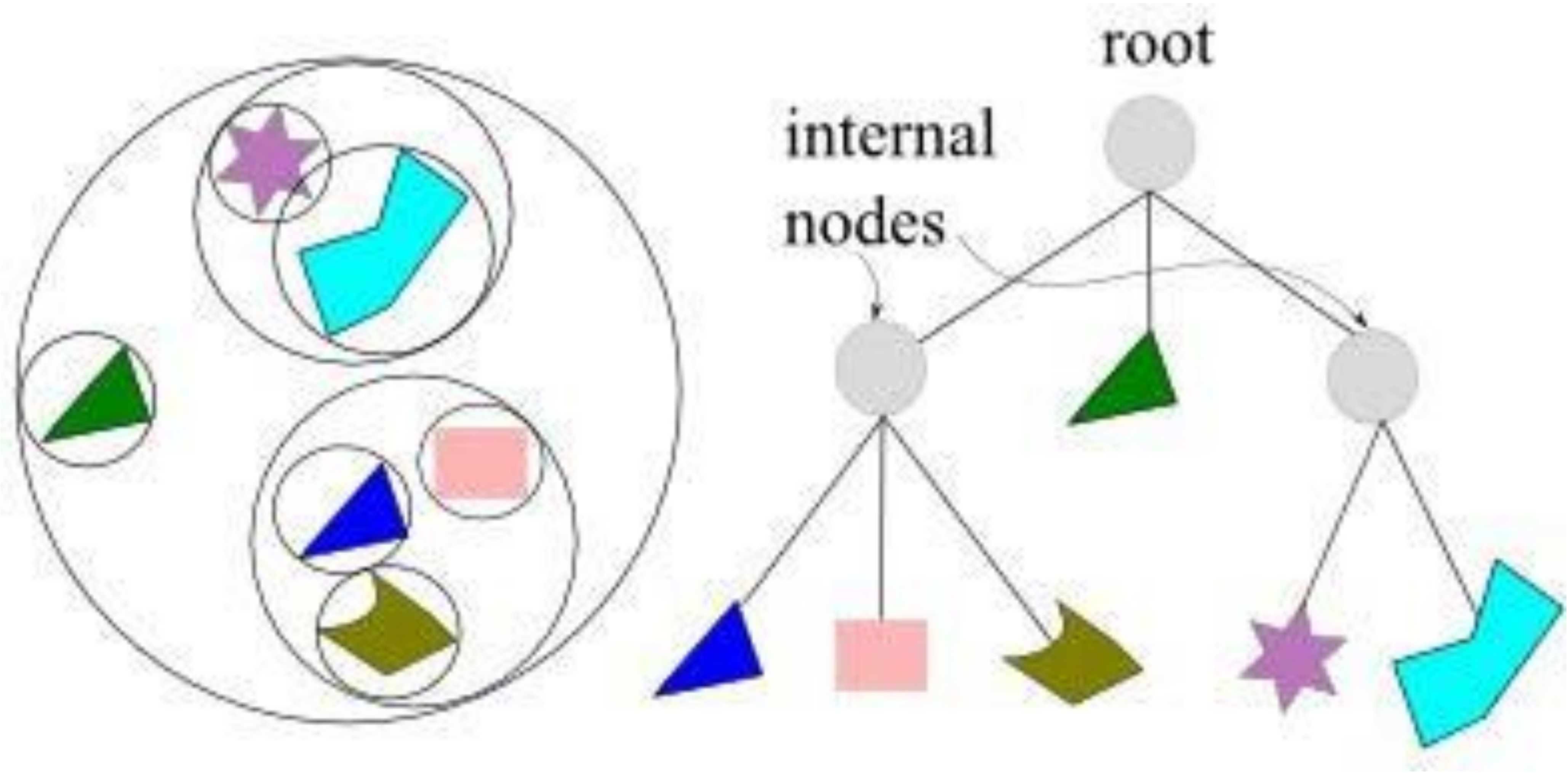
Object based: sort the objects along the longest dimension and divide them equally.

AABB Tree Construction (object based)



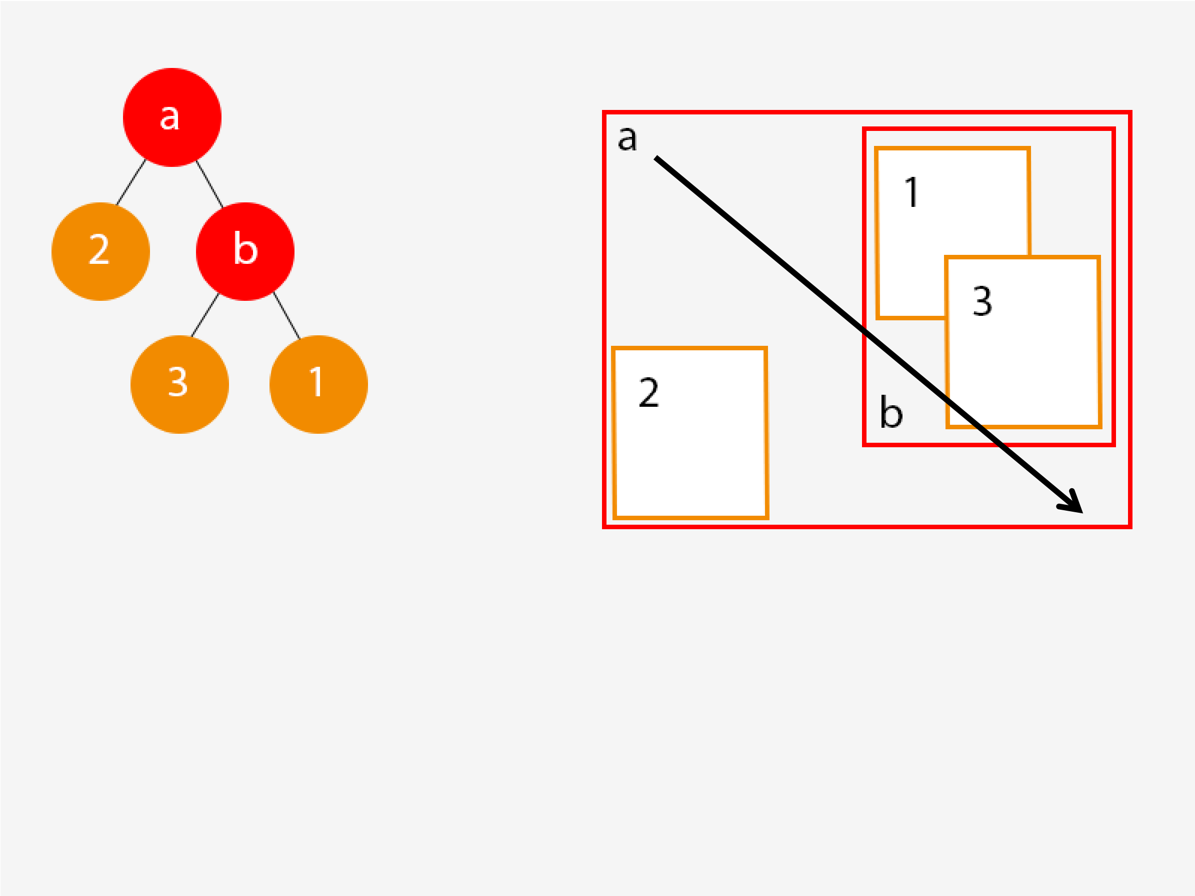


Sphere Trees



www.bogotobogo.com/Games/spatialdatastructure.php

Ray and AABB tree Intersection




```
bvh::intersect(ray,t)
{
   if (aabb== null || !aabb.intersect(ray,t))
          return false;
   else
   {
      i1=left.intersect(ray,t1);
      i2=right.intersect(ray,t2);
      if (i1 && i2) {t=min(t1,t2); return true;}
      if (i1) {t=t1; return true;}
      if (i2) {t=t2; return true;}
      return false;
   }
}
```

DFS traversal of tree nodes!

BVH Distance Queries

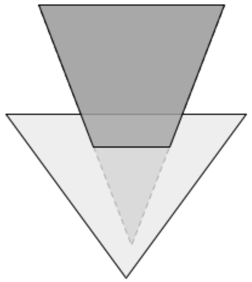
```
minDistance(bvNode, point, currentMin)
{
    if (isLeaf(bvNode))
            d1=d2=minDist(bvNode.object, point);
    else {
    d1=minDistance(bvNode.left, point, currentMin);  d2=minDistance(bvNode.right,
    point, currentMin);}

     if (min(d1,d2) > currentMin) {  return currentMin;
     return min(d1,d2)
}
```

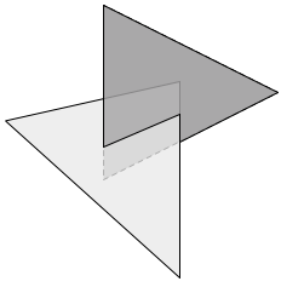
Is DFS traversal of tree nodes efficient?
BFS with a priority queue!

BVH Intersection Queries

```
leaf_pairs ← {};
if (root_A.box ∩ root_B.box) Q.insert(root_A, root_B );
while Q not empty {
        {nodeA,nodeB} ← Q.pop;
        if (nodeA and nodeB are leaves) leaf_pairs.insert( node_A, node_B );
        else if (node_A is a leaf) {  /* symmetrically for node_B */
                if (node_A.box ∩ node_B.left.box) Q.insert( node_A, node_B.left );  /* symmetrically for node_B.right */
        else {
            if (node_A.left.box ∩ node_B.box) Q.insert( node_A.left, node_B);
            if (node_A.right.box ∩ node_B.box) Q.insert( node_A.right, node_B);
            if (node_A.box ∩ node_B.left.box) Q.insert( node_A, node_B .left);
            if (node_A.box ∩ node_B.right.box) Q.insert( node_A.left, node_B.right);
        }
}
```

Triangle-Triangle intersection





T_1 intersects T_2 <=> at least one tri edge intersects the other tri.

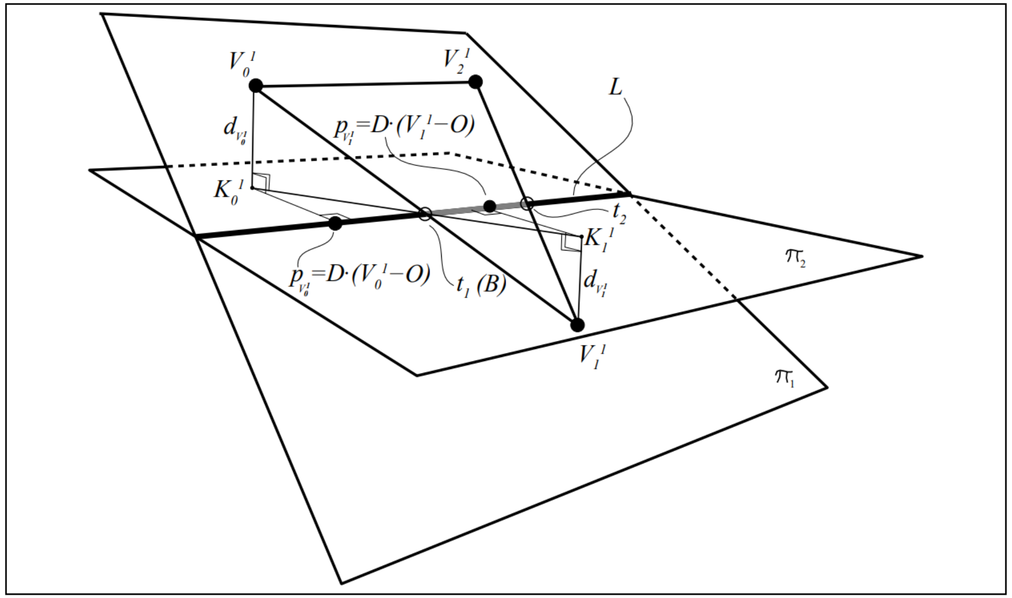
Algorithm 1: Test edge-tri intersection for all 6 edges.

T_1 intersects T_2 => Vertices of T_1 , T_2 straddle plane of T_2 , T_1 respectively.

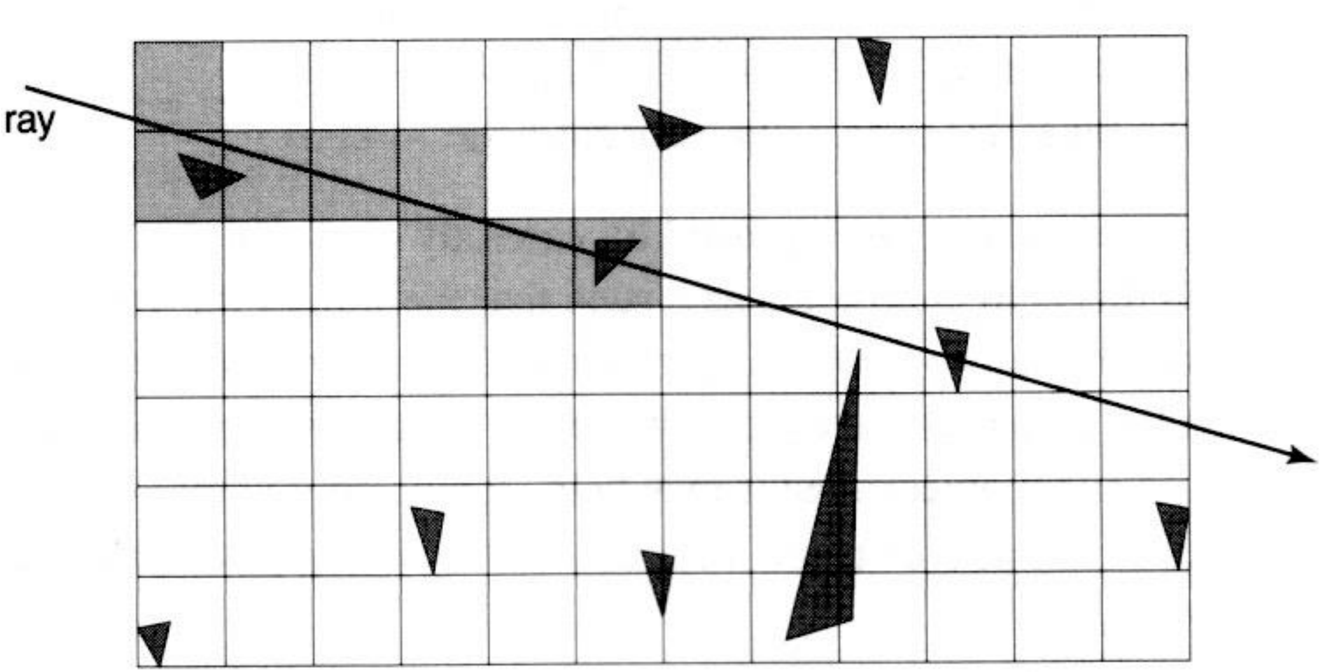
What if T_1 , T_2 are co-planar?

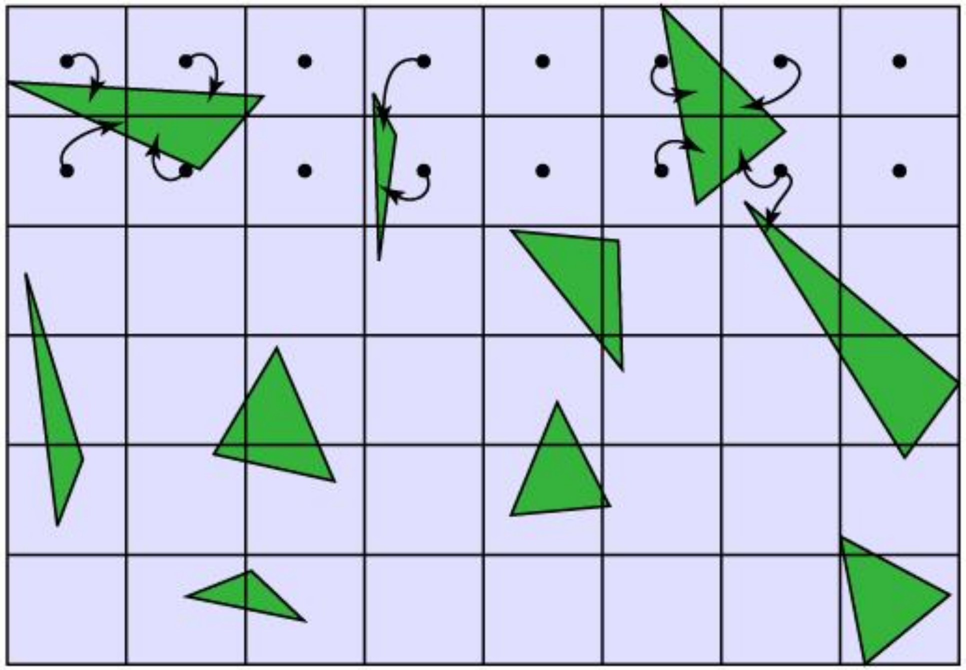
Algorithm 2: if (vertices of T_1 , T_2 on the same side of plane of T_2 , T_1) return false;

The triangles intersect iff the triangle intervals along line of intersection do.



Regular space subdivision

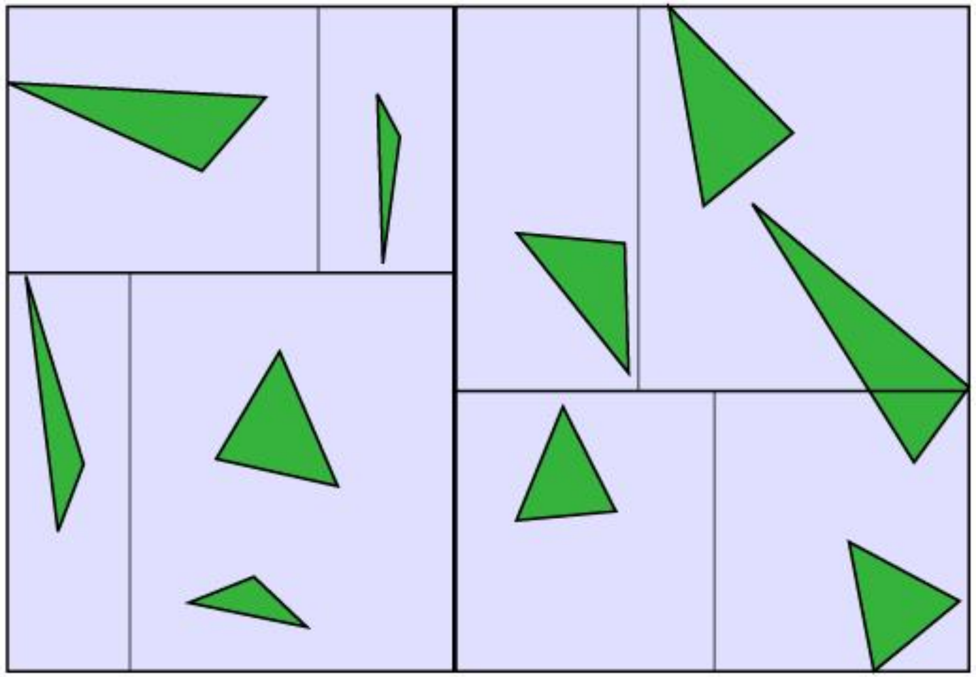


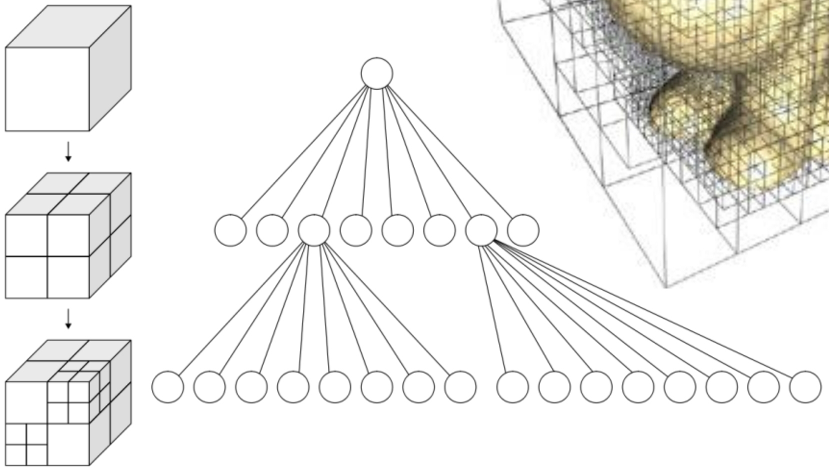


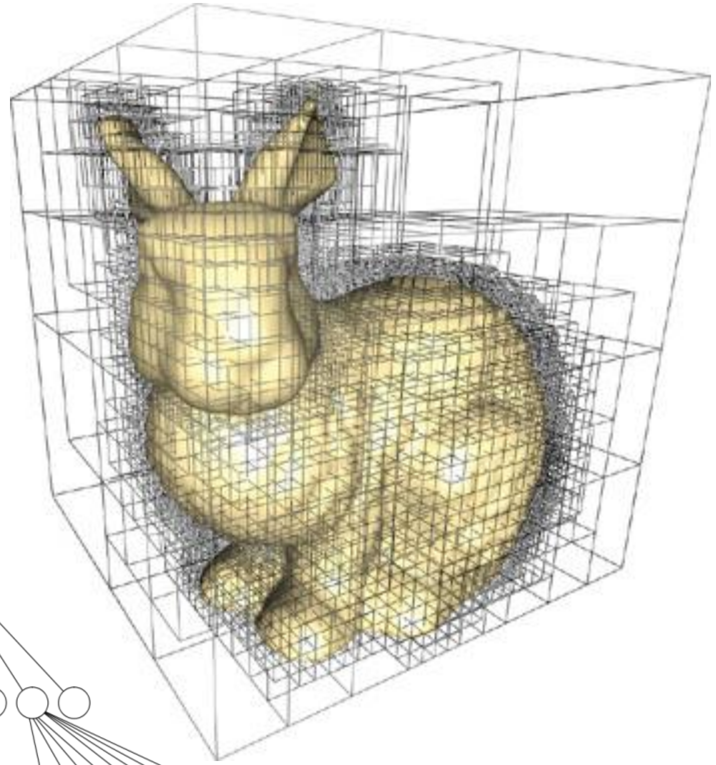
Grid divides space, not objects.

Non-regular space subdivision

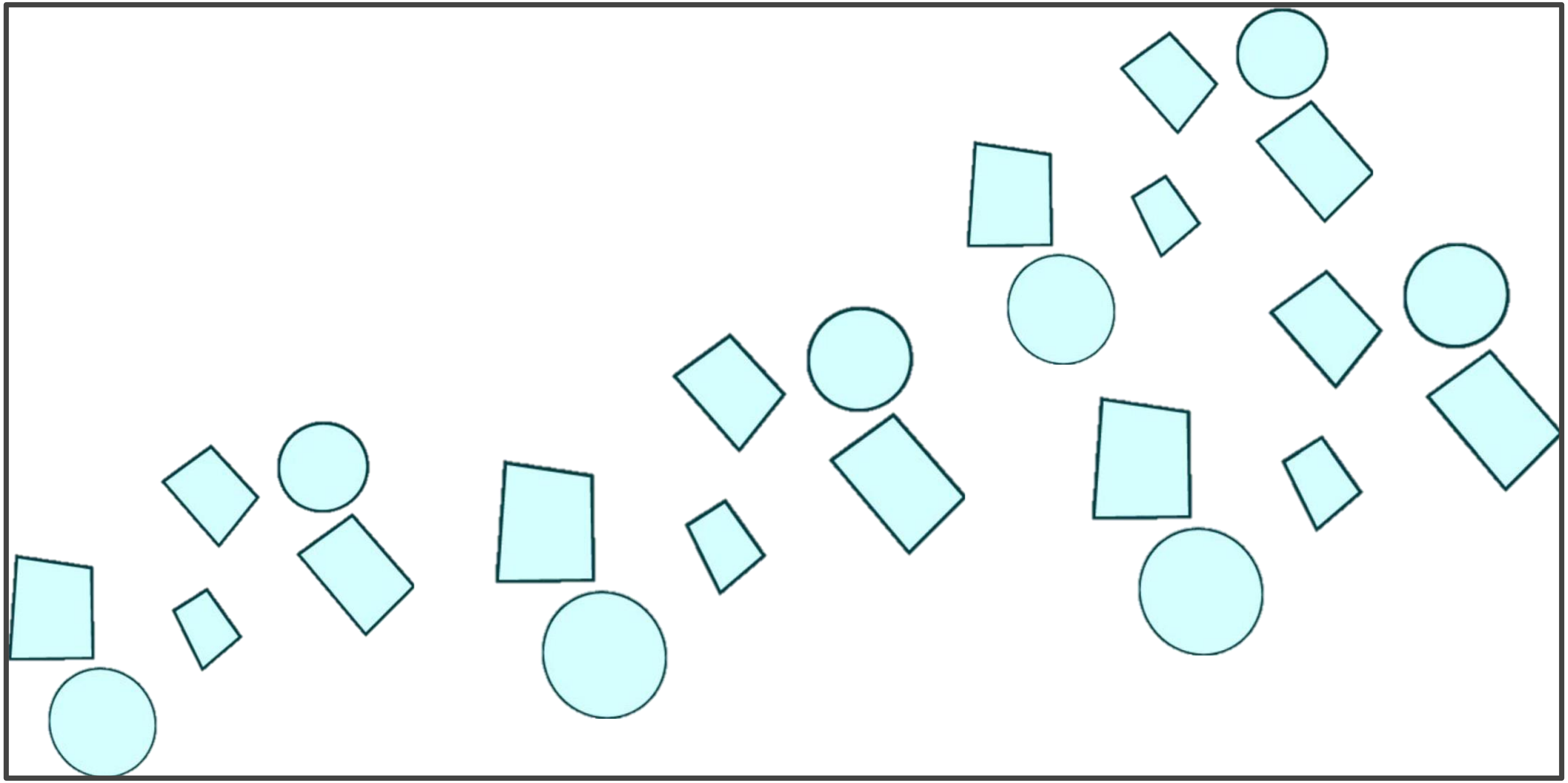
k-d Tree
Octree



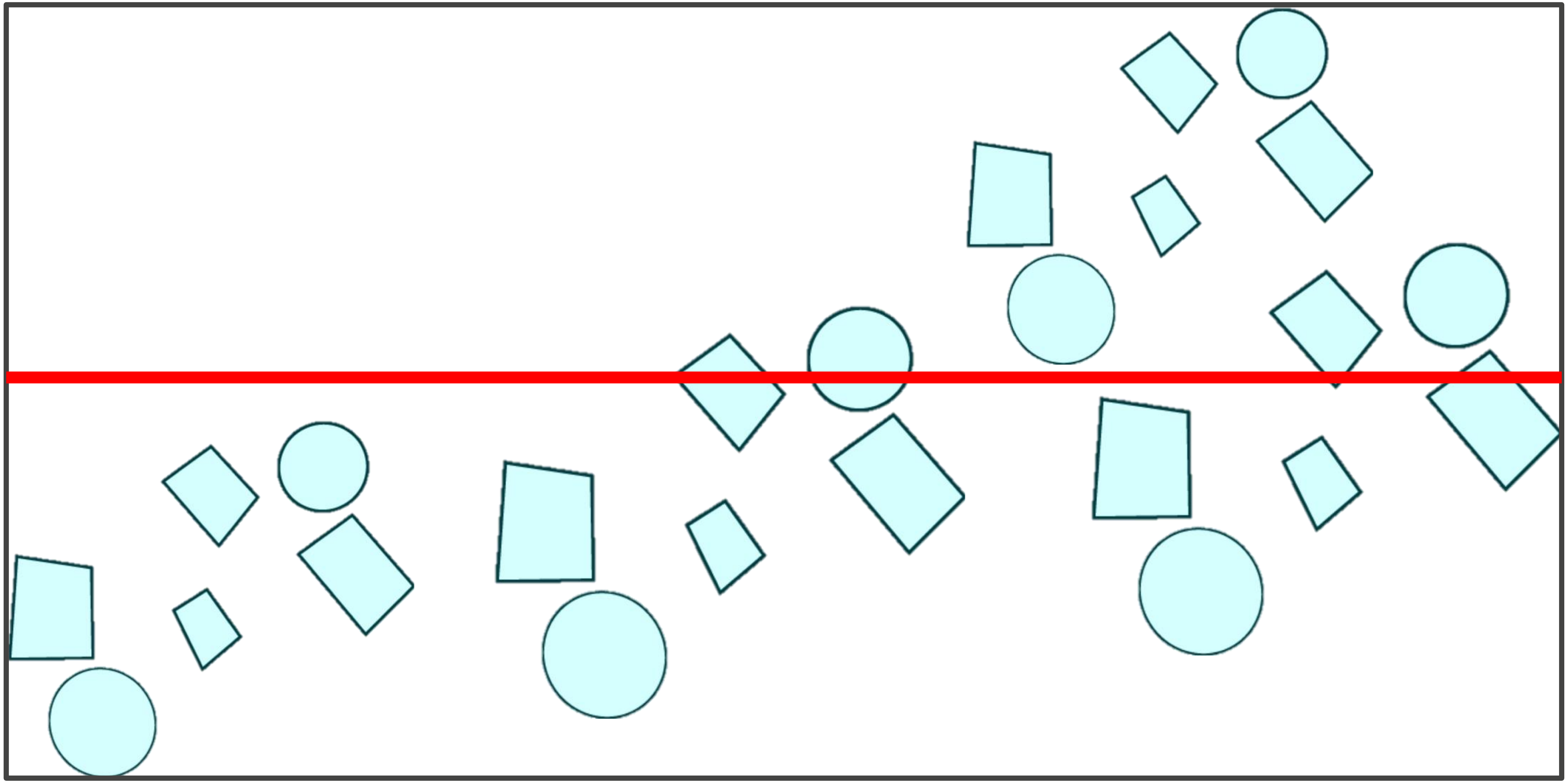




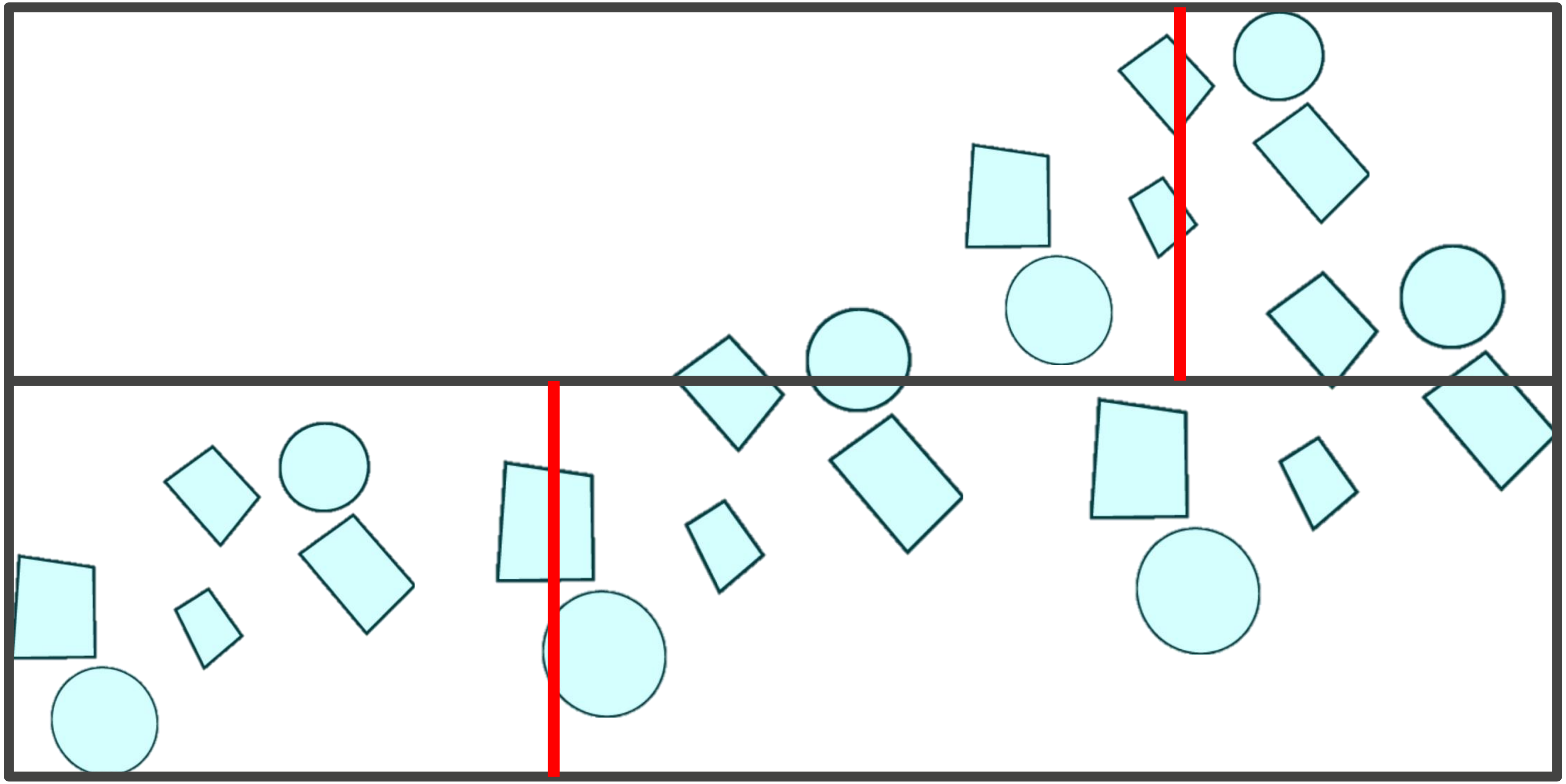
Constructing a k-d Tree



Constructing a k-d Tree



Constructing a k-d Tree



Constructing a k-d Tree