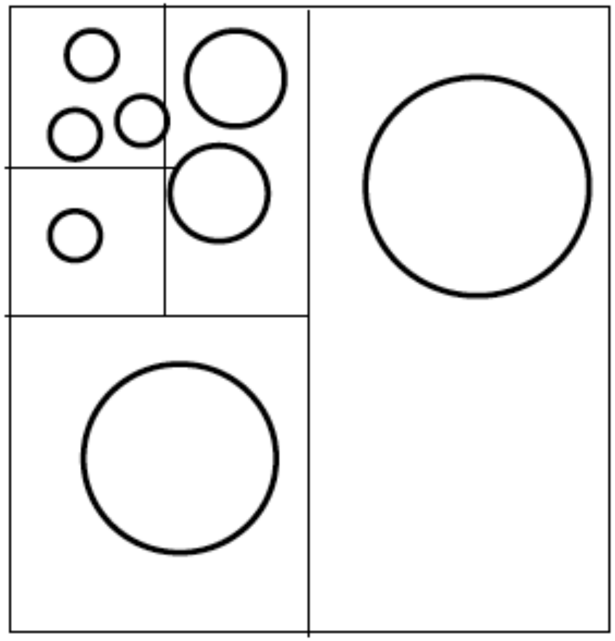
# CS 419: Production Rendering

## Octrees
## Bounding Volume Hierarchies

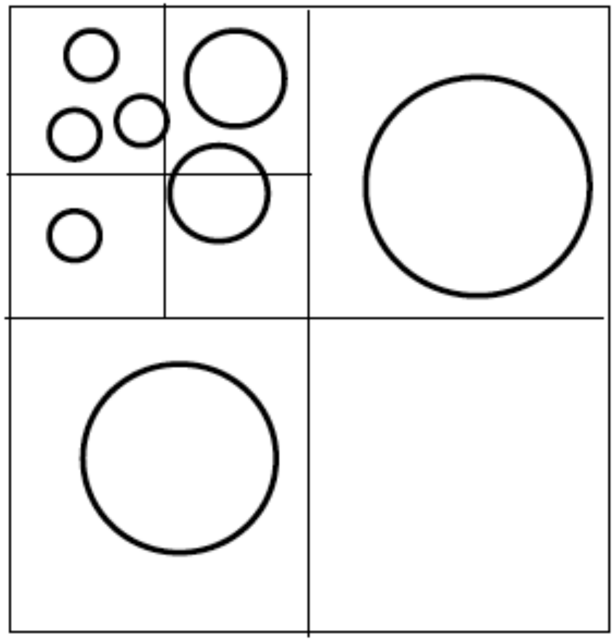
Eric Shaffer

Some Content Taken from *Physically Based Rendering* by Pharr et al.

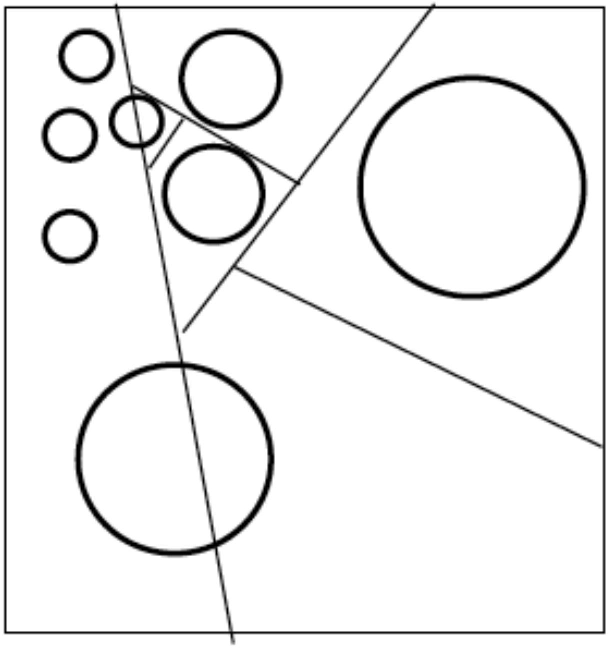
# Lots of types of Spatial Hierarchies



KD-Tree



Oct-Tree

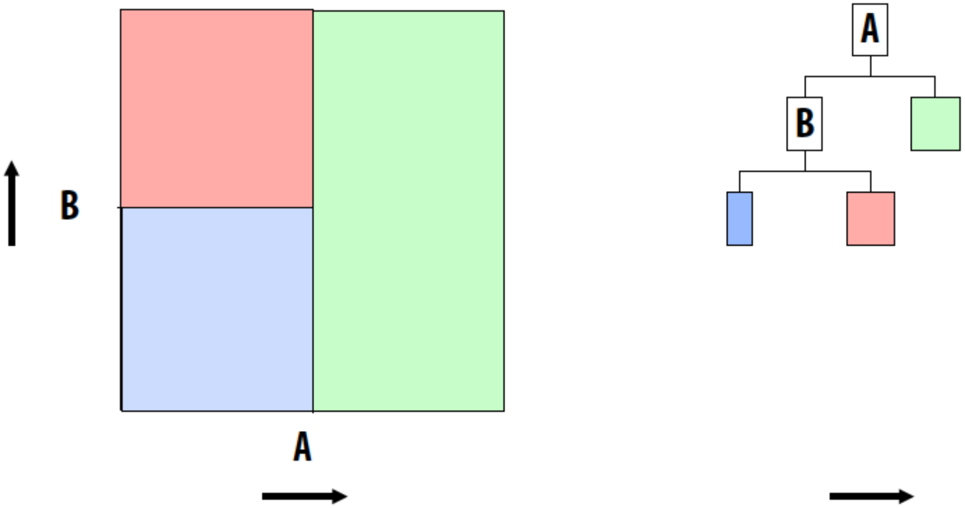


BSP-Tree

Taken from *Physically Based Rendering* by Pharr et al.

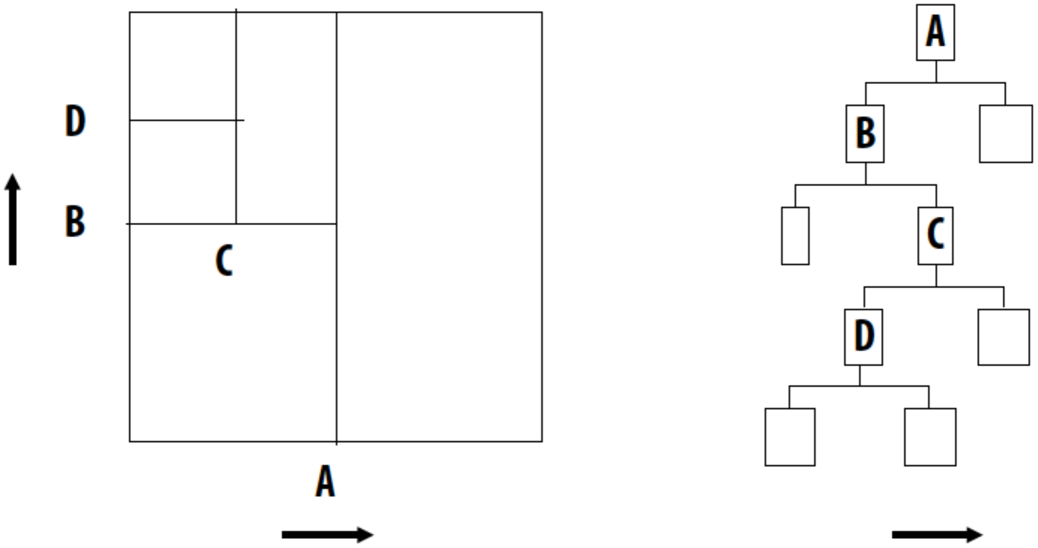
# Spatial Hierarchies

- Decompose space into partitions
- Use a tree-like structure
- Point location computed by recursive search



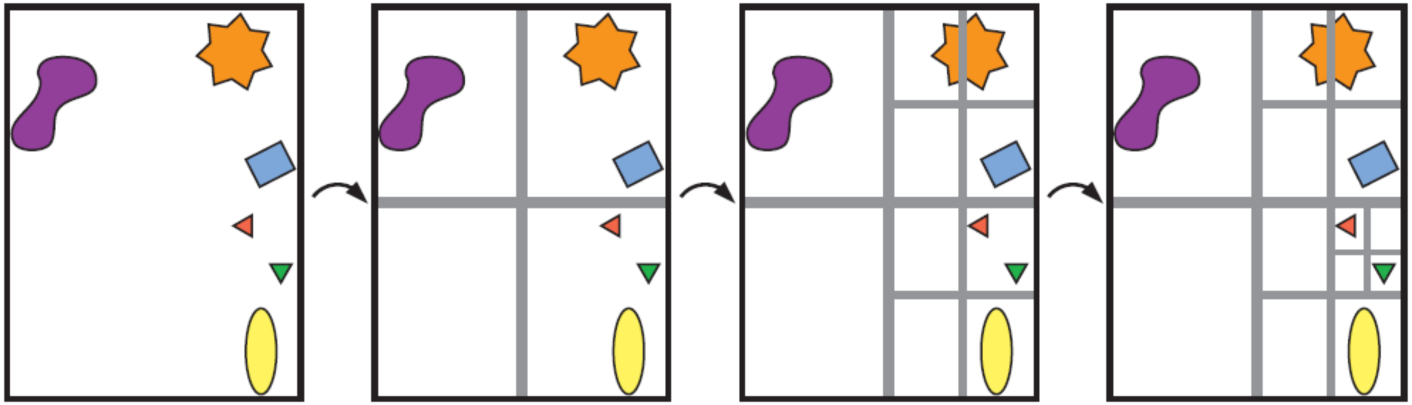
# Spatial Hierarchies

- Leaves correspond to regions of space
- Leaves contain list of all objects in that region



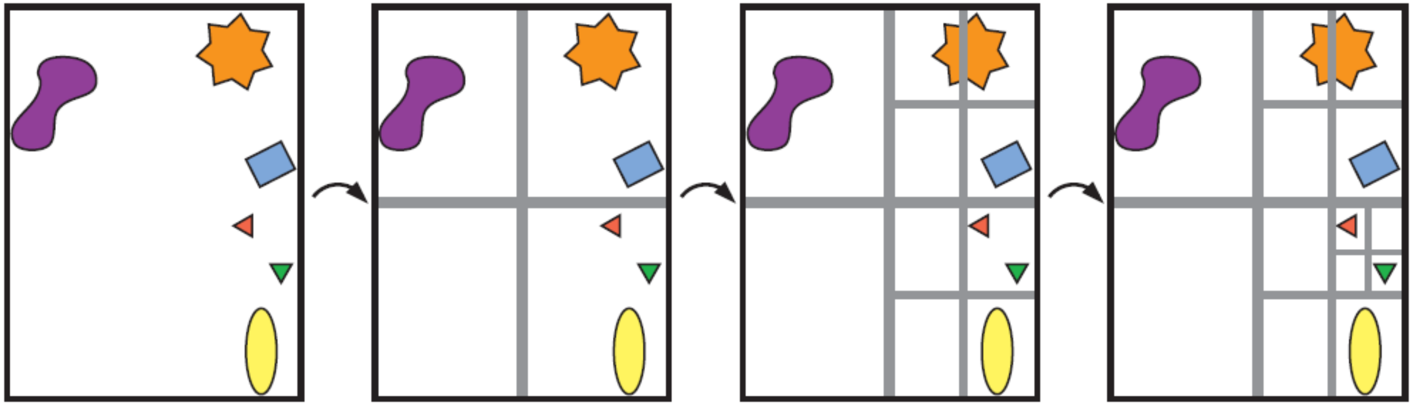
# Octrees

- Start with axis aligned bounding box
  - If termination criteria not reached
  - Recursively split current cell into octants

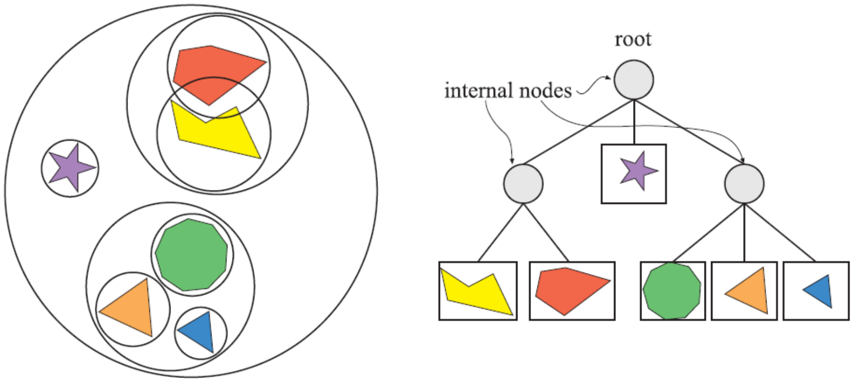


# Flipped Octree

- Construct fine uniform grid using $2^k$ cells along each axis
  - Use sparse data structure: hash table
- We will have k+1 levels in the octree
- For occupied cell at level k
  - Right shift cell indices
  - Add parent cell and any child info at level k-1



# Bounding Volume Hierarchies (BVHs)

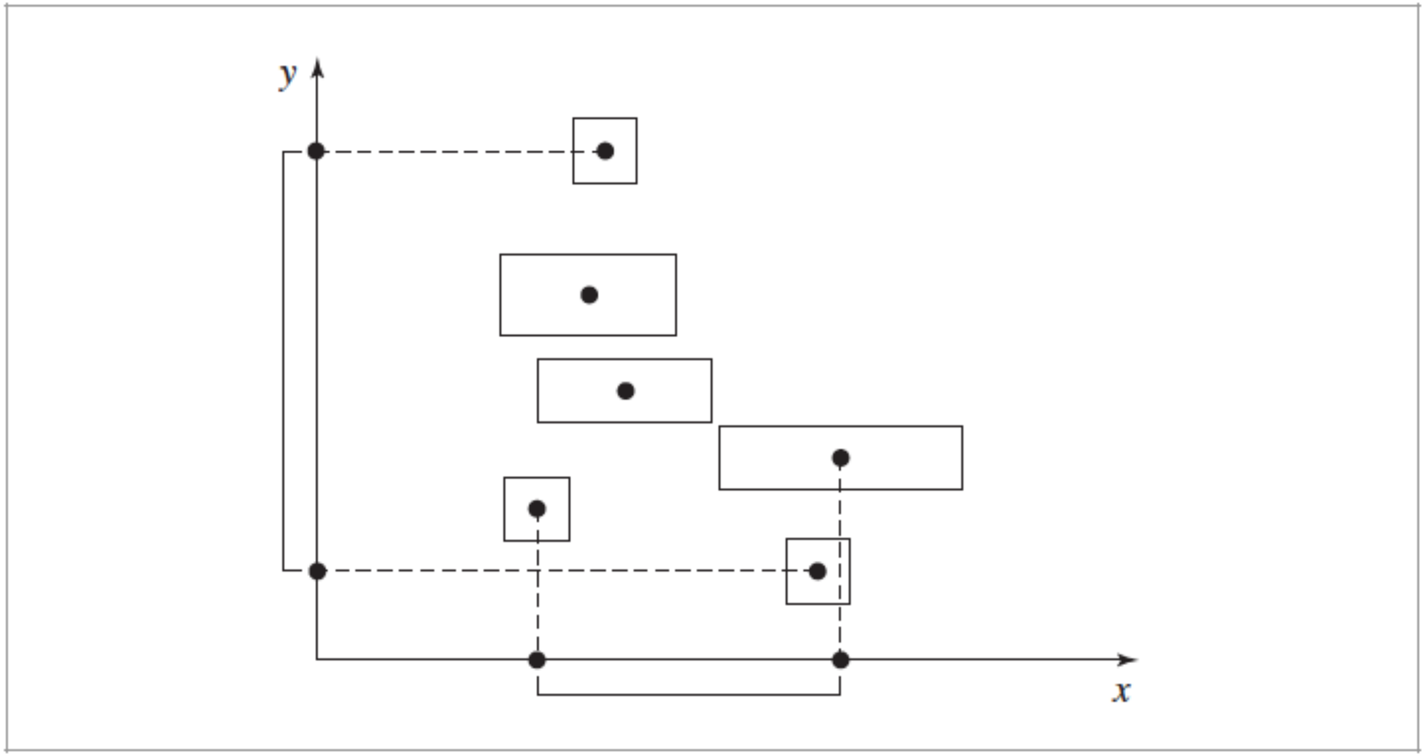


Three stages to construction:

1. Bound each primitive
2. Build a tree using recursive splitting
3. Convert tree to pointerless structure…more compact

- BVH nodes store BV extent plus centroid
- Leaves store primitives
- Each primitive appears in only one node
- If bounding boxes overlap, must traverse more than one subtree
- Make internal nodes compact
  - 8 byte representation means 4 fit in 32 byte cache line
  - In *PBR*, moving from 16 bytes to 8 resulted in 20% speedup

# Which axis to split?

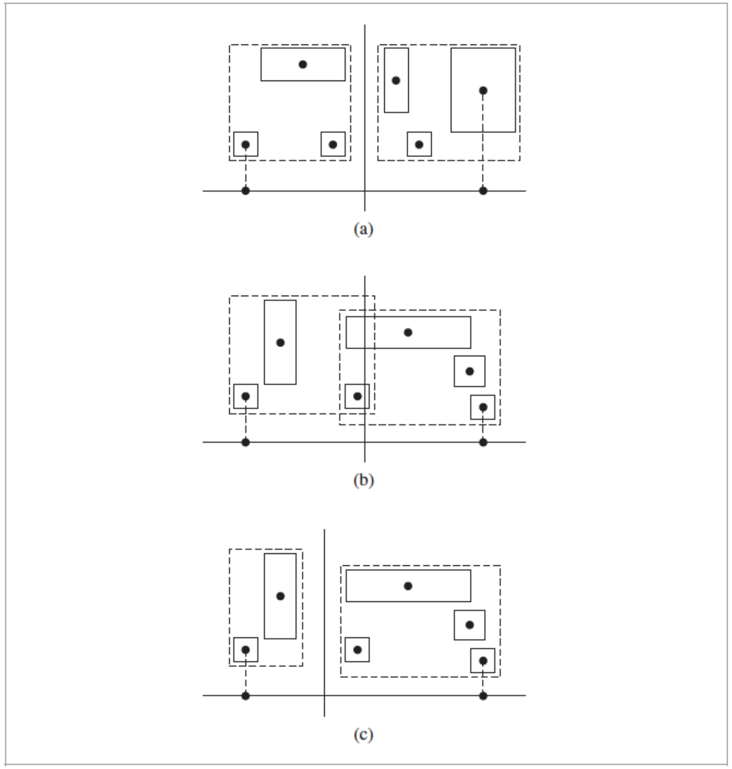


# How to split? Some simple heuristics…

- **Mid-point**
  - Compute mid-point of the primitives centroids
  - Split along a principal axis

- **Split-equal**
  - partition primitive into 2 sets
  - n/2 objects with smallest coordinate centroids
  - n/2 objects with largest coordinate centroids

# Problems with simple solutions



Midpoint works well

Midpoint and Equal  are suboptimal

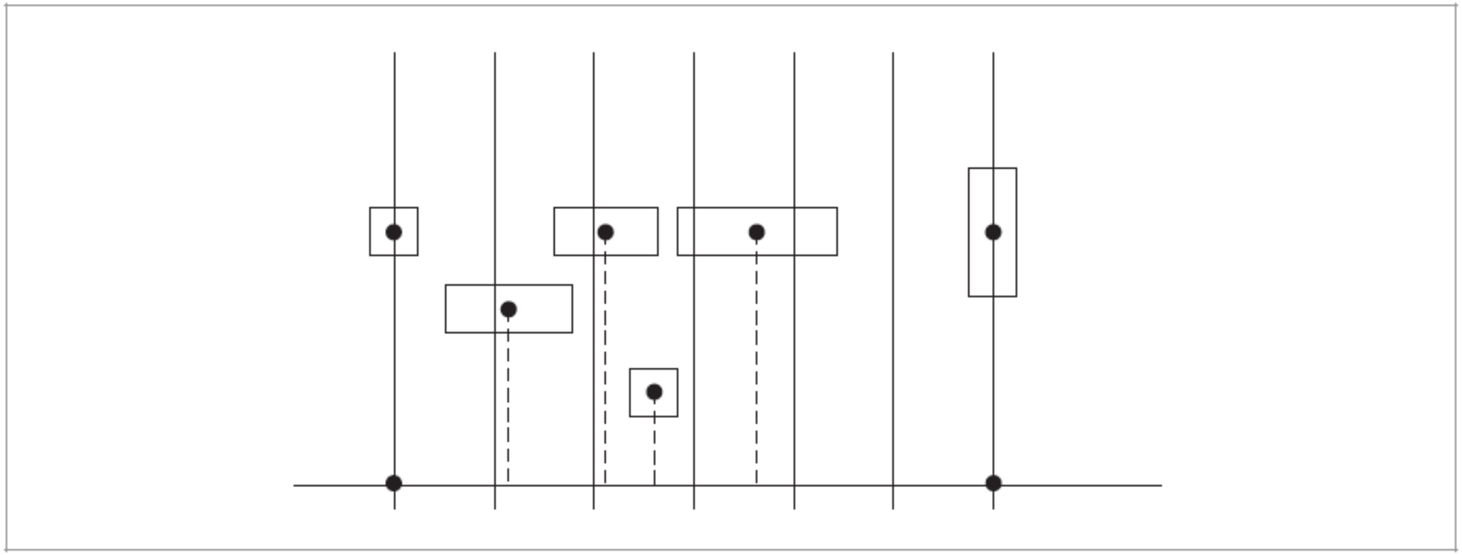
A better split

# Surface Area Heuristic (SAH)

- If we choose not to split a node, the ray-tracing cost is $\sum_{i=1}^{N} t_{\text{isect}}(i)$
- If we split: $c(A, B) = t_{\text{trav}} + p_A \sum_{i=1}^{N_A} t_{\text{isect}}(a_i) + p_B \sum_{i=1}^{N_B} t_{\text{isect}}(b_i),$
- Probability of ray going through convex volume A if it goes through enclosing convex volume W is $S_A/S_W$
  - $S_x$ surface area in volume X…

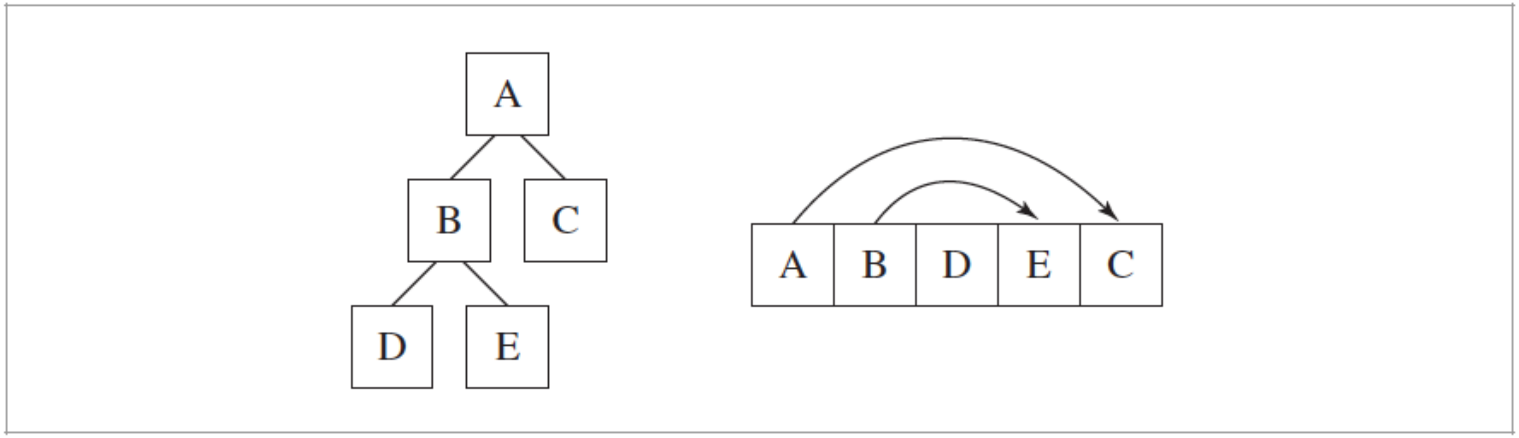
# Surface Area Heuristic (SAH)

- Bucket the axis…measure the cost of splitting at bucket boundaries
  - What would be a more principled alternative? Why not do that?
- Traversal cost 1/8 of intersection cost…intersection cost is 1



# Compact Layout

- Stored in depth-first order
  - First child immediately after parent node
  - Offset to second child stored explicitly



# Traversal of BVHs for Ray Tracing

- Test ray against root BV for intersection…
  - Recurse (but not really…)
  - For a leaf, test against primitives
  - For a parent, test 2 children….