Université Concordia University
FACULTY OF ENGINEERING AND COMPUTER SCIENCE
Department of Computer Science and Software Engineering

COMP371
COMPUTER GRAPHICS

LECTURE 17
SPATIAL DATA STRUCTURES

Lecture Overview

- Review of last class
- Spatial Data Structures
  - Hierarchical Bounding Volumes
  - Regular Grids
  - Octrees
  - BSP Treea

Ray Tracing Acceleration

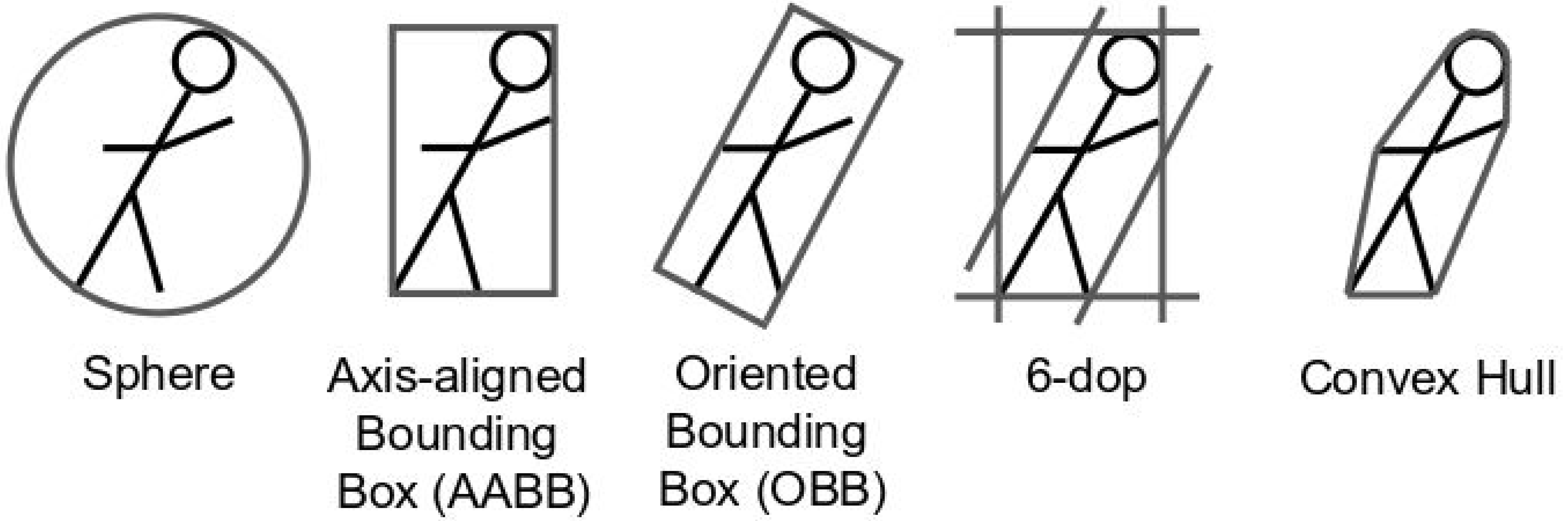
- **Faster intersections**
  - Faster ray-object intersections
    - Object bounding volume
    - Efficient computation of intersections
  - <u>Fewer ray-object intersections</u>
    - Hierarchical bounding volumes (boxes, spheres)
    - Spatial data structures
    - Directional techniques
- **Fewer rays**
  - Adaptive tree-depth control
  - Stochastic sampling
- **Generalized rays (beams, cones)**

Spatial Data Structures

- Data structures to store geometric information
- Sample applications
  - Collision detections
  - Location queries
  - Simulations
  - Rendering
- Spatial data structures for ray tracing
  - Object-centric data structures (bounding volumes)
  - Space subdivision (grids, octrees, BSP trees)
  - Speed-up of 10x, 100x, or more

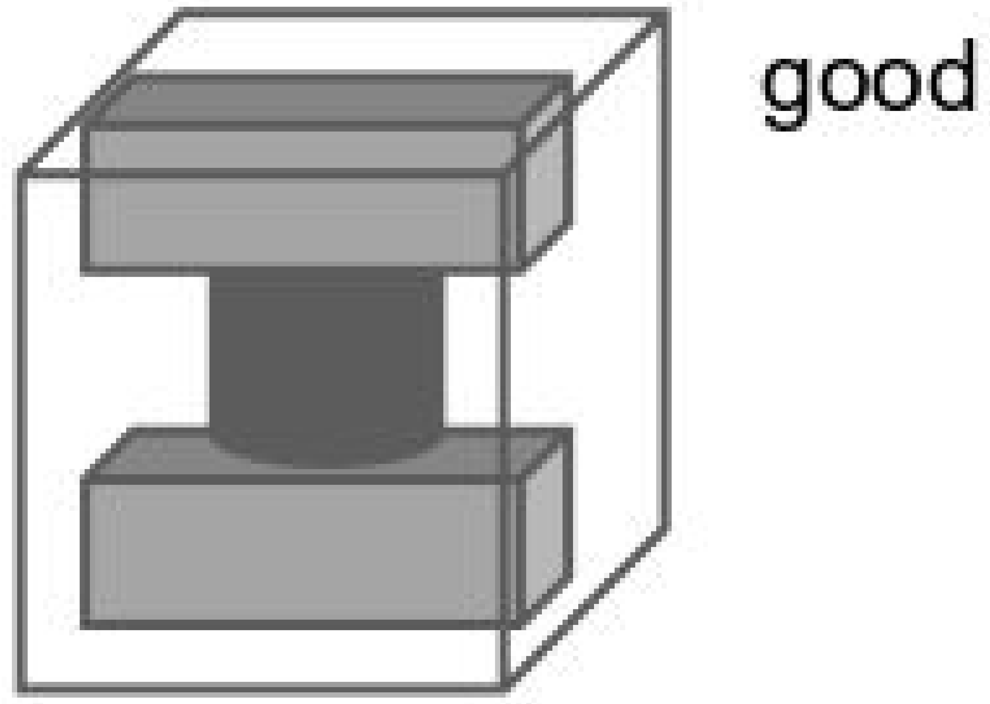
Bounding Volumes

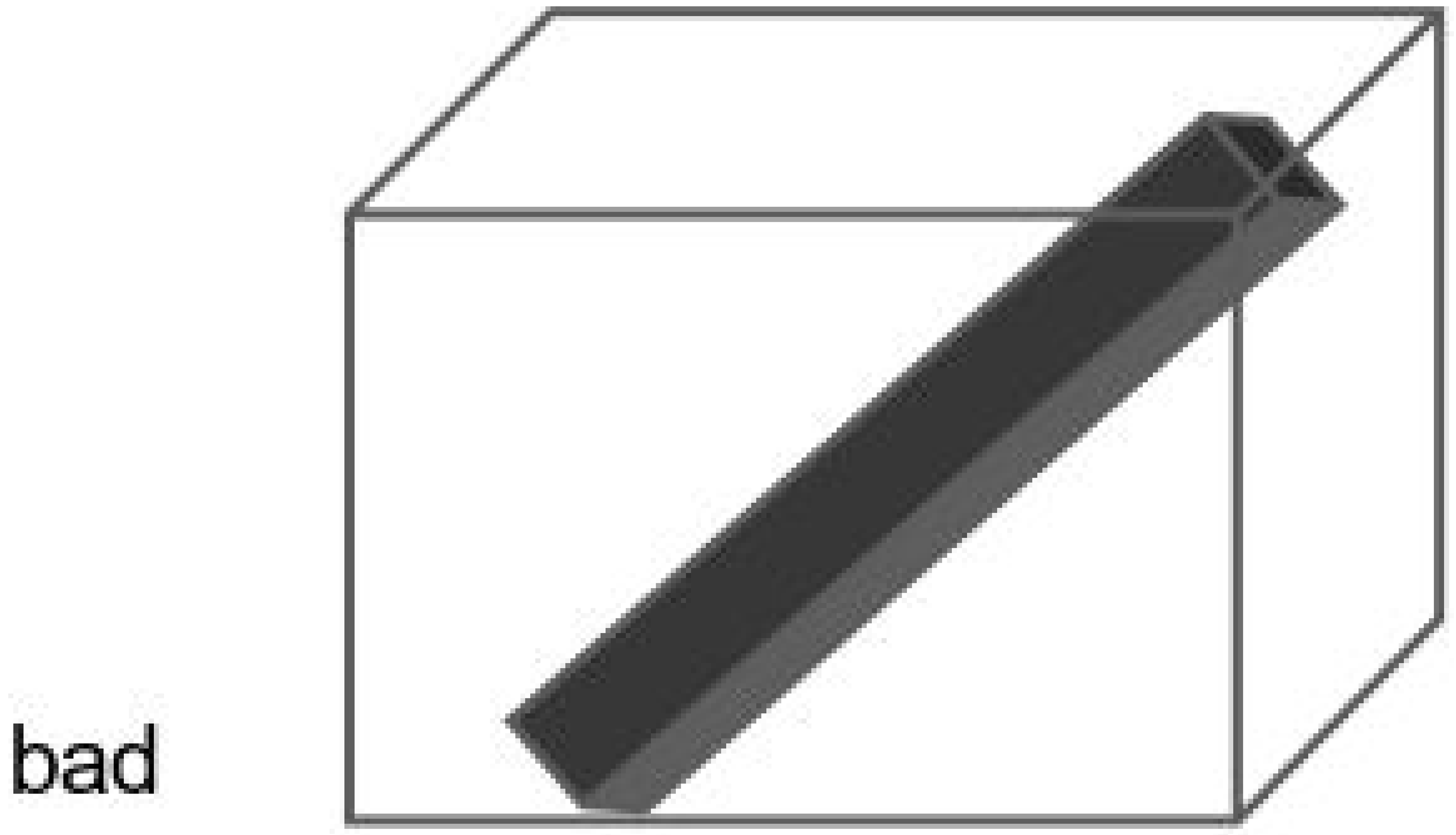
- Wrap complex objects in simple ones
- Does ray intersect bounding box?
  - No: does not intersect enclosed objects
  - Yes: calculate intersection with closest objects
- Common types:



Selection of Bounding Volumes

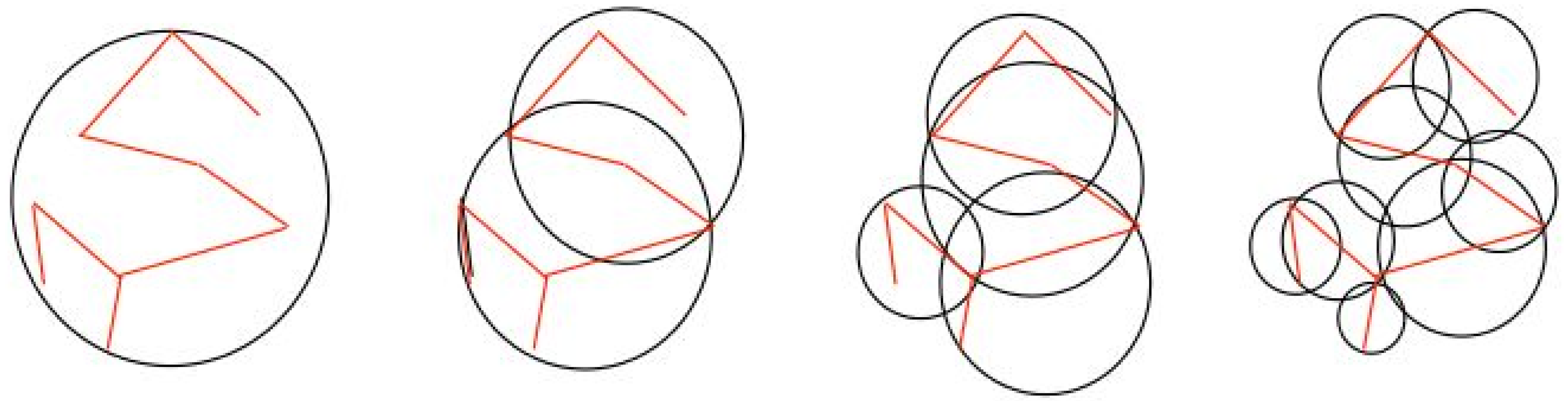
- Effectiveness depends on:
  - Probability that ray hits bounding volume, but not enclosed objects (tight fit is better)
  - Cost of calculating intersections with bounding volume and enclosed objects
- Break-down into steps the calculations of bounding volumes
- Use heuristics





Hierarchical Bounding Volumes

- With simple bounding volumes, ray casting still requires O(n) intersection tests
- Idea: use tree data structure
  - Larger bounding volumes contain smaller ones etc
  - Sometimes naturally available (e.g. human figure)
  - Sometimes difficult to compute
- Often reduces complexity to O(log(n))

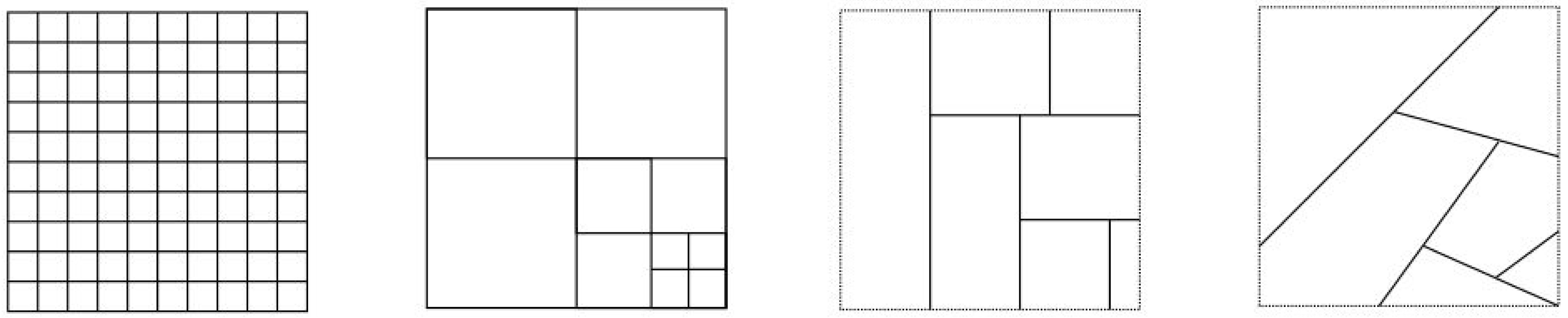


Ray Intersection Algorithm

- Recursively descend down the tree
- If ray misses bounding volume, no intersection
- If ray intersects bounding volume, recurse with the enclosed volumes and objects
- Maintain near and far bounds to prune further
- Overall effectiveness depends on model and constructed hierarchy

Spatial Subdivision

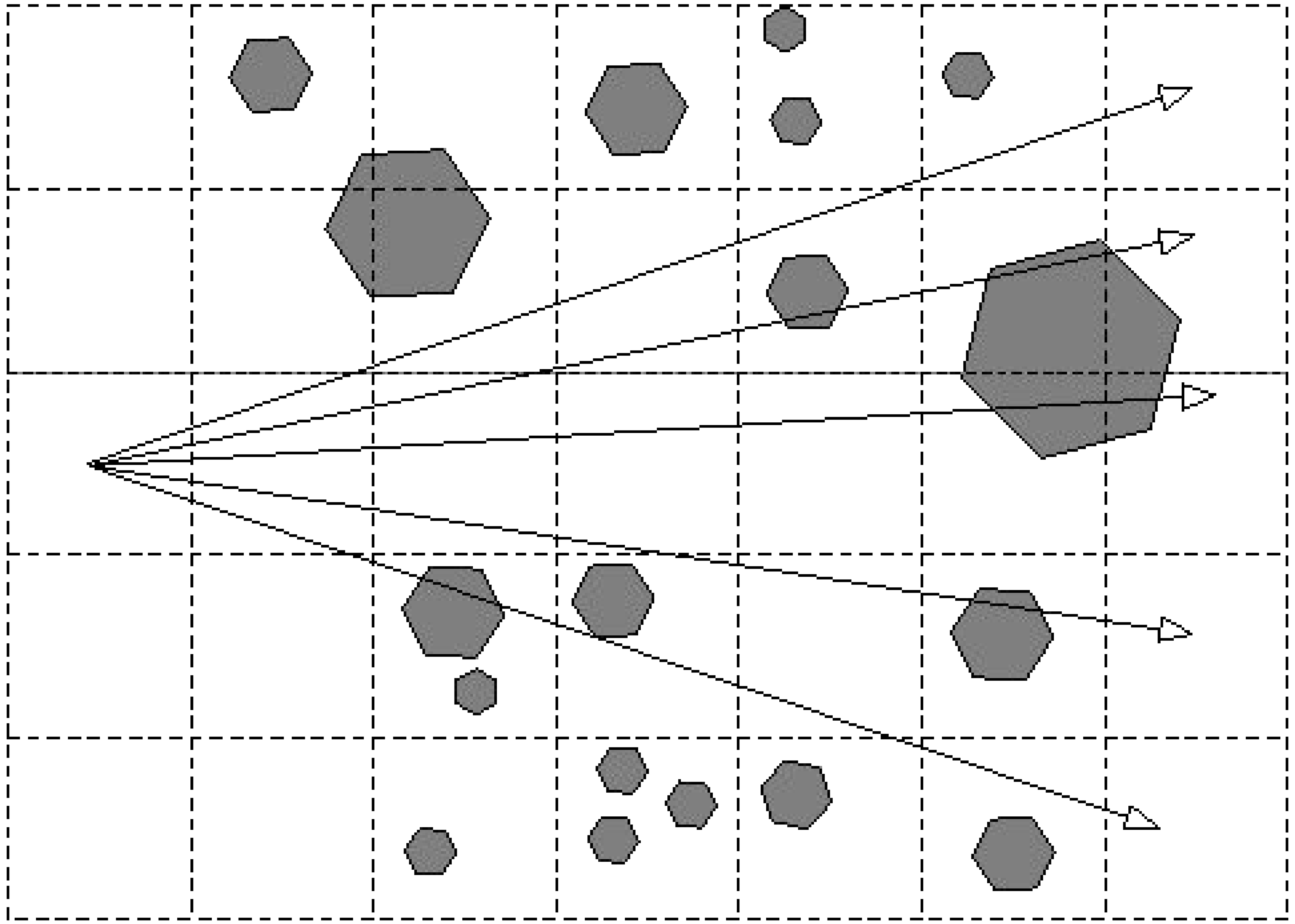
- Bounding volumes enclose objects, recursively i.e.object-centric
- Alternatively, divide space (as opposed to objects)
  - For each segment of space, keep a list of intersecting surfaces or objects
- Basic techniques:



Uniform Spatial Sub   Quadtree/Octree   kd-tree   BSP-tree

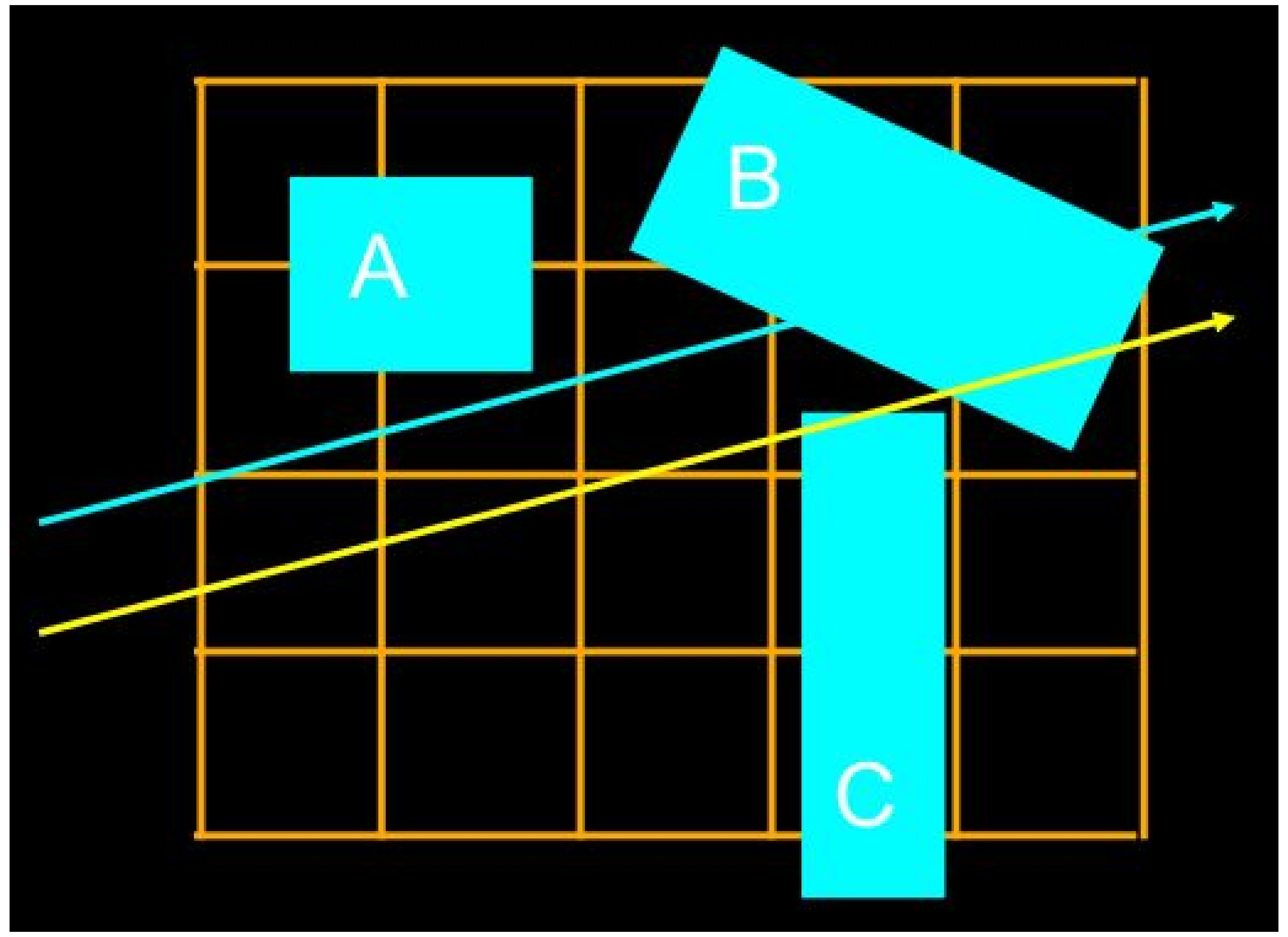
Grids

- 3D array of cells (voxels) that tile space
  - Each cell keeps a list of all all intersecting surfaces
  - Intersection algorithm steps from cell to cell



Caching Intersection Points

- Objects can span multiple cells
- For A need to test intersection only once
- For B need to cache intersection and check next cell for any closer intersection with other objects
  - If not, C could be missed (yellow ray)

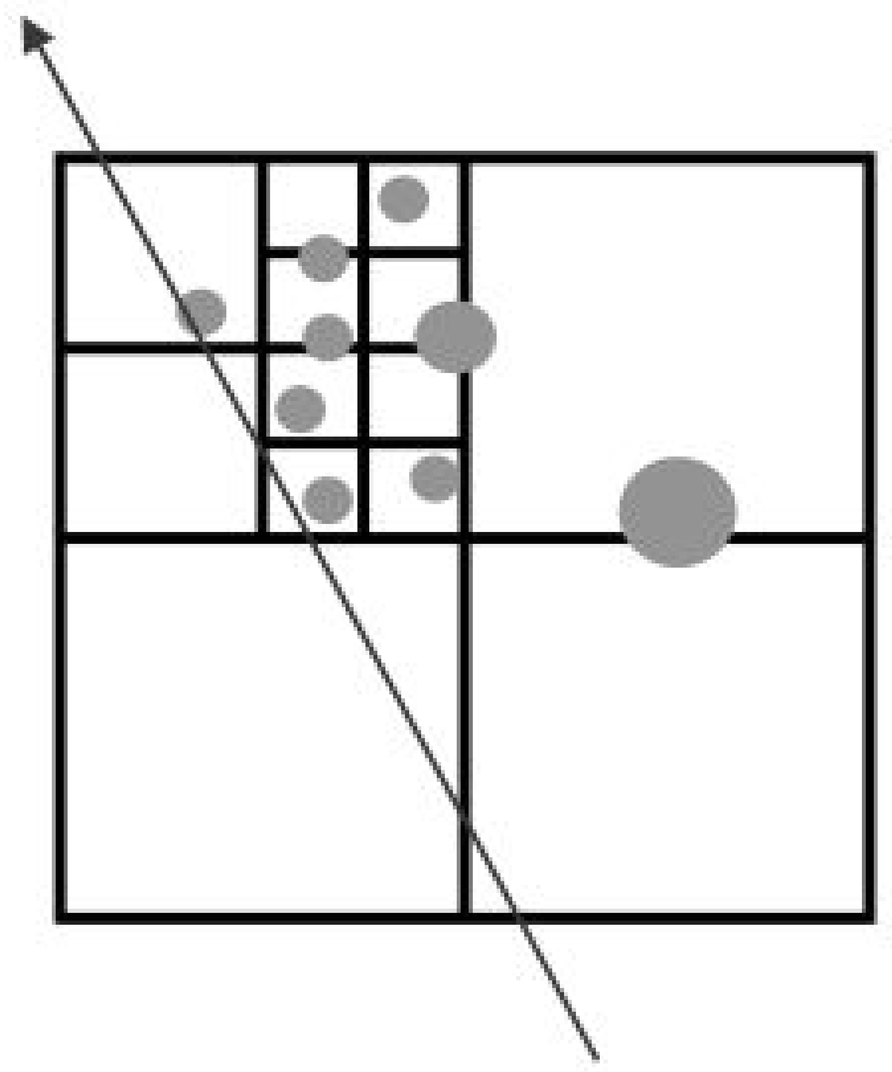


Assessment of Grids

- Poor choice when world is non-homogeneous
- Grid resolution:
  - Too small: too many surfaces per cell
  - Too large: too many empty cells to traverse
  - Can use algorithms like Bresenham's for efficient traversal
- Non-uniform spatial subdivision more flexible
  - Can adjust to objects that are present

Quadtrees

- Generalization of binary trees in 1D
  - Node (cell) is a square
  - Recursively split into 4 equal sub-squares
  - Stop subdivision based on number of objects
- Ray intersection has to traverse quadtree
- More difficult to step to next cell



Octrees

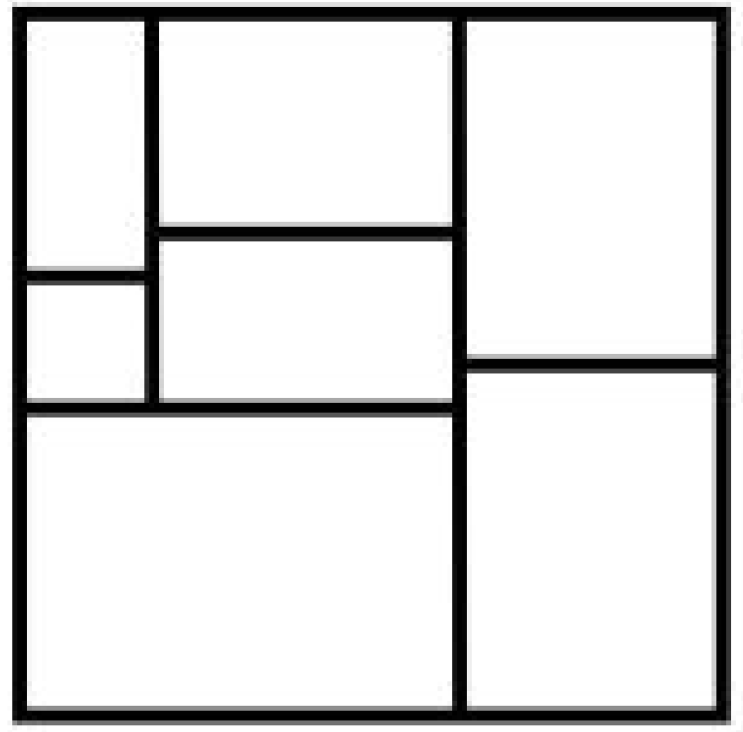
- Generalization of quadtree in 3D
- Each cell may be split into 8 equal sub-cells
- Internal nodes store pointers to children
- Leaf nodes store list of surfaces
- Adapts well to non-homogeneous scenes

Assessment for Ray Tracing

- Grids
  - Easy to implement
  - Require a lot of memory
  - Poor results for non-homogeneous scenes
- Octrees
  - Better on most scenes (more adaptive)
- Alternative: nested grids
- Spatial subdivision expensive for animations
- Hierarchical bounding volumes
  - Natural for hierarchical objects
  - Better for dynamic scenes

Other Spatial Subdivision Techniques

- Relax rules for quadtrees and octrees
- k-dimensional tree (k-d tree)
  - Split at arbitrary interior point
  - Split one dimension at a time
- Binary space partitioning tree (BSP tree)
  - In 2 dimensions, split with any line
  - In k dims. split with k-1 dimensional hyperplane
  - Particularly useful for painter’s algorithm
  - Can also be used for ray tracing

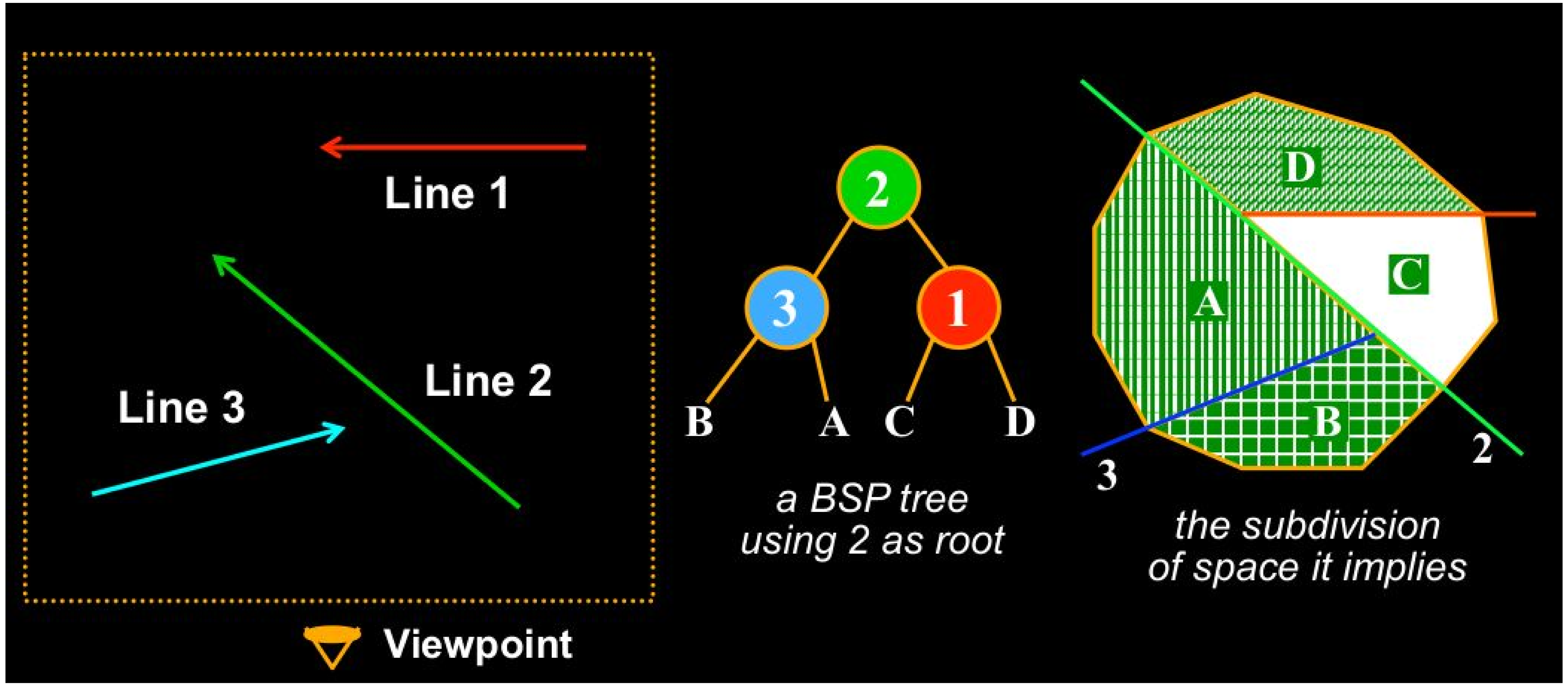


BSP Trees

- Split space with any line (2D) or plane (3D)
- Applications
  - Painter's algorithm for hidden surface removal
  - Ray casting
- Inherent spatial ordering given viewpoint
  - Left subtree: in front, right subtree: behind
- Problem: finding good space partitions
  - Proper ordering for any viewpoint
  - How to balance the tree

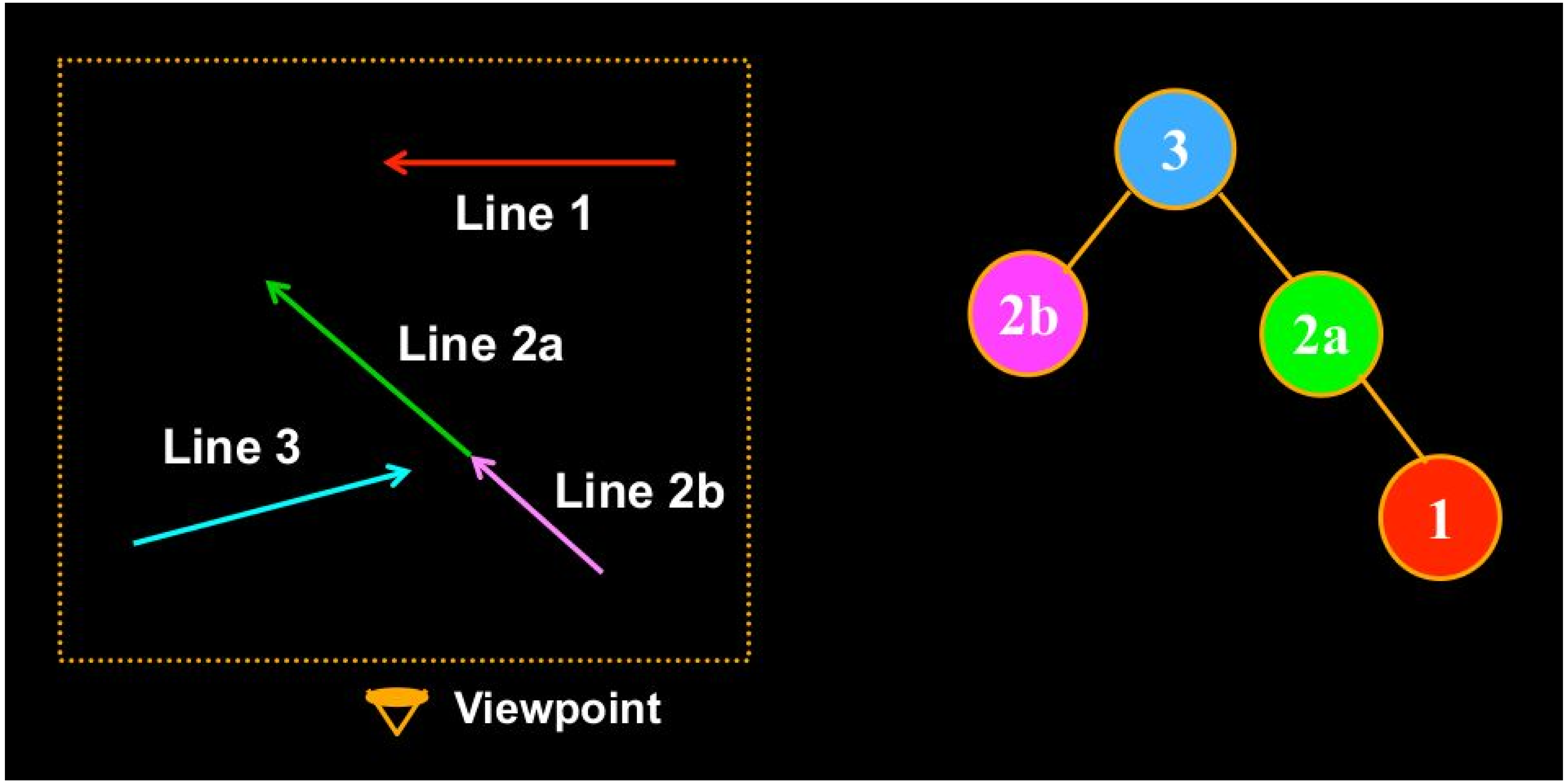
Building a BSP Tree

- Use hidden surface removal as intuition
- Using line 1 or line 2 as root is easy



Splitting of Surfaces

- Using line 3 as root requires splitting



Building a Good Tree

- Naive partitioning of n polygons yields $O(n^3)$ polygons (in 3D)
- Algorithms with $O(n^2)$ increase exist
  - Try all, use polygon with fewest splits
  - Do not need to split exactly along polygon planes
- Should balance tree
  - More splits allow easier balancing
  - Rebalancing?

Painter's Algorithm with BSP Trees

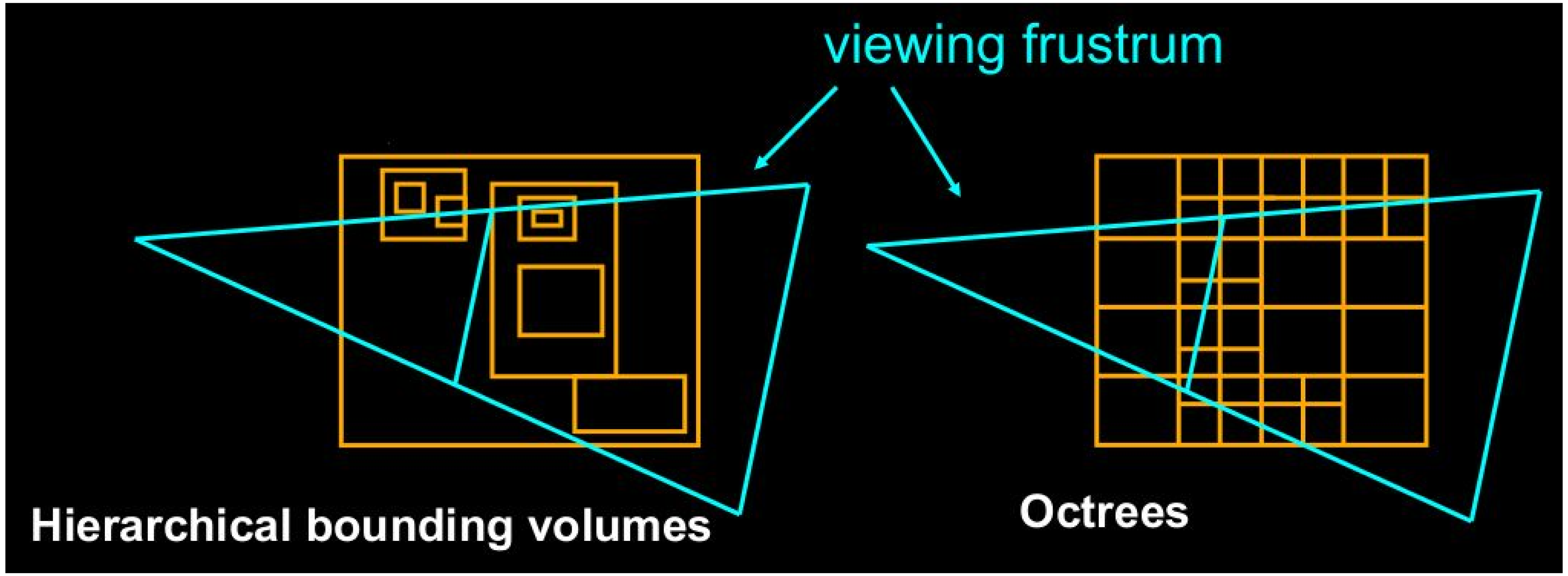
- Building the tree
  - May need to split some polygons
  - Slow, but done only once
- Traverse back-to-front or front-to-back
  - Order is viewer-direction dependent
  - What is front and what is back of each line changes
  - Determine order on the fly

Details of Painter's Algorithm

- Each plane has form Ax + By + Cz + D
- Plug in coordinates and determine
  - Positive: front side
  - Zero: on plane
  - Negative: back side
- Back-to-front: inorder traversal, farther child first
- Front-to-back: inorder traversal, near child first
- Do backface culling with same sign test
- Clip against visible portion of space (portals)

Clipping with Spatial Data Structures

- Accelerate clipping
  - Goal accept or reject whole sets of objects
  - Can use any spatial data structures
- Scene should be mostly fixed
  - Terrain fly-through
  - Gaming



Data Structures Demos

BSP Tree construction

http://symbolcraft.com/graphics/bsp/index.html

KD Tree construction

http://donar.umiacs.umd.edu/quadtree/points/kdtree.html

Real-Time and Interactive Ray Tracing

Interactive ray tracing via space subdivision

http://www.cs.utah.edu/~reinhard/egwr/

State of the art in interactive ray tracing

http://www.cs.utah.edu/~shirley/irt/

Review

- Spatial Data Structures

Next Lecture

- Keyframe Animation

CONCORDIA.CA

Studying Support Slides