Bounding Volume Hierarchies and Spatial Partitioning

Kenneth E. Hoff III

COMP-236 lecture
Spring 2000

Introduction

- Bounding Volume Hierarchies vs. Spatial Partitioning
  - What are they and how do they compare?
- Motivation: *Need for Speed!*
  - Demonstration through applications:
    View-frustum culling, ray-tracing, collision detection
- How can hierarchies help?
  - Apply to example applications
- Building bounding volume hierarchies
- Building spatial partitionings
- What's the best choice?
- Can we do better?

What are they? How do they Compare?

Bounding Volume Hierarchies

- Hierarchical object representation
- Object subdivision
- Hierarchical clustering of objects
- Object levels of detail
- Classifies regions of space around objects

Examples:

- OBB-trees
- AABB-trees
- Sphere-trees
- k-DOPs

Spatial Partitioning

- Hierarchical spatial representation
- Spatial subdivision
- Hierarchical clustering of space
- Spatial levels of detail
- Classifies objects around regions of space

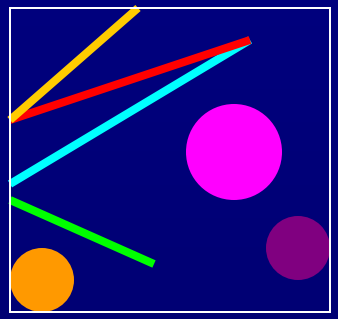
Examples:

- Uniform grids
- Quadtrees & Octrees
- BSP-trees
- kD-trees

Examples

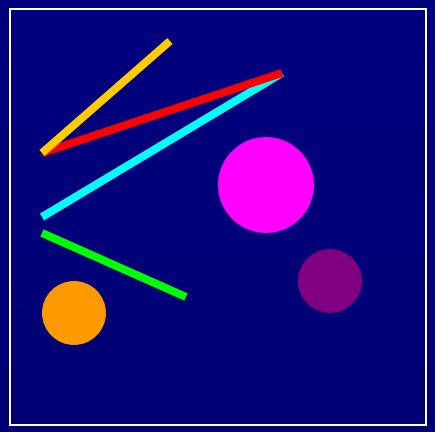
Bounding Volume Hierarchies

- Tightly fits objects
- Redundant spatial representation



Spatial Partitioning

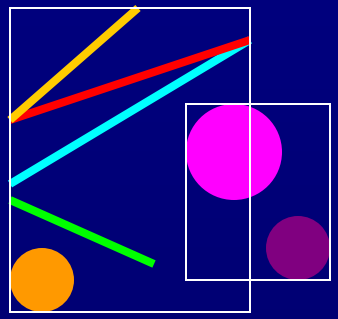
- Tightly fills space
- Redundant object representation



Examples

Bounding Volume Hierarchies

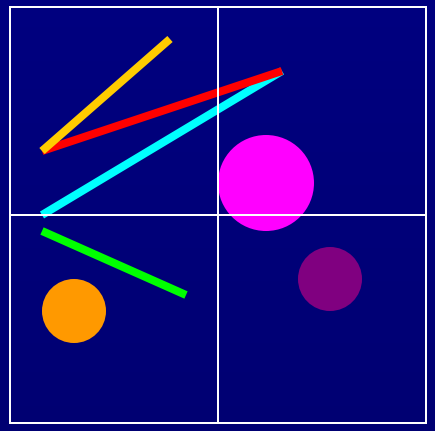
- Tightly fits objects
- Redundant spatial representation



Volumes overlap multiple objects

Spatial Partitioning

- Tightly fills space
- Redundant object representation

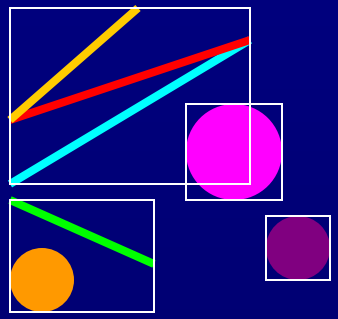


Objects overlap multiple volumes

Examples

Bounding Volume Hierarchies

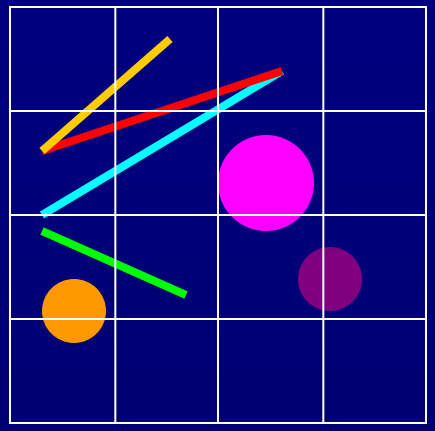
- Tightly fits objects
- Redundant spatial representation



Volumes overlap multiple objects

Spatial Partitioning

- Tightly fills space
- Redundant object representation

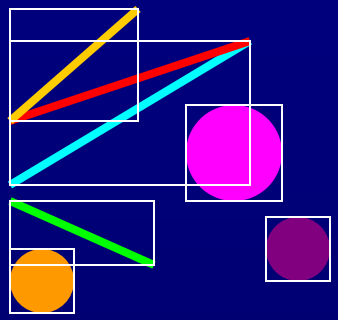


Objects overlap multiple volumes

Examples

Bounding Volume Hierarchies

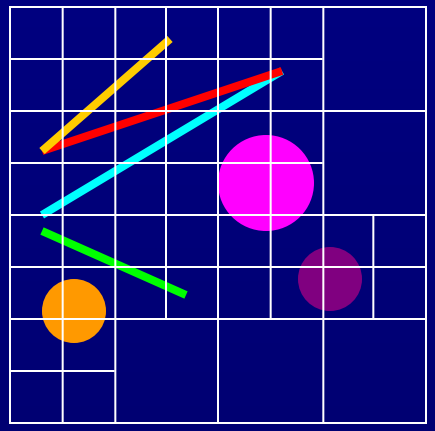
- Tightly fits objects
- Redundant spatial representation



Volumes overlap multiple objects

Spatial Partitioning

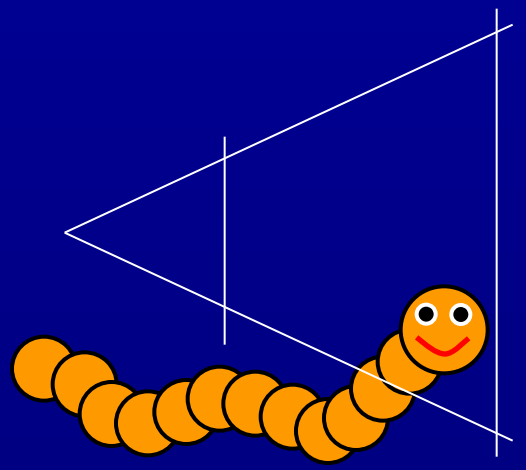
- Tightly fills space
- Redundant object representation



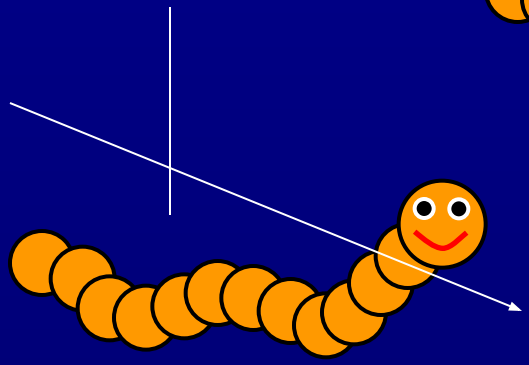
Objects overlap multiple volumes

Motivation: Example Applications

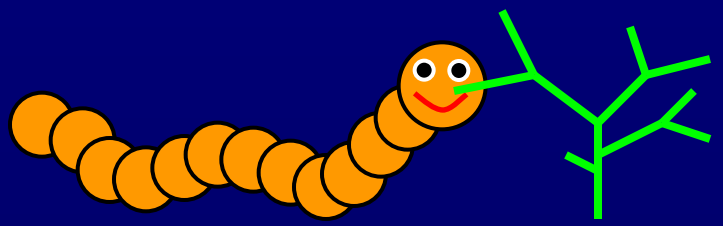
View-frustum culling
O(n)



Ray-tracing
O(n) per ray



Collision detection
$O(n^2)$

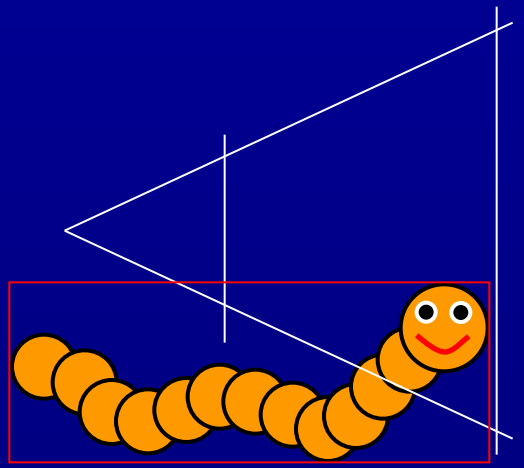


How do we speed it up?

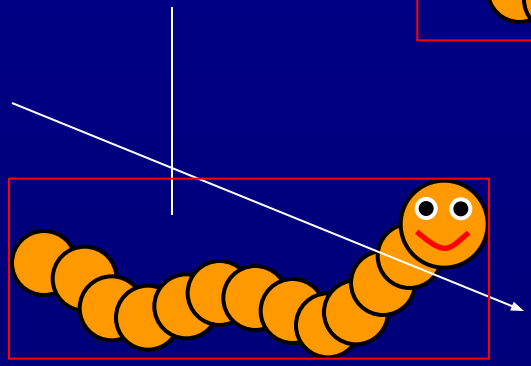
- More efficient intersection calculations
- Avoid intersection calculations
  - Make a single intersection calculation to decide for an entire cluster of objects or space
  - Cluster hierarchically

How can bounding volume hierarchies help?

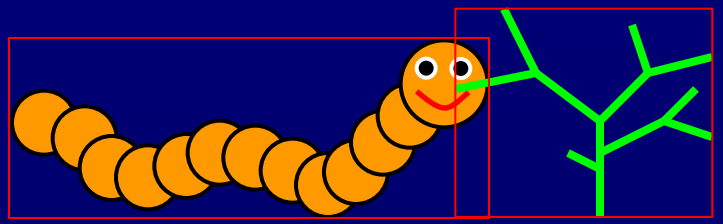
View-frustum culling



Ray-tracing

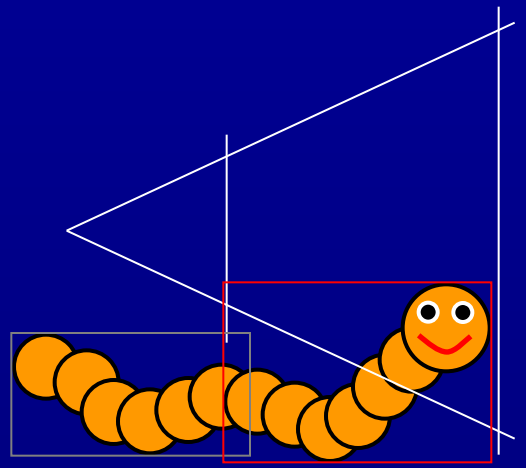


Collision detection

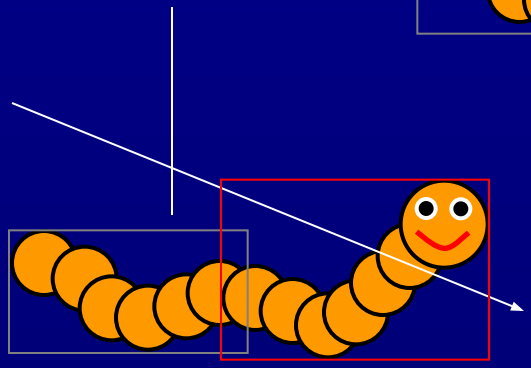


How can bounding volume hierarchies help?

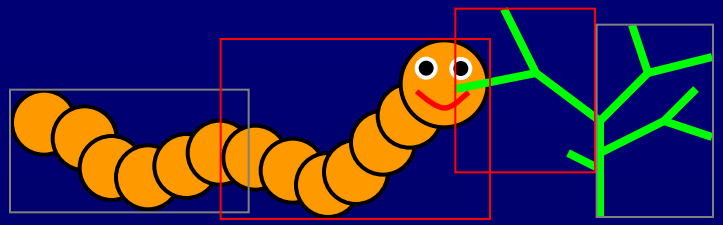
View-frustum culling



Ray-tracing

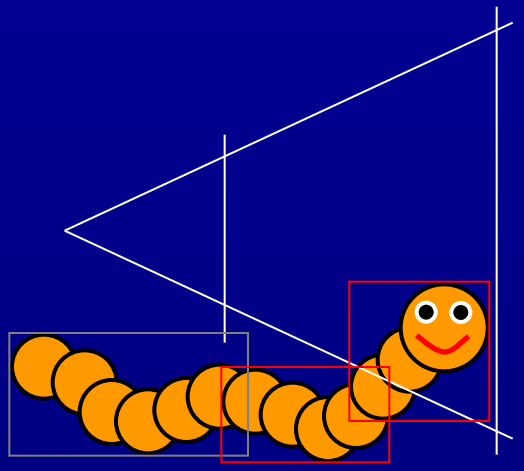


Collision detection

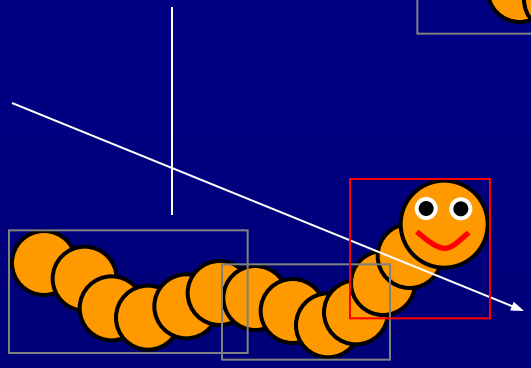


How can bounding volume hierarchies help?

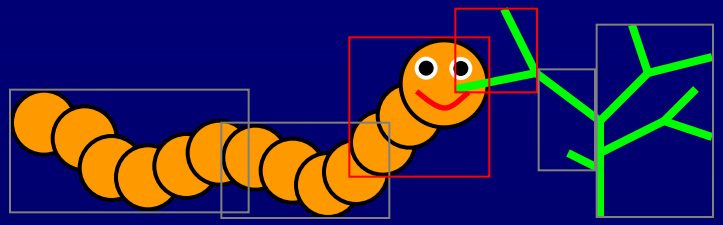
View-frustum culling



Ray-tracing

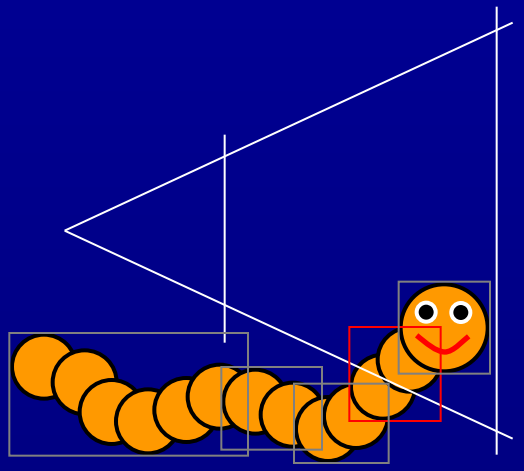


Collision detection

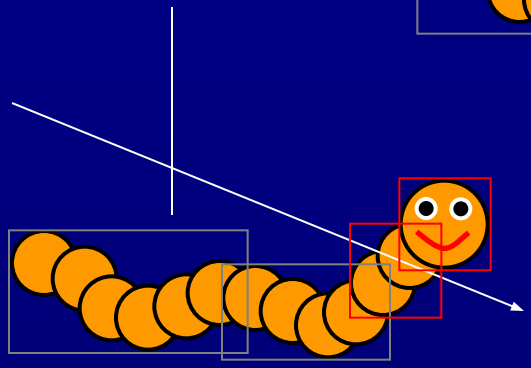


How can bounding volume hierarchies help?

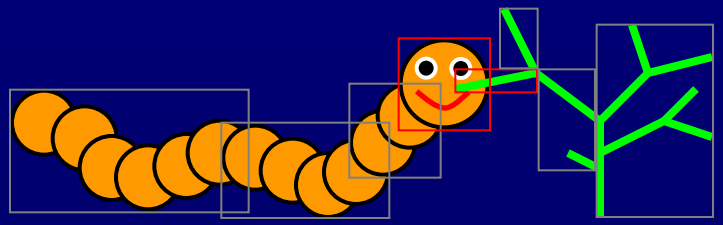
View-frustum culling



Ray-tracing



Collision detection

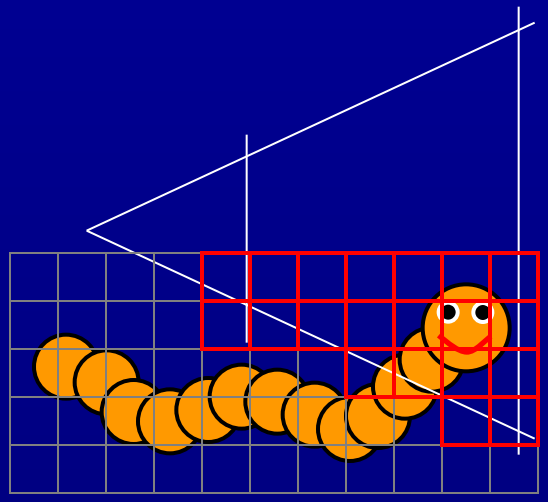


How can bounding volume hierarchies help?

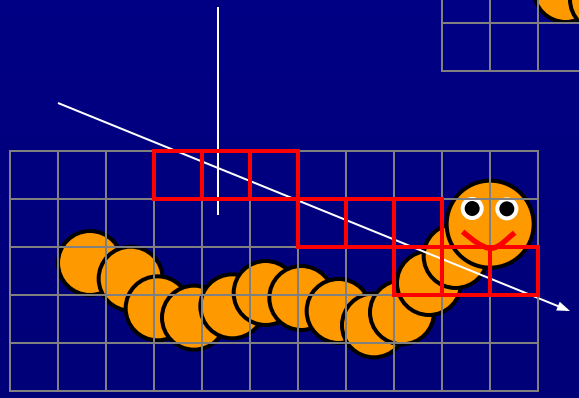
Logarithmic search for intersecting primitives!

How can spatial partitioning help?

View-frustum culling

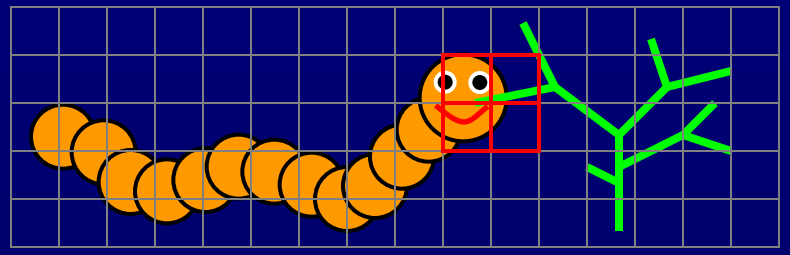


Ray-tracing



Uniform spatial partitioning

Collision detection



How can spatial partitioning help?

Performance varies for uniform partitioning, but hierarchical approaches also give logarithmic search for intersecting primitives!

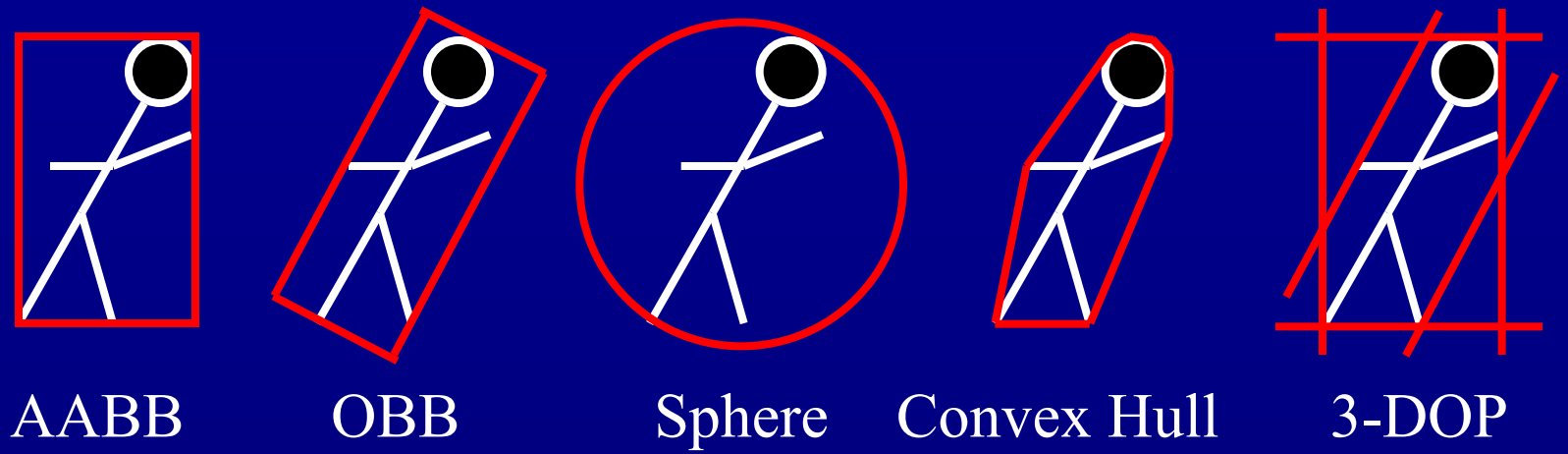
What are the potential problems?

- What are the hidden costs?
  - When nothing intersects?
  - When nearly everything intersects?
  - What are the worst cases?
- Is it worth it?
- What applications get the most benefit?
- What about just using my modeling hierarchy?
  - Too shallow (not fine enough level of detail
  - Designed for object manipulation rather than minimizing intersections
  - Insensitive to actual positions of objects

Building Bounding Volume Hierarchies

- Choose a bounding volume type
  - Axis-aligned bounding box (AABB)
  - Oriented bounding box (OBB)
  - Sphere
  - Convex Hull
  - k-DOPs
- Choose a clustering strategy
  - Top-down:
    how do we partition objects among children?
  - Bottom-up:
    how do we find leaf clusters and merge into parents?

Bounding Volume Type



- Intersection cost vs. tightness of fit vs. storage overhead vs. implementation complexity
- How do we find the best fit for a particular bounding volume?
  - AABBs and convex hulls are clear.
    What about spheres, k-DOPs, and OBBs?
- How do we compare the quality of fit between different BVs?
  - Min volume, min surface area, etc.

Hierarchical Clustering Strategy

Top-down: how do we partition objects among children?

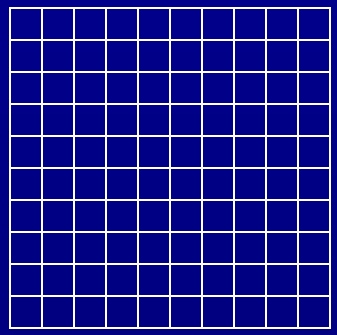
- Choosing splitting axis
  - longest dimension, largest spread of objects, etc.
- Choosing split point
  - mean, median, largest gap, etc.

Bottom-up: how do we find leaf clusters and merge into parents?

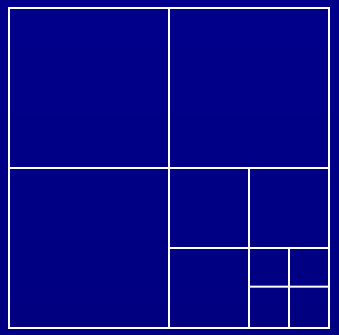
- Leaf object clusters
  - single primitive, specific minimum size cluster, etc.
- Merging children into parent
  - Nearest neighbors: uniform subdivision, Voronoi diagram

Building Spatial Partitionings

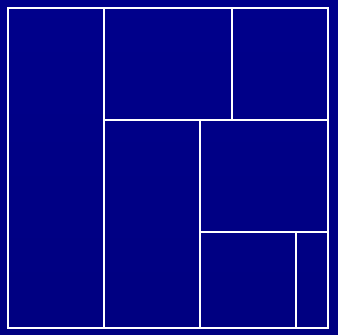
Decide how to recursively subdivide space (top-down)



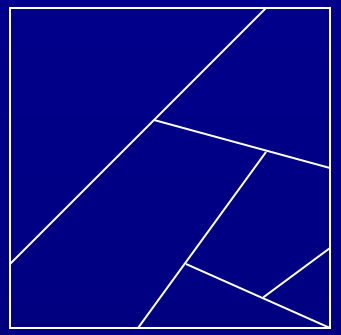
Uniform subdivision



Quadtree

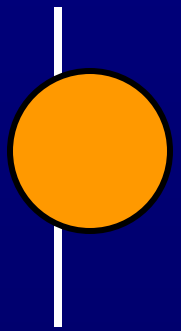


kD-tree



BSP-tree

Decide how to classify objects into regions of space with respect to partitioning plane



Store in both regions,
Store with partition, or
Split geometry

What's the best choice?

- Depends on the application
  - trial and error?
  - "Gut" feeling?
  - Careful analysis based on a cost function?
- Factors:
  - Complexity of implementation
  - Storage overhead
  - Computational overhead
  - Type of geometry: static or dynamic

Can we do better?

- Combining bounding volume hierarchies and spatial partitioning
  - Examples:
    - Occlusion culling: octrees of BSP-trees
    - Radiosity: 3D BSP-trees of 2D BSP-trees
- Hybrid bounding volume hierarchies
  - adaptive nodes
  - adaptive trees
  - performance driven metrics

Conclusion

- These hierarchical data structures are fundamental in a wide variety of graphics problems
  - most common way of obtaining high performance
- Important to be very familiar with the possible variations
  - these structures appear everywhere!
- Despite tremendous amount of previous publications, still lots of room for further research
  - warning: difficult problems ahead!

References

- Andrew Glassner
  An Introduction to Ray Tracing
- David Rogers
  Procedural Elements for Computer Graphics
- Tomas Möller and Eric Haines
  Real-Time Rendering
- Alan and Mark Watt
  Advanced Animation and Rendering Techniques
- Foley, van Dam, Feiner, and Hughes
  Computer Graphics: Principles and Practice
- Stefan Gottschalk's Dissertation: OBB-tree