Spatial Data Structures

Hierarchical Bounding Volumes
Regular Grids
Octrees
BSP Trees
Constructive Solid Geometry (CSG)
[Angel 9.10]

Outline

- Ray tracing review – what rays matter?
- Ray tracing speedup
  - faster intersection tests: simple enclosing geometry
    - bounding volumes
  - fewer intersection tests: avoid testing many objects
    - hierarchical bounding volumes
    - regular grid
    - octree
    - BSP tree
- CSG and ray tracing

Spatial Data Structures

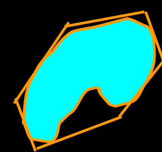
- Data structures to store geometric information
- Sample applications
  - Height field representation
  - Collision detection (hierarchical bounding volumes)
  - Surgical simulations (finite element method)
  - Rendering
- Spatial data structures for ray tracing
  - Object-centric data structures (bounding volumes)
  - Space subdivision (grids, octrees, BSP trees)
  - Speed-up of 10x, 100x, or more

Bounding Volumes

- Suppose you are ray tracing teapots...
  - need to intersect ray with a collection of Bezier patches...
  - ...or a large number of triangles

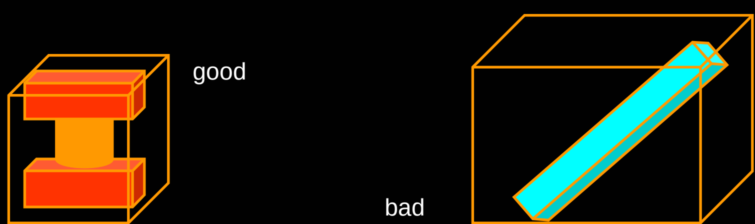
Bounding Volumes

- Wrap complex objects in simple ones
- Does ray intersect bounding box?
  - No: does not intersect enclosed objects
  - Yes: calculate intersection with enclosed objects
- Common types
  - Boxes, axis-aligned
  - Boxes, oriented
  - Spheres
  - Finite intersections or unions of above



Selection of Bounding Volumes

- Effectiveness depends on:
  - Probability that ray hits bounding volume, but not enclosed objects (tight fit is better)
  - Expense to calculate intersections with bounding volume and enclosed objects
- Use heuristics / your best judgment



Hierarchical Bounding Volumes

- With simple bounding volumes, ray casting still requires O(n) intersection tests...

Hierarchical Bounding Volumes

- With simple bounding volumes, ray casting still has 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 tree
- If ray misses bounding volume, no intersection
- If ray intersects bounding volume, recurse with enclosed volumes and objects
- Maintain near and far bounds to prune further
- Overall effectiveness depends on model and constructed hierarchy

Focus on Objects

- Bounding volumes are object centric
  - place simple bounding volume around each object
  - group these simple bounding volumes into a hierarchy

Focus on Objects

- Bounding volumes are object centric
  - place simple bounding volume around each object
  - group these simple bounding volumes into a hierarchy
- Problems:
  - finding a good grouping can be difficult
  - if objects are moving, a group that is compact now may not be compact later ..
  - (logical groupings such as an object or animated character, however, are easy to group and maintain)

Focus on Objects → Focus on the Space

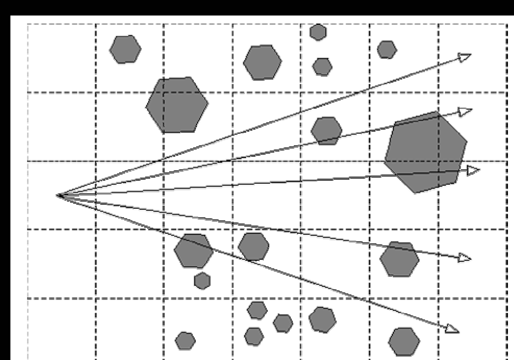
- Bounding volumes are object centric
  - place simple bounding volume around each object
  - group these simple bounding volumes into a hierarchy
- Problems:
  - finding a good grouping is difficult
  - if objects are moving, a group that is compact now may not be compact later..
- If there are many distinct objects, dividing up space and registering objects in that space may be a better option

Spatial Subdivision

- Regular grids
- Octrees
- BSP trees

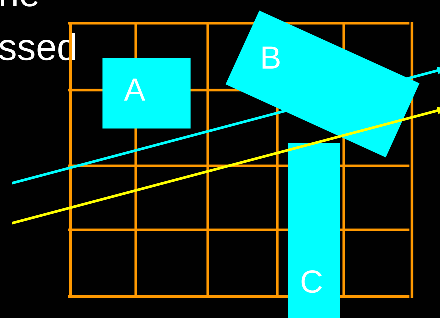
Grids

- 3D array of cells (voxels) that tile space
- Each cell points to 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 closer one
- If not, C could be missed (yellow ray)



Assessment of Grids

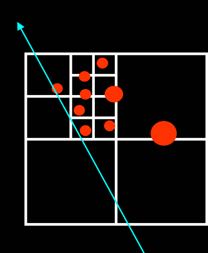
- Poor choice when world is non-homogeneous
- Size of grid
  - Too small: too many surfaces per cell
  - Too large: too many empty cells to traverse
  - Can use alg like Bresenham's for efficient traversal
- Non-uniform spatial subdivision more flexible
  - Can adjust to objects that are present

Quadtrees

- Goal: a hierarchical subdivision of an entire (bounded) 2D space

Quadtrees

- Generalization of binary trees to 2D
  - 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 to 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 inhomogeneous scenes

Assessment for Ray Tracing

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

BSP Trees

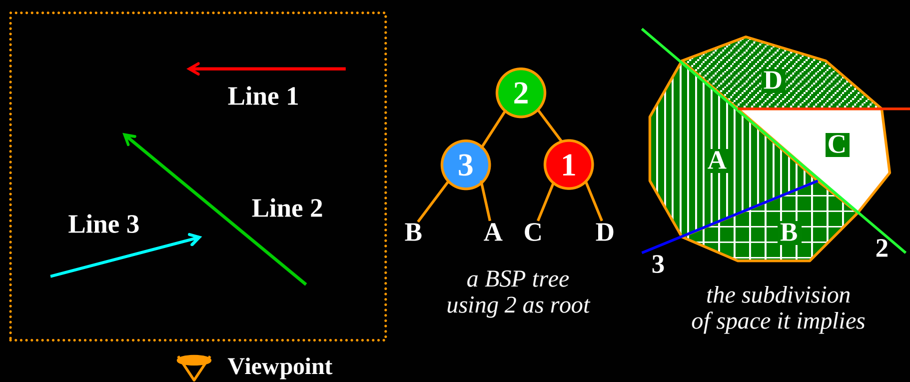
- Binary space partitioning
- Goal: divide space in a more efficient way, with results depending on the particular scene

BSP Trees

- Split space with any line (2D) or plane (3D)
- Applications
  - Painters 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
  - Balance tree

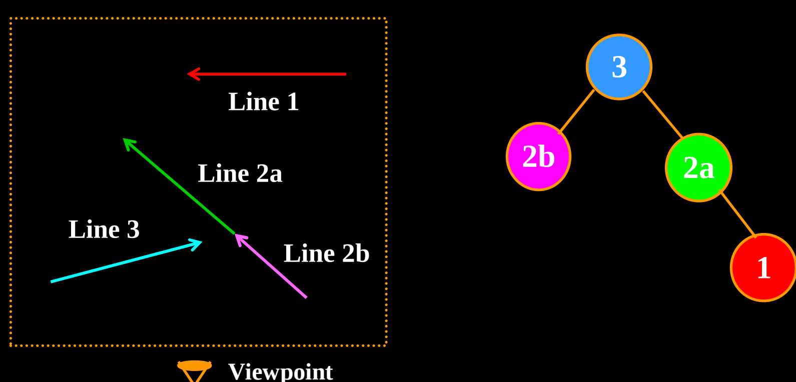
Building a BSP Tree

- 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



Painter’s Algorithm with BSP Trees

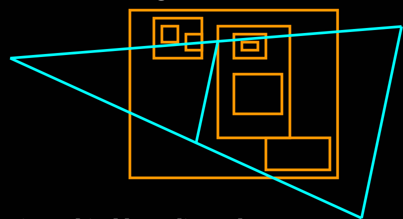
- Building the tree
  - May need to split some polygons
  - Slow, but done only once
- Traverse back-to-front or front-to-back
  - Order is viewer-direction dependent
  - What is front and what is back of each line changes
  - Determine order on the fly
- 2D example of traversal

Details of Painter’s Algorithm

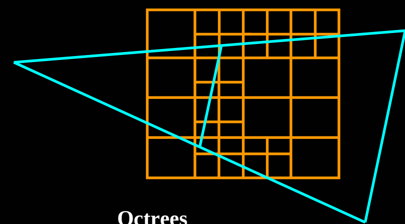
- Each face has form Ax + By + Cz + D
- Plug in coordinates and determine
  - Positive: front side
  - Zero: on plane
  - Negative: back side
- Back-to-front: inorder traversal, farther child first
- Front-to-back: inorder traversal, near child first
- Clip against visible portion of space (portals)

Clipping With Spatial Data Structures

- Accelerate clipping
  - Goal: accept or rejects whole sets of objects
  - Can use a spatial data structure
- Scene should be mostly fixed
  - Terrain fly-through
  - Gaming



Hierarchical bounding volumes



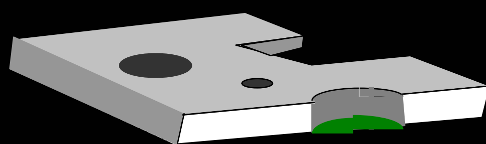
Octrees

Data Structure Demo

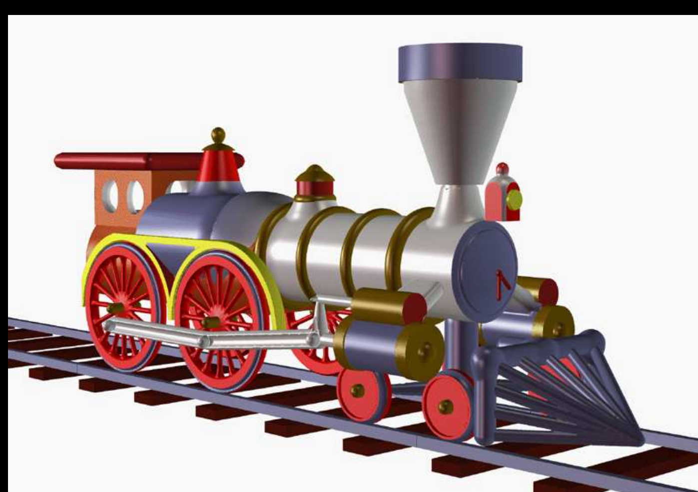
- BSP Tree construction
  http://symbolcraft.com/graphics/bsp/

Constructive Solid Geometry (CSG)

- Generate complex shapes with simple building blocks (boxes, spheres, cylinders, cones, ...)
- Particularly applicable for machined objects
- Efficient with ray tracing



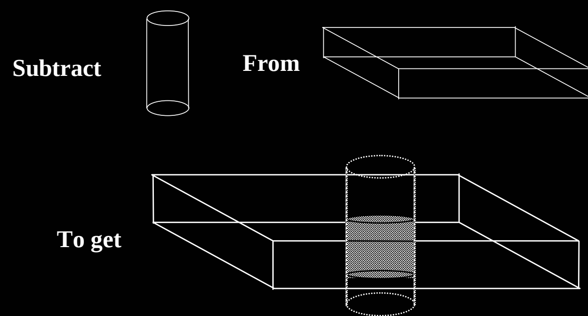
Example: A CSG Train



Brian Wyvill et al., U. of Calgary

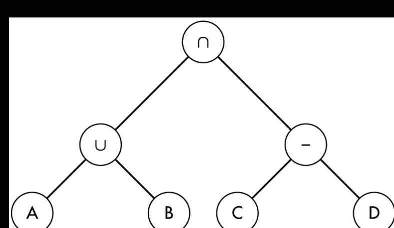
Boolean Operations

- Intersection and union
- Subtraction
  - Example: drilling a hole



CSG Trees

- Set operations yield tree-based representation



- Use these trees for ray/objects intersections
- Think about how!

Implicit Functions for Booleans

- Solid as implicit function, $F(x,y,z)$
  - $F(x, y, z) < 0$ interior
  - $F(x, y, z) = 0$ surface
  - $F(x, y, z) > 0$ exterior
- For CSG, use $F(x, y, z) \in \{-1, 0, 1\}$
- $F_{A \cap B}(\mathbf{p}) = \max(F_A(\mathbf{p}), F_B(\mathbf{p}))$
- $F_{A \cup B}(\mathbf{p}) = \min(F_A(\mathbf{p}), F_B(\mathbf{p}))$
- $F_{A - B}(\mathbf{p}) = \max(F_A(\mathbf{p}), -F_B(\mathbf{p}))$

Summary

- Hierarchical Bounding Volumes
- Regular Grids
- Octrees
- BSP Trees
- Constructive Solid Geometry (CSG)

Announcements

- Ray tracing project – there have been some changes to the grammar and starter code
  - check newsgroup periodically
  - get new assignment handout
- Ray tracing project – there will be a help session later in the week
  - watch the newsgroup for an announcement