Johns Hopkins
Engineering for Professionals

605.767 Applied Computer Graphics

Brian Russin

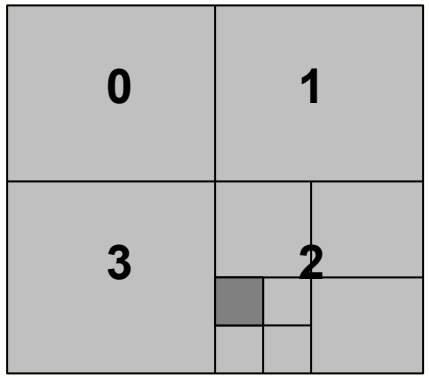
Module 5B
Bi-Spatial Partitioning

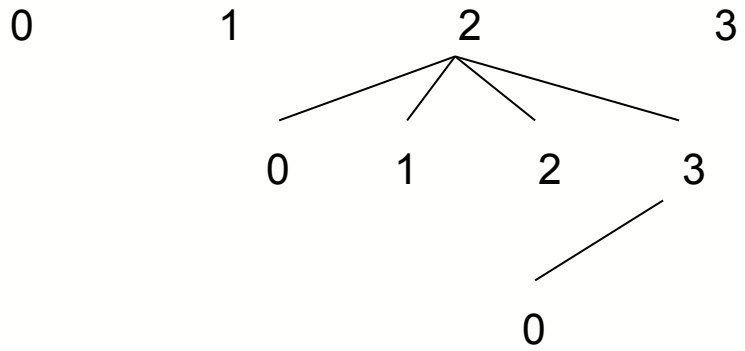
Spatial Subdivision

- Label points in space based on occupancy by objects
  - Based on a single cubic element of space known as a voxel
    - Volumetric element – smallest cube used in the representation
  - Brute force – divide world space into uniform voxels
    - Costly in terms of memory
- Efficient structures like quadtrees (2D), octrees (3D)
  - Subdivide space into squares (2D) or cubes (3D)
- Used for volume rendering
  - Medical imaging applications
    - Cross sections of scans – Marching Cubes
- Often used as secondary data structures in 3D computer graphics
  - e.g., ray-tracing to minimize ray-object intersection search
    - What objects are encountered as a ray traverses voxel space?

Quadtrees

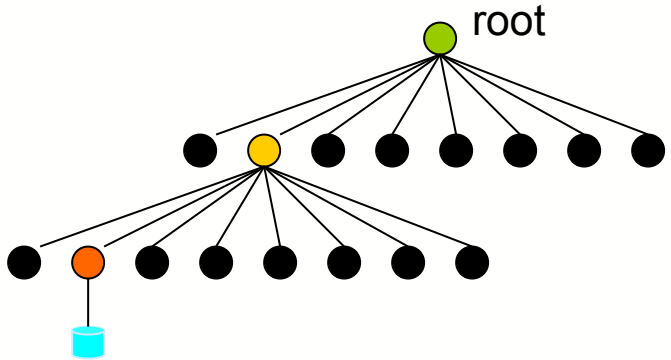
- Generated by successively dividing a 2D space (square region representing the scene space) into quadrants
  - Any subregion that is occupied is further subdivided until the size of the subregion reaches the desired resolution
- Each node in the quadtree has 4 elements, one for each of the quadrants in the region
- Two types of terminal nodes
  - Subregions unoccupied by objects
  - Subregions of minimum size occupied by part of an object



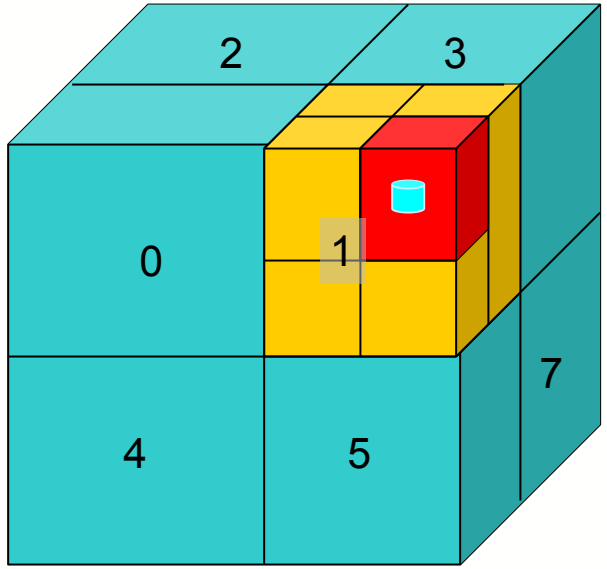


Octrees

- Hierarchical data structure specifying occupancy of cubic regions
- Similar tree structure to quadtree
  - 8 nodes instead of 4
  - Split each box at the center of the box
- https://www.gamedeveloper.com/programming/octree-partitioning-techniques



Indicates a terminal node

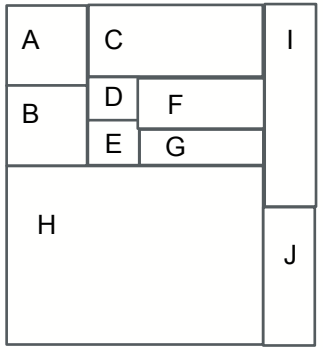


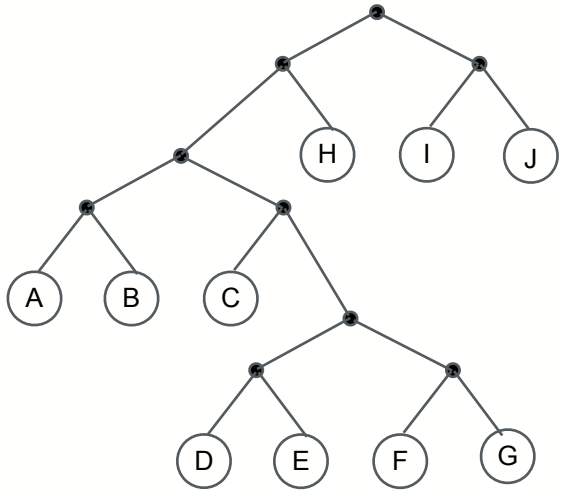
Constructing Octrees

- Enclose scene in AABB
- Recursively subdivide each box along each axis
- Ends when specific criteria area met
  - Maximum recursion level
    - Fewer than n (threshold) primitives in the box
  - Binds primitives to the box
- Options if an object overlaps a box boundary:
  - Store object in both boxes
  - Store object in smallest box that contains the entire object
  - Store one level above where overlap occurs
    - Small object may be bounded by box enclosing entire scene
  - Split primitive
  - Store pointers to objects in each box
    - Editing is more difficult – but OK for static scene
  - “Loose” octrees
    - Relaxes box size – uses same center point but increases box size
      - Figure 19.6 in Haines and Moller (14.6 in 3rd Edition)

k-D Trees

- Generalized version of quadtrees and octrees
- k is the number of dimensions subdivided
- Freely select the next dimension to subdivide
  - Instead of alternating x, y, z, x, … , as in octrees
- Searching is usually much more efficient
  - Useful for detailed tests
    - Finding nearest point on a mesh
    - Ray-intersection with complex geometry with thousands of triangles



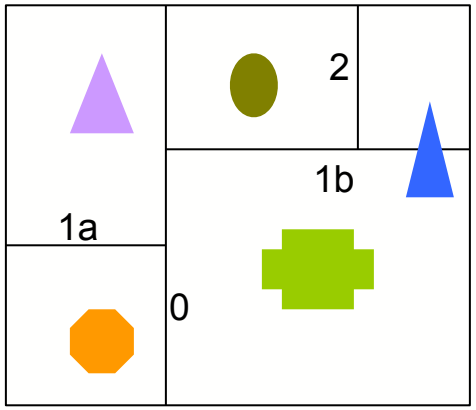


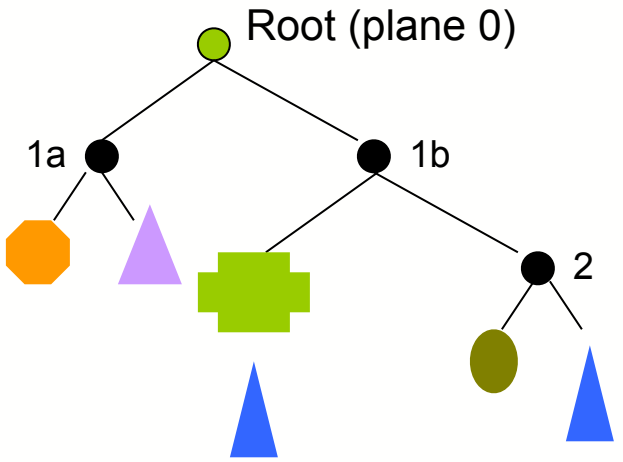
Binary Space Partitioning

- **Binary Space Partitioning** (BSP) trees recursively divide space into 2 regions at each node
  - Use a plane to divide space into 2 halves
    - Geometry is sorted into the 2 regions
  - Can provide a more efficient partitioning since the subdivision planes can be adjusted based on spatial distribution of objects
    - Reduces depth of the tree representation
    - Reduces search time
- Two variants
  - Axis-aligned
    - Division is a plane aligned to a coordinate axis
  - Polygon aligned
    - Division is a plane of a polygon
- Proposed by Henry Fuchs (University of North Carolina) as a means of sorting objects to aid in hidden surface removal
  - Polygon aligned – provides an exact sorting

Axis-Aligned BSPs

- Enclose scene in AABB
- Recursively subdivide each box into smaller boxes
  - Choose an axis of the box to generate a perpendicular plane that divides the box in 2
    - Can allow the plane to vary in position
  - Ends when specific criteria area met
    - Maximum recursion level
    - Fewer than n (threshold) primitives in the box
    - Binds primitives to the box





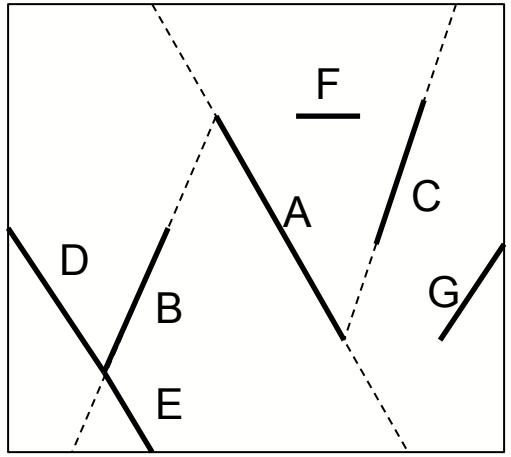
Axis-Aligned BSPs

- Strategies for splitting a box
  - Cycles through the axes
    - Split root along x, children along y, grandchildren along z
    - Called **k-d trees**
  - Split along the longest dimension of the box
    - At some distance d
    - To balance the tree split such that equal numbers of primitives appear on each side of the plane at d
      - Can be costly
      - Often split at an average center point of the primitives
- Can be used for sorting
  - Only provides rough front-to-back sorting
    - Since contents of leaf nodes are not sorted
    - Objects may be in multiple nodes of the tree
  - Sorting may be useful in occlusion culling

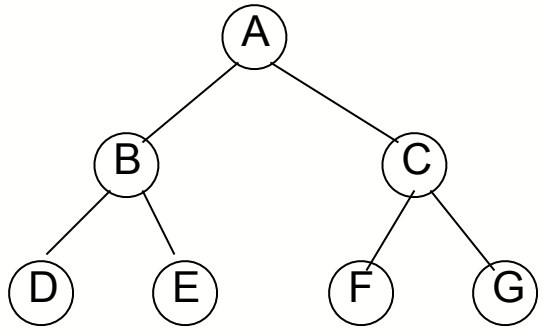
Polygon-Aligned BSP Trees

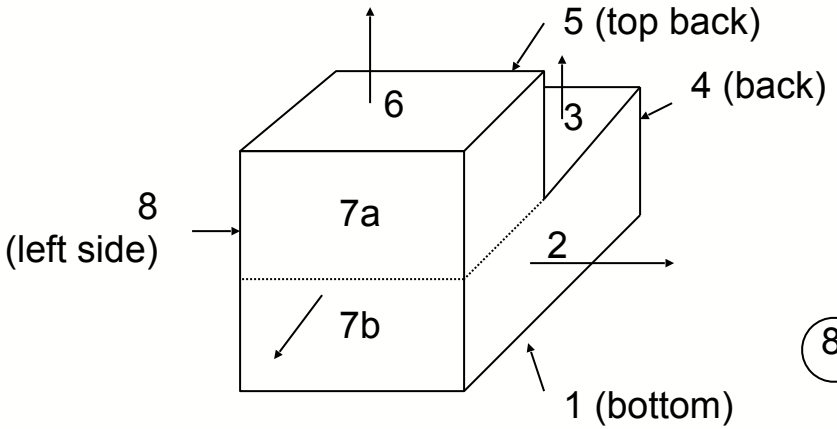
- Choose the partitioning planes to coincide with polygon planes
  - Divides space into half-space containing all remaining faces in front and half-space containing all remaining faces in back of the current polygon
  - Faces intersecting a partitioning plane are split into 2
  - Continue to subdivide using a face from each front and back as its front and back children
    - Terminates when each node contains a single face
- Tree's internal nodes are the partitioning planes
  - Leaves are regions in space, faces are represented as terminal nodes
    - Front polygons are left branches and back polygons are right branches
- Creating an efficient BSP tree is a time consuming process
  - Best to form a balanced tree
  - Creation is often done once and stored for reuse
- Recommend looking at:
  - https://web.cs.wpi.edu/~matt/courses/cs563/talks/bsp/document.html

BSP Examples

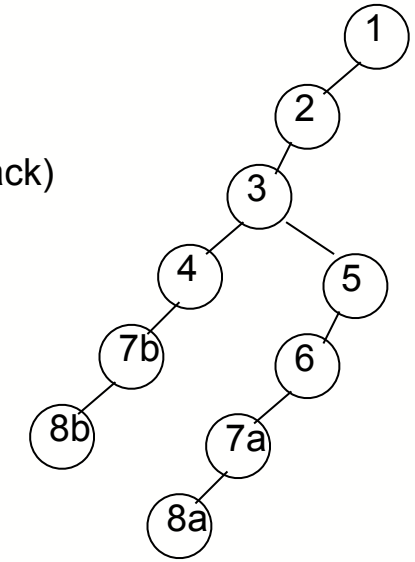


Balanced tree





Unbalanced tree



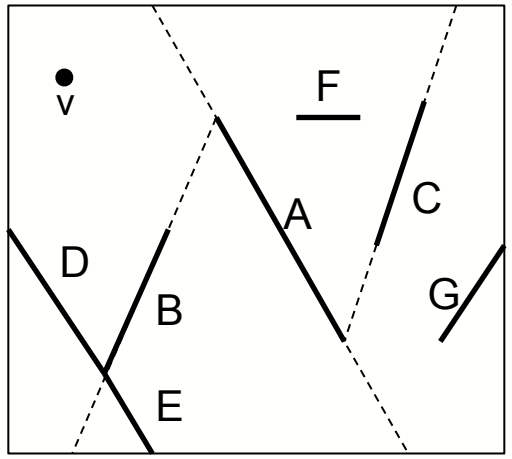
Hidden Surface Removal using BSP Trees

- Can traverse the BSP tree to obtain a drawing order for polygons
  - A list-priority hidden surface removal algorithm
  - Painter's method (back-to-front) rendering can be performed
    - No need for Z buffer
- Very efficient for calculating visibility relationships among a static group of polygons
  - Useful when view reference point changes but objects in scene are at fixed positions
  - Trades intensive pre-processing step against efficient algorithm
  - BSPs used in early generation game engines

Traversing the BSP Tree

- Traverse the tree to yield a correct priority ordered face list
  - Determine on which side of the root plane the view position is located
    - Polygon set in the far side of this plane is beyond those in the near side
      - Recursively take the next level's plane and test camera position
  - Back-face culling
    - Do not display a polygon if the eye is in its rear half-space

```
// From: http://www.exaflop.org/docs/naifgfx/naifbsp.html (dead-link)
void WalkBspTree(BSP_ENTRY* this)
{
   BSP_ENTRY* front  = this->front;
   BSP_ENTRY* back  = this->back;
   if (NormalFacingViewer(&this->normal, &this->origin)
   {
      if (back != NULL)   WalkBspTree(back);
      DrawPolygon(this->id );
      if (front != NULL)  WalkBspTree(front);
   }
    else
   {
      if (front != NULL)  WalkBspTree( front );
      DrawPolygon(this->id );    // Remove if backface culling is desired
      if (back != NULL)   WalkBspTree(back);
   }
}
```



Drawing order from camera point v:
G,C,F,A,E,B,D