AA Example: Visibility

- The problem: Given a geometric model (list of polygons) and a view specification, generate the image which represents that scene viewed in that way.
- Many ways to approach the problem
  - Ivan Sutherland, A Characterization of Ten Hidden Surface Algorithms, 1974
  - More approaches in the decades since
**Visibility Algorithm Taxonomy**

- **Continuous**
  - Object space
    - Presort: Water-Allerton
    - Nonpresort: BSP
  - Image space
    - Subdivide by scissoring: Scanline
    - Subdivide uniformly: Octree
    - Subdivide hierarchically: Warnock

- **Point-sampling**
  - Object space
  - Image space

---

**Painter’s Algorithm**

- **Approach**
  - Sort polygons
  - Draw polygons in order: furthest to closest
Painter’s Algorithm Problems

- **Depth sort is hard**
  - Time-consuming
  - Dealing with ambiguities
- **For interactive 3D applications**
  - View point changes as viewer moves through scene
  - Requires new sort of pgon list with every new view position

Binary Space Partition (BSP) Tree

- **Concept**
  - Create a view-independent organization of pgons that can accelerate sorted drawing of primitives
- **Approach**
  - Build binary tree capturing spatial pgon relationships (do once per model)
  - Traverse tree for particular view position (do once per frame)
BSP Tree: Polygons in Scene
(2D top view, view position at left)

BSP Tree Algorithm: Building the Tree

BSPTree MakeBSPTree( PolygonList list ) {
    PolygonList frontlist, backlist;
    Polygon root, p;
    if (list.isEmpty()) return null;
    else {
        root = list.somePolygon();
        list.remove(root);
        for (Polygon p : list) {
            if (root.inFront(p) // p in front of root
                frontlist.add(p);
            else if (root.inBack(p)) // p in back of root
                backlist.add(p);
            else{ // p crosses plane of root
                p.split(root, frontpart, backpart);
                frontlist.add(frontpart);
                backlist.add(backpart);
            }
        }
        return (BSPTree(MakeBSPTree(frontlist), root,
                        MakeBSPTree(backlist)));
    }
}
BSPTree: Building the Tree (1)

BSPTree: Building the Tree (2)
BSPTree: Building the Tree (3)

BSPTree: Root choice
BSP Tree Algorithm: Displaying the Tree

```c
void DisplayBSP( BSPTree tree, ViewPoint viewer) {
    if (!tree.isEmpty()) {
        if (root.inFront(viewer)) {
            DisplayBSP(tree->back);
            DisplayPolygon (tree->root);
            DisplayBSP(tree->front);
        }
        else {
            DisplayBSP(tree->front);
            DisplayPolygon (tree->root);
            DisplayBSP(tree->back);
        }
    }
}
```

BSPTree: The Tree
BSPTree: Displaying the Tree (1)

- For view point at C:
  - At 3: viewpoint in front -> display back first

---

BSPTree: Displaying the Tree (2)

- For view point at C:
  - At 3: viewpoint in front -> display back first
  - At 4: viewpoint in back -> display front first
BSPTree: Displaying the Tree (3)

For viewpoint at C:
At 3: viewpoint in front -> display back first
At 4: viewpoint in back -> display front first (none)
    display self

BSPTree: Displaying the Tree (4)

For viewpoint at C:
At 3: viewpoint in front -> display back first
At 4: viewpoint in back -> display front first (none)
    display self
display back (5b)
at 5b: viewpoint in back -> display front (none)
    display self
display back (none)
BSPTree: Displaying the Tree (5)

For viewpoint at C:
At 3: viewpoint in front -> display back first // done with 4, 5b
display self

BSPTree: Displaying the Tree (6)

For viewpoint at C:
At 3: viewpoint in front -> display back first // with 4, 5b
display self
display front (2)
At 2: viewpoint in back -> display front first
BSPTree: Displaying the Tree (7)

For view point at C:
At 3: viewpoint in front -> display back first  // done with 4, 5b
    display self
display front (2)
At 2: viewpoint in back -> display front first
    at 5a: viewpoint in back -> display front (none)
    display self
display back (none)

BSPTree: Displaying the Tree (8)

For view point at C:
At 3: viewpoint in front -> display back first  // done with 4, 5b
    display self
display front (2)
At 2: viewpoint in back -> display front first  // done with 5a
    display self
BSPTree: Displaying the Tree (9)

For view point at C:
At 3: viewpoint in front -> display back first // done with 4, 5b
   display self
   display front (2)
At 2: viewpoint in back -> display front first // done with 5a
   display self
   display back (1)
At 1: viewpoint in back -> display front first (none)
   display self
   display back (none)