Binary Space Partition Trees

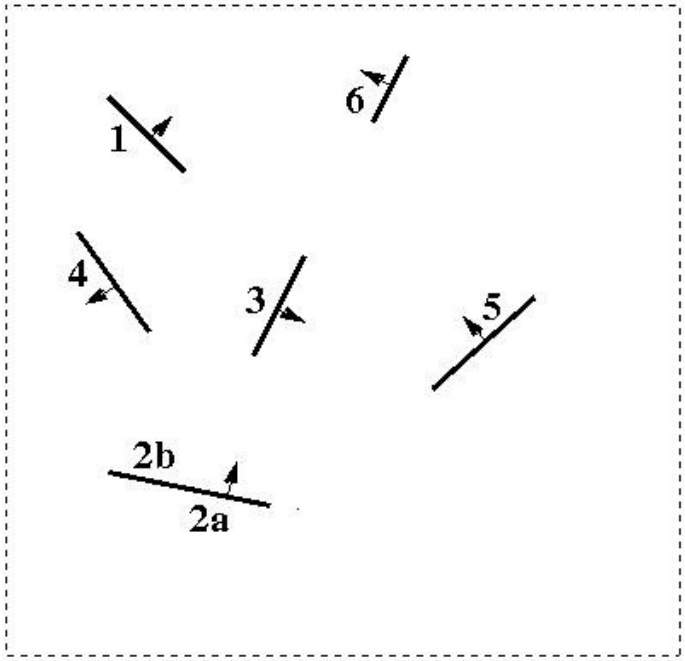
Overview

- Previous list priority algorithms fail in a number of cases, non of them is completely general
- BSP tree is a general solution, but with its own problems
  - Tree size
  - Tree accuracy

Binary Space Partitioning Trees
(Fuchs, Kedem and Naylor `80)

- More general, can deal with inseparable objects
- Automatic, uses partition planes defined by the scene polygons
- Method has two steps:
  - building of the tree independently of viewpoint
  - traversing the tree from a given viewpoint to get visibility ordering

Building a BSP Tree (Recursive)

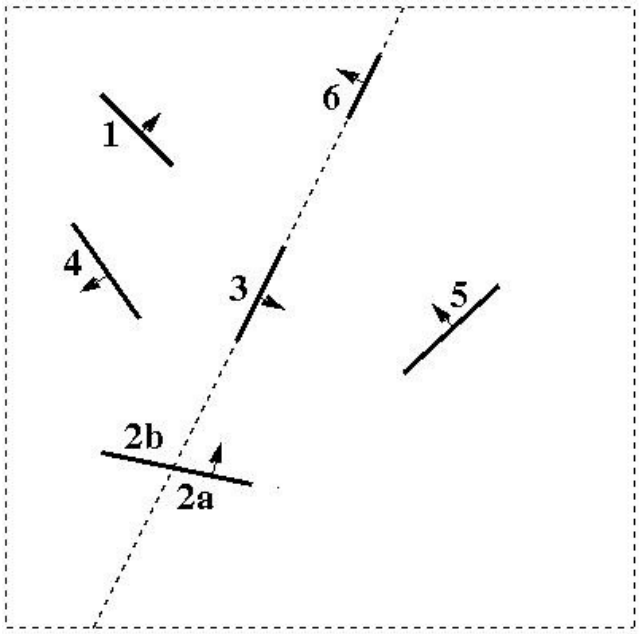


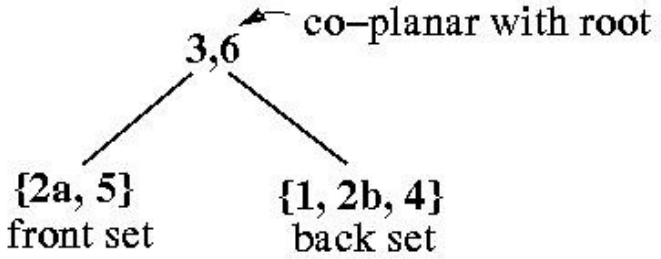
A set of polygons

{1, 2, 3, 4, 5, 6}

The tree

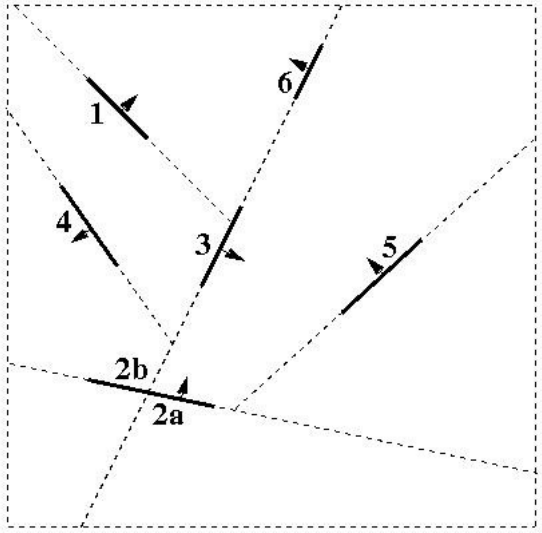
Building a BSP Tree (Recursive)

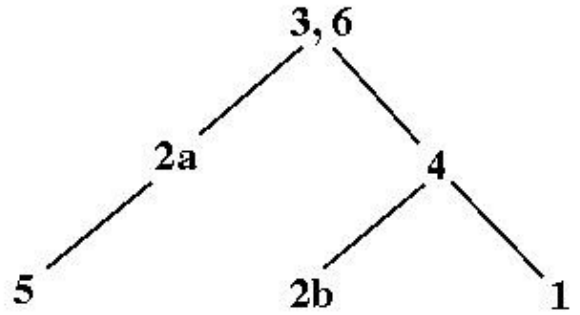




Select one polygon and partition the space and the polygons

Building a BSP Tree (Recursive)





Recursively partition each sub-tree until all polygons are used up

Building a BSP Tree (Recursive)

- Start with a set of polygons and an empty tree
- Select one of them and make it the root of the tree
- Use its plane to divide the rest of the polygons in 3 sets: *front, back, coplanar*.
  - Any polygon crossing the plane is split
- Repeat the process recursively with the *front* and *back* sets, creating the front and back subtrees respectively

Building a BSP Tree (Incremental)

- The tree can also be built incrementally:
  - start with a set of polygons and an empty tree
  - insert the polygons into the tree one at a time
  - insertion of a polygon is done by comparing it against the plane at each node and propagating it to the right side, splitting if necessary
  - when the polygon reaches an empty cell, make a node with its supporting plane

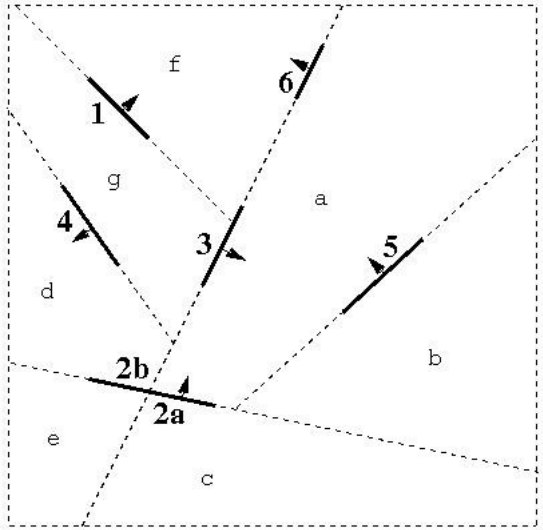
Back-to-Front Traversal

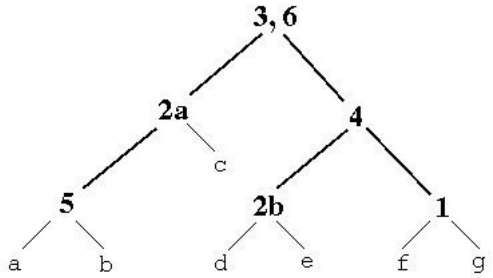
```
void traverse_btf(Tree *t, Point vp)
{
    if (t = NULL) return;
    endif

    if (vp in-front of plane at root of t)
        traverse_btf(t->back, vp);
        draw polygons on node of t;
        traverse_btf;(t->front, vp);
    else
        traverse_btf(t->front, vp);
        draw polygons on node of t;
        traverse_btf(t->back, vp);
    endif
}
```

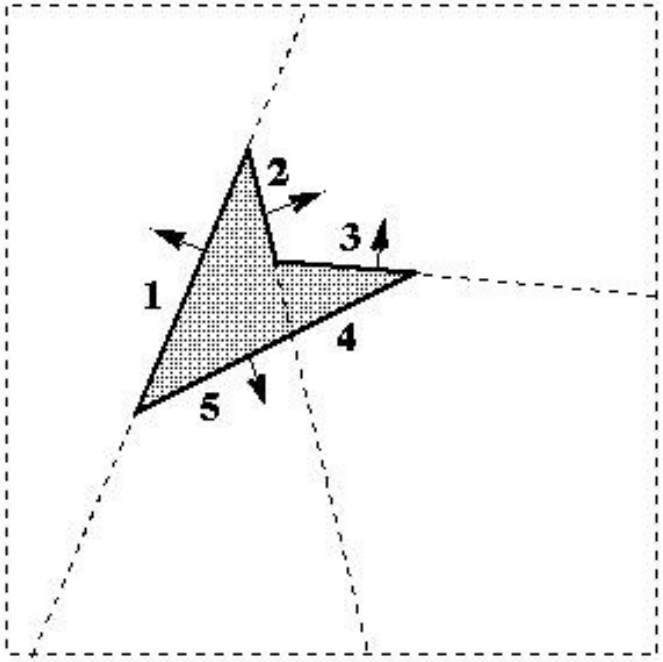
BSP as a Hierarchy of Spaces

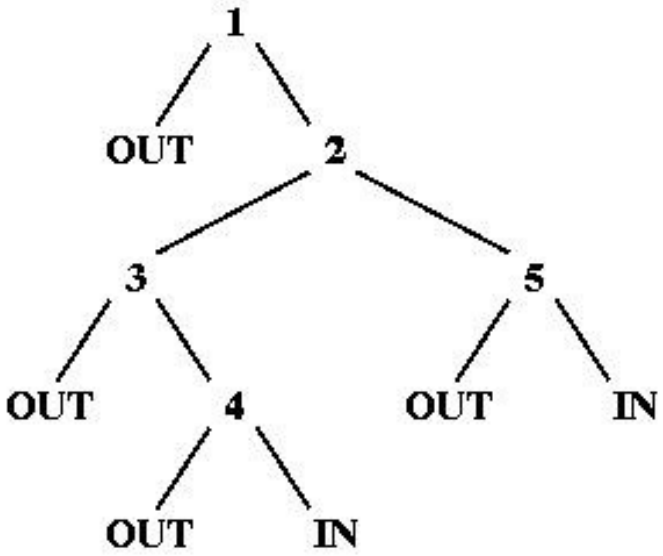
- Each node corresponds to a region of space
  - the root is the whole of R^n
  - the leaves are homogeneous regions



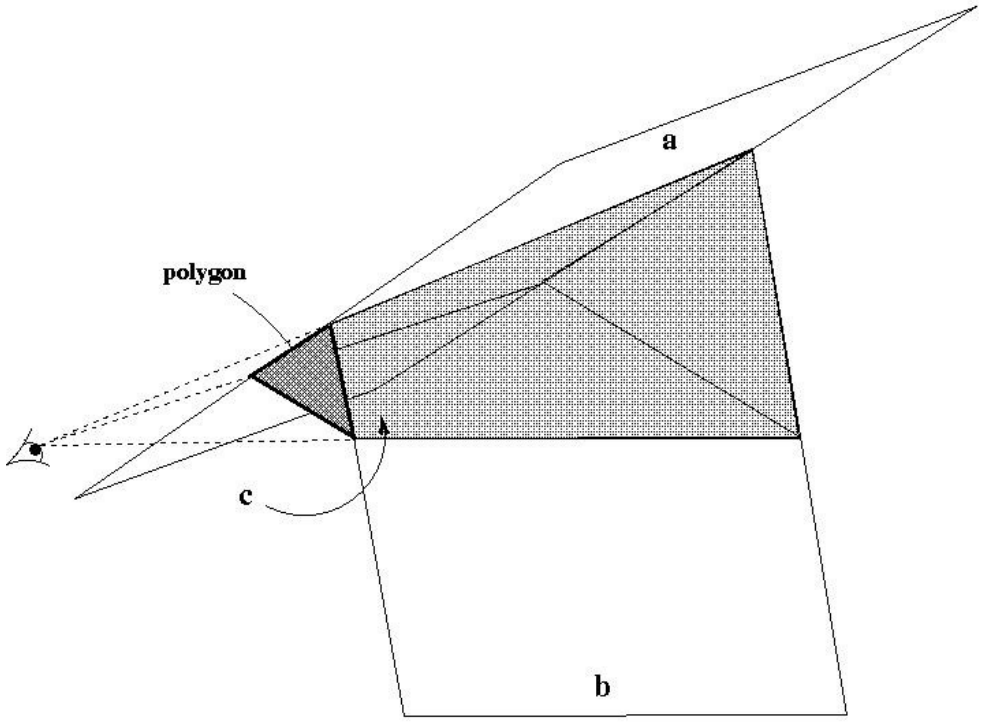


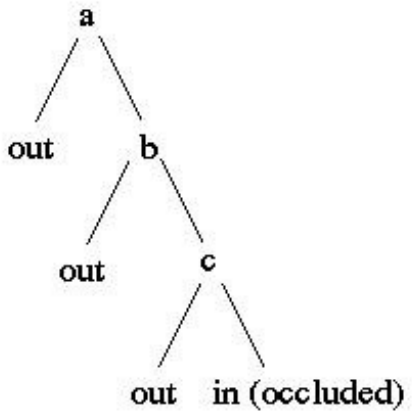
Representation of Polygons





Representation of Polyhedra





BSP Trees for Dynamic Scenes

- When an object moves the planes that represent it must be removed and re-inserted
- Some systems only insert static geometry into the BSP tree
- Otherwise must deal with merging and fixing the BSP cells (see the book!)

Recap

- A BSP is a sequence of binary partitions of space
- Can be built recursively or incrementally
- Choice of plane used to split is critical
- BSP trees are hard to maintain for dynamic scenes