Binary Space Partitioning Trees

Cluster priority

(a) Clusters 1 through 3 are divided by partitioning planes $P_1$ and $P_2$, determining regions A through D in which the eyepoint may be located. Each region has a unique cluster priority.

(b) The binary-tree representation of (a).
Face priority

(a) Faces in a cluster and their priorities. A lower number indicates a higher priority.

(b) Priorities of visible faces.
Pseudocode for building a BSP tree

type
  BSP_tree = record
    root: polygon;
    backChild, frontChild: ^BSP_tree
  end;

function BSP_makeTree ( polyList : listOfPolygons ) : ^BSP_tree;

var
  root: polygon;
  backList, frontList: listOfPolygons;
  p, backPart, frontPart: polygon {We assume each polygon is convex.}

begin
  if polyList is empty then
    BSP_makeTree := nil
  else
    begin
      root := BSP_selectAndRemovePoly ( polyList );
      backList := nil;
      frontList := nil;
    end;
end;
for each remaining polygon \( p \) in \( \text{polyList} \)
begin
  if \( \text{polygon } p \text{ in front of root} \) then
    \( \text{BSP\_addToList}( p, \text{frontList} ) \)
  else if \( \text{polygon } p \text{ in back of root} \) then
    \( \text{BSP\_addToList}( p, \text{backList} ) \)
  else { Polygon \( p \) must be split.}
    begin
      \( \text{BSP\_splitPoly}( p, \text{root}, \text{frontPart}, \text{backPart} ) \);
      \( \text{BSP\_addToList}( \text{frontPart}, \text{frontList} ) \);
      \( \text{BSP\_addToList}( \text{backPart}, \text{backList} ) \);
    end
  end;
\( \text{BSP\_makeTree} := \text{BSP\_combineTree}( \text{BSP\_makeTree}( \text{frontList} ),\)
                 \( \text{root}, \text{BSP\_makeTree}( \text{backList} ) ) \)
end
end;
BSP Tree

(a) front back
1 2 5a 4 5b

(b) front back
front 3 back
front 2 back

(c) front back
front 3 back
front 2 back

(d) front back
front 3 back
front 2 back

5 back
4 back
front 1 back

The University of Texas at Austin
(a) Top view of scene with BSP tree before recursion with polygon 3 as root.

(b) After building left subtree.

(c) Complete tree.

(d) Alternate tree with polygon 5 as root.
Pseudocode for displaying a BSP tree

procedure BSP_displayTree ( tree : ^ BSP_tree );
beginsk
  if tree is not empty then
    if viewer is in front of root then
      begin
        { Display back child, root, and front child. } 
        BSP_displayTree ( tree ^ .backChild );
        displayPolygon ( tree ^ .root );
        BSP_displayTree ( tree ^ .frontChild )
      end
    else 
      begin
        { Display front child, root, and back child. } 
        BSP_displayTree ( tree ^ .frontChild );
        displayPolygon ( tree ^ .root ); { Only if back-face culling not desired}
        BSP_displayTree ( tree ^ .backChild )
      end
  end;
Two traversals of the BSP tree corresponding to two different projections. Projectors are shown as thin lines. White numbers indicate drawing order.
Homogeneous Region

Partitioned Region

Initial Tree

New Tree

Elementary operation used to construct Partitioning Trees
Example intra-object Partitioning Tree