List Priority Algorithms

Determines a visibility ordering for objects ensures that a correct picture results if objects are rendered in that order.

e.g. if no object overlaps in ‘z’ then we need only to sort objects by increasing ‘z’ and render them in that order.

Farther objects are obscured by closer ones as pixels from the closer polygons overwrite those of more distant ones.

If objects overlap in ‘z’, we may still be able to determine a correct order.

If objects cyclically overlap, or penetrate each other, then there is no correct order.

Hybrids that combine both object and image precision operations.

Depth comparison and object splitting are done with object precision.

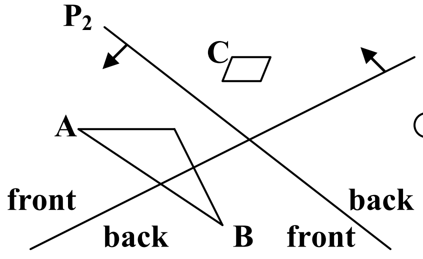
Scan conversion(which relies on ability of graphics device to overwrite pixels of previously drawn objects) is done with image precision.

A. <u>BSP TREE METHOD</u>

Binary Space partitioning (BSP) tree is an efficient method for determining object visibility by painting surfaces onto the screen from back to front, as in painter’s algorithm

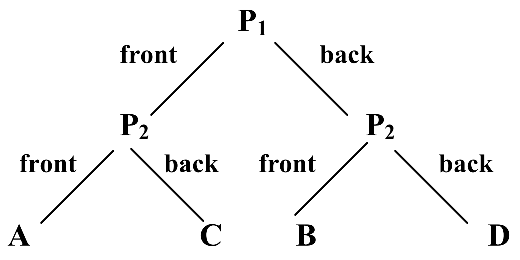
Useful when view reference point changes but the objects in a scene are at fixed position

Identifying surfaces that are “inside” and “outside” partitioning plane at each step of space sub division relative to viewing direction is required for visibility testing.



With plane p1, we first partition the space into two sets of objects. One set of object is behind plane p1 relative to the viewing direction, and the other set is in-front of p1.

Since one object is intersected by plane p1, we divide that object into two separate objects labeled A and B. Objects A and C are in-front of P1 and objects B and D are behind p1.



Next partition the space again with plane p2 and construct the binary tree representation Objects are represented as terminal nodes, with Front objects as left branches and back objects as right branches.

For objects described with polygon facets, we choose the partitioning planes to coincide with polygon planes. Make use of polygon equations to identify “inside” and ”outside” polygons and the tree is constructed with one partitioning plane for each polygon face.

Any polygon intersected by partitioning plane is split into two parts.
When BSP tree is complete, we process the tree by selecting the surfaces for display in the order back to front so that foreground objects are painted over background objects.

B. Depth sorting Method

Makes use of both image and object space operations to:

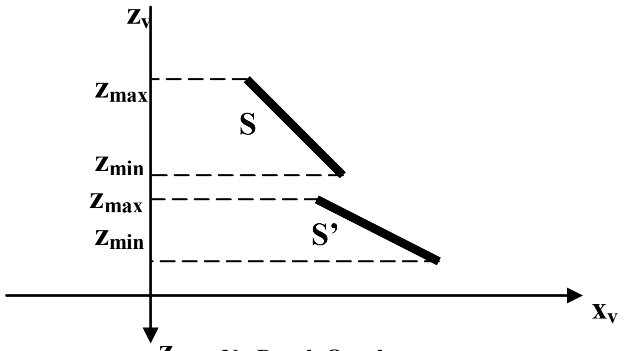
i. **Sort surfaces in decreasing depth order(using both image and object space)**
ii. **Scan converted in order starting with surface with greatest depth(using image space)**

Also referred to as painters algorithm because an artist first paints background color then most distant object then nearer object and so on. In the end the foreground objects are painted on canvas over the background. Each layer of paint covers up previous layer

Similarly, we first sort surfaces according to their distance from view plane.

Intensity values for farthest surface are entered into refresh buffer.

Taking each surface in turn (in decreasing depth order) we paint surface intensities onto frame buffer over intensity of previously processed surface



No Depth Overlap

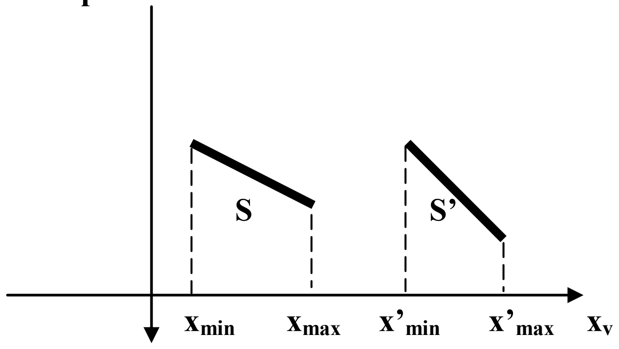
We assume we're viewing along 'z' direction. Surfaces are ordered according to largest 'z' value on each surface

Surface with greatest depth is compared with other surface in list to see if there are any overlaps in depth

If no depth overlap then surface S is scan converted

Additional test are required for each surface in case of depth overlap with surface S:

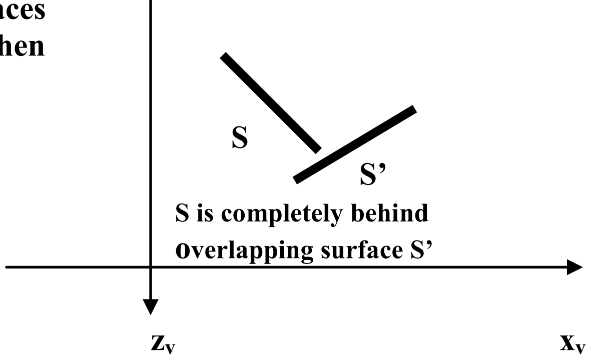
i. **Bounding rectangle in xy plane for two surfaces don't overlap**
ii. **Surface S is completely behind overlapping surface relative to viewing position**
iii. **Overlapping surface S' is completely in-front of S relative to viewing position**
iv. **Projection of 2 surfaces onto view plane don't overlap.**



Depth overlap but no overlap in x direction

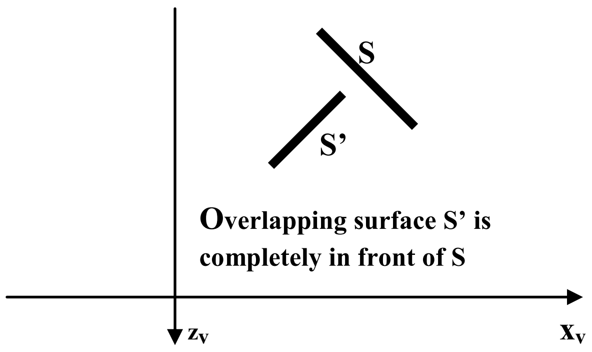
If any of these conditions is true then no reordering of surfaces is necessary. i.e. if all surfaces pass at least one of the tests then none of those surfaces are behind S.

For test (i) we perform two steps to check overlap first in 'x' then in 'y' direction. If either of these directions show no overlap then the two planes cannot obscure one another.



For tests (ii) and (iii), we can perform inside outside polygon test. Substitute coordinates for all vertices into plane equation for overlapping surfaces and check the sign of the result obtained.

Based on plane equation if all vertices of S are inside S' then S is completely behind S'. Similarly, S' is completely in-front of S if all vertices of S are outside of S'.



For test (iv) we can use line equation in xy plane for checking intersections between bounding edges of two surfaces.