CH-7

Visibility

Hidden Lines and Surfaces → while displaying 3D image/ object through projection some of parts of object are not visible from viewer or viewing position. These parts are known as Hidden Lines or Surfaces. There are no. of Method to detect the visible Surfaces known as **Visible Surface detection methods** or **hidden surface elimination methods.**

Classification of Visible Surface detection Algorithm

These are classified according to whether they deal with object definitions directly or with their projected images These two approaches are

1. **Object space** → (1) It compares objects and parts of objects to each other within scene def^{n} to determine which surfaces, as a whole, we should label as visible.

[It deals with world co-ordinates]

(2) **For eg** → Back face removal algorithm, Binary space position, Area subdivision

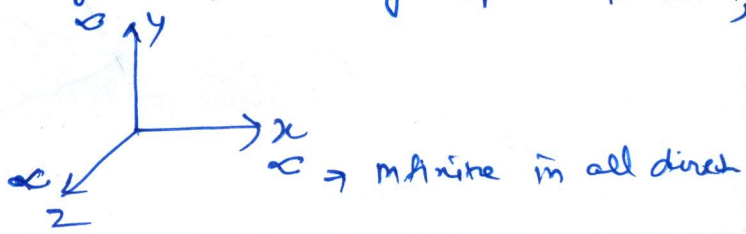


Image space Method → (a) visibility is decided point by point at each pixel position on the projection plane.
[It deals with screen co-ordinates]

(b) This Method deals with projected ~~with~~ image

(c) For eg → z-Buffer algorithm, Scan line algorithm, Painter algorithm (Depth sorting)

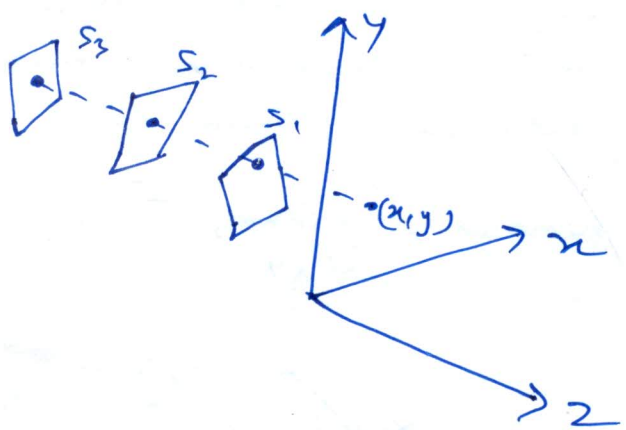
Most of visible surface detection Method use the concept of sorting and coherence.

Sorting → It is used to enable depth comparisons which helps in ordering various surfaces of scene according to their distance from view plane.

Coherence → It is used to take advantage of regularities in scene. [we have to specify the viewing transformation for HSR]

Depth Buffer Algorithm / Z-Buffer algorithm

(1) It consider each and every pixel of scene and compares the depth value or z-value of the various objects to determine which surface is in the front and which is on the rear. This depth value is measured along the z-axis from the view plane. Hence name of algo z-buffer or depth buffer algorithm.



(1) In figure three surfaces at varying distance (2) along the orthographic projection line from position (x,y) in a view plane taken as xy plane. Surface S_1 has smallest depth from view plane and so is visible at that position.

Implementation → (1) It uses two frame buffers.

depth buffer → for storing z-values of each pixel

frame/ Refresh buffer → for storing intensity value of each pixel

(2) Initially all the cells of depth buffer are set to minimum z-values i.e z-values can range from 0 at the back clipping plane to z_{max} at the front clipping plane.

All the cell of refresh buffer are set to background Intensity.

(3) Algorithm → (1) $Refresh(x,y) = I_{background}$

(2) $depth(x,y) = 0$

(3) for each pixel position (x,y) calculate the depth value (z-value)

(4) Compare the calculated new value with value previously stored in z-buffer at that location

if $z > depth(x,y)$ then

Set $depth(x,y) = z$, $refresh(x,y) = I_{surface(x,y)}$

if $z < depth(x,y)$ then

do nothing

$I_{surface(x,y)}$ → Projected intensity value for surface at pixel position (x,y)

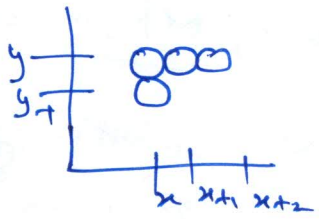
depth value →

plane eqn of surface

$$Ax + By + Cz + D = 0$$

$$z = \frac{-Ax - By - D}{C}$$

for any scan line, if depth of position (x, y) is z then the depth z' for next pixel position $(x+1, y)$ along the scan line is



$$z' = \frac{-A(x+1) - By - D}{C}$$

$$= \frac{-Ax - A - By - D}{C}$$

$$\boxed{z' = z - \frac{A}{C}}$$

Illy for position $(x, y+1)$

$$\boxed{z' = z - \frac{B}{C}}$$

The ratios $-\frac{A}{C}$ & $-\frac{B}{C}$ is constant for each surface,

Advantages → ① It is easy to implement
② It does not involve any sorting

Disadvantages → ① It requires large storage space. Z-buffer memory is proportional to the no. of pixels on the screen.

② It can be applied to scene containing only polygonal surfaces. It can't be applied to non planner surfaces.

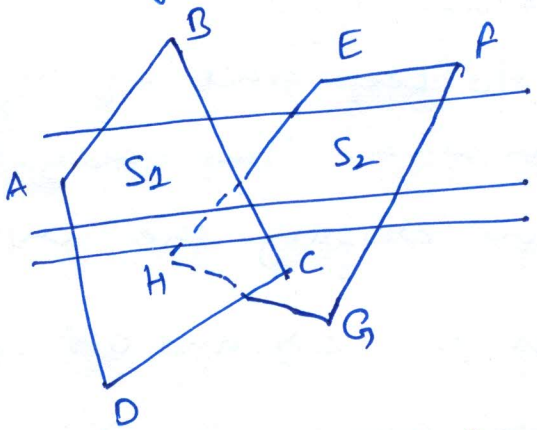
③ It works only with opaque surfaces & not with transparent surfaces.

Scan line Method → ① It is another image space algo and is an extension of scan line polygon filly algorithm.

② It processes the image one scan line at a time rather than one pixel at a time.

③ Z-buffer for only one scan line

[AEL → during scanning scan line intersecting the edge such as AB, BC, EH & FG are active edge]



Scan line 1 → AB, BC, EH & FG

for edge AB & BC, the position b/w AB & BC, the flag for S1 is ON & no depth calculation are necesary.

Illy b/w EH & FG, flag for S_2 is ON. No other positions along scan line 1 intersect surfaces, so intensity values in the other areas are set to the background intensity

Scan line 2 → edge list contains edges AD, EH, BC & FG.

→ from AD to EH, only the flag for S_1 is ON,

→ b/w EH & BC, the flag for Both S_1 & S_2 are ON So depth calculation is required.

for eg. depth of S1 is less than S_2, so intensities for S1 are loaded into refresh buffer until boundary BC is encountered.

Then flag for Surface S_1 goes off and intensities for Surface S_2 are stored until edge FG is passed.

Implementation → (1) It uses edge table and Polygon table.

(2) edge table contains all non-horizontal edges of all polygons projected on the view plan.

(3) Polygon table contains information abt all the polygons in scene.

(4) Active edge list to keep track of surfaces crossing a particular scan line. This table contains only those edges that are crossing the current scan line.

(5) Boolean flag is set ON and OFF to indicate whether a position along a scan line is inside or outside of the surface.

Depth Sort Algorithm →

① It can applied to both image space and Object space Method.

② Using image and object space operation, it perform following operation such as

(a) Surfaces are sorted in order of decreasing depth

(b) Surfaces are scan converted in order, starting with surface of greatest depth.

③ sorting operation are carried out in object and image space and scan conversion is carried out in image space.

④ This Method is also known as painter algorithm.

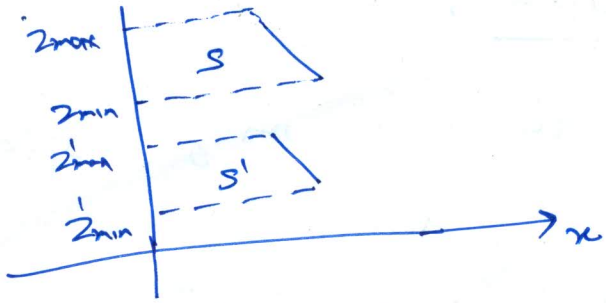
⑤ for eg → painter paint the picture and this picture can be consist of many colours and layers i.e one layer is above other layer. So each layer has different depth value.

⑥ So using this technique we first sort surfaces acc to their distance from the view plane.

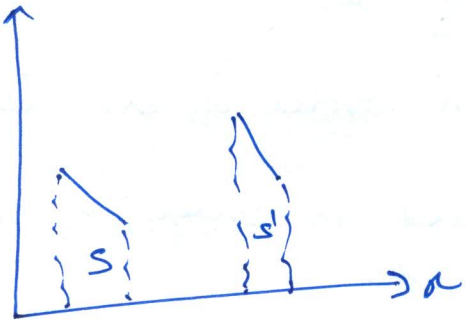
⑦ Taking each succeeding surface in turn (decreasing depth order), we paint the surface intensities on to the frame buffer over the intensities of the previously processed surfaces.

⑧ Painting polygon surface on frame Buffer acc to depth is carried out in several steps.

for eg →

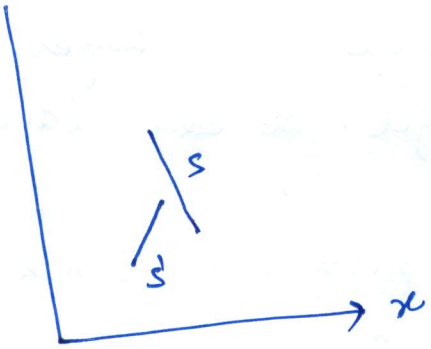


Two Surfaces with no depth overlap



Two Surfaces with depth overlap but no overlap in x direction

Surface S is completely behind the overlapping surface S'



overlapping surface S' is completely in front of surface S but S is not completely behind S'.