Hidden Surface Removal (Visible Surface Detection)

In case of 3D viewing, additional effort should be made to specify the viewing direction. So major concern for realistic graphics is identifying those parts of a scene that are visible from a chosen viewing position

Several algorithms have been devised some require more memory , some involve more processing time and some apply only to special types of objects.

Various algorithms are referred to as "*Visible–Surface Detection***" also referred to as "***Hidden Surface Elimination Method***" algorithms broadly classified approaches are "***Object Space Methods (OSM)***" and "***Image Space Method (ISM)***"**

OSM **compares objects and parts of object to each other within the scene definition to determine which surfaces as a whole we should label as visible e.g.** *Back Face Detection method*

ISM **decides point by point at each pixel position and is most widely used.**

BACK-FACE DETECTION

A fast and simple object-space method for identifying the back faces of a polyhedron is based on the "inside-outside" tests.

A point (x, y, z) is "inside" **a polygon surface with plane parameters A, B, C, and D if**

$$Ax + By + Cz + D < 0$$

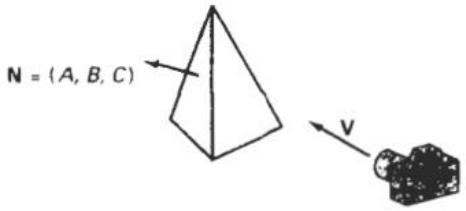
When an inside point is along the line of sight to the surface, the polygon must be a back face (we are inside that face and cannot see the front of it from our viewing position).

We can simplify this test by considering the normal vector N to a polygon surface, which has Cartesian components (A, B, C).

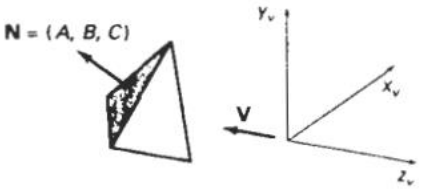
In general, if V is a vector in the viewing direction from the eye (or "camera") position, then this polygon is a back face if

$$V.N > 0$$ **as** ***both vectors are point in the same direction, angle between them is 0***

If object descriptions have been converted to projection coordinates and our viewing direction is parallel to the viewing z_v axis, then $V = (0, 0, V_z)$ and so that we only need to consider the sign of C, the z component of the normal vector N



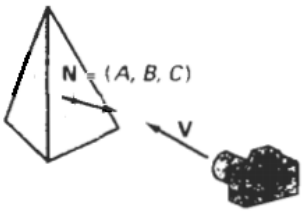
Vector **V** in the viewing direction and a back-face normal vector **N** of a polyhedron



A polygon surface with plane parameter $C < 0$ in a right-handed viewing coordinate system is identified as a back face when the viewing direction is along the negative z_v axis.

If the object gets rotated towards the camera them both vectors start facing towards each other then

$$V.N < 0$$ **as** ***vectors are point in opposite direction, angle between them is 180***



Vector **V** in the viewing direction and normal vector **N** of a polyhedron

In a right-handed viewing system with viewing direction along the negative z_v axis, the polygon is a back face if C < 0. Also, we cannot see any face whose normal has z component C = 0, since our viewing direction is grazing that polygon. Thus, in general, we can label any polygon as a back face if its normal vector has a z component value

C <= 0

<u>Z-buffer or Depth buffer method:</u>
Commonly used *ISM*

Since object depth is usually measured from the view plane along the z axis of a viewing surface. Each surface of a scene is processed separately , one point at a time across the surface

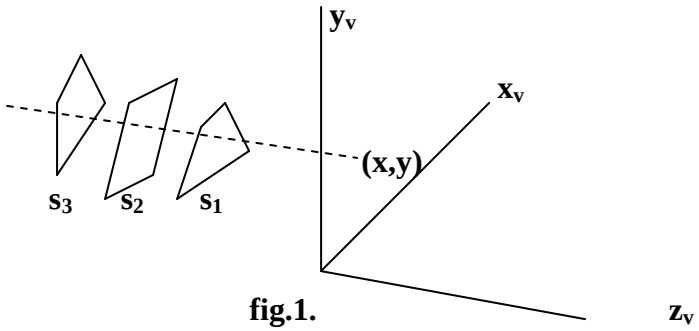


fig.1.

Usually for polygon surface, because depth values can be computed very quickly and the method is easy to implement and can apply to non planar surfaces.

Implementing the algorithm in normalized coordinates , z value ranges from 0 at the back clipping plane to z_{max} at the front clipping plane.
Algorithm

```
i. Initialize the depth buffer and refresh buffer so that for all buffer position(x,y)
          depth(x,y) = 0  or (zmin) and   refresh(x,y) = Ibackground
ii. For each position on each polygon surface compare depth values to previously stored
    values in the depth buffer to determine visibility
          Calculate the depth 'z' for each (x,y) position on the polygon
iii. If  z  > depth(x,y) then set
          depth = z  and   refresh(x,y)  =  Isurface (x,y)
      where   Ibackground (x,y) is background intensity ,  Isurface (x,y)  is projected intensity
      value for surface at    pixel position (x,y)
```

After all surfaces have been processed the depth buffer contains depth values for the visible surfaces and the refresh buffer contains the corresponding intensity values for those surfaces.

Depth value for a surface position (x, y) is z = (-Ax –By – D)/c ……(i)
Let depth z' at (x + 1 , y) z' = -A(x+1) – By – D/c or z' = z – A/c……..(ii)

Now since (-A/c) is constant for each surface, so succeeding depth values across a scan-line are obtained from preceding values with a simple mathematics.

On each scan-line , we start by calculating depth on a left edge of the polygon that intersect that scan-line (fig ii) 'z' at each successive position across the scan

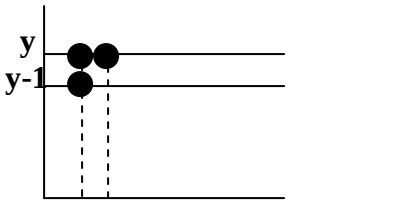
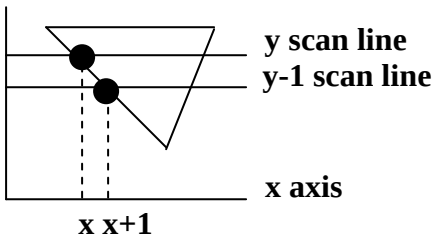


fig. ii.

line are the calculated by equation(ii)
first determine y coordinate extents of each polygon and process from top scan to bottom scan starting at a top vertices we can recursively calculate 'x' position down a left edge of the polygon as
x' = x - 1/ m where m is the slope of edge fig(iii)



'z' values down the edge are then obtained recursively as **fig iii.**
$z' = z + (a/m + b)/c$ or $z' = z + b/c$ as $m \rightarrow$ for vertical edge

Scan-line Method

ISM **for removing hidden surfaces is an extension of the scan-line algorithm for filling polygon interiors.**

Instead of filling just one surface we consider multiple surfaces.

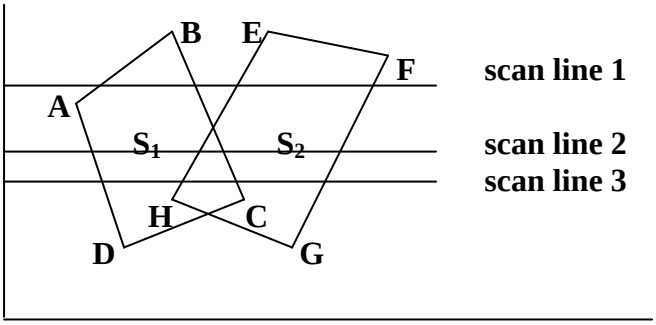
As each scan-line is processed all polygon surfaces intersecting that line are examined to determine which are visible

We assume that tables (*polygon***) are set up for the various surfaces (***edge, polygon etc***)**

Polygon table contains *coefficients of the plane equation* **for each surface intensity information for the surfaces and** *possible pointers into the edge table*

Active edge list **for scan-line1 contains information from the edge table for edges AB,BC,EH AND FG**

For positions along this scan-line between edges AB AND BC only the flags for surfaces S_1 is on.



No depth calculation is necessary and intensity information for surface S_1 is entered from the polygon table into the refresh buffer

Between edges EH and FG only the flag for surface S_2 is on.

No other position along scan-line1 intersect surfaces so intensity values in the other areas are set to background intensity

For scan-line 2 and 3 the active edge list contains edges AD, EH, BC and FG

Along scan-line2 from edge AD to edge EH only the flag for surface S_1 is on.

But between edges EH and BC the flags for both surfaces are on and depth calculations must be made using plane coefficients for the two surfaces,

e.g if 'z' of surface S_1 is less than that of S_2 intensities for surface S_1 are loaded into the refresh buffer until boundary BC is encountered.

Then the flag for surface S_1 goes off and intensities for surface S_2 are stored until edge FG is passed
Any number of overlapping polygon surfaces can be processed with this scan-line method.