Visible-Surface Detection

INTRODUCTION

- Major consideration in the generation of realistic graphics displays is identifying those parts of a scene that are visible from a chosen viewing position.
- The various algorithms are referred to as visible-surface detection methods.
- Sometimes these methods are also referred to as hidden-surface elimination methods, although there can be a subtle differences between identifying visible surfaces and eliminating hidden surfaces.

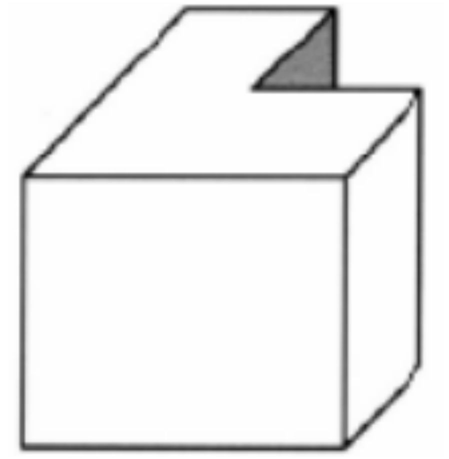
Visible-Surface Detection Methods

- Two approaches to determine what is visible within a scene from a chosen viewing position
  - ***Object-space* methods**: Decide which object, as a whole, is visible.
  - ***Image-space* methods**: The visibility is decided point-by-point.

Approaches

- Back-Face Detection
- Depth Buffer
- A-Buffer
- Scanline
- Depth Sorting
- BSP Tree
- Area Subdivision
- Octree
- Raycasting

Back Face Detection

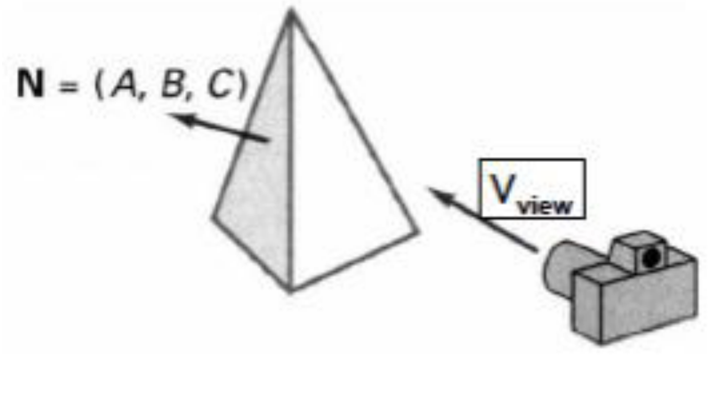


Back Face Detection

- 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.

Back Face Detection Algorithm

- Back-face test by considering the direction of the normal vector **N** to a polygon surface, which has Cartesian components (A, B, C).
- In general, if $\mathbf{V}_{\text{view}}$ is a vector in the viewing direction from the eye (or "camera") position, then this *polygon is a back face if* $\mathbf{V}_{\text{view}} \cdot \mathbf{N} > 0$



Depth –Buffer Method

- A commonly used image-space approach to detecting visible surfaces is the depth-buffer method, which compares surface depths at each pixel position on the projection plane
- This procedure is also referred to as the z-buffer method, since object depth is usually measured from the view plane along the z axis of a viewing system.

Depth –Buffer Method

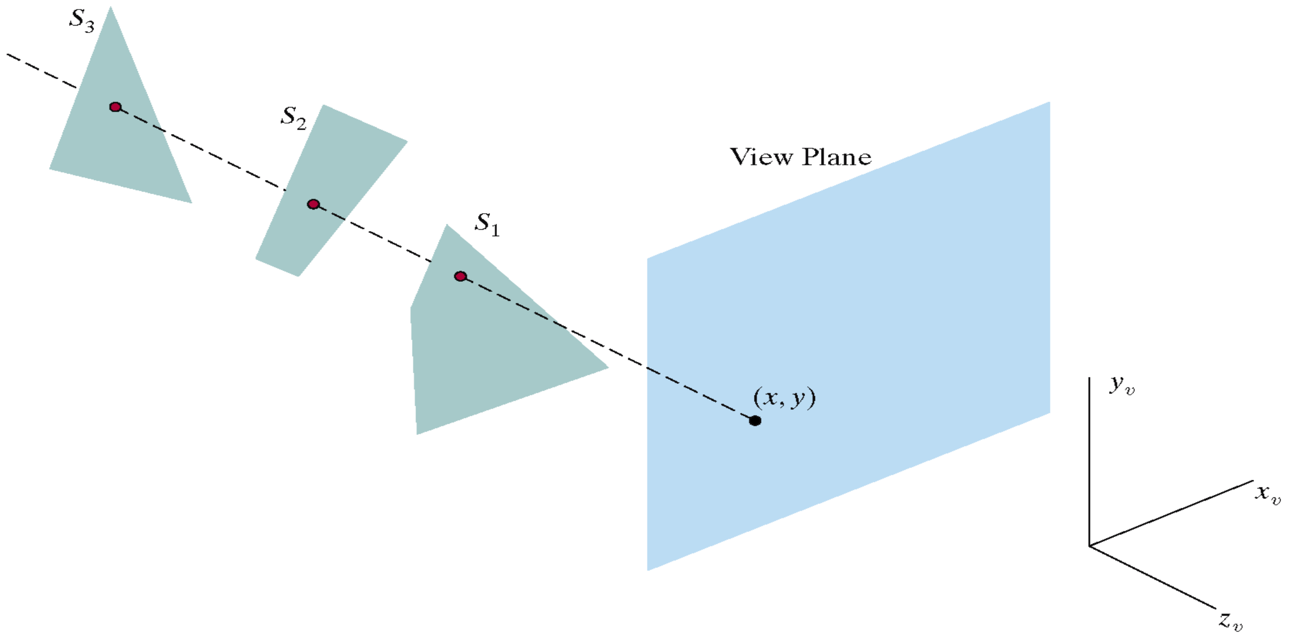
- Each surface of a scene is processed separately, one point at a time across the surface.
- The method is usually applied to scenes containing only polygon surfaces, because depth values can be computed very quickly and the method is easy to implement.

Depth –Buffer Method

- With object descriptions converted to projection coordinates, each (x,y,z) position on a polygon surface corresponds to the orthographic projection point (x,y) on the view plane.
- Therefore, for each pixel position (x,y) on the View plane, object depths can be compared by comparing z values.

Depth –Buffer Method

- Figure shows three surfaces at varying distances along the orthographic projection line from position *(x,* y) in a view plane taken as the *Xv Yv* plane.
- Surface S1 is closest at this position, so its surface intensity value at *(x,* y) is saved.



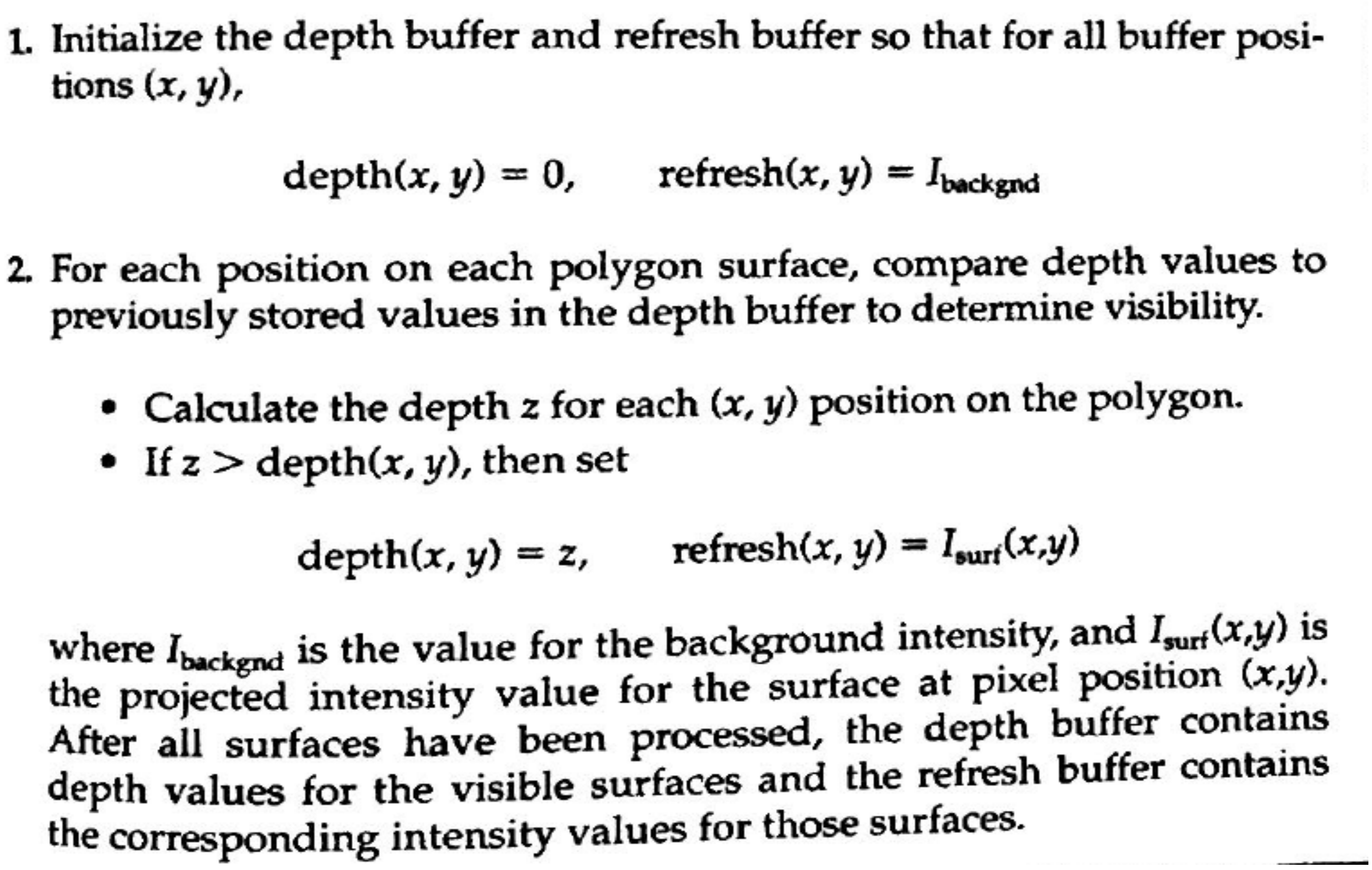
Depth –Buffer Method

- We can implement the depth-buffer algorithm in normalized coordinates, so that z values range from 0 at the back clipping plane to Zmax at the front cliping plane.
- The value of Zmax can be set either to 1 or to the largest value that can be stored on the system.

Depth –Buffer Method

- As implied by the name of this method, two buffer areas are required. A depth buffer is used to store depth values for each *(x.* y) position as surfaces are processed, and the refresh buffer stores the intensity values for each position.
- Initially all positions in the depth buffer are set to 0 (minimum depth) and the refresh buffer is initialized to the background intensity.
- Each surface listed in the polygon table is then processed, one scan line at a time calculating the depth at each pixel position.

Depth Buffer Algorithm



1. Initialize the depth buffer and refresh buffer so that for all buffer positions (x, y),

$$\text{depth}(x, y) = 0, \qquad \text{refresh}(x, y) = I_{\text{backgnd}}$$

2. 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.
   - If $z > \text{depth}(x, y)$, then set

$$\text{depth}(x, y) = z, \qquad \text{refresh}(x, y) = I_{\text{surf}}(x,y)$$

where I_{backgnd} is the value for the background intensity, and $I_{\text{surf}}(x,y)$ is the projected intensity value for the 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 Calculation

- Calculate the z-value on the plane

$$Ax + By + Cz + D = 0 \quad \Rightarrow \quad z = \frac{-Ax - By - D}{C}$$

- Incremental calculation

$$z_{(x,y)} : \text{the depth of position } (x, y)$$

$$z_{(x+1,y)} = \frac{-A(x+1) - By - D}{C} = z_{(x,y)} - \frac{A}{C}$$

$$z_{(x,y+1)} = \frac{-Ax - B(y+1) - D}{C} = z_{(x,y)} - \frac{B}{C}$$

A- Buffer Method

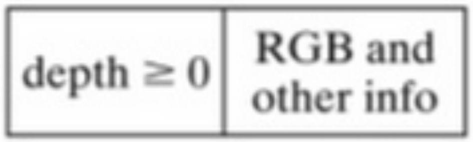
- The drawback of the depth-buffer method is that it can only find one visible surface at each pixel position.
- In other words, it deals only with opaque surfaces and cannot accumulate intensity values for more than one surface, as is necessary if transparent surfaces are to be displayed.
- The A-buffer method expands the depth buffer so that each position in the buffer can reference a linked list of surfaces.
- Thus, more than one surface intensity can be taken into consideration at each pixel position, and object edges can be anti aliased.

A- Buffer Method

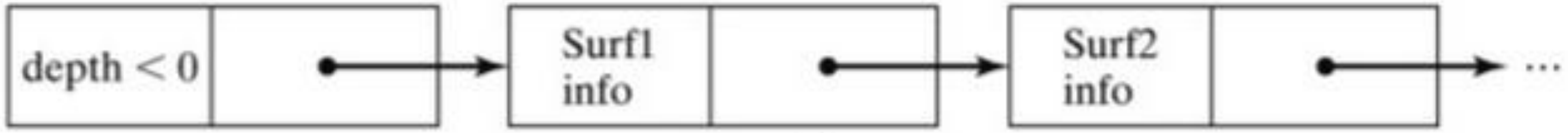
- Each position in the A-buffer has two fields:
  - Depth field - stores a positive or negative real number
  - Intensity field - stores surface-intensity information or a pointer value.
- If the depth field is positive, the number stored at that position is the depth of a single surface overlapping the corresponding pixel area.
- The intensity field then stores the RGB components of the surface color at that point and the percent of pixel coverage as in figure(a).

A- Buffer Method

- If the depth field is negative, this indicates multiple-surface contributions to the pixel intensity.
- The intensity field then stores a pointer to a linked list of surface data, as in Fig.(b).



(a)



(b)

A- Buffer Method

- Data for each surface in the linked list includes
  - RGB intensity components
  - opacity parameter (percent of transparency)
  - Depth
  - percent of area coverage
  - surface identifier
  - other surface-rendering parameters
  - pointer to next surface

A- Buffer Method

- The A-buffer can be constructed using methods similar to those in the depth-buffer algorithm.
- Scan lines are processed to determine surface overlaps of pixels across the individual scan lines.
- Surfaces are subdivided into a polygon mesh and clipped against the pixel boundaries.
- Using the opacity factors and percent of surface overlaps, we can calculate the intensity of each pixel as an average of the contributions from the overlapping surfaces.

Scan Line Method

- This method is is an extension of the scan-line algorithm for filling polygon interiors.
- Instead of filling just one surface, we now deal with multiple surfaces. As each scan line is processed, all polygon surfaces intersecting that line are examined to determine which are visible.
- Across each scan line, depth calculations are made for each overlapping surface to determine which is nearest to the view plane.
- When the visible surface has been determined, the intensity value for that position is entered into the refresh buffer.

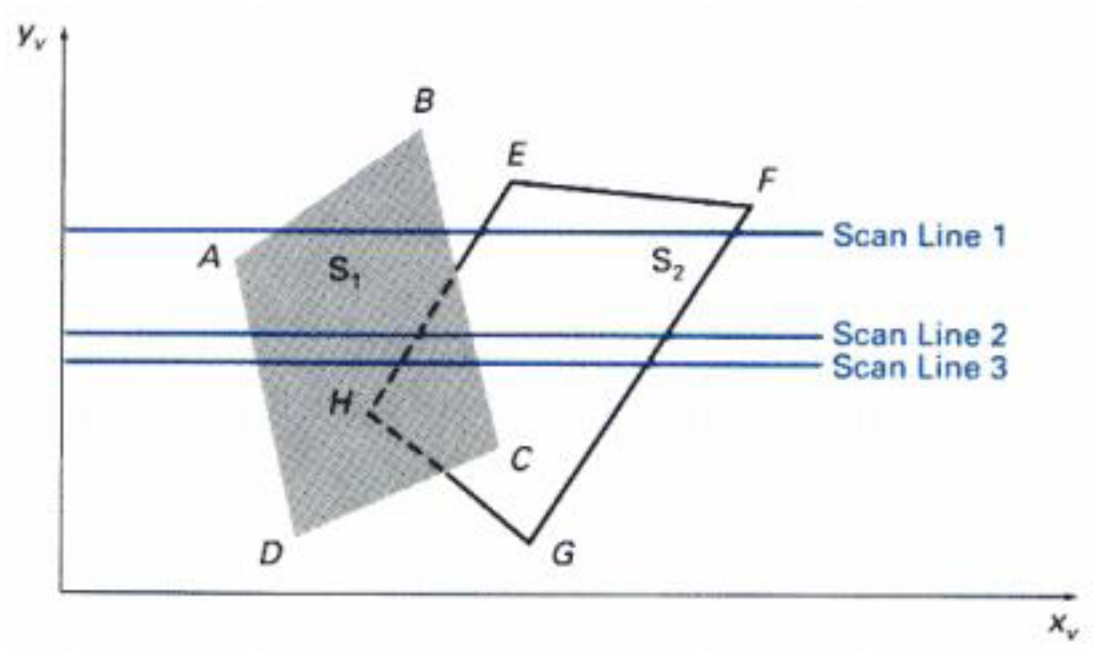
Scan-Line Method

- The tables are set up for the various surfaces which include both an edge table and a polygon table.
- The edge table contains coordinate endpoints for each line in the scene, the inverse slope of each line, and pointers into the polygon table to identify the surfaces bounded by each line.
- The polygon table contains intensity information of the surface.

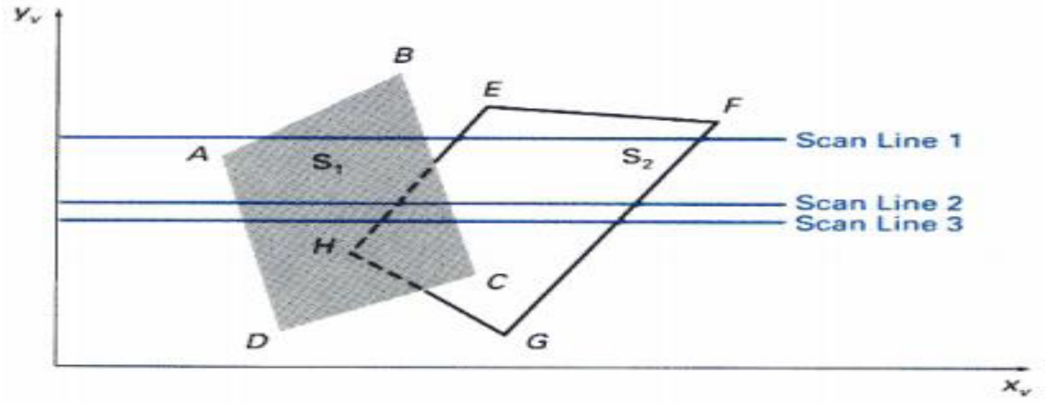
Scan-Line Method

- Active edge list is setup to facilitate the search for surfaces crossing a given scan line.
- This active list will contain only edges that crosses the current scan line sorted in increasing x.
- In addition a flag for each surface is set on or off to indicate the position along a scan line is inside or outside of surface.
- Scan lines are processed left to right. At leftmost boundary the flag is turned on and at rightmost boundary it is turned off.

Scan-Line Method

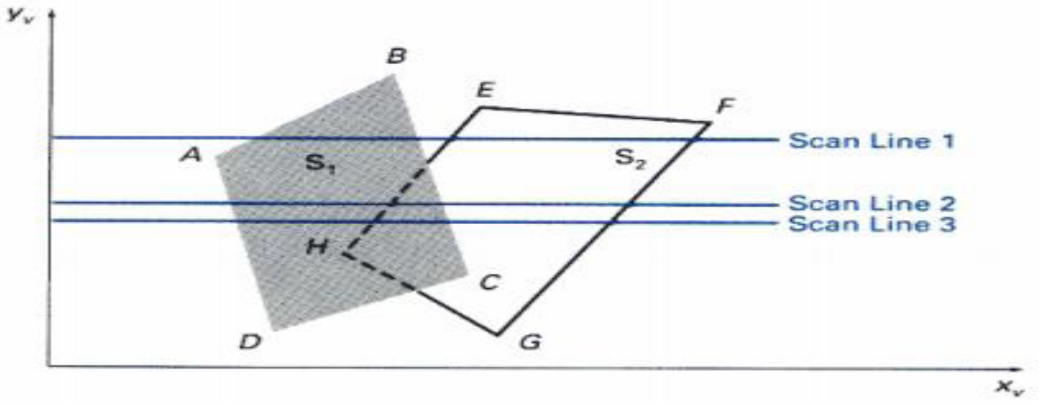


Scan-Line Method



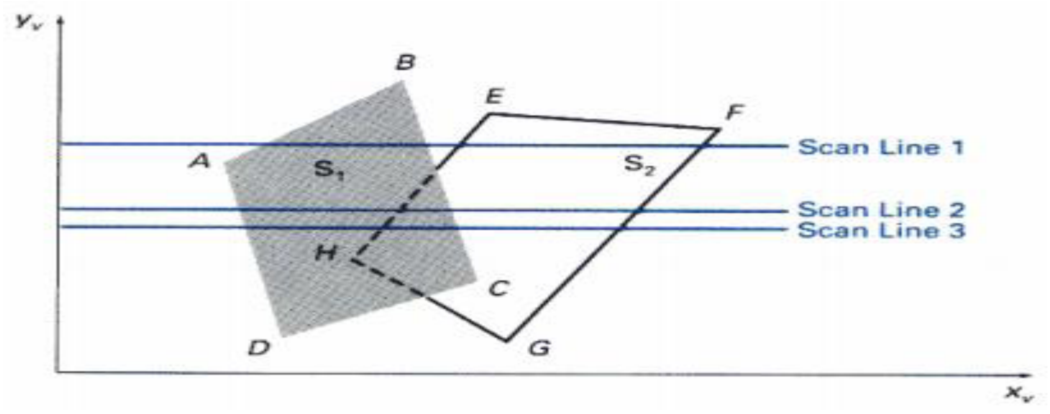
- The active list from the scan line 1 contains information from the edge table for edges AB, BC, EH and FG.
- For positions along this scan line between edges AB and BC, only flag for surface s1 is on. Therefore no depth calculations are necessary and intensity information for surface S1 is entered from the polygon table into refresh buffer.
- Between Edges EH and FG, only the flag for surface S2 is on.
- No other positions along scan line 1 intersect the surfaces, so the intensity values in the other areas are set to the background intensity.
- The background intensity can be loaded throughout the buffer in an initialization routine.

Scan-Line Method



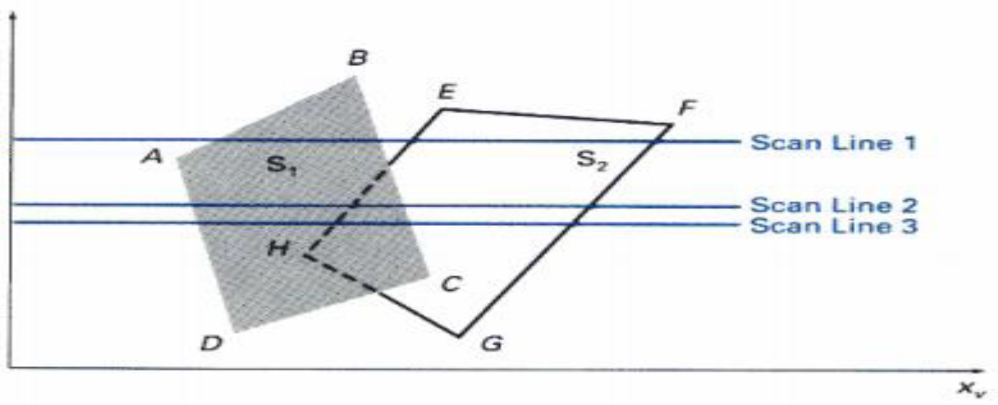
- For scan lines 2 and 3 in Fig, the active edge list also contains edges *AD, EH, BC,* and *FG.*
- Along scan line 2 from edge *AD* to edge *EH,* only the flag for surface S1 is on. But between edges EH and *BC,* the flags for both surfaces are on.
- In this interval, depth calculations must be made using the plane coefficients for the two surfaces.
- For this example, the depth of surface S1 is assumed to be less than that of S2, so intensities for surface S1 are loaded into the refresh buffer until boundary BC is encountered.
- Then the flag for surface S1 goes off, and intensities for surface *S2* are stored until edge FG is passed.

Scan-Line Method



- We can take advantage of-coherence along the scan lines as we pass from one scan line to the next.
- In Fig scan line 3 has the same active list of edges as scan line 2.
- Since no changes have occurred in line intersections, it is unnecessary again to make depth calculations between edges *EH* and *BC.*
- The two surfaces must be in the same orientation as determined on scan line 2, so the intensities for surface S1 can be entered without further calculations.

Scan-Line Method



- Any number of overlapping polygon surfaces can be processed with this scan-line method.
- Flags for the surfaces are set to indicate whether a position is inside or outside, and depth calculations are performed when surfaces overlap.
- When these coherence methods are used, we need to be careful to keep track of which surface section is visible on each scan line.