Chapter-6 Visible Surface Detection

6.1 Visible Surface Detection Methods (OR, Hidden surface elimination)

- When a picture that contains the Opaque (non-transparent) objects and surfaces are viewed, the objects that are behind the objects that are closer cannot be viewed.
- Surfaces which are obscured by other opaque surfaces along the line of sight (projection) are invisible to the viewer.
- To obtain a realistic screen image, these hidden surfaces need to be removed.
- This process of identification and removal of these surfaces is known as Hidden-surface problem.
- Visible surface detection or Hidden surface removal is major concern for realistic graphics for identifying those parts of a scene that are visible from a chosen viewing position.
- Several algorithms have been developed. Some require more memory, some require more processing time and some apply only to special types of objects.
- Deciding upon a method for a particular application can depend on such factors as the complexity of the scene, type of objects to be displayed, available equipment, and whether static or animated displays are to be generated.
- Visible surface detection methods are broadly classified according to whether they deal with objects or with their projected images.
    - **Object-Space method and**
    - **Image-space method**
- In physical coordinate system, object-space method is implemented and in case of screen coordinate system, image-space method is implemented.
- When a 3D object need to be displayed on the 2D screen, the parts of the screen that are visible from the chosen viewing position is identified.

Object Space Method (OSM)

- In this method, various parts of objects are compared.
- After comparison visible, invisible or hardly visible surface is determined.
- These methods generally decide visible surface.

- In the wireframe model, these are used to determine a visible line. So these algorithms are line based instead of surface based.
- Method proceeds by determination of parts of an object whose view is obstructed by other object and draws these parts in the same color
- An object-space method compares objects and parts of objects 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***

Image Space Method (ISM)

- In this method, positions of various pixels are determined.
- It is used to locate the visible surface instead of a visible line.
- Each point is detected for its visibility. If a point is visible, then the pixel is on, otherwise off. So the object close to the viewer that is pierced by a projector through a pixel is determined. That pixel is drawn is appropriate color.
- In an image-space algorithm, visibility is decided point by point at each pixel position on the projection plane. Most visible-surface algorithms use image-space methods.
- **E.g. Depth-buffer method, Scan-line method, Area-subdivision method**

List Priority Algorithms

- This is a hybrid model that combines both object and image precision operations.
- Here, depth comparison & object splitting are done with object precision and scan conversion (which relies on ability of graphics device to overwrite pixels of previously drawn objects) is done with image precision.
- **E.g. Depth-Sorting method, BSP-tree method**

Difference between OSM and ISM

Object Space	Image Space
1. Image space is object based. It concentrates on geometrical relation among objects in the scene.	1. It is a pixel-based method. It is concerned with the final image, what is visible within each raster pixel.

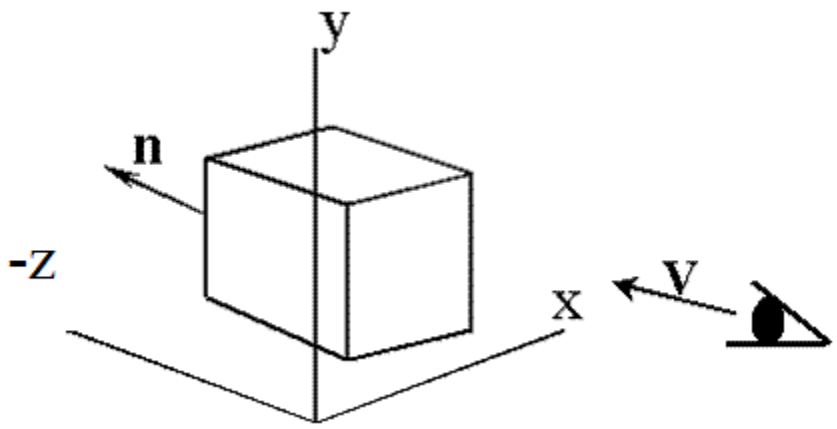
2. Here surface visibility is determined.	2. Here line visibility or point visibility is determined.
3. It is performed at the precision with which each object is defined, No resolution is considered.	3. It is performed using the resolution of the display device.
4. Calculations are not based on the resolution of the display so change of object can be easily adjusted.	4. Calculations are resolution base, so the change is difficult to adjust.
5. These were developed for vector graphics system.	5. These are developed for raster devices.
6. Object-based algorithms operate on continuous object data.	6. These operate on object data.
7. Vector display used for object method has large address space.	7. Raster systems used for image space methods have limited address space.
8. Object precision is used for application where speed is required.	8. There are suitable for application where accuracy is required.
9. It requires a lot of calculations if the image is to enlarge.	9. Image can be enlarged without losing accuracy.
10. If the number of objects in the scene increases, computation time also increases.	10. In this method complexity increase with the complexity of visible parts.

6.2 Back Face Detection Method

- When we project 3-D objects on a 2-D screen, we need to detect the faces that are hidden on 2D.
- Back-Face detection, also known as **Plane Equation method**
- It is an object space method in which objects and parts of objects are compared to find out the visible surfaces.

- Let us consider a triangular surface that whose visibility needs to decide. The idea is to check if the triangle will be facing away from the viewer or not. If it does so, discard it for the current frame and move onto the next one.
- Each surface has a normal vector. If this normal vector is pointing in the direction of the center of projection, then it is a front face and can be seen by the viewer.
- If this normal vector is pointing away from the center of projection, then it is a back face and cannot be seen by the viewer.
- A fast and simple object-space method for locating back faces
- A point (x,y,z) is "inside" a polygon surface with plane parameters A, B, C, 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 and so cannot be seen.

HOW TO DETECT?

- BY A SIMPLE TEST

If the z component of the normal vector is positive, then, it is a back face. If the z component of the vector is negative, it is a front face.

Principle:

i. Remove all surfaces pointing away from the viewer

ii. Eliminate the surface if it is completely obscured by other surfaces in front of it Render only the visible surfaces facing the viewer
iii. Back facing and front facing faces can be identified using the sign of V •N where V is the view vector and N is normal vector (N= Ai+Bj+Ck)

- We can simplify this test by considering the normal vector N to a polygon surface which has Cartesian components (A, B, C).
- There are two types of algorithm system to calculate V.N (V DOT N), i.e. ***Right handed system and left handed system.*** You can choose any one of them.

Algorithm For left Handed system

1. Start
2. Compute N for every face of the polygon
3. Check following

If (V • N > 0) then back face

else if (V • N < 0) then front face

else if (V • N= 0) then on line of view

else (Undefined)

4. End

Conclusion: if the Z component of the normal vector is positive, then it is a back face. If the Z component of the vector is negative, then it is a front face.

Algorithm for right handed system

1. Start
2. Compute N for every face of the polygon
3. Check following condition

If (V • N < 0) then back face

else if (V • N >0) then front face

else if (V • N= 0) then on line of view

else (Undefined)

4. End

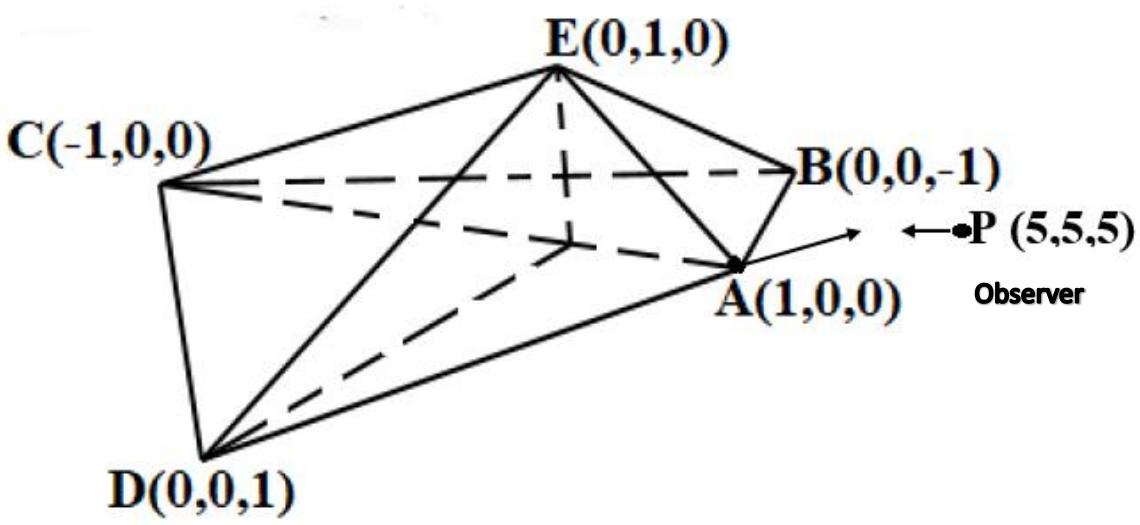
Conclusion: if the Z component of the normal vector is negative, then it is a back face. If the Z component of the vector is positive, then it is a front face.

Limitations of BDM

- This method works fine for convex polyhedra, but not necessarily for concave polyhedra.
- This method can only be used on solid objects modeled as a polygon mesh.

Numerical:

Q. Find the visibility for the surface AED where an observer is at P (5, 5, and 5).



Solution:

Here,

AE = (0-1) i + (1-0) j + (0-0) k = - i + j
AD= (0-1) i + (0-0) j + (1-0) k = - i + k

Step-1: Normal vector N for AED

Thus, N = AE x AD is calculated as

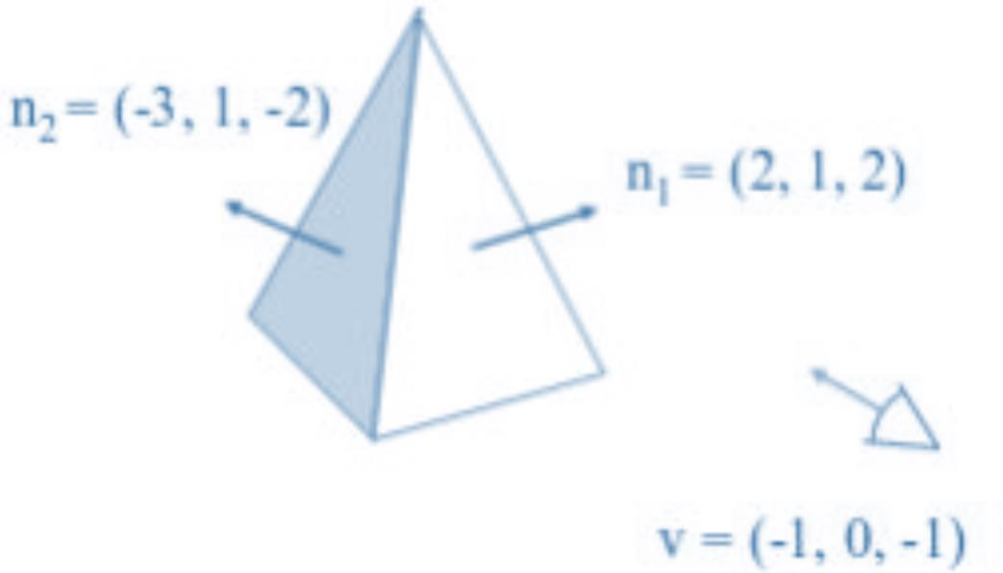
$$\begin{pmatrix} i & j & k \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} = i(1-0) - j(-1+0) + k(0+1)$$
$$= i + j + k$$

Step-2: If observer at P (5, 5, 5) so we can construct the view vector V from view point A (1, 0, 0) as: V = PA = (1-5) i + (0-5) j + (0-5) k = -4i - 5j - 5k

Step-3: To find the visibility of the object, we use dot product of view vector V and normal vector N as: V • N = (-4i - 5j – 5k) • (i + j + k) = -4-5-5 = -14< 0

This shows that the surface is visible for the observer.

Q. Find the visibility of n1 and n2 from V.



Solution

$N_1 \bullet v = (2, 1, 2) \cdot (-1, 0, -1) = -2 - 2 = -4,$

i.e $N_1 \bullet v < 0$

So, N_1 front facing polygon

$N_2 \bullet v = (-3, 1, -2) \cdot (-1, 0, -1) = 3 + 2 = 5$

i.e $N_2 \bullet v > 0$

So N_2 back facing polygon

6.3 Depth-Buffer Method

- It is also known as **z-buffer method**.
- Commonly used image-space method for detecting visible surface
- Developed by Edwin Catmull
- It is an image-space approach.
- The basic idea is to test the Z-depth of each surface to determine the closest (visible) surface.
- It compares surface depths at each pixel position on the projection plane.
- It is called z-buffer method since object depth is usually measured from the view plane along the z-axis of a viewing system.

  OR

  In this method each surface is processed separately one pixel position at a time across the surface. The depth values for a pixel are compared and the closest (smallest z) surface determines the color to be displayed in the frame buffer.

How this method works?

- ✓ With object description converted to projection co-ordinates, each (x,y,z) position on 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 depth is compared by z-values.

- ✓ It is applied very efficiently on surfaces of polygon. Surfaces can be processed in any order. To override the closer polygons from the far ones, two buffers are used.

  1. **Depth buffer / (z- Buffer)** is used to store depth values for (x, y) position, as surfaces are processed ($0 \leq$ depth ≤ 1).

2. **The frame buffer/ (refresh / image buffer)** is used to store the intensity value for each position (x, y).

The z-coordinates are usually normalized to the range [0, 1]. The 0 value for z-coordinate indicates back clipping pane and 1 value for z-coordinates indicates front clipping pane.

Process:

i. Initially all the positions in depth buffer are set to 0, and refresh buffer is initialize to background color.
ii. Each surfaces listed in polygon table are processed one scan line at a time by calculating the depth (z-value) for each position (x, y).
iii. The calculated depth is compared to the value previously stored in depth buffer at that position.
iv. If calculated depth is greater than stored depth value in depth buffer, new depth value is stored and the surface intensity at that position is determined and placed in refresh buffer.

Algorithm:

Start

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

i.e. Depthbuffer (x,y) =0, Refreshbuffer (x, y) = $I_{background}$.

2. For each position on each polygon surface, compare depth values to previously stored value in depth buffer to determine visibility.

2.1 Calculate the depth ***Z*** for each (x, y) position on polygon

2.2 If *Z* > depth (x, y) then set depth (x, y) = Z, refresh (x, y) = $I_{surface(x, y)}$

Where, $I_{background}$ = Intensity value for background color
$I_{surface\ (x, y)}$ = Intensity value for surface color at pixel position (x, y) on projected plane.

3. Repeat step 2 for all surfaces

4. After all surfaces are processed, the depth buffer contains the depth value of the visible surface and refresh buffer contains the corresponding intensity values for those surfaces.

5. End

How to calculate depth value / Z- value?

The depth values of the surface position (x, y) are calculated by plane equation (Ax + By + Cz + D = 0) of surface then,

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

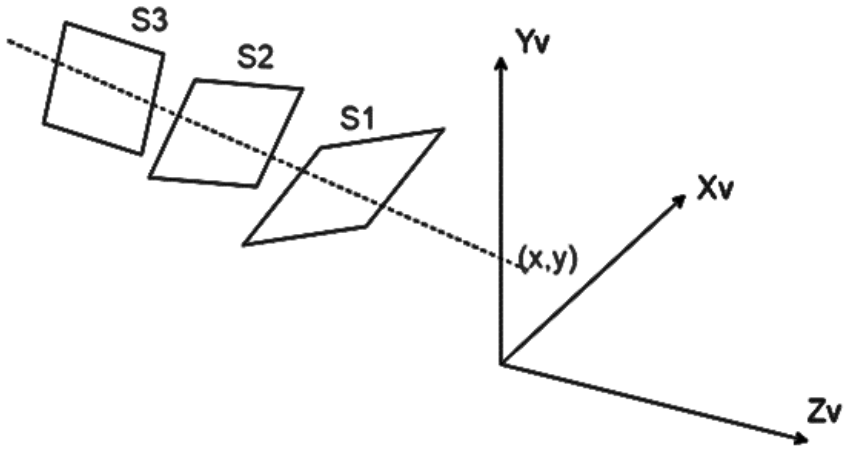
Let Depth *Z'* at position (x+1, y)

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

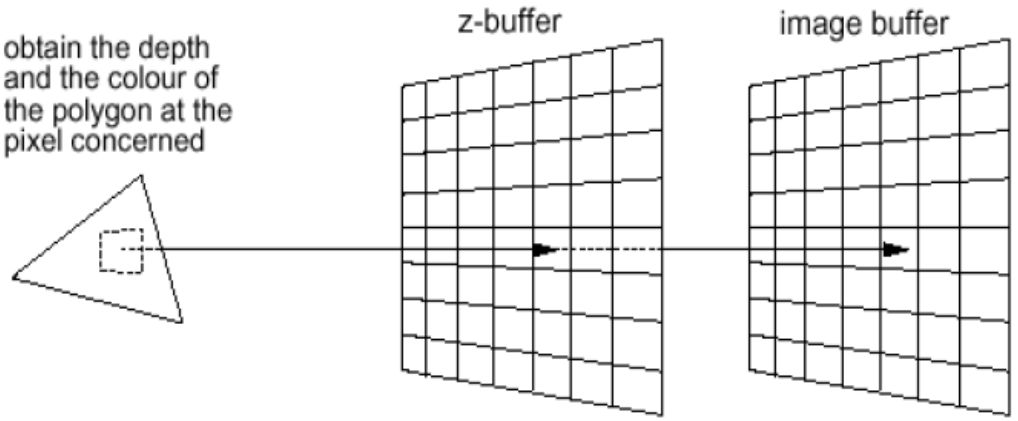
$$\Rightarrow Z' = Z - A/C$$

Where, A/C is constant for each surface so succeeding depth value across a scan line are obtained from preceding values by simple calculation.

Example.



- In above figure, at view plane position (x, y), the surface S_1 has the smallest depth from the view plane so is visible at that position.
- Also in figure, Three surfaces at varying distance from view plane (x_v, y_v), the projection along (x, y) surface S1 is closest to the view-plane so surface intensity value of S1 at (x, y) is saved.



Advantages of Z-buffer method

- It is easy to implement.
- It reduces the speed problem if implemented in hardware.
- It processes one object at a time.

Disadvantages of Z-buffer method

- It requires large memory.
- It is time consuming process

6.4 Scan Line Method

- It is an image space algorithm.
- It processes one line at a time rather than one pixel at a time.
- This algorithm records **edge list, active edge list**. So accurate bookkeeping is necessary.
- The edge list or edge table contains the coordinate of two endpoints.
- Active Edge List (AEL) contain edges a given scan line intersects during its sweep.
- The active edge list (AEL) should be sorted in increasing order of x. The AEL is dynamic, growing and shrinking.
- It is an extension of the scan-line algorithm for filling polygon interiors where, we deal with multiple surfaces rather than one.
- Each scan line is processed with calculating the depth for nearest view for determining the visible surface of intersecting polygon.
- When the visible surface has been determined, the intensity value for that position is entered into the refresh buffer.
- To facilitate the search for surfaces crossing a given scan line, we can set up an **active list** of edges from information in the edge table that contain only edges that cross the current scan line, sorted in order of increasing x.
- In addition, we define a **flag** for each surface that is set on or off to indicate whether a position along a scan line is inside or outside of the surface.
- Scan lines are processed from left to right. At the leftmost boundary of a surface, the surface flag is turned on; and at the rightmost boundary, it is turned off.

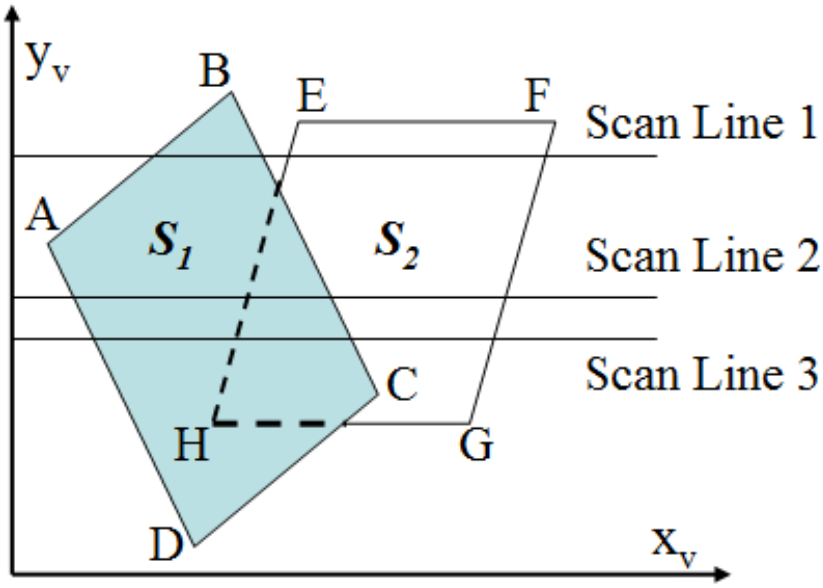


Fig. Scan lines crossing the projection of two surfaces, S_1 and S_2 in the view plane. Dashed lines indicate the boundaries of hidden surfaces.

- **The active list for 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 the 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 the refresh buffer.
- Similarly, between edges EH and FG, only the flag for surface S2 is on. NO other positions along scan line-1 intersect surfaces, so the intensity values in the other areas are set to the background intensity.

For scan lines 2 and 3 (line 2 and 3 have same intersection point and plane so no need to proceed 3 if 2 is done)

- The active edge list 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.

6.5 A-Buffer Method:

- Also known as **anti-aliased** or **area-averaged** or **accumulation buffer**
- Hidden face detection method suited to medium scale virtual memory computers.
- Extension of **depth-buffer (or Z Buffer)** method.
- Developed at Lucas film Studios for the rendering system "***Renders Everything You Ever Saw***" (REYES).
- As the depth buffer method can only be used for opaque object but not for transparent object, the A-buffer method provides advantage in this scenario.
- A buffer method requires more memory, but different surface colors can be correctly composed using it.
- A buffer method is slightly costly than Z-buffer method because it requires more memory in comparison to the Z-buffer method.
- It proceeds just like the depth buffer algorithm.
- Being a descendent of the Z-buffer algorithm, each position in the buffer can reference a linked list of surfaces.
- In A-buffer method, each pixel is made up of a group of sub-pixels. The final color of a pixel is computed by summing up all of its sub-pixels. Due to this accumulation taking place at sub-pixel level it is called **accumulation buffer**.
- The key data structure in the A buffer is the **accumulation buffer**.

Each position in the A buffer has 2 fields

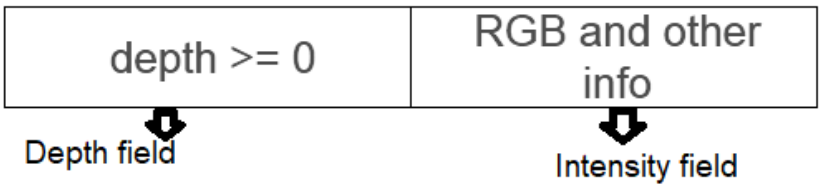
1. ***Depth field***

   - stores a positive or negative real number

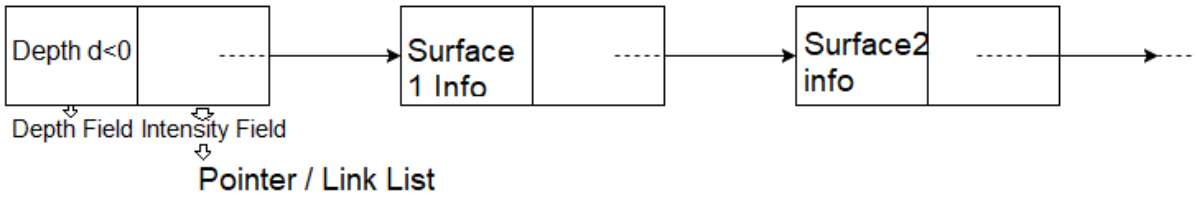
2. ***Surface data field or Intensity field***

   - Stores surface intensity information or a pointer value to a linked list of surfaces that contribute to that pixel position.

- **If depth >= 0,** single surfaces exist i.e. the number stored at that position is the depth of a single surface overlapping the corresponding pixel area.



- **If depth < 0,** multiple surface exist i.e. multiple-surface contributions to the pixel intensity. The intensity field then stores a pointer to a linked list of surface data



- As shown in the above figure, **multiple-surface contributions** to the pixel intensity is indicated by depth < 0. The 2nd field, i.e., the intensity field then stores a pointer to a linked list of surface data

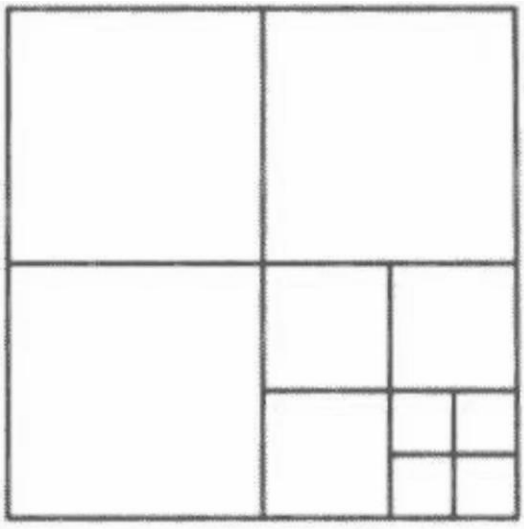
The surface buffer in the A buffer method includes:

- Depth
- Surface Identifier
- Opacity Parameter
- Percent of area coverage
- RGB intensity components
- Pointer to the next surface

6.6 Area Subdivision Method

- Uses image space method
- Also called a **Warnock Algorithm**
- It is based on a divide & conquer method.
- It uses fundamental of area coherence.
- It is used to resolve the visibility of algorithms.

How it works?



1. Divide the total viewing area into smaller and smaller rectangles until each small area is the projection of part of a single visible surface or no surface at all.

2. Continue this process until the subdivisions are easily analyzed as belonging to a single surface or until they are reduced to the size of a single pixel.

Process

- Successively divide the area into four equal parts at each step.
- There are **four possible relationships** that a surface can have with a specified area boundary.

    1. **Surrounding surface** − One that completely encloses the area.
    2. **Overlapping surface** − One that is partly inside and partly outside the area.
    3. **Inside surface** − One that is completely inside the area.
    4. **Outside surface** − One that is completely outside the area.

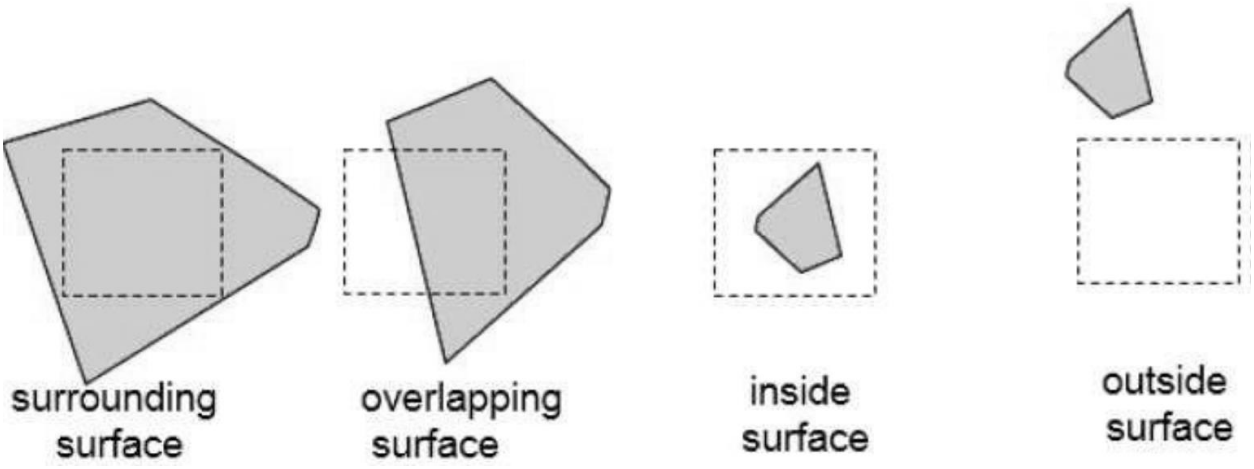


Figure: Area boundary relationship in Area Subdivision Method

The tests for determining surface visibility within an area can be stated in terms of these four classifications. No further subdivisions of a specified area are needed if one of the following conditions is true:

1. All surfaces are outside surfaces with respect to the area.
2. Only one inside, overlapping or surrounding surface is in the area.
3. A surrounding surface obscures all other surfaces within the area boundaries

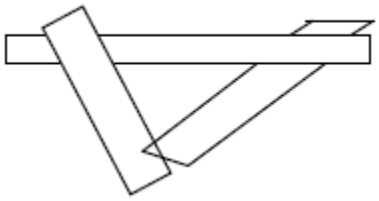
6.7 Depth Sorting Method

- This method uses both object space and image space methods.
- In this method the surface representation of 3D object are sorted in of decreasing depth from viewer.
- Then sorted surface are scan converted in order starting with surface of greatest depth for the viewer.
- This algorithm is also called "**Painter's Algorithm**" as it simulates how a painter typically produces his painting by starting with the background and then progressively adding new (nearer) objects to the canvas.

The conceptual steps that performed in depth-sort algorithm are

1. Surfaces are sorted in order of decreasing depth (z-coordinate).
2. Resolve any ambiguity this may cause when the polygons z-extents overlap, splitting polygons if necessary.
3. Surfaces are scan converted in order, starting with the surface of greatest depth.

Problem: One of the major problems in this algorithm is intersecting polygon surfaces. As shown in fig. below.



- Different polygons may have same depth.
- The nearest polygon could also be farthest.
- We cannot use simple depth-sorting to remove the hidden-surfaces in the images.

Solution:
For intersecting polygons, we can split one polygon into two or more polygons which can then be painted from back to front. This needs more time to compute intersection between polygons. So it becomes complex algorithm for such surface existence

Example (view details in the book of Hearn and Baker)

Assuming we are viewing along the z axis. Surface S with the greatest depth is then compared to other surfaces in the list to determine whether there are any overlaps in depth. If no depth overlaps occur, S can be scan converted. This process is repeated for the next surface in the list. However, if depth overlap is detected, we need to make some additional comparisons to determine whether any of the surfaces should be reordered.

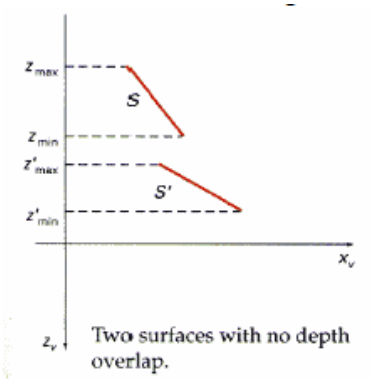
We make the following tests for each surface that overlaps with S. If any one of these tests is true, no reordering is necessary for that surface. The tests are listed in order of increasing difficulty.

1. The bounding rectangles in the xy plane for the two surfaces do not overlap

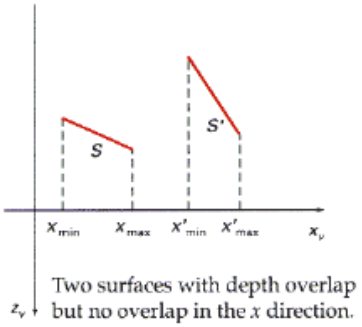
2. Surface S is completely behind the overlapping surface relative to the viewing position.

3. The overlapping surface is completely in front of S relative to the viewing position.
4. The projections of the two surfaces onto the view plane do not overlap.

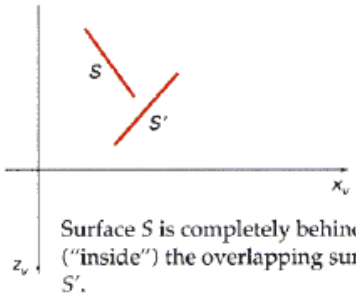
We perform these tests in the order listed and proceed to the next overlapping surface as soon as we find one of the tests is true. If all the overlapping surfaces pass at least one of these tests, none of them is behind S. No reordering is then necessary and S is scan converted.



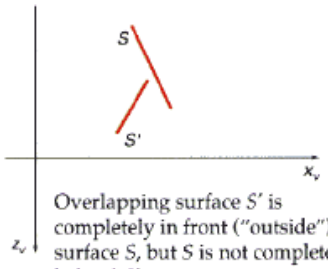
Two surfaces with no depth overlap.



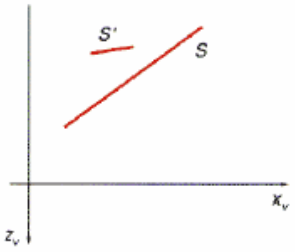
Two surfaces with depth overlap but no overlap in the x direction.



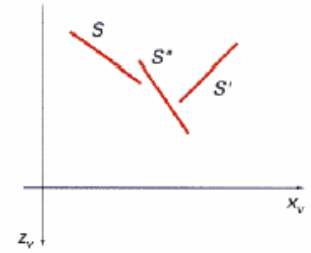
Surface S is completely behind ("inside") the overlapping surface S'.



Overlapping surface S' is completely in front ("outside") of surface S, but S is not completely behind S'.



Surface S has greater depth but obscures surface S'.



Three surfaces entered into the sorted surface list in the order S, S', S'' should be reordered S', S'', S.

****End of Chapter****