Chapter 6
Visible Surface Detection (Hidden Surface Elimination)

Introduction

- A scene to be displayed consists of collection of objects such that objects which are in line of sight of other objects will block those objects from view.
- For generating **a realistic display**, it is necessary to display only those surfaces/parts of scene that are visible and eliminate the hidden surface/parts from a chosen viewing position. This in turn, reduces the size of **memory requirement**, because no need of store all surfaces, only the visible surface is stored.
- The problem of identification of such surfaces and removal of these surfaces is called as **hidden surface problem**. The algorithm that deals with such problem are **called visible surface detection or hidden surface elimination.**
- Several algorithms have been developed for this. Some require more memory, some require more processing time and some apply only to special types of objects.

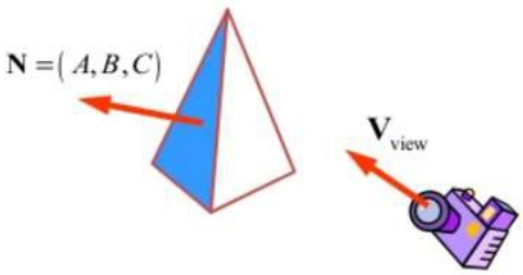
Fig: Hidden surface displayed Fig: Hidden surface Removed

- Hidden surface detection methods are broadly classified into following two categories.
  - **i. Object-space Method**
    - This algorithm deals with object definition i.e. considers the geometrical relationships between the actual objects in world coordinate.
    - Compares objects and parts of objects to each other within the scene definition to determine which surface as a whole we should label as visible
    - Object space methods are generally used inline display algorithm.
  - **ii. Image-space Method**
    - This algorithm deals with the projected images
    - Visibility is decided point by point at each pixel position on the projection plane.
    - Most visible surface detection algorithm uses image space method.

Back-Face Detection (Plane Equation Method)

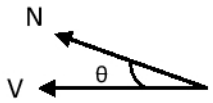
- Back-face detection is a fast **object space algorithm** based upon the **inside outside test** for identifying the **back face** of a solid object.
- In a solid object, there are some surfaces which are facing toward the viewer (front faces) and there are some surfaces which are opposite to the viewer (back faces).
- Suppose a point P (X, Y, Z) is inside a polygon surface with plane parameter A, B, C and D if

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



- Now, in this test, we consider the normal vector N to a polygon surface which has Cartesian components (A, B, C). Then, if V is the vector in viewing direction from the eye position then this polygon surface is
  - **back face** if, V.N > 0
  - front face if, V. N< 0
  - on line of view if V.N = 0

  where V. N= |V| . |N| . Cos(θ), and θ is the angle between V and N.
- So if V and N are **orthogonal/Perpendicular** (the angle between vectors is 90°) i.e. V and N are in line of view then due to cos(90°), **V. N=0**



- Hence if the dot product is positive, we can say that polygon faces away from the viewer and should be removed.

Where applicable?

- For a single convex polyhedron, such as the pyramid like in above figure, this test identifies all the hidden surfaces on the object, since each surface is either completely visible or completely hidden.
- Also, if a scene contains only non-overlapping convex polyhedron, then again all hidden surfaces are identified with the back-face method.

Where not applicable?

- This method works fine for convex polyhedron, but not necessarily for concave. For other objects, such as the concave polyhedron, more tests need to be carried out to determine whether there are additional faces that are totally or partially obscured by other faces. And a general scene can be expected to contain overlapping objects along the line of sight. We then need to determine where the obscured objects are partially or comp1etely hidden by other objects.

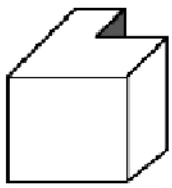


Fig: View of a concave polyhedron with one face partially hidden by other faces

Numerical:

Consider a surface defined by vertex A (1,0,0), B (0, 0, -1), C (-1, 0, 0) and D (0, 0, 1). If the observer is at (2, 2, 2), is the face ABCD is visible or back face?

For solution, see class Notes:

Depth Buffer Algorithm / Z-Buffer Algorithm

- Depth buffer algorithm is a commonly used image space approach **to determine visible surface**.
- This algorithm compares the surface depths at each pixel position on the projection plane.
- Since object depth is usually measured from the view plane along the z-axis of a viewing system, this procedure is also called as Z-buffer method.
- In this method two buffer are required: Z buffer and Frame buffer

- Z-buffer is extension of frame buffer (store the intensity of each pixel) that stores the z-coordinate or depth of every pixel in the image space.

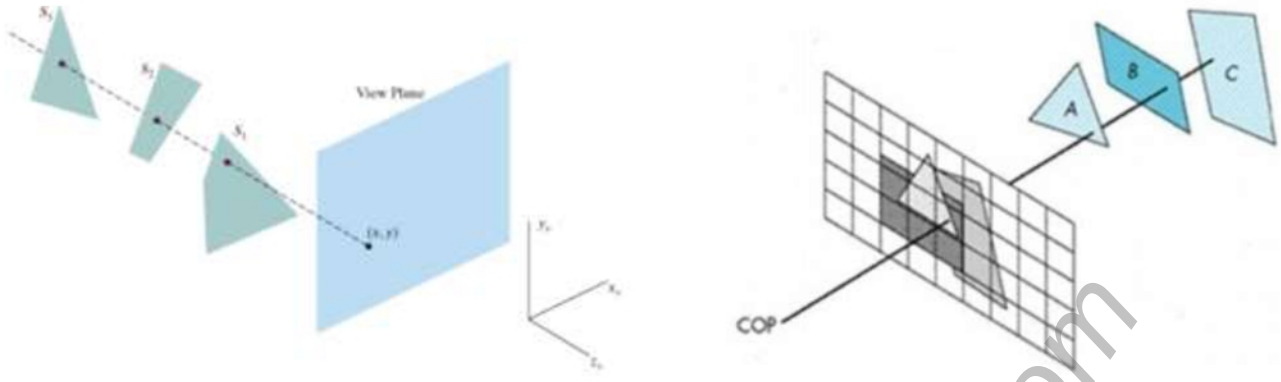


Fig: A view plane position (x,y), surface s1 has
Smallest depth from the view plane so it is visible
At that position

Fig: Z-buffer Technique

- In this algorithm a buffer of same size as the frame buffer is set up. Each element of depth buffer corresponds to a pixel in the frame buffer and initially holds the maximum depth in the scene.
- As each polygon is scan converted, the depth at each pixel is calculated and compared with the corresponding depth in the depth buffer. If the depth is less than that stored in the depth buffer (i.e. nearer to viewer) then that pixel is set in the frame buffer with the polygon color at that point and the depth buffer is set to the polygon depth. If the polygon depth is greater (i.e. farther away from the viewer) then the depth buffer depth at that point, then that pixel in not written to the frame buffer.
- If the equation of plane is ax + by + cz + d=0, then the depth at any point (x, y) where y is the scan line and x is the position in scan line, is given as

$$Z = -\frac{(ax+by+d)}{c}$$

- Figure 1 shows three surfaces: s1, s2 and s3, at varying distances along the orthographic projection line from position (x, y) in view plane taken as the xy plane. Surface s1, is closest at this position, so its surface intensity value at (x, y) is saved for display.
- The z-coordinates are usually normalized to the range [0, 1].

Algorithm

- Step-1: Initialize the buffer values so that for all buffer position (x,y)
  Depthbuffer (x, y) = 1.0
  Framebuffer (x, y) = background color
- Step-2: Process each polygon surface one scan line at a time.
  For each projected (x, y) pixel position of a polygon, calculate depth z.
  If Z < Depthbuffer (x, y)
  - set depthbuffer (x, y) = z
  - compute surface color at positon (x, y)
  - framebuffer (x, y) = surfacecolor (x, y)

Advantage

- Overcome the problem associated with back face detection method.
- It is easy to implement.
- It can be implemented in hardware to overcome the speed problem.

- Diversity of primitives: not only polygon
- Applicable to complex scene with overlapping object.

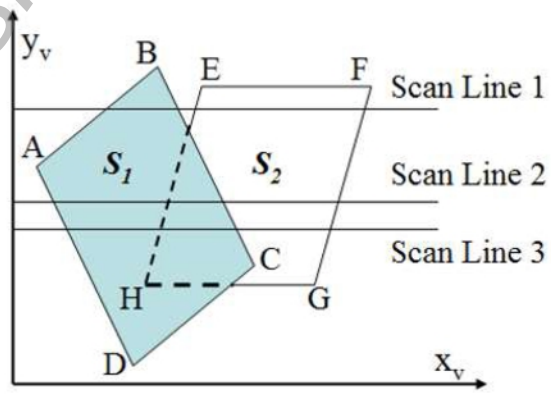
Disadvantage

- It requires and additional buffer and hence large memory.
- It is time consuming process as it draws hidden objects also.
- Since the algorithm processes objects one at a time, the total number of polygon in a picture can be arbitrary large and therefore may require lots of computation.
- Applicable to only opaque surface (Solution: A-buffer method)

Scan Line Method

- In Z buffer method, we process each surface one at a time but using scan line we **can process more than one surface at a time.**
- Scan line method is image space method for **removal of hidden surface** which is an extension of 2D scan line filling algorithm. This method deals with **multiple surfaces**.
- In this method, 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 positons is entered into the refresh or frame buffer.
- In this technique, three tables are used for the various surfaces which includes edge table, active edge table and polygon surface table.
- **Edge table** stores the edge information of the all the polygon surfaces in the scene such as endpoint for each line in scene, slope of line, pointers into the polygon table to identify the surface bounded by line.
- **Polygon table stores** surface information such as coefficient of plane equation, intensity information and pointer information into the edge table.
- **Active edge** table stores active list of edges from information in the edge table. Active list will 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.

How it works?



- In the figure above, three scan line named scanline1, scanlin2 and scanline3 are considered.
- The **active edge table** for **scan line 1** contains information from edge table for edges AB, BC, EH and FG.
- For the position along the scan line between edge AB and BC, only the flag for surface S1 is on, so no depth calculation is necessary. Hence only the information for surface s1 is entered form the polygon table into the refresh buffer. Similarly, between edge EH and FG, only the flag for surface s2 is on, so no

depth calculation is necessary. Hence only the information for surface s2 is entered from the polygon table. For rest positions along the scan line doesn't intersect any surfaces, so the intensity values for other areas are set to background intensity.

- The **active edge table for scan line 2** and scan line 3 contains information from edge table for edges AD, EH, BC, FG.
- Along scan line 2, from edge AD to EH, only the flag for surface s1 is on but between the edge EH and BC, the flag for both the surfaces are on. In this interval, depth calculations must be made using the plane coefficient for the two surfaces. Here, 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 only surface s2 goes on and intensities for surface S2 are stored until FG edge is passed.

A-Buffer (Accumulation Buffer) Method

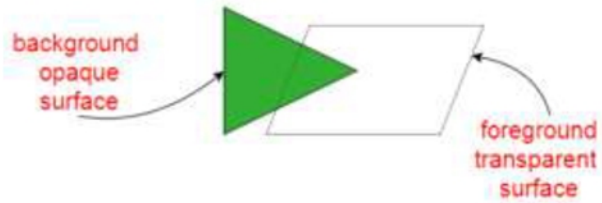
- A-Buffer method in computer graphics is a general hidden face detection mechanism **which extends the algorithm of Z-buffer method.**
- This method is also known as **anti-aliased** or **area-averaged** or **accumulation buffer**.
- As the depth buffer method can only be used for **opaque** object but not for **transparent object**, the A-buffer method provides advantage in this scenario.
- Being a descendent of the Z-buffer algorithm, each position in the buffer can reference a **linked list** of surfaces.
- The key data structure in the A buffer is the accumulation buffer.
- Each position in the A buffer has two fields
  1. Depth field
  2. Surface data field or Intensity field
- A depth field stores a positive or negative real number. A surface data field can store surface intensity information or a pointer to a linked list of surfaces that contribute to that pixel position

(a) When a pixel overlap by only one surface

- As shown in the above figure, if the value of depth is >= 0, the number stored at that position is the **depth of single surface** overlapping the corresponding pixel area. The 2nd field, i.e., the intensity field then stores the RGB components of the surface color at that point and the percent of pixel coverage

(b) When a pixel overlaps by multiple surfaces

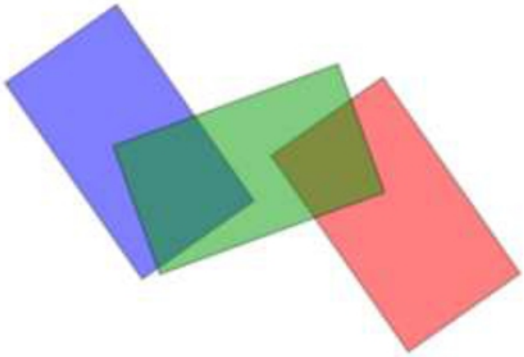
- As shown in the above figure**, multiple-surface contributions** to the pixel intensity is indicated by depth < 0. The 2nd field, i.e., the intensity field then stores a pointer to a linked list of surface data.
- A buffer method is slightly costly than Z-buffer method because it requires more memory in comparison to the Z-buffer method. It proceeds just like the depth buffer algorithm. Here, the depth and opacity are used to determine the final color of the pixel. As shown in the figure below, the A buffer method can be used to show the transparent objects.



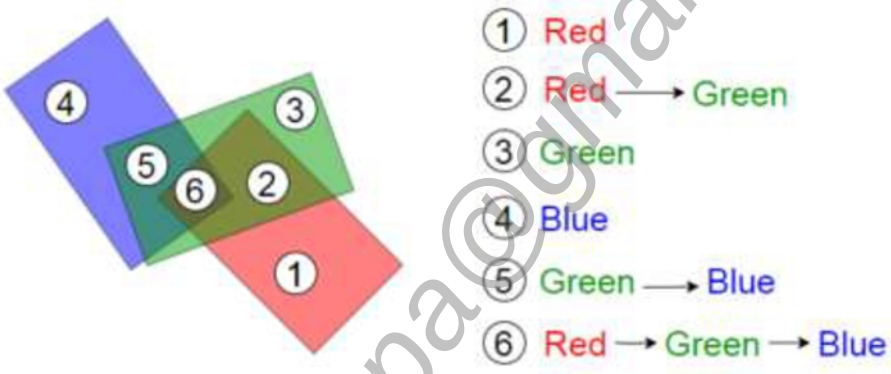
- The surface buffer in the A buffer method includes:
  1. Depth
  2. Surface Identifier
  3. Opacity Parameter

4. Percent of area coverage
5. RGB intensity components
6. Pointer to the next surface

- The other advantage of A buffer method is that it provides anti-aliasing in addition to what Z-buffer does. The usage of A-buffer algorithm for the **transparent surfaces** is as shown below:



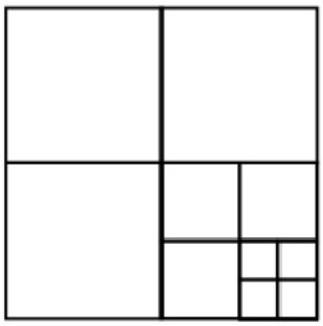
- On applying the A-buffer method on all the six surfaces indicated below, the corresponding colors are as:



- 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, A-buffer method gets the name **accumulation buffer**.

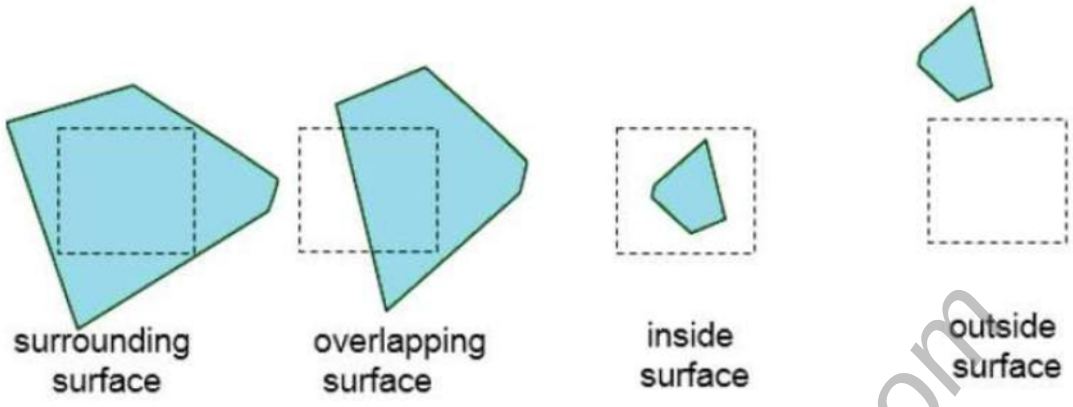
Area subdivision Method (Warnock Algorithm)

- Area subdivision method is a **divide and Conquer based** image space approach to hidden surface problem developed by Warnock.
- In this technique, each area is subdivided into quadrants (Represented by **Quad tree**) until rectangle contain part of one projected surface or rectangle contains part of no surfaces or rectangle is size of pixel.



- At each stage in the **recursive subdivision** process, the relationship between projection of each polygon and the area of interest is checked for four possible relationships.
  1. Surrounding Polygon: one that completely encloses the area of interest.
  2. Overlapping or intersection polygon: one that is partly inside and partly outside that area.

3. Inside or contained polygon: one that is completely inside the area.
4. Outside or disjoint polygon: one that is completely outside the area.



- After checking relationship, we can handle each relationship as follows.
  1. If all polygons are disjoint from area, then background color is displayed in the area.
  2. If there is only one intersecting or only one contained polygon, then the area is first filled with the background color and then the part of the polygon contained in the area is filled with the color of polygon.
  3. If there is a single surrounding polygon but no intersection or contained polygons, then the area is filled with the color of surrounding polygon.
  4. If there are more than one polygon intersection, contained in, or surrounding the area then all the polygon surfaces are ordered/sorted according to their depth from view plane. The polygon surface with maximum depth is scan converted first and then the nearer one and repeated until all the polygon surfaces are processed.
     Otherwise, the area is subdivided into four smaller parts and decision is made recursively.

Advantage

- Extra memory buffer is not required.
- It follows the divide and conquer strategy, therefore, parallel computer can be used to speed up the process.

List Priority Algorithms

- Determines a visibility ordering for objects ensures that a correct picture results if objects are rendered in that order.
- Depth sorting method and BSP tree methods fall under this.

Depth sorting method (Painters' Algorithm)

- As the name suggest, the algorithm follows the standard practice for painting. While painting, the artist first paints the background color, next the most distant objects are added then the nearer objects and so forth. The new painted layer covers the old so that only newest layer of the paint is visible.
- So this concept can be implemented on frame buffer for covering/hiding the hidden polygon surface by choosing the correct order to draw them.
- This method uses both object space and image space method.
- The basic idea of depth sort algorithm is to paint the polygons into the frame buffer in order of decreasing distance from view point i.e. first draw the distance object than the closer objects such that pixels of each object overwrites the previous object. This involves the following steps.
  - Sorting of polygons in order of decreasing depth.
  - Resolve any ambiguity this may cause when the polygons Z extents overlap, splitting polygons if necessary.
  - Scan conversion of polygon in order, starting with the polygon of greatest depth.

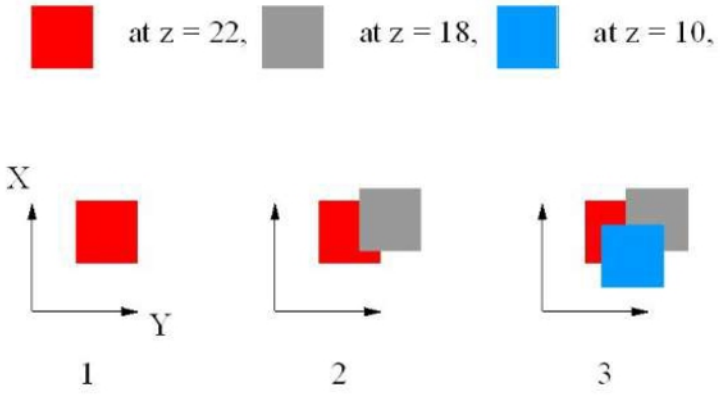
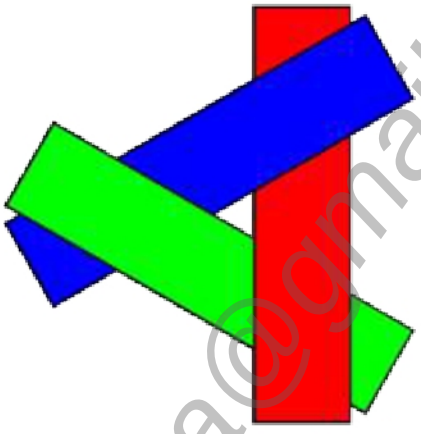


Fig: Painters algorithm

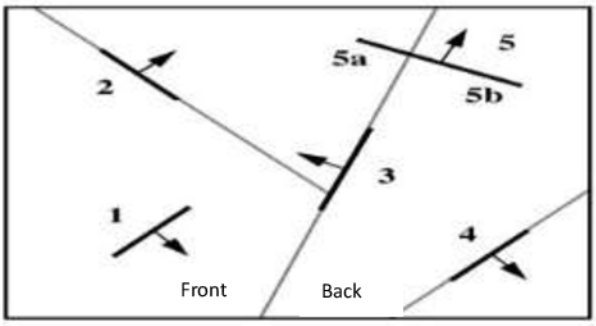
- A problem arises in this algorithm in case of intersecting polygon surfaces. as shown in fig. below.



- So in the above figure we see that different polygons may have same depth and the nearest polygon could also be farthest. So simple depth sorting technique can't be used to remove hidden surface.
- In order to solve this problem, we have to 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.

BSP tree method

- A binary space partitioning (BSP) tree is an efficient **object space** method for determining object visibility by painting surfaces onto the screen from back to front as in the painter's algorithm.
- The BSP tree is particularly useful when the view reference point changes, but object in a scene are at fixed position.
- Applying a BSP tree to visibility testing involves identifying surfaces that are "front" or "back" the partitioning plane at each step of space subdivision relative to viewing direction.
- Following is the procedure to build a BSP tree in the object space.
  i. Select a polygon as the root of the tree i.e. the first partition plane
  ii. Partition the object space into two half spaces, so that some object polygons lie in the front half while the other in the rear(back) half with respect to the particular plane.
  iii. If the polygon is intersected by the partition plane, split it into two sub polygon so that they belong to different half spaces.
  iv. Select one polygon on the roots front as the left node and another on the roots back as the right node.
  v. Recursively subdivide the space considering the plane of the children's as partition planes until a subspace contains a single polygon.



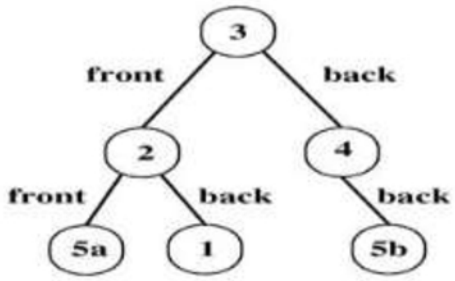


Fig: BSP tree considering plane 3 as root node

- So once the BSP tree is complete, we follow the painting order i.e. back to front to process the tree.
- If the viewer is in the root polygons front half space, then the algorithm first displays all the polygon in the roots rear half space then the root itself and finally all the polygons in its front half i.e. the order of processing polygons in BSP tree is

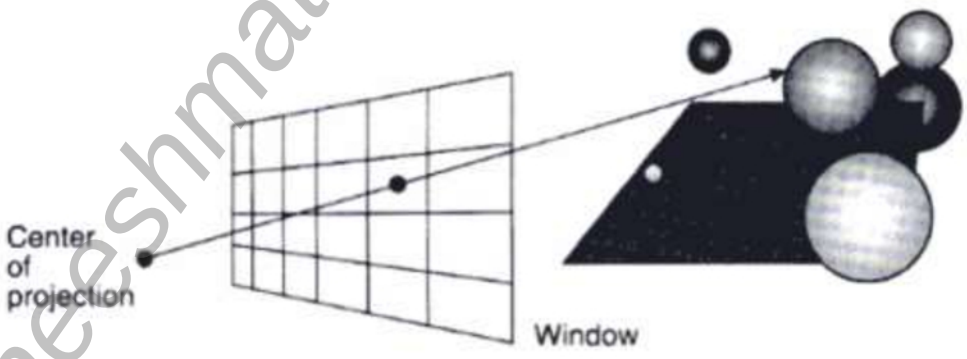
  i.e. 5b, 4, 3, 1, 2 and 5a
- Hence foreground object are painted over background object.

Assignment

1. Image Space Vs Object Space Method for Hidden Surface Detection

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.

2. Briefly explain Ray tracing algorithm for visible surface detection.
   - Ray tracing also known as ray casting is efficient method for visibility detection in the objects.
   - It can be used effectively with the object with both flat(polygon) and curved surface.
   - The algorithm is based on the principle of light and optics.



   - A ray is fired from the center of projection(eye) through each pixel in windows to which the window map to determine the closest surface intersected. So we set the pixel to the color of the surface at the point where the light ray strikes it.
   - The main problem with this technique is that it is too slow for real time application.