Unit-7

Visible Surface Detection:

Visible surface detection is the process of identifying those parts of a scene that are visible from a chosen viewing position. There are numerous algorithms for efficient identification of visible objects for different types of applications. These various algorithms are referred to as visible-surface detection methods or also referred as hidden-surface elimination methods.

When a picture contains non-transparent objects, the surfaces behind the objects can not be viewed. To obtain a realistic screen image, the hidden surfaces need to be removed. This process of identification and removal of these surfaces is known as hidden-surface problem. These processes helps the computer to decrease processing time for rendering image, eliminating extra information of image.

The hidden surface problems can be solved by two methods:- 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.

(less imp concept only)

Coherence for visibility:

Coherence denotes similarities between items or entities. It describes the extent to which these items or entities are locally constant. We may use knowledge about similarities between areas within the scene to make large reductions in the number of calculations. Such an approach is known as 'coherence' and it occurs in many forms.

i) Object coherence → If two objects are completely seperate, then we may apply global tests when comparing them rather than testing each vertex of an object.

ii) Face coherence → Even when the faces are general surfaces and not plane, it is usual for properties of the face to vary smoothly across it.

iii) edge coherence → In edge coherence object changes visibility only when it crosses behind a visible edge or intersects a visible face.

iv) Frame coherence → Pictures of the same scene at two successive times will usually have small changes with much of the scene unchanged. This can be used to reduce the amount of data that has to be calculated each time.

ⓐ Object-space Methods:- 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. This method can be used efficitively to locate visible surfaces in some cases. Line-display algorithms generally use object-space methods to identify visible lines in wire-frame displays. This method is easy to implement but slow than other methods. This method is implemented in physical coordinate system.

Basic Procedure:

For each object in the scene do
1. Begin
   - Determine those parts of the object whose view is unblocked by other parts of it or any other object with respect to the viewing specification.
   - Draw those parts in the object color.
2. End.

Characteristics:

i) If there are n objects in the scene, complexity = $O(n^2)$.
ii) Calculations are performed at the resolution in which the objects are defined.
iii) Display is more accurate but computationally more expensive compared to image-space method.
iv) It is suitable for scene with small number of objects with simple relationship with each other.
v) Process is unrelated to display resolution or the individual pixel in the image.

(b). Image-space method:- In this method visibility is decided by point at each pixel position on the projection plane. This method can be easily implemented to visible-line detection. It is implemented in screen coordinate system. Most of the hidden line/surface algorithms use image-space methods.

Basic Procedure

For each pixel in the image do

1. Begin
   - Determine the object closest to that pixel.
   - Draw the pixel in the object colour.
2. End.

Characteristics:-

i) If there are p pixels in the image, complexity depends on n and p O(np).

ii) For each pixel, determine all n objects to determine the one closest to the viewer.

iii) Accuracy of the calculation is bounded by the display resolution.

iv) A change of display resolution requires re-calculation.

object space method

(*). Back Face Detection:- (Plane Equation method)

In this method objects and parts of objects are compared to find out the visible surfaces. The idea is to check if the object will be facing away from the viewer or not. If it does so, discard it from current frame and move onto the next one.

Each surface has a normal vector. If this normal points in the direction of center of projection then it is front face and can be seen by viewer. If this normal is pointing away from the direction of center of projection then it is back face and cannot be seen by viewer.

A point (x, y, z) is inside a polygon surface if;

$$A_x + B_y + C_z + D < 0.$$

We can simplify this test by considering normal vector $\ddot{N}$ to a polygon surface which has cartesian components (A, B, C). If V is the vector in viewing direction from the eye position then this polygon is a back face if, $V \cdot N > 0$.

In the equation $A_x + B_y + C_z + D = 0$, if A, B, C remains constant, then varying value of D results in a whole family of parallel planes. One of which $(D=0)$ contains the origin of the coordinate system and,

- If $D>0$, plane is away from the observer.
- If $D<0$, plane is towards the observer.

If the object is centered at origin, the all surface that are viewable will have negative D and un-viewable surface have positive D.

<u>Limitations</u>

→ 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.

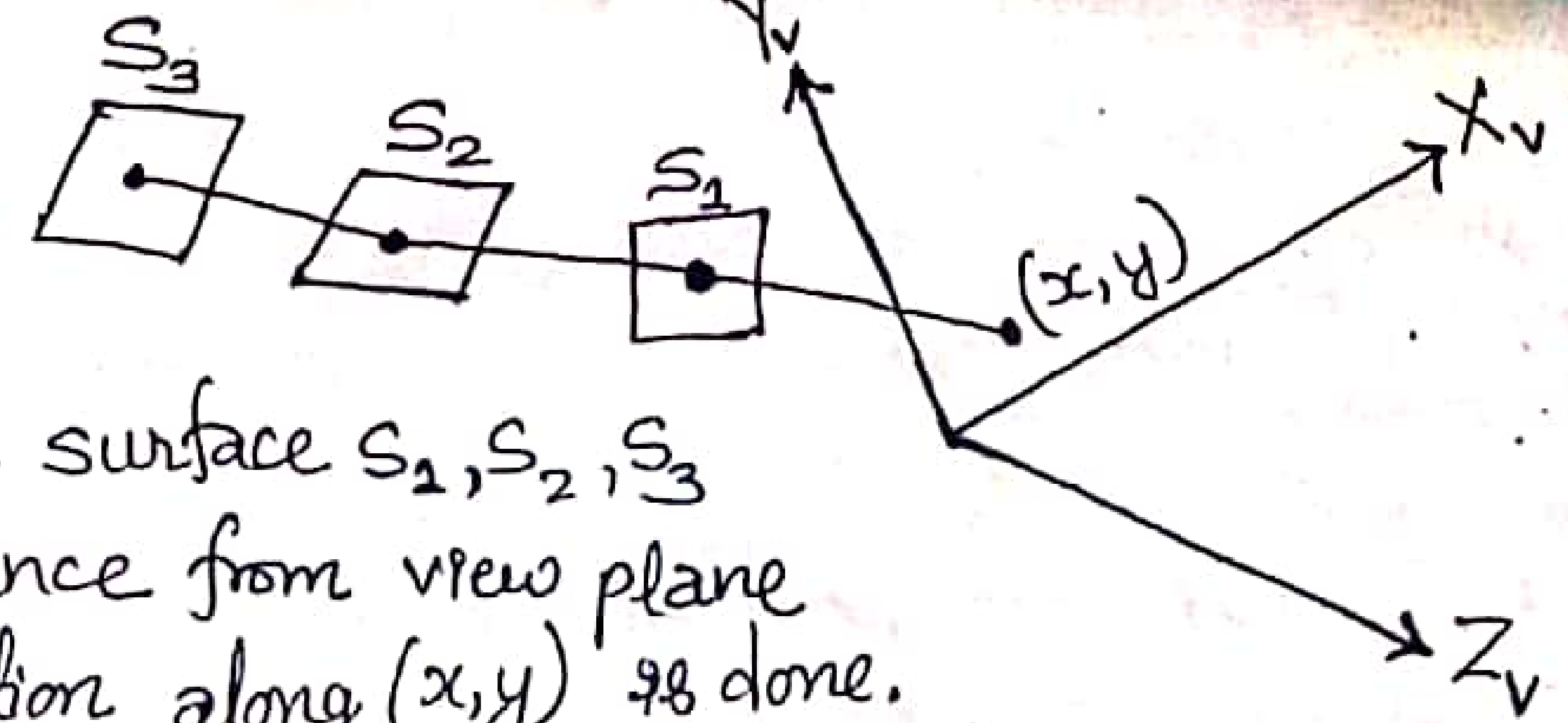
→ With this method we can not identify visible surface in case of overlapping objects

⊛. Most Popular Methods for Image-Space Method:-

1) Depth Buffer (Z-buffer): (Imp)

In this method two buffers are used named as frame buffer and depth buffer. Depth buffer is used to store depth values for (x, y) position, as surfaces are processed $(0 \leq depth \leq 1)$. The frame buffer is used to store the intensity value of color value at each position (x, y).

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



In figure, three surface S_1, S_2, S_3 at varying distance from view plane $X_v Y_v$ the projection along (x,y) is done.
Surface S_1 is closest to the view-plane so surface intensity value of S_1 at (x,y) is saved.

Initially all the positions in depth buffer is set to 0 and refresh buffer is initialized to background color. Each surface listed in polygon table is processed one scan line at a time, calculating the depth (z-value) for each position (x,y). The calculated depth is compared to the value previously stored in depth buffer at that position. 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.

After all surfaces are processed, the depth value of surface position (x,y) is calculated by plane equation surface.

$$Z = \frac{-A_x - B_y - D}{C}$$

Let depth Z' at position $(x+1, y)$

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

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

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

Algorithm:

1. Start
2. Set the buffer values
   - Depthbuffer $(x,y)=0$
   - framebuffer (x,y) = background color.
3. Process each polygon (One at a time).
   i) For each projected (x,y) pixel position of a polygon, calculate depth z.
   ii) If $Z >$ depth buffer(x,y).
      - Compute surface color.
      - set depthbuffer $(x,y)=z$.
      - framebuffer (x,y) = surfacecolor(x,y).
4. Stop.

Implementation steps for Depth buffer

1. Initially each pixel of the z-buffer is set to the maximum depth value.
2. The image buffer is set to the background color.
3. Surfaces are rendered (provided) one at a time.
4. For the first surface, the depth value of each pixel is calculated.
5. If this depth value is smaller than the corresponding depth value in the z-buffer then both the depth value in the z-buffer and the color value in the image buffer are replaced by the depth value and the color value of this surface calculated at the pixel position.
6. Repeat step 4 and 5 for the remaining surfaces.
7. After all the surfaces have been processed, each pixel of the image buffer represents the color of a visible surface at that pixel.

Advantages:

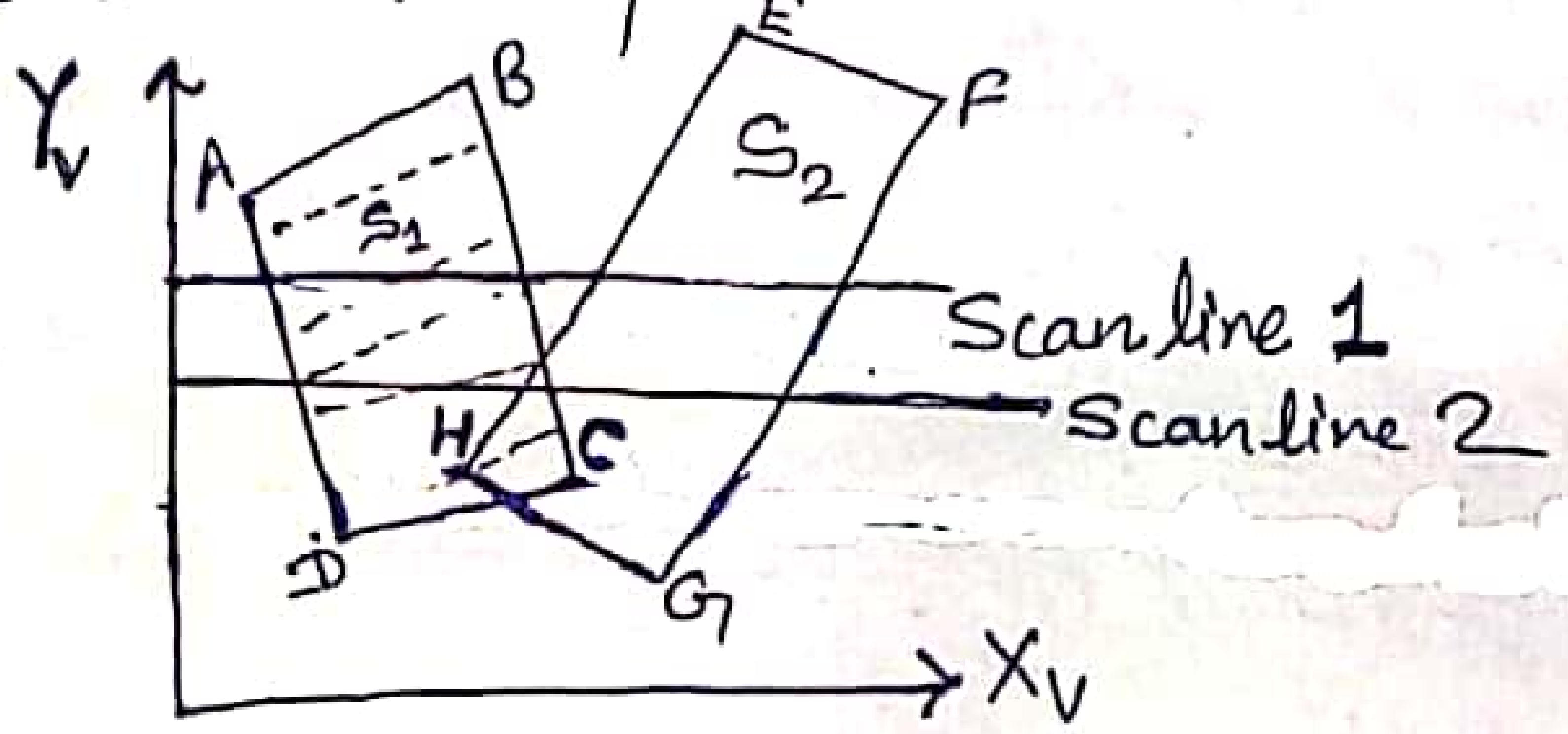
- → Simple and does not require additional data structures.
- → No object-comparision required.
- → Processes one object at a time.
- → Can be applied to non-polygonal objects.

Disadvantages:

- → This method requires additional buffer
- → It requires large memory.
- → It is time consuming process.

2) Scan-Line Method:

In this method, 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 ~~line~~ the view plane.



active edge → Scan line ले काटेको side हेर्नु

ON → indicate a scan line is inside of surface

OFF → indicate a scan line is outside of surface.

Note: – Select surface according to choosen sides

Working Procedure:-

The active list for scan line one contains information from the edge table for edges AD, BC, EH and FG.

→ For scan line 1, between edges AD and BC, flag (S_1) = ON and flag (S_2) = OFF. Since no overlapping case so, no depth calculations are required and intensity information for surface S_1 is entered from the polygon table into the refresh buffer.
Similarly, between edge EH and FG flag (S_1) = OFF and flag (S_2) = ON.

→ For scan line 2, the active edges AD, EH, BC and FG. Along the line 2, from edge AD to edge EH, flag (S_1) = ON and flag (S_2) = OFF. But between edges EH and BC, flag (S_1) = ON and flag (S_2) = ON, Since this is the case of overlapping. Therefore depth calculation must be made using the plane coefficient for the two surfaces.

For this example the depth surface S_1 is assumed to be less than that of S_2. So intensity of surface S_1. ~~is assumed to be less than~~ are loaded into the refresh buffer until boundry BC is encountered. Then the flag for the surface S_1 goes off. The intensity for surface S_2 is stored until edge FG is passed.

Algorithm:

1. Start
2. Established data structure.
   a. Edge table with line endpoints.
   b. Polygon table with plane coefficients, color and pointer to the edge table.
   c. Active edge list sorted in increasing order of x.
   d. A flag for each surface.
3. Repeat for each scan line.
   a. Update AEL (active edge list) for each scan line y.
   b. When flag is ON, enter intensity of that surface to refresh buffer.
   c. When two or more flags are ON, calculate depth and store the intensity of the surface nearest to view plane.
4. Stop.

Characteristics:-

i) Processing of the scan line start from the top to the bottom of the display.
ii) It requires an edge table, polygon table, active edge list and flag.
iii) It deals with multiple polygons.
iv) This method calculates the z-value for only the overlapping surface which is tested by scan line.
v) Across each scan line, depth calculations are made for each overlapping surface to determine which surface is nearest to the view plane.

Advantages:-

→ Any number of overlapping polygon surfaces can be processed with this method.
→ Deals with transparent, translucent, and opaque surfaces.
→ Can be applied to non-polygonal objects.

Disadvantages:-

→ Depth calculations are performed only when there are overlapping polygons.
→ Additional memory buffer is required.
→ Complex.

⊛. Depth Sorting Method (Painter's Algorithm):

This method uses both object space and image space method. The depth sorting method performs following two basic functions:

i) First, the surfaces are sorted in order of decreasing depth.

ii) Second, the surfaces are scan-converted in order, starting with the surface of greatest depth.

The intensity values for farthest surface are entered into the refresh buffer. The farthest polygon is displayed first, then the second farthest polygon and so on. After all surfaces have been processed, the refresh buffer stores the final intensity values for all visible surfaces.

When there are only a few objects in the scene, then this method can be very fast. However, as the number of objects increases, the sorting process can become very complex and time consuming.

Advantages:

→ The sorting computation is very fast, so quick calculation of the image display is possible.

→ The finished image data remains object based and can be edited in terms of line weight, exploded views etc.

→ It will also print at the highest resolution available on devices such as postscript laser printers.

Disadvantages:

→ Different polygons may have same depth.

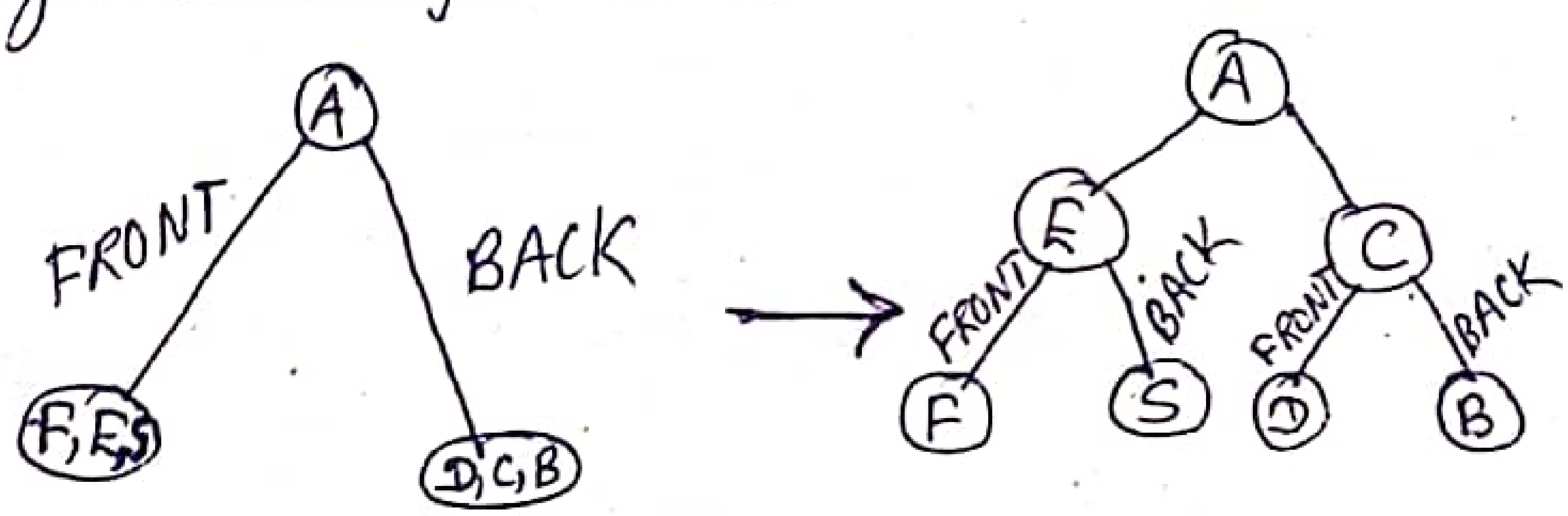
→ The nearest polygon could also be farthest.

→ We can not use simple depth-sorting to remove the hidden surfaces in the images.

Binary Space Partitioning (BPS) Tree Method:-

It is an enhancement of depth sorting method/algorithm that divides entire 3D scene into two parts by selecting any polygon surface as a dividing partitioning plane. This method is best suited when a 3D scene is static but, view port changes frequently. This method first displays polygon surfaces that are far from viewport and enters their intensity into frame buffer. It can be divided into two steps:

i) Construction of BSP tree → A space that contains 3D scene is recursively divided into two parts i.e, FRONT and BACK with respect to viewport, by selecting any surface as a dividing plane. The recursive subdivision process is repeated until each half space contains exactly one object or zero. This recursive division process is represented as binary tree with initial partitioning plane as a root node. FRONT object is represented as a left node of a tree and BACK object is represented as a right node of a tree.



ii) Display of BSP tree → If view port is infornt of root node, then BSP tree is displayed in reverse inorder traversal i.e, right, root and left.

B, C, D, A, S, E, F

If view port is in back of root node, then BSP tree is displayed in inorder traversal. i.e, left, root and right.

F, E, S, A, D, C, B

A-Buffer Method:-

It is the extension of Z-Buffer method that deals with all kinds of surfaces. It is called accumulation method because it can accumulate color information of all background surfaces that are visible through foreground transparent surface.

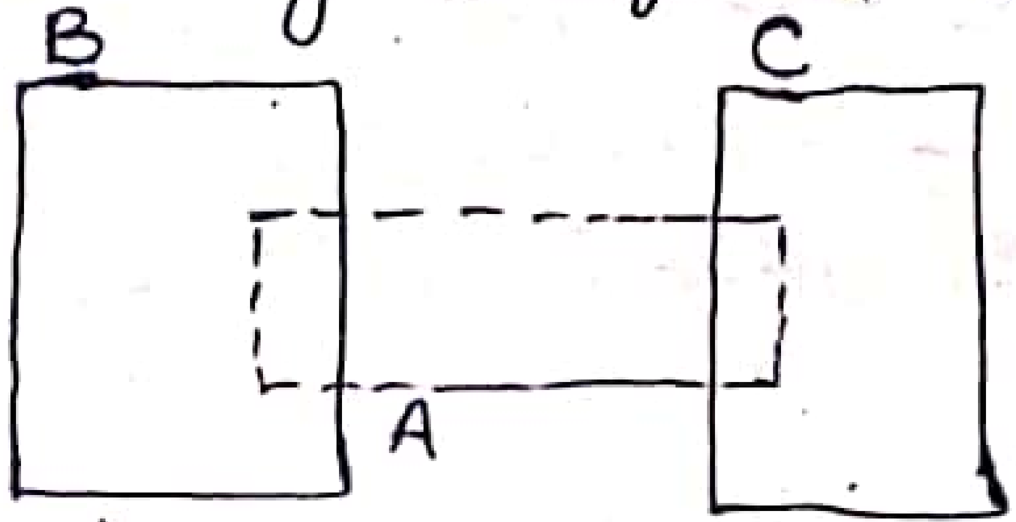


fig: Viewing two background surfaces B & C through a transparent surface A.

It extends frame buffer of foreground surface hence it can point all the background surfaces that are visible. A buffer has two fields;

Depth Field	Intensity field

i) Depth Field → It stores +ve and -ve depth value of a particular surface. If it is +ve then that surface is opaque surface otherwise it is translucent surface.

ii) Intensity Field → It stores surface intensity value or pointer to background surfaces.
- → If depth is +ve then intensity field contains color value of that surface.
- → If depth value is -ve then intensity field contains pointer to all visible background surfaces.
- → Intensity field can contain
  - • RGB information
  - • Surface Opacity
  - • Spatial orientation
  - • Pointer value to other surfaces.

Pros and Cons of A-Buffer

- → It deals with all types of surfaces.
- → Requires more computation than Z-buffer method.
- → Depth computation is always performed if single surface is visible surface.

Scanned with CamScanner

⊛ Ray Tracing Method:-

In this method rays are casted from each pixel to the scene and visible surfaces are identified based on nearest intersection point. The number of casted light rays depends on the no. of pixels in the viewplane i.e, resolution. This method is capable of producing high degree of visual realism which ~~makes~~ suits best to applications where image can be rendered slowly ahead of time such as still images, television, visual effects etc. It is capable of wide variety of optical effects such as reflection and refraction, scattering and dispersion phenomena.

Advantages:

→ Conceptually simple.
→ Automatic hidden surface removal
→ Ray-object insertion is fundamental in computer graphics like visibility testing.

⊛ Octree Method:-

An octree is a tree data structure in which each internal node has exactly eight children. Octrees are used to partition 3D space recursively subdividing it into eight octants. Octrees are often used in 3D graphics and 3D game engines.

When an octree representation is used for viewing volume, hidden surface elimination is accomplished by projecting octree nodes into viewing surface in a front to back order.

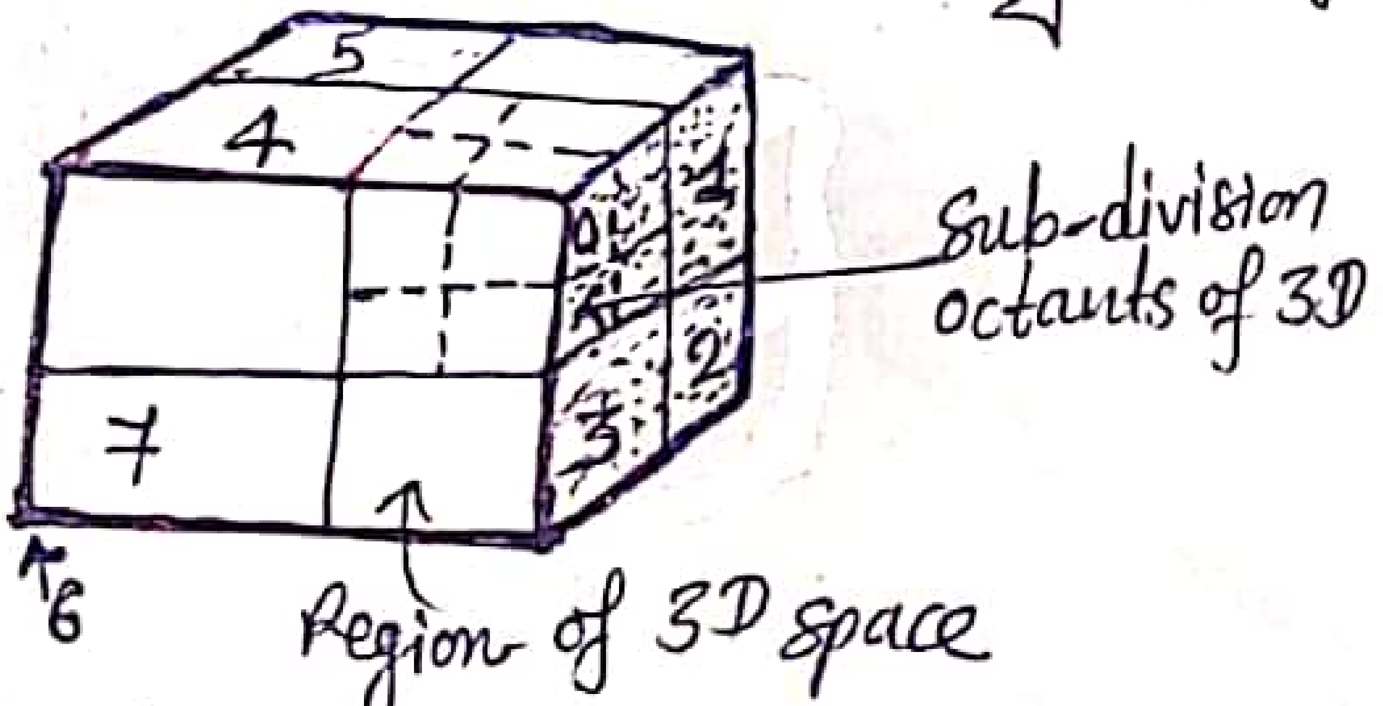


fig Front face with octants 0,1,2,3. (Visible to viewer).

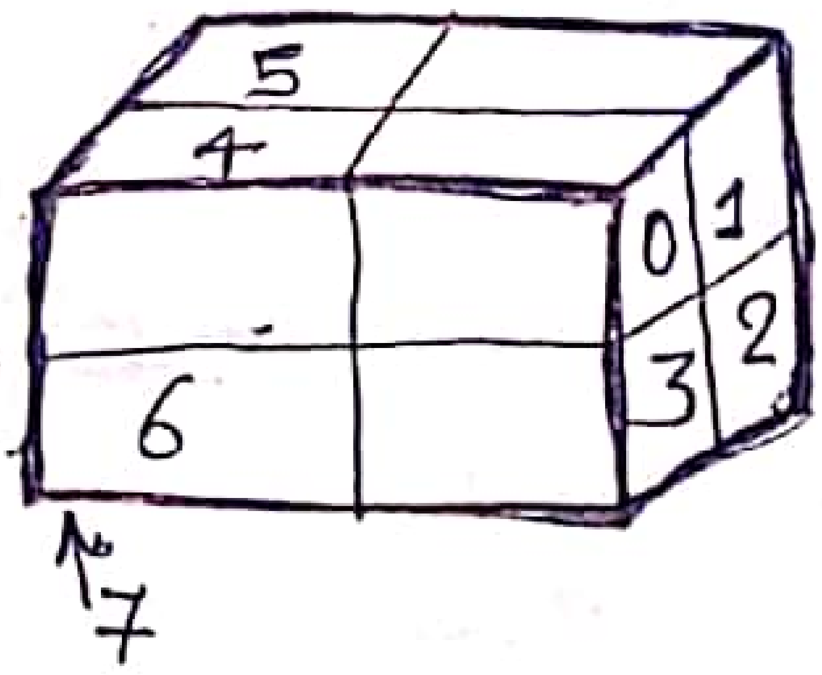


fig. Back face with octants 4,5,6,7 (Not visible to viewer)