

CGA Unit-4 notes

Q.1

Explain Z-buffer algorithm with advantages and disadvantages.

Depth Buffer Z – Buffer Method

This method is developed by Cutmull. It is an image-space approach. The basic idea is to test the Z-depth of each surface to determine the closest *visible* surface.

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 *smallestz* surface determines the color to be displayed in the frame buffer.

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 named **frame buffer** and **depth buffer**, are used.

Depth buffer is used to store depth values for x, y position, as surfaces are processed $0 \leq \text{depth} \leq 1$.

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.

Algorithm

Step-1 – Set the buffer values –

Depthbuffer $x, y = 0$

Framebuffer $x, y = \text{background color}$

Step-2 – Process each polygon *Oneatatime*

For each projected x, y pixel position of a polygon, calculate depth z .

If $Z > \text{depthbuffer } x, y$

 Compute surface color,

 set depthbuffer $x, y = z$,

 framebuffer $x, y = \text{surfacecolor } x, y$

Advantages

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

Disadvantages

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

Q.2

What are the basic tests in Warnock's algorithm? Explain.

Area sub-division method

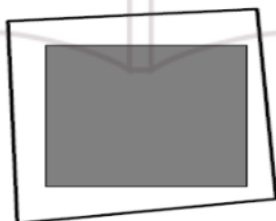
- Works in image-space
- Area Coherence is exploited
- Divide-and-conquer strategy.

Examine and divide if necessary

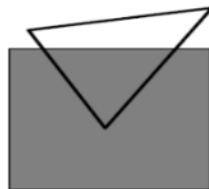
WARNOCK'S Algorithm

Possible relationships of the area of interest (rectangular) and the projection of the polygon:

1. Surrounding polygons are completely inside the area of interest.
2. Intersecting polygons intersect the area.
3. Contained polygons are completely inside the area
4. Disjoint polygons are completely outside the area.



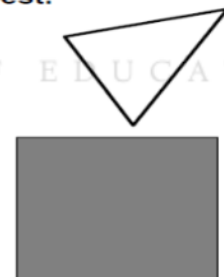
Surrounding



Intersecting



Contained



Disjoint

Treat interior part of the intersection and contained as equivalent. Disjoint is irrelevant. The decisions about division of an area is based on:

1. All polygons are disjoint from the area. Display background color.
2. Only one Intersecting part or interior (contained): Fill the area with background color and then scan-convert the polygon.
3. Single surrounding polygon, and no intersecting or contained polygon. Use color of surrounding polygon to shade the area.
4. More than one polygon intersect, contained in and surrounding the area. But the surrounding polygon is in front of all other polygons. Then fill the area with the color of the surrounding polygon.

Step 4 is implemented by comparing the Z-coordinates of the planes of all polygons (involved) at the four corners of the area

If all four tests fail, divide the rectangular area into four equal parts. Recursively apply this logic, till you reach the lowest minimum area or maximum resolution – pixel size. Use nearest surface in that case to paint/shade.

Q.3

Compare all visible surface detection methods.

<u>Comparison of VSD (HSR) techniques</u>				
Algorithms/ Methods	Memory	Speed	Issues in Implementation	Remarks
Z-Buffer	Two arrays	Depth complexity	Scan conversion, Hardware	Commonly used
Painter's	One array	Apriori sorting helps speed-up	Scan conversion	Splitting and sorting the major bottleneck
Ray casting	Object database	$O(\#pixels, \#surfaces \text{ or } objects)$	Spatial data structures help speedup	Excellent for CSG, shadows, transparency
Scanline, Area sub- division			Hard	Cannot be generalized for non- polygonal models

Q.4**11.1 Back-Face Detection**

In a solid object, there are surfaces which are facing the viewer (front faces) and there are surfaces which are opposite to the viewer (back faces).

These back faces contribute to approximately half of the total number of surfaces. Since we cannot see these surfaces anyway, to save processing time, we can remove them before the clipping process with a simple test.

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

The test is very simple, suppose the z axis is pointing towards the viewer, 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, it is a front face.

Note that this technique only caters well for nonoverlapping convex polyhedra.

For other cases where there are concave polyhedra or overlapping objects, we still need to apply other methods to further determine where the obscured faces are partially or completely hidden by other objects (eg., using Depth-Buffer Method or Depth-Sort Method).



Q.5

Types of coherence:

1. Object Coherence:

Visibility of an object can often be decided by examining a circumscribing solid (which may be of simple form, eg. A sphere or a polyhedron.)

2. Face Coherence:

Surface properties computed for one part of a face can be applied to adjacent parts after small incremental modification. (eg. If the face is small, we sometimes can assume if one part of the face is invisible to the viewer, the entire face is also invisible).

3. Edge Coherence:

The Visibility of an edge changes only when it crosses another edge, so if one segment of a non-intersecting edge is visible, the entire edge is also visible.

4. Scan line Coherence:

Line or surface segments visible in one scan line are also likely to be visible in adjacent scan lines. Consequently, the image of a scan line is similar to the image of adjacent scan lines.

5. Area and Span Coherence:

A group of adjacent pixels in an image is often covered by the same visible object. This coherence is based on the assumption that a small enough region of pixels will most likely lie within a single polygon. This reduces computation effort in searching for those polygons which contain a given screen area (region of pixels) as in some subdivision algorithms.

6. Depth Coherence:

The depths of adjacent parts of the same surface are similar.

7. Frame Coherence:

Pictures of the same scene at successive points in time are likely to be similar, despite small changes in objects and viewpoint, except near the edges of moving objects.

Most visible surface detection methods make use of one or more of these coherence properties of a scene.

To take advantage of regularities in a scene, eg., constant relationships often can be established between objects and surfaces in a scene.

Q.6

When we view a picture containing non-transparent objects and surfaces, then we cannot see those objects from view which are behind from objects closer to eye. We must remove these hidden surfaces to get a realistic screen image. The identification and removal of these surfaces is called **Hidden-surface problem**.

There are two approaches for removing hidden surface problems – **Object-Space method** and **Image-space method**. The Object-space method is implemented in physical coordinate system and image-space method is implemented in screen coordinate system.

When we want to display a 3D object on a 2D screen, we need to identify those parts of a screen that are visible from a chosen viewing position.

Two Main Approaches

Visible surface detection algorithms are broadly classified as:

- **Object Space Methods:** Compares objects and parts of objects to each other within the scene definition to determine which surfaces are visible
- **Image Space Methods:** Visibility is decided point-by-point at each pixel position on the projection plane

Image space methods are by far the more common

Object-Space Method

- Back-Face Detection
- BSP-Tree Method
- *Area-Subdivision Method*
- Octree Methods
- Ray-Casting Method

Image-Space Method

- Depth-Buffer Method
- A-Buffer Method
- Scan-Line Method
- *Area-Subdivision Method*

Classification of Visible-Surface Detection Algorithms:

1. Object-space Methods

Compare objects and parts of objects to each other within the scene definition to determine which surfaces, as a whole, we should label as visible:

For each object in the scene do

Begin

1. Determine those parts of the object whose view is unobstructed by other parts of it or any other object with respect to the viewing specification.
2. Draw those parts in the object color.

End

- Compare each object with all other objects to determine the visibility of the object parts.
- If there are n objects in the scene, complexity = $O(n^2)$
- Calculations are performed at the resolution in which the objects are defined (only limited by the computation hardware).
- Process is unrelated to display resolution or the individual pixel in the image and the result of the process is applicable to different display resolutions.
- Display is more accurate but computationally more expensive as compared to image space methods because step 1 is typically more complex, eg. Due to the possibility of intersection between surfaces.
- Suitable for scene with small number of objects and objects with simple relationship with each other.

2. Image-space Methods (Mostly used)

Visibility is determined point by point at each pixel position on the projection plane.

For each pixel in the image do

Begin

1. Determine the object closest to the viewer that is pierced by the projector through the pixel
2. Draw the pixel in the object colour.

End

- For each pixel, examine all n objects to determine the one closest to the viewer.
- If there are p pixels in the image, complexity depends on n and p ($O(np)$).
- Accuracy of the calculation is bounded by the display resolution.
- A change of display resolution requires re-calculation.

Application of Coherence in Visible Surface Detection Methods:

- Making use of the results calculated for one part of the scene or image for other nearby parts.
- Coherence is the result of local similarity
- As objects have continuous spatial extent, object properties vary smoothly within a small local region in the scene. Calculations can then be made incremental.

Q.7

Syllabus Topic : Curve Representation

9.1 Curve Representation

Q. Write a note on curve representation.

The objects which we can see around us is of different shapes either visible as 2D or 3D. Modeling of objects in computer graphics uses the primitives such as lines, circles, ellipse etc. Modeling of geometric objects generally combines these basic primitives to create another object. For example, to generate curves in computer graphics, multiple lines are connected with each other using some data points (dots) as shown in Fig. 9.1.1.

Dots + lines > curves




Fig. 9.1.1 : Modeling of curves

As per Fig. 9.1.1 we can see that in graphics modeling, the curves generated by connecting multiple lines are not smooth but in real world curves appears smooth. To design curve which also have some smoothness we require high design approximation, which can be represented as explicit, implicit, parametric and non-parametric.

Q. Explain implicit and explicit curve representation in detail with example.

Curves are represented mainly into two ways.

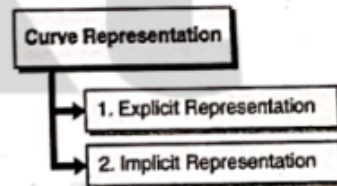


Fig. C9.1 : Curve Representation

→ **1. Explicit Representation**

The explicit form of a curve in two dimensions gives the value of one variable i.e. the dependent variable, in terms of other independent variable.

In x, y space, it is written as, $y = f(x)$.

→ **2. Implicit Representations**

In two dimensions, an implicit curve can be represented by the equation $f(x, y) = 0$

The implicit form is less coordinate-system dependent than is the explicit form.

In three dimensions, the implicit form $f(x, y, z) = 0$

Curves in three dimensions are not as easily represented in implicit form.

We can represent a curve as the intersection, if it exists, of the two surfaces: $f(x, y, z) = 0, g(x, y, z) = 0$.

Q.8

Back face removal

In a solid object, there are surfaces which are facing the viewer (front faces) and there are surfaces which are opposite to the viewer (back faces).

These back faces contribute to approximately half of the total number of surfaces. Since we cannot see these surfaces anyway, to save processing time, we can remove them before the clipping process, with a simple test.

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

- If this vector is pointing in the direction of the center of projection, then it is a front face and can be seen by the viewer denoted by viewing vector V_{view} Fig. 8.4.1

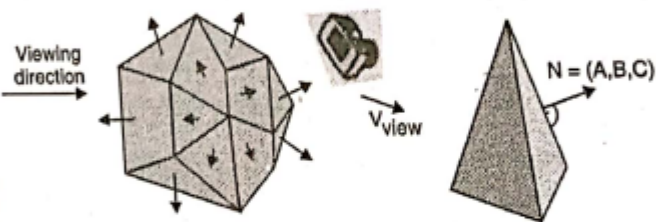


Fig. 8.4.1 : Vectors of back face removal on polyhedron

- If it is pointing away from the center of projection then it is a back face and cannot be seen by the viewer denoted by normal vector,

$$N = (A, B, C).$$

A Polygon (in 3D) is a back face if :

$$V_{\text{view}} \cdot N > 0.$$

- If object descriptions are converted to projection coordinates and viewer's viewing direction is parallel to the viewing z-axis,

Then,

$$V_{\text{view}} = (0, 0, V_z) \text{ and}$$

$V_{\text{view}} \cdot N$ with $N = (A, B, C)$ will be,

$$V_{\text{view}} \cdot N = (0, 0, V_z) \cdot (A, B, C)$$

$$\therefore V_{\text{view}} \cdot N = V_z \cdot C$$

Here, V_z is positive (i.e., view is along +ve Z-direction)

Hence, we only need to check the sign of C where, C is known as the z component of the normal vector N .

Thus,

In right handed viewing system with direction along the negative V_z axis,

- If $C < 0$: Back face of the polygon
- If $C = 0$: Gazing the polygon, not visible from viewing position
- If $C > 0$: Face visible from viewing position

Similarly, in left hand viewing system back faces are identified by $C \geq 0$, when the viewing direction is at positive V_z axis.

Note that this technique only caters well for non-overlapping convex polyhedra.

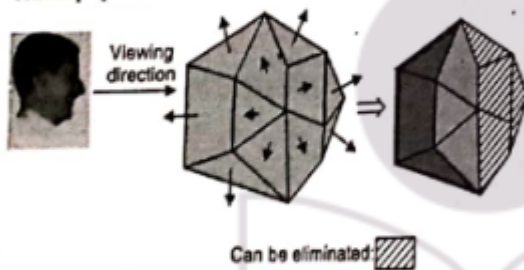


Fig. 8.4.2 : Back-Face Removal

For other cases where there are concave polyhedra or overlapping objects, we still need to apply other methods to further determine where the obscured faces are partially or completely hidden by other objects (e.g., using Depth-Buffer Method or Depth-Sort Method).

Disadvantages of Back Face Removal algorithm

1. It Requires a lot of memory.
2. It can only be used on solid objects.
3. Partially hidden faces cannot be determined by this method.
4. It works fine for convex polyhedra but not necessarily for concave polyhedra.
5. It is not useful for ray tracing, or photometry/radiosity.