# UNIT-III

## Introduction :-

In 3D objects, we wish to determine which lines or surfaces of the objects are visible, so that we can display only the visible lines or surfaces, this process is known as hidden surfaces or hidden line elimination or visible surface determination.

In 3D, what is hidden and what is visible depends upon the point of view. The removal of hidden object is essential to producing realistic looking images. It is broadly classified into two categories -

1) object-space methods
2) Image-space methods

In object space method, compare objects and part of objects within scene to determine which surfaces are visible.

In image space method, visibility is decided point by point at each pixel position on the projection plane.

## 1) Hidden Surface Removal Algorithm :-

A very small sized ^ plane, triangle is chosen as it has the least set of vertices which can form a plane. If any three points on plane surface are known, the unknown parameter A, B, C and D of the plane surface equation

$$Ax + By + Cz + D = 0$$

can be found as follows -

All three points $(x_1, y_1, z_1)$, $(x_2, y_2, z_2)$ and $(x_3, y_3, z_3)$ should satisfy the equation $Ax + By + Cz + D = 0$, as it lies on the surface.

A unique solution of equation for A, B, C, D can be obtained if -

$$\begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x & y & z & 1 \end{vmatrix} = 0$$

$$\begin{vmatrix} x-x_1 & y-y_1 & z-z_1 & 0 \\ x_1-x_2 & y_1-y_2 & z_1-z_2 & 0 \\ x_2-x_3 & y_2-y_3 & z_2-z_3 & 0 \\ x & y & z & 1 \end{vmatrix} = 0 \quad \begin{cases} R_1 = R_4 - R_1 \\ R_2 = R_1 - R_2 \\ R_3 = R_2 - R_3 \end{cases}$$

using column 4 -

$$1 \cdot \begin{vmatrix} x-x_1 & y-y_1 & z-z_1 \\ x_1-x_2 & y_1-y_2 & z_1-z_2 \\ x_2-x_3 & y_2-y_3 & z_2-z_3 \end{vmatrix} = 0$$

$$\underbrace{\begin{vmatrix} y_1-y_2 & z_1-z_2 \\ y_2-y_3 & z_2-z_3 \end{vmatrix}}_{A} x + \underbrace{\begin{vmatrix} z_1-z_2 & x_1-x_2 \\ z_2-z_3 & x_2-x_3 \end{vmatrix}}_{B} y + \underbrace{\begin{vmatrix} x_1-x_2 & y_1-y_2 \\ x_2-x_3 & y_2-y_3 \end{vmatrix}}_{C} z$$

$$+ \underbrace{\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}}_{D} = 0$$

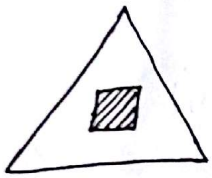
If any point inside the Polygon surface, then $Ax + By + Cz + D < 0$

## 2.) Area Subdivision Method :-

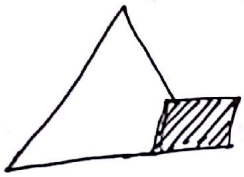
This technique for hidden surface removal is an image space method. We apply this method by successively dividing the total viewing area into smaller and smaller rectangles until each small area is the part of a single visible surface.

Starting with the total view, we apply the tests to determine whether we should subdivide the total area into smaller rectangles If the tests indicate that the view is complex we subdivide it.

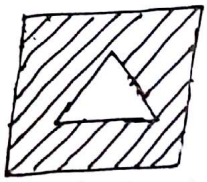
We continue this process until the subdivision are easily analyzed or until they are reduced to the size of a single pixel. We can describe the surface characteristics in following way –



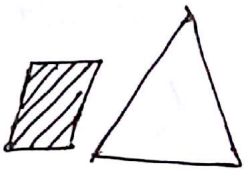
Surrounding surfaces



Overlapping surfaces



Inside surfaces

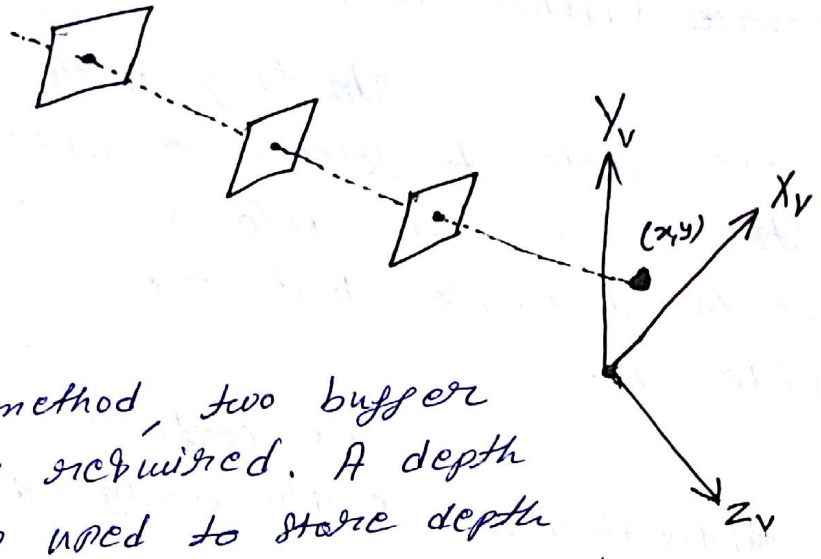


Outside surfaces

## 3) Depth Buffer Method :-

This method is image space approach to detecting visible surfaces, 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. For each pixel position (x,y) on the view plane, object depths can be compared by comparing z-values.




In this method, two buffer areas are required. A depth buffer is used to store depth value for each (x,y) position and the refresh buffer stores the intensity values for each position. Initially all position in the depth buffer are set to zero and refresh buffer is initialized to the background intensity.

Depth values for a surface at position (x,y) are calculated from the plane equation for each surface -

$$z_{xy} = \frac{-Ax - By - D}{C}$$

For each horizontal scan line, the successive depth value can be calculated as -

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

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

$$\boxed{z_{x+1,y} = z_{x,y} - \frac{A}{C}}$$

where A/C is a constant for each surface.

The scan line can be vertical and move horizontally, and $z_{x,y+1}$ can be calculated successively for each x -

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

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

$$\boxed{z_{x,y+1} = z_{x,y} - \frac{B}{C}}$$

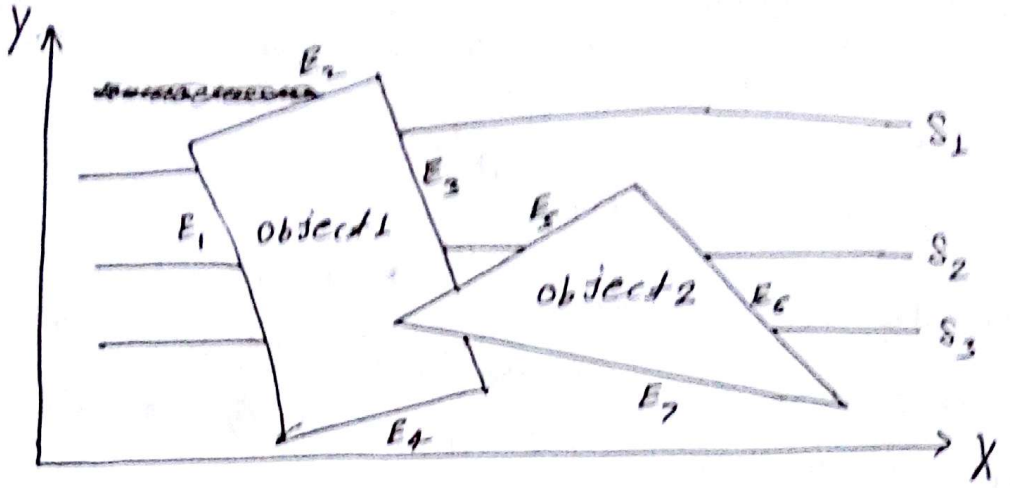
## 4) Scan-Line Algorithm :-

The scan-line is an image-space algorithm. It processes the image one scan line at a time rather than one pixel at a time.

→ When it enters the projection of a plane, an IN flag goes on and the beam switches from the background colour to the colour of the polygon.

→ After the beam leaves the polygon's edge, the colour switches back to the background colour.

→ When scan-line beam is in two or more polygons, it calculate a z-depth and select the colour of the nearest polygon.




→ In figure, scan-line $S_1$ must deal only with the left-hand object; $S_2$ must plot both objects, but there is no depth conflict. $S_3$ find the z-depth of both objects in the region between edge $E_3$ and $E_5$.

→ For Scan-line S1: The active edge list contains edges $E_1$ and $E_3$. At edge $E_1$, the IN flag goes up and the beam switches to object1 colour until it crosses edge $E_3$; At $E_3$ IN flag goes off and colour returns to background.

→ For Scan-line S2: The active edge list contains $E_1$, $E_3$, $E_5$ and $E_6$. The IN flag goes up and down twice and the beam switches to appropriate object colour twice.

→ For Scan-line S3: The active edge list contains the same edges as for S2, but the order is altered, namely $E_1$, $E_5$, $E_3$ and $E_6$. The IN flag goes up two times continuously at $E_1$ and again at edge $E_5$. So calculate z-depth, it is smaller for object 2, indicating it is closer. Therefore the beam colour switches to object 2 colour at edge $E_5$ and it keeps until edge $E_6$.

5) A Buffer :-

A-buffer method is extension of ~~idea in the z-method~~ z-buffer method. z-buffer has a drawback that it can only define only one visible surface and cannot accumulate intensity values for more than one surfaces.

In A-buffer method, each position in the buffer has reference to a linked list of surfaces. Due to this more than one surface intensity can be taken into consideration at each pixel position.

A-buffer has two fields :-

1) Depth field : Holds a +ve or -ve real number
2) Intensity field : Holds surface intensity or a pointer value

The depth field in the A-buffer shows whether a single or multiple surfaces. If depth value is -ve, then multiple surfaces contribute the intensity of the corresponding pixel.

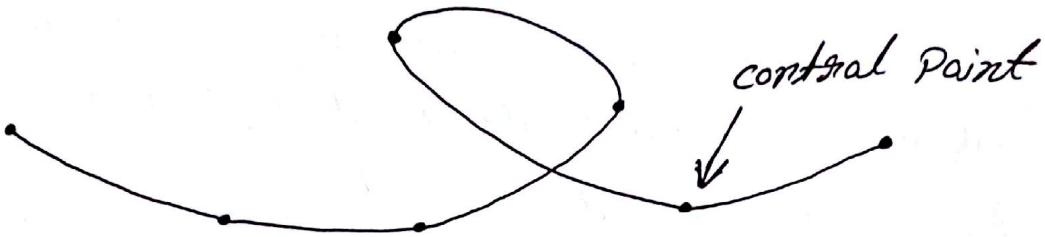
The information of multiple surface intensity is stored in the linked-list. If there is only single surface, then the intensity field stores the intensity of corresponding pixel.

A-buffer is implemented similar to z-buffer, however in A-buffer the intensity of pixel is determined by considering percentage of overlapping surfaces.

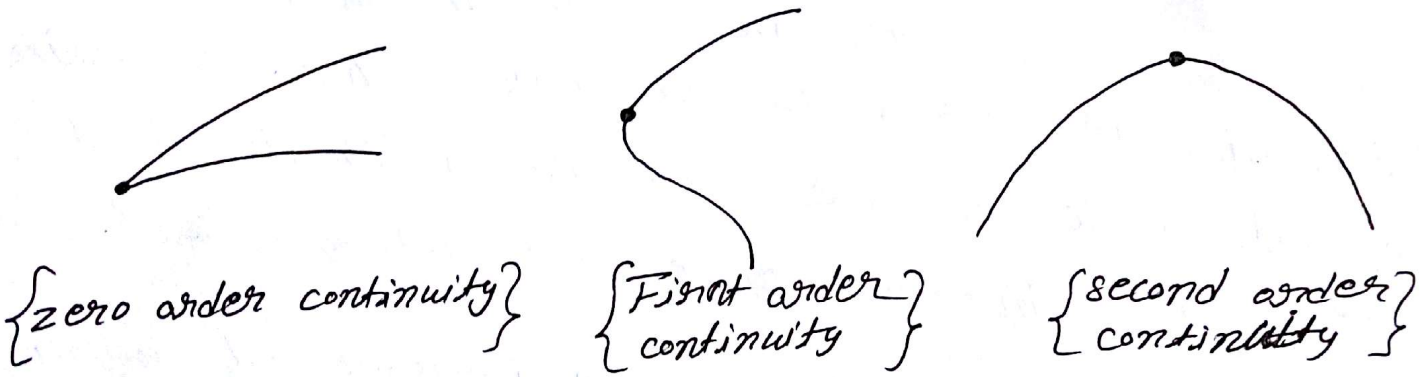
## ✳ CURVE :-

The use of curve is to allows for a higher levels of modeling especially for the construction of highly realistic models. we model an object by using small, curved surface placed (con) next to each other.

Generally a curve is specified by a set of points called control points. These points control the shape of curve interactively.




If curves simply meet, then it is called zero order continuity and if the tangents of both the curve at that point are not equal, then it is first order continuity and if tangent of both the curves at that point are equal, then it is second order continuity.




## ✳ SPLINE Representation :-

The term spline curve referred to a curve with a piecewise cubic polynomial function whose first and second derivation continuous across the various curve sections.

We specify a spline curve by giving a set of coordinate positions, called control points, which indicates the general shape of the curve. These control points are then fitted with piecewise continuous polynomial functions in one of two ways –

1) Interpolation Spline
2) Approximation Spline

When polynomial functions are fitted such that curve passes through each control point, called interpolation spline.

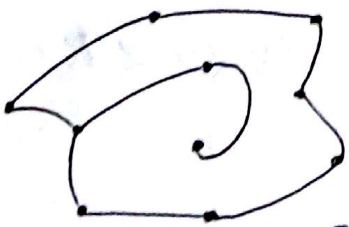
When the polynomial functions are fitted to the general path without necessarity passing through any control points, called approximation spline.

[Interpolation Spline]

[Approximation Spline]

A spline curve is defined, modified and manipulated with operations on the control points.

A convex polygon boundary, that enclose all control points is called convex hull.



[convex Hull]

## * Spline Specification :-

Suppose we have the following parametric cubic polynomial representation for x coordinate along the path of a spline section -

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x, \quad 0 \le u \le 1$$

$$\therefore x(u) = [u^3 \; u^2 \; u \; 1] \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix}$$

$$\Rightarrow \boxed{x(u) = U \cdot C} \quad - (1)$$

where U is the row matrix and C is the coefficient column matrix.

Boundary conditions for this curve might be set, on the end point coordinate $x(0)$ and $x(1)$, and the first derivatives at the end points $x'(0)$ & $x'(1)$. These four boundary conditions are sufficient to determine the value of the four coefficient $a_x, b_x, c_x$ and $d_x$.

We can write the boundary conditions in matrix form and solve for coefficient matrix C as -

$$\boxed{C = M_{spline} \cdot M_{geom}} \quad - (2)$$

where $M_{spline}$ is the 4x4 matrix and $M_{geom}$ is a four element column matrix containing the geometric constraint (boundary) values on the spline.

For example -

$$P(u) = au^3 + bu^2 + cu + d, \quad (K \le u \le K+1)$$

where k & k+1 are the end points.

where x component of P is -

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$

and similarly for the y and z component.

$$P(u) = [u^3 \; u^2 \; u \; 1] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

$$P'(u) = [3u^2 \; 2u \; 1 \; 0] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

So $M_{spline} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}$

By expanding eq$^n$ ② -

$$x(u) = \sum_{k=0}^{3} g_k \cdot BF_k(u)$$

where $g_k$ are the constraint parameters and $BF_k(u)$ are the polynomial blending functions.

## * Bezier Curve :-

A Bezier curve can be fitted to any number of control points. and their relative position determine the degree of Bezier polynomial.

Suppose we are given (n+1) control points, their position is defined as -

$$P_k = (x_k, y_k, z_k); \; k \text{ varying from } 0 \text{ to } n.$$

These coordinate points blended to produce position vector P(u), using Bezier blending function $BEZ_{k,n}(u)$.

A Bezier polynomial function between $P_0$ and $P_n$ is defined as -

$$P(u) = \sum_{k=0}^{n} P_k \cdot BEZ_{k,n}(u)$$

where $P_k$ : Position of control point and

$BEZ_{k,n}$ : Bezier blending function.

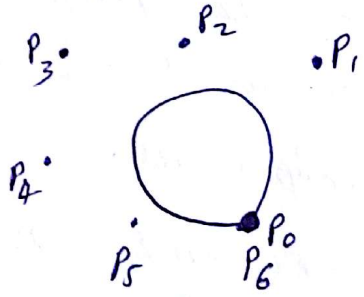
$BEZ_{k,n}(u)$ is defined as -

$$BEZ_{k,n}(u) = C(n,k)\, u^k (1-u)^{n-k} \quad ; \quad C(n,k) = \frac{\lfloor n}{\lfloor n-k \; \lfloor k}$$

## * Bezier B-Spline curve :-

The Bezier B-spline approximation provides for the local control of curve shape. If a single control point is changed; portions of the curve that lie far away are not disturbed.

A Bezier can be made to pass closer to a given coordinate position by assigning multiple control points to that position.





deviation of curve is greater where more than one control points.

we can fit a Bezier curve to any number of control points but this required the calculation of polynomial functions of higher degree.

For cubic Bezier curve, we require 4 control points -

So $n+1 = 4$

$\Rightarrow n = 3$

$$P(u) = \sum_{k=0}^{n} P_k \, BEZ_{k,n}(u)$$

where $BEZ_{k,n}(u) = C(n,k) u^k (1-u)^{n-k}$

A Bezier curve is a polynomial of degree one less than the number of control points used.

$$BEZ_{0,3}(u) = C(3,0) u^0 (1-u)^3$$
$$= (1-u)^3$$

$$BEZ_{1,3}(u) = C(3,1) u^1 (1-u)^2$$
$$= 3u(1-u)^2$$

$$BEZ_{2,3}(u) = C(3,2) u^2 (1-u)^1$$
$$= 3u^2(1-u)$$

$$BEZ_{3,3}(u) = C(3,3) u^3 (1-u)^0$$
$$= u^3$$

Vector equation represents a set of three parametric equation -

$$x(u) = \sum_{k=0}^{n} x_k \, BEZ_{k,n}(u)$$

$$y(u) = \sum_{k=0}^{n} y_k \, BEZ_{k,n}(u)$$

$$z(u) = \sum_{k=0}^{n} z_k \, BEZ_{k,n}(u)$$

———— * ————