Unit - 2

3D concepts

1) parallel and Perspective Projection

2) 3D object Representation - Polygons, curved lines, splines, Quadric surfaces, visualization of datasets, 3D transformation, viewing visible surface identification

3D-transformation:

1) Translation: In three dimensional homogeneous coordinate representation a point is translated from position $P=(x,y,z)$ to position $P'(x',y',z')$ with the matrix operation

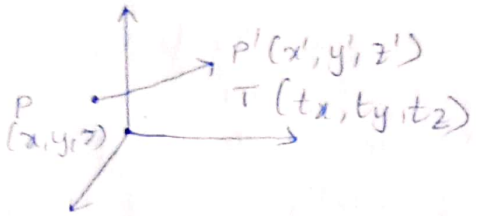
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$P' = T \cdot P$

$x' = x + t_x$

$y' = y + t_y$

$z' = z + t_z$



we obtain the inverse of the translation matrix by negating the translation distance (t_x, t_y, t_z)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

2) Rotation

A three dimensional z-axis rotation is given as

$x' = x\cos\theta - y\sin\theta$

$y' = x\sin\theta + y\cos\theta$

$z' = z$

In homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$P' = R_z(\theta) \cdot P$

x-axis

$y' = y\cos\theta - z\sin\theta$

$z' = y\sin\theta + z\cos\theta$

$x' = x$

y-axis

$z' = z\cos\theta - x\sin\theta$

$x' = z\sin\theta + x\cos\theta$

$y' = y$

x → y → z → x

x-axis: $P' = R_x(\theta) P$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

y-axis: $P' = R_y(\theta) P$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

General 3D Rotation:

In special case where an object is to be rotated about an axis that is parallel to one of the coordinate axes we can attain the desired rotation with the following transformation sequence.

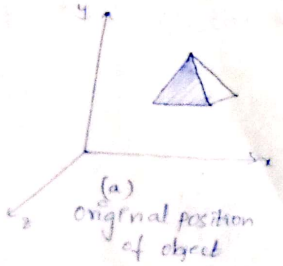
i) Translate the object so that rotation axis coincides with the parallel coordinate axis

ii) Perform the specified rotation about that axis

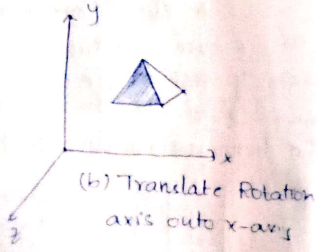
iii) Translate the object so that the Rotation axis moved back to its original position.

$P' = T^{-1} R_x(\theta) T P$

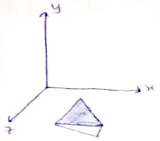
③ ② ①



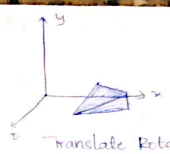
(a) Original position of object



(b) Translate Rotation axis onto x-axis



© Rotate object through angle



Translate Rotation axis at original Position

When the object is to be Rotated about an axis that is not parallel to one of the coordinate axis we need to perform additional transformation. The five steps required are

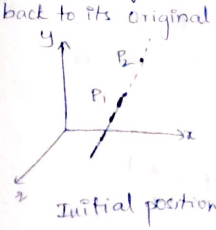
Step-1: Translate the object so that the rotation axis passes through coordinate origin.

2) Rotate the object so that the axis of Rotation coincide with one of the coordinate axis

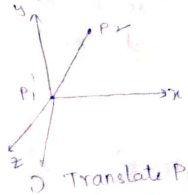
3) Perform the ~~special~~ specified rotation about coordinate axis.

4) Apply inverse Rotation to bring the rotation axis back to it's original orientation.

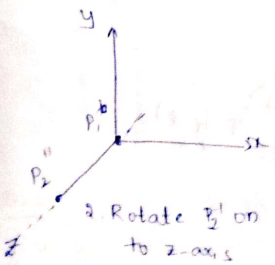
5) Apply inverse translation to bring the rotation axis back to its original position.



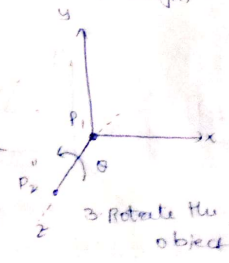
Initial position



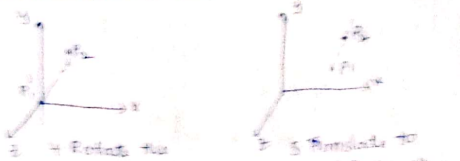
① Translate P_1 to origin



② Rotate P_2' on to z-axis



③ Rotate the object



4 Rotate the axis to original position

5 Translate to original position

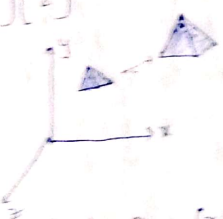
3 Scaling The matrix expression for the scaling transformation of a position $P=(x, y, z)$ relative to the coordinate origin can be written as

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$x' = x \cdot s_x$

$y' = y \cdot s_y$

$z' = z \cdot s_z$

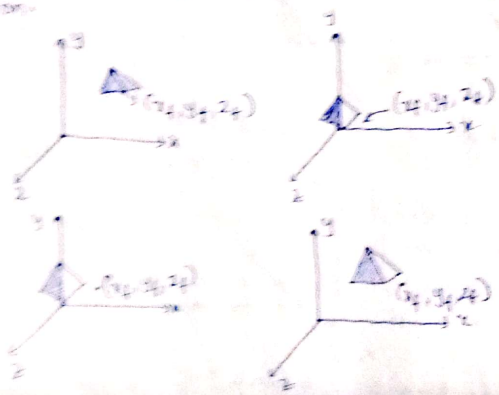


Scaling with respect to selected fixed position $=(x_f, y_f, z_f)$ can be represented with the following transformation sequence

Step 1: Translate the fixed point to origin

Step 2: Scale the object relative to the coordinate origin

Step 3: Translate the fixed point back to its original position.



$T(x_f, y_f, z_f) \cdot S(s_x, s_y, s_z) \cdot T(-x_f, -y_f, -z_f)$

$$= \begin{bmatrix} s_x & 0 & 0 & (1-s_x)x_f \\ 0 & s_y & 0 & (1-s_y)y_f \\ 0 & 0 & s_z & (1-s_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Other transformations.

Reflection In general 3D reflection matrices are setup similar to those for 2D reflections relatively to given axis are equivalent to 180° rotation about the axis. The matrix representation for reflection of points relative to xy plane is RF_z

$$RF_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

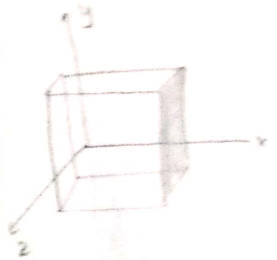
yz plane is $RF_x = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

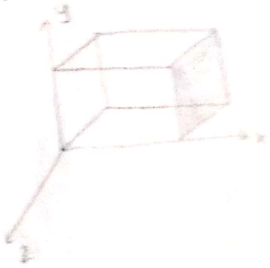
xz plane is $RF_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

a) Shearing Shearing transformations can be used to modify objects shapes. In 3D shearing relative to z-axis

$$SH_z = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

a and b are some real values



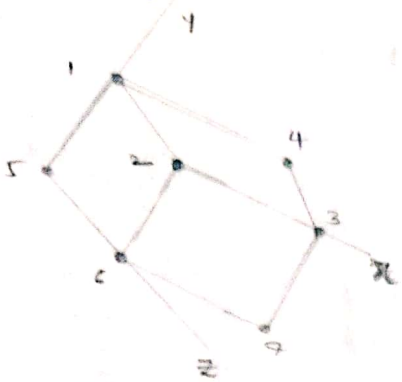


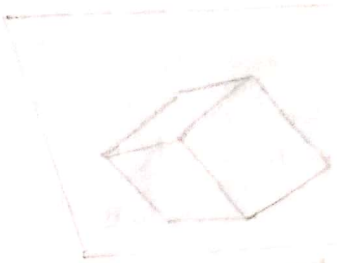
3D-viewing.

Viewing in 3D involves the following considerations

Parallel projections:

* Orthographic projections that display more than one face of an object
* Such views are called axonometric orthographic projections
* The most commonly used axonometric projection is the isometric projection



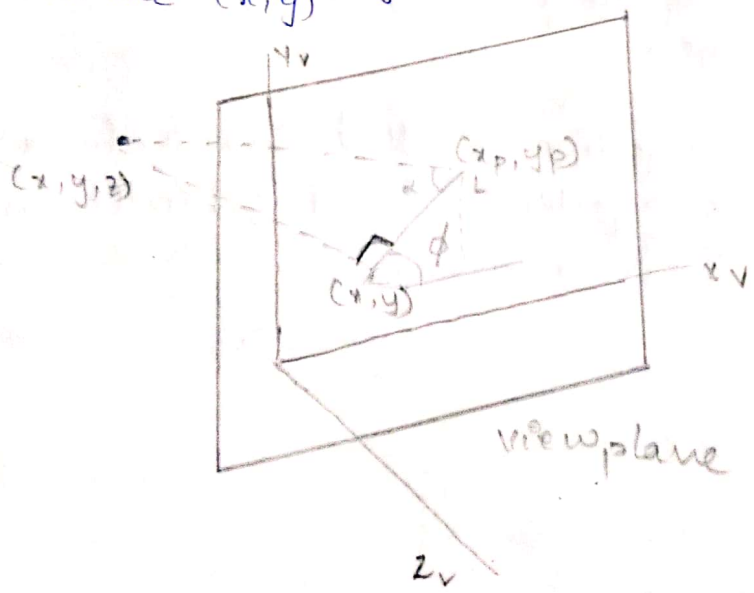


* Oblique projection:

→ An oblique projection is obtained by projecting points along parallel lines that are not perpendicular to the projection plane.

→ In some applications packages, an oblique projection vector is specified with two angles, α and ϕ

→ Point (x, y, z) is projected to position (x_p, y_p) on the view plane. Orthographic projection coordinates on the plane are (x, y)



$$\cos\phi = \frac{\text{Base}}{\text{Hyp}}$$

$$\cos\phi = \frac{x_p - x}{L}$$

$$L\cos\phi = x_p - x$$

$$x_p = x + L\cos\phi$$

$$\sin\phi = \frac{y_p - y}{L}$$

$$L\sin\phi = y_p - y$$

$$\boxed{y_p = y + L\sin\phi}$$

Similarly, $\tan\alpha = \frac{\text{Perpendicular}}{\text{Base}}$

$$\tan\alpha = \frac{z}{L}$$

$$L = \frac{z}{\tan\alpha}$$

$$\boxed{L = zL_1}$$ where L_1 is inverse of $\tan\alpha$

Hence

$$x_p = x + L\cos\phi = x + z(L_1\cos\phi)$$

$$y_p = y + k\sin\phi = y + z(L_1\sin\phi)$$

Observe if the orthogonal projection $L_1 = 0$.

→ The transformation matrix for producing any parallel projection onto the $x_v y_v$ plane can be written as

$$M_{parallel} = \begin{bmatrix} 1 & 0 & L_1\cos\phi & 0 \\ 0 & 1 & L_1\sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

→ The orthogonal projection will be obtained when $L_1 = 0$

→ oblique projections are generated with nonzero values for L_1

Cavalier and Cabinet:

→ Common choices for angle ϕ are 30° and 45°, which display a combination view of the front, side, and top (or front, side and bottom) of an object.

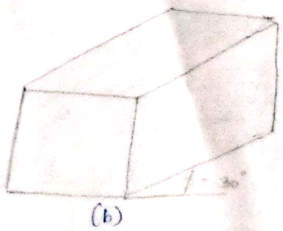
→ When

→ $\tan\alpha = 1$ then the projection is called Cavalier ($\alpha = 45°$)

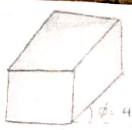
→ $\tan\alpha = 2$ then the projection is called cabinet ($\alpha = 63°$)

→ ϕ usually takes the value 30° or 45°

Cavalier example



(b)



(a)

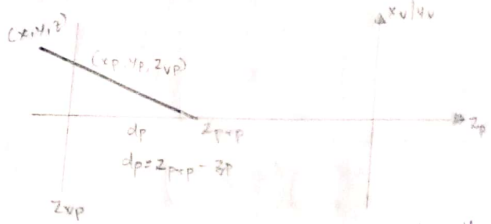
Perspective projection

* To obtain a perspective projection of a three-dimensional object, we transform points along projection lines that meet at the projection reference point.

* Suppose we set the projection reference point at position z_{prp} along the z_v axis, and we place the view plane at z_{vp}

* We can write equations describing coordinate positions along this perspective projection line in parametric form as

- $x' = x - xu$
- $y' = y - yu$
- $z' = z - (z - z_{prp})u$



* On the view plane $z' = z_{vp}$ and we can solve the z' equation for parameter u at this position along the projection line:

$$u = z_{vp} - z / z_{prp} - z$$

* Substituting this value of u into the equations for x' and y', we

$$x' = x - x\left(\frac{z_{vp} - z}{z_{prp} - z}\right)$$

$$x' = \frac{x(z_{prp} - z) - x(z_{vp} - z)}{z_{prp} - z}$$

$$x' = \frac{x z_{prp} - xz - x z_{vp} + xz}{z_{prp} - z}$$

$$\boxed{x' = \frac{x z_{prp} - x z_{vp}}{z_{prp} - z}} \quad \frac{-x d_p}{z_{prp} - z}$$

$$y_p = y\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right)$$

$$y_p = \frac{y(z_{prp} - z) - y(z_{vp} - z)}{z_{prp} - z}$$

$$y_p = \frac{y z_{prp} - y z - y z_{vp} + y z}{z_{prp} - z}$$

$$\boxed{y_p = \frac{y z_{prp} - y z_{vp}}{z_{prp} - z}} = \frac{y d_p}{z_{prp} - z}$$

where $d_p = z_{prp} - z_{vp}$

Using a three dimensional homogeneous coordinate representation we can write the perspective projection transformation 12-13 in matrix form -

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -z_{prp}/d_p & z_{vp}(z_{prp}/d_p) \\ 0 & 0 & -1/d_p & z_{prp}/d_p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

In this representation the homogeneous factor is

$$\boxed{h = \frac{z_{prp} - z}{d_p}}$$

Here $x_h = x_p \cdot h$

$$= x\left(\frac{d_p}{z_{prp} - z}\right)\left(\frac{z_{prp} - z}{d_p}\right)$$

$$\boxed{x_h = x}$$

$$y_h = y_p \cdot h = y\left(\frac{d_p}{z_{prp} - z}\right)\left(\frac{z_{prp} - z}{d_p}\right)$$

$$\boxed{y_h = y}$$

h should be 1

Here in this representation,

$$h = \frac{z_{prp} - z}{d_p}$$

Since we are view plane $z = z_{vp}$

$$= \frac{z_{prp} - z_{vp}}{d_p}$$

$$= \frac{z_{prp} - z_{vp}}{z_{prp} - z_{vp}}$$

$$h = 1$$

There are number of special cases for the perspective transformation equation shown in fig :

$$x_p = x\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right)$$

$$y_p = y\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) \quad \text{---} (1)$$

If a view plan is taken to be the uv plane then $z_{vp} = 0$ and the projection coordinates are

$$x_p = x\left(\frac{z_{prp} - 0}{z_{prp} - z}\right) = x\left(\frac{z_{prp}}{z_{prp} - z}\right)$$

$$x_p = x\left(\frac{1}{1 - z/z_{prp}}\right)$$

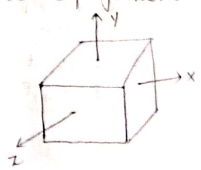
Similarly $y_p = y\left(\frac{z_{prp}}{z_{prp} - z}\right) = y\left(\frac{1}{1 - z/z_{prp}}\right)$

And in some graphic packages, the projection reference point is always taken to be at the viewing coordinate origin in this case $z_{prp} = 0$ and the projection coordinates are

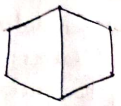
$$x_p = x\left(\frac{z_{vp}}{z}\right) = x\left(\frac{1}{z/z_{vp}}\right)$$

$$y_p = y\left(\frac{z_{vp}}{z}\right) = y\left(\frac{1}{z/z_{vp}}\right)$$

Vanishing point: The point at which a set of projected parallel lines appears to converge is called a vanishing point. The vanishing point for any set of lines that are parallel to one of the principle axis of an object is referred as "principle vanishing point". These are accordingly classified as one point, 2., 3. perspective projections



1 point perspective projection



x-axis vanishing point

z-axis vanishing point

Two point perspective projection

3D object representation:

Representation schems for solid objects are often divided into two broad board categories: ① Boundary Representations ② Space partitioning representations

① Boundary Repres: It describes a three dimensional object as a set of surfaces that seperate the object interior from the environment. Ex: Splines

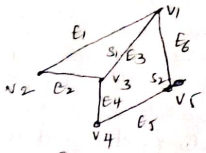
② Spaw Partitioning Representation: Used to describe interior properties by partitioning the spacial region containing an object into a set of small non-overlapping contiguous solid Ex: oc tree

*Polygon surfaces: Many graphic systems store all object descriptions as sets of surface polygons. The most common used boundary representation for a three dimensional graphics object is a set of surface polygons that enclose the object interior

Example:

Wireframe work representation of a cylinder with backlines removes

Polygon tables:



① Vertex table

$V_1 : x_1, y_1, z_1$
$V_2 : x_2, y_2, z_2$
$V_3 : x_3, y_3, z_3$
$V_4 : x_4, y_4, z_4$
$V_5 : x_5, y_5, z_5$

② Edge table

$E_1 : V_1, V_2$
$E_2 : V_2, V_3$
$E_3 : V_3, V_1$
$E_4 : V_3, V_4$
$E_5 : V_4, V_5$
$E_6 : V_5, V_1$

③ Polygon Surface Table

S_1	E_1, E_2, E_3
S_2	E_3, E_4, E_5, E_6

We specified a polygon surface with a set of vertex coordinates and associated attribute parameters. Polygon data tables can be organized into two groups: ① Geometric tables: Contain vertex coordinates and parameters to identify the spacial orientation of the polygon surfaces

② Attribute information: It includes parameters specifying the degree of transparency of obj[illegible] nd its surface reflectivity and texture characteristics

Plane Equation To produce a display of 3D object we must process the input data representation for the object through several procedures for some of these procedures we need information about the special orientation of individual surface components of the object. This information is obtained from vertex coordinate values and the equations that describe the polygon planes. The equation for a plane surface can be expressed in the form $Ax+By+Cz+D=0$

where (x,y,z) is the any point on the plane and the coefficients A,B,C and D are constants describing the special properties of the plane. To obtain the values of A,B,C and D we select three successive polygon vertex ex (x_1,y_1,z_1) (x_2,y_2,z_2) (x_3,y_3,z_3) and solve the following set of simultaneous linear plane equations for the ratio A/D

$$(A/D)x_k + (B/D)y_k + (C/D)z_k = -1 \quad \text{where } k=1,2,3$$

The solutions for this set of equations can be obtained in determinent form using Cramer's rule

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \quad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}$$

$$C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \quad D = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

Now

$$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$$
$$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$$
$$D = -x_1(y_2 z_3 - y_3 z_2) - x_2(y_3 z_1 - y_1 z_3) - x_3(y_1 z_2 - y_2 z_1)$$
$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$

As vertex values and other information are stored into the polygon data structure. Values for A,B,C and D are computed for each polygon and stored with other polygon data

* Plane equations are used to identify the positions of spacial points relative to the plane surface of an object. For any point (x,y,z) not on a plane with parameters A,B,C and D we have the following conditions:

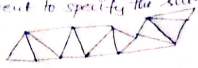
① $Ax+By+Cz+D \neq 0$ ie, the point is either inside or outside the plane surface

② if $Ax+By+Cz+D < 0$ ie, the point is inside the surface

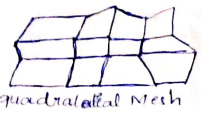
(3) If $Ax+By+Cz+D>0$ the point is outside the surface

Polygon representations

Polygon Meshes: When object surfaces are to be tiled it is more convenient to specify the surface facets with a mesh function



→ polygon mesh (triangle strip)



quadrilateral Mesh

One type of polygon mesh is a triangle strip and this function produces $n-2$ connected triangles given the coordinates for n vertices. Another similar function is quadrilateral mesh which generates a mesh of $\frac{n-1}{m-1}$ quadrilaterals. Given n/m array of vertices.

Quadratic Surfaces

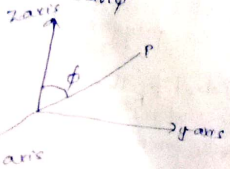
A frequently used class of objects are quadratic surfaces, quartic surfaces which are described with second degree equations. They include spheres, ellipsoid, Torus, paraboloid's and Hyperbola

1) Spheres: In cartesian coordinates a spherical surface with radius r centered on the coordinate origin is defined as set of points (x,y,z) that satisfy the equation

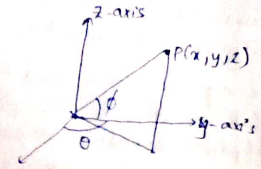
$$x^2+y^2+z^2=r^2$$

We can also describe the spherical surface in parametric form.

$$x=r\cos\phi\cos\theta$$
$$y=r\cos\phi\sin\theta \qquad -\pi/2 \le \phi \le \pi/2$$
$$z=r\sin\phi \qquad -\pi \le \theta \le \pi$$



Spherical coordinates parameters (r,ϕ,θ)



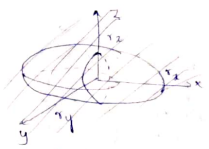
Quadratic Surfaces

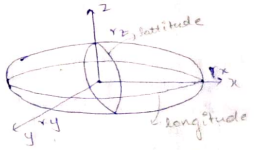
2) Ellipsoid: The cartesian representation for points over the surface of an ellipsoid centered on the origin is

$$\left(\frac{x^2}{r_x^2}+\frac{y^2}{r_y^2}+\frac{z^2}{r_z^2}\right)=1$$

And the parametric representation of ϕ (latitude angle) and θ (longitude angle) is

$$x=r_x\cos\phi\cos\theta \qquad -\pi/2 \le \phi \le \pi/2$$
$$y=r_y\cos\phi\sin\theta \qquad -\pi \le \theta \le \pi$$
$$z=r_z\sin\phi$$



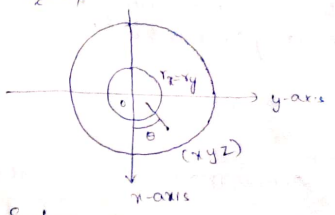


3) Torus: It can be generated by rotating a circle (or) other conic about a specified axis. The cartesian representation of points over a surface of torus can be written in the form,

$$\left[r-\sqrt{\frac{x^2}{r_x^2}+\frac{y^2}{r_y^2}}\right]^2+\frac{z^2}{r_z^2}=1$$

In parametric form, (using latitude (ϕ) and longitude (θ) angles)

$$x=r_x(r+\cos\phi)\cos\theta$$
$$y=r_y(r+\cos\phi)\sin\theta \qquad -\pi \le \phi \le \pi$$
$$z=r_z\sin\phi \qquad -\pi \le \theta \le \pi$$



Super Quadric Surfaces:

Super Quadric surfaces are formed by incorporating additional parameters into Quadric equations to provide increased flexibility for adjusting object shapes

Superellipse:

cartesian coordinates,

$$\left(\frac{x}{r_x}\right)^{2/s_1}+\left(\frac{y}{s_y}\right)^{2/s_1}=1$$

parametric equations

$x = r_x \cos^{s_0}\theta \qquad -\pi \le \theta \le \pi$

$y = r_y \sin^{s_0}\theta$

Super ellipsoid:

Cartesian:

$$\left[\left(\frac{x}{r_x}\right)^{2/s_2} + \left(\frac{y}{r_y}\right)^{2/s_2}\right]^{s_2/s_1} + \left(\frac{z}{r_z}\right)^{2/s_1} = 1$$

Parametric

$x = r_x \cos^{s_1}\phi \cos^{s_2}\theta \qquad -\pi/2 \le \phi \le \pi/2$

$y = r_y \cos^{s_1}\phi \sin^{s_2}\theta \qquad -\pi \le \theta \le \pi$

$z = r_z \sin^{s_1}\phi$

Spline representation:

A spline is a flexible strip used to produce a smooth curve through a designated set of points. 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 can be fitted in two ways:

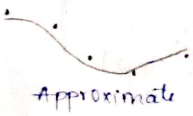
① When polynomial sections are fitted so that curve passes through each control point, the resultant curve is said to be interpolate the set of control points

two methods for smoothing

interpolate approx

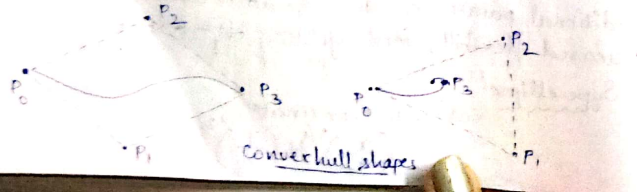
*interpolate

② When the polynomial are fitted to the general control point path without necessarily passing through any control point. The resulting curve is said to approximate the set of control point



Approximate

* The convex polygon boundary that encloses a set of control points is called convex hull



Convex hull shapes

* A polyline connecting the sequence of control points is often referred to as control graph of the curve.

Control graph shapes

Curve is steering from first end point to last end point

We generally move from

$P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3$

Curve from $P_0 \rightarrow P_3$

Parametric continuity conditions

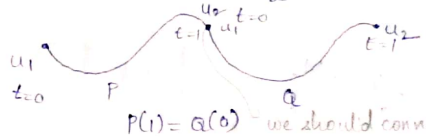
* To ensure a smooth transition from one section of piece wise parametric curve to the next we can impose various continuity conditions at the connection points

→ two audio's connection which must be smooth without any disturbance

example: Connecting one part of brain with another

→ We set parametric continuity by matching the parametric derivatives of adjoining curve sections at their common boundary

① Zero-order parametric continuity (C^0): The values of x, y and z evaluated at u_2 for the first curve section are equal to the values of x, y and z evaluated at u_1 for the next curve section



$P(1) = Q(0)$ we should connect the curve P and Q

$u_2 = u_1$ where the value of u_1 and u_2 becomes equal

* First order parametric continuity: It means that first parametric derivatives for the two successive curve sections are equal at their joining point

$P'(1) = Q'(0)$ → smooth joining will be provided.

* It is denoted by C^1

→ Second order parametric continuity: (C^2): It means both first and second parametric derivatives of the two curve sections are same at the intersection

$P'(1) = Q'(0)$
$P''(1) = Q''(0)$ } → when both happens the curve will be very smooth there will be no disturbance

* An alternative method for joining two successive curve sections is to specify the ~~coordinates~~ conditions for geometric continuity. In this case we only require parametric derivatives of

The two sections to be proportional to each other at boundary instead of equal to each other

1) Zero-order geometric continuity (C^0)

2) Second order geometric continuity (C^2)

In simple terms,

3) Third order geometric continuity

* The only difference between geometric and parametric continuity is in geometric continuity we specify that ... are proportional

* Spline Specifications: The parametric cubic polynomial ... for the x-coordinate along the path of a spline section

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

* The four boundary conditions $x(0), x(1), x'(0), x'(1)$ are sufficient to determine the values of the 4 coefficients a_x, b_x, c_x and d_x

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

$= u \cdot C$ → $C = M_{spline} \cdot M_{geom}$

$x(u) = u \cdot M_{spline} \cdot M_{geom}$

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

polynomial blending function

constraint parameters such as control point coordinates

* Bezier Curves: (one of the approximate splines)

→ Bezier splines have number of properties that make them highly useful and convient for curve and surface design.

→ Suppose we are given (n+1) control point positions

$P_k = (x_k, y_k, z_k)$ with k varing from 0 to n

$P_0, P_1, P_2, P_3, \cdots P_n$ (n control point positions)

These coordinate points can be blended to produce the following position vector

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

control points

Blending function

Blending function can be defined as,

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

where $C(n,k) = \frac{n!}{k!(n-k)!}$

We can define Bezier blending function with recursive calculations,

$$BEZ_{k,n}(u) = (1-u) BEZ_{k,n-1}(u) + u\, BEZ_{k-1,n-1}(u) + \cdots \quad n > k \geq 1$$

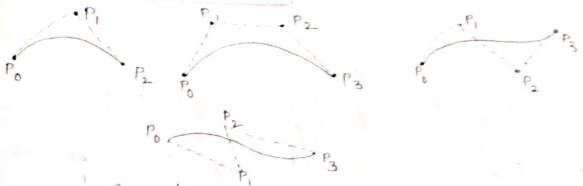
* A set of three parametric equations to individual curve coordinates,

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

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

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

* Bezier curve matches more smoothness when compared to spline



example of Bezier curve (category of approximate set, these are most similar to splines)

Properties of Bezier curves

① A very useful property of Bezier curve is that it always pass through the first and last control points

② Bezier curve will always be within the convex hull (or) convex polygon boundary of the control points

③ Polynomial equation depends on control points that is polynomial equation must have degree less than one control points → degree should be (n-1)

n number of control points

Example cubic Bezier curves:

generates equation with 3 degree polynomial

Cubic Bezier curves are generated with 4 control points

The four blending functions for cubic Bezier curves obtained by substituting n=3 into equation

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

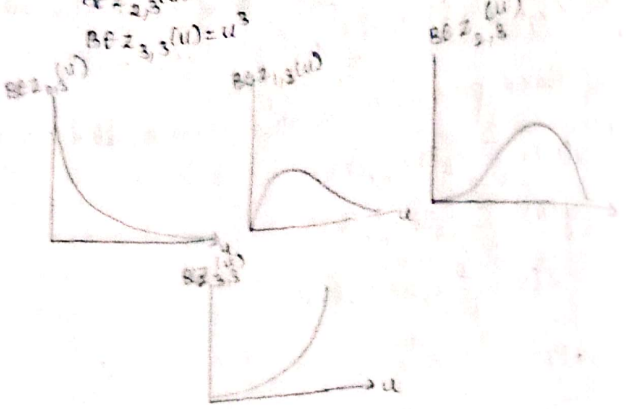
obtained functions

$BF z_{0,3}(u) = (1-u)^3$

$BF z_{1,3}(u) = 3u(1-u)^2$

$BF z_{2,3}(u) = 3u^2(1-u)$

$BF z_{3,3}(u) = u^3$



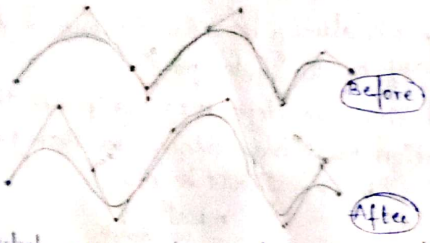
B-spline curve and surface

→ It provides some advantages for the Bezier curves while we have some disadvantages and corrected by B-spline curves.

→ These are the most widely used class of approximating splines. B-splines have two advantages, over Bezier splines.

1) The degree of ~~these~~ Bsplines polynomial can be set independently ~~not~~ of the number of control points

2) B splines allow local control over the shape of a spline curve or surface.



Local control → means when we change a particular control point from a to a' only that particular curve area will be effected

* Bezier curves are having global control (i.e, when a particular control point is changed the whole curve will be changed)

B spline curves.

We can write a general expression for the calculation of coordinate positions along a Bspline curve in a blending function formulation as:

$$P(u) = \sum_{k=0}^{n} P_k B_{k,d}(u)$$ where $u_{min} \le u \le u_{max}$, $2 \le d \le n+1$

where P_k are input set of $n+1$ control points and $B_{k,d}$ blending function,

$B_{k,d}$ are polynomials of degree $d-1$ where parameter d can be choosen any integer value in range 2 to $n+1$ control points.

* The blending functions for B-spline curves are defined as,

$$B_{k,1}(u) = \begin{cases} 1, & \text{if } u_k \le u < u_{k+1} \\ 0, & \text{otherwise} \end{cases}$$

$$B_{k,d}(u) = \frac{u-u_k}{u_{k+d-1}-u_k} B_{k,d-1}(u) + \frac{u_{k+d}-u}{u_{k+d}-u_{k+1}} B_{k+1,d-1}(u)$$ } 10.4

* The above equation calculates the value recursively based on the previous values, where each blending function $B_{k,d}$ is defined over d sub intervals of total range of u. The selected set of sub intervals end points u_j is referred to as knot vector.

Some types of B-splines.

Q* Uniform Periodic B-spline When the spacing between the knot values is constant. The resulting curve is called uniform B-spline.

For example: The calculation of B-spline blending function for a uniform integer knot vector, we select parameters $d=n=3$. The knot vector must then contain ~~n+1~~ $n+d+1 = 7$ knot values. And the values will be $\{0,1,2,3,4,5,6\}$

→ Using recursive knot relation on equation 10.5 we obtain the first blending function as,

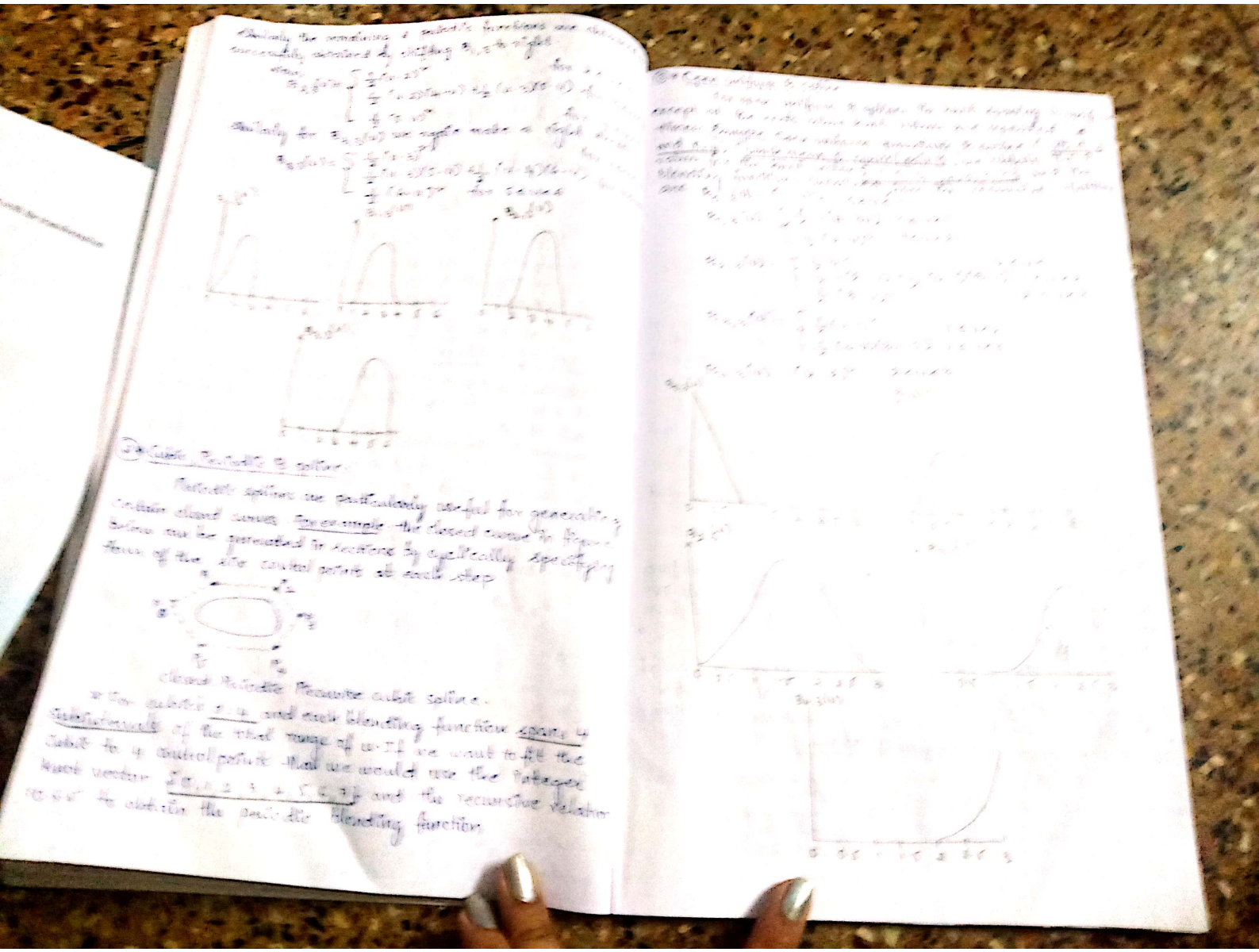
$$B_{0,3}(u) = \begin{cases} \frac{1}{2}u^2 & \text{for } 0 \le u < 1 \\ \frac{1}{2}u(2-u) + \frac{1}{2}(u-1)(3-u) & \text{for } 1 \le u < 2 \\ \frac{1}{2}(3-u)^2 & \text{for } 2 \le u < 3 \end{cases}$$

u is divided into intervals by using knot values ($0 \to 1$, $1 \to 2$, $2 \to 3$)

u value of $B_{0,3}(u)$ divided into 3 intervals

Be apply the next periodic blending function by substituting $u-1$ for u in $B_{0,3}$ and shifting the starting position up by 1.

$$B_{1,3}(u) = \begin{cases} \frac{1}{2}(u-1)^2 & \text{for } 1 \le u < 2 \\ \frac{1}{2}(u-1)(3-u) + \frac{1}{2}(u-2)(4-u) & \text{for } 2 \le u < 3 \\ \frac{1}{2}(4-u)^2 & \text{for } 3 \le u < 4 \end{cases}$$



Scanned by CamScanner

formulation of ... point function over a ... Spline, we can obtain a vector point function over a ... surface using cartesian product of Bspline blending functions in the form

$$P(u,v) = \sum_{k_1=0}^{n_1} \sum_{k_2=0}^{n_2} P_{k_1,k_2} B_{k_1,d_1}(u) B_{k_2,d_2}(v)$$

where the vector values for P_{k_1,k_2} specifies position of the (n_1+1) by (n_2+1) control points

Visible Surface detection Algorithm/methods:

Hidden (or) Surface elimination method.

-A major consideration in the generation of realistic graphics display is identifying those parts of a scene that are visible from a chosen viewing position there are many approaches to solve this problem and are referred as visible surface detection methods and also as hidden surface elimination methods. These algorithms are classified into 1) objects-space Methods

2) Image-Space Methods

1) Object-space Method: It compares objects and parts of objects to eachother within the scene to determine which surfaces should label as visible

2) Image-space Method: Visibility is decided point by point at each pixel position on the projection plane

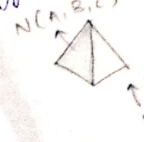
Algorithm-1:

Back-FACE Method

It is an object space method for identifying the backfaces of a polyhedron and is based on inside outside tests. A point (x,y,z) is inside a polygon surface with plain parameters A, B, C and D,

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

We are inside that face and cannot see the front of it from our viewing position. In general, V is a vector in the viewing direction from the eye (or) Camera position than this polygon is a backface if $V \cdot N > 0$

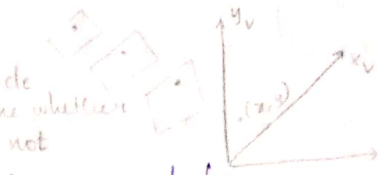


Further, if object description has been converted to projection coordinates and our viewing direction is parallel to viewing z_v axis than $V = (0,0,V_z)$ and $V \cdot N = V_z C$ we can label any polygon as a backspace if its normal vector has a z component value $C \leq 0$

Depth Buffer Method

↳ Image space method

↳ Based on the pixels position we determine whether it is inside visible (or) not



* It is an image space approach for detecting visible surfaces. This procedure is also refered as z-buffer method since object depth is usually measured from view plane along the z-axis of a viewing system

Algorithm:

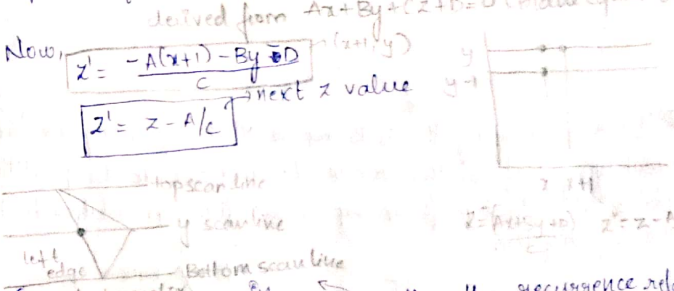
Step-1: Initilize the depth Buffer and refresh the Buffer so that for all Buffer positions (x,y) $depth(x,y) = 0$, $refresh(x,y) = I_{backgnd}$

Step-2: For each position on each polygon surface compare depth values to previously stored values in the depth Buffer to determine the visibility.

(i) Calculate the depth z for each (x,y) position on the polygon and if $z > depth(x,y)$ than set $depth(x,y) = z$ and $refresh(x,y) = I_{surf}(x,y)$

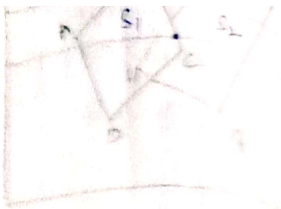
Step-3: After processing all surfaces we will get visible surface in $depth(x,y)$ and intensity value in $refresh(x,y)$

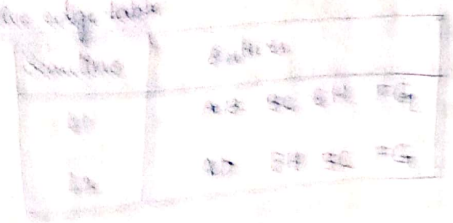
depth value for a surface position (x,y) can be calculated from the plane equation for each surface, $z = \frac{-Ax - By - D}{C}$



If the polygon is of this form then the recurrence relation becomes

$$z' = z + \frac{B}{C}$$





Area subdivision Method:

- This technique of hidden surface removal is an image space method and apply this method by successively dividing the total viewing area into smaller and smaller rectangles until each small area is projection of part of single visible surface (or) no surface at all

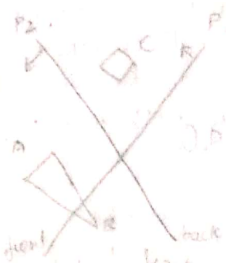
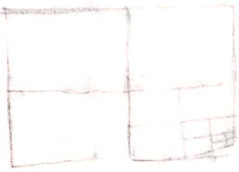
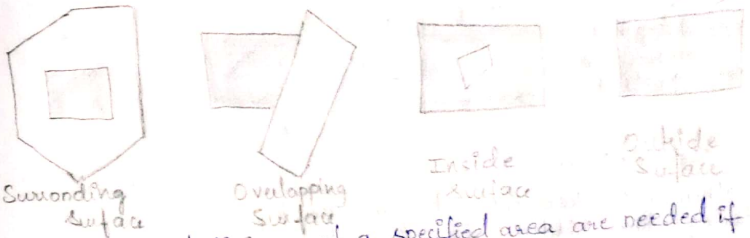


fig: for BSP tree



The test for determining surface visibility within an area can be stated in terms of four classifications

1. Surrounding Surface: One that completely encloses the area.
2. Overlapping Surface: One that is partially inside and partially outside the area.
3. Inside Surface: One that is completely inside the area.
4. Outside Surface: One that is completely outside the area.



* 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 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.

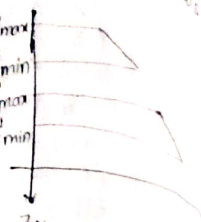
Depth-sorting Method: (or) Painters Algorithm

It uses both image space and object space operations the depth sorting method performs the following basic operations:

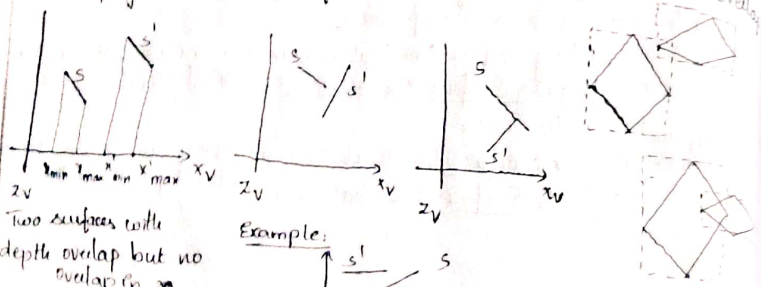
1. Surfaces are sorted in order of decreasing depth.
2. Surfaces are scanned converted in order starting with the surface of greatest depth.

Surface S with greatest depth is compared to the others surfaces

If no depth overlaps occurs o is
surfaces that have no depth overlap.
If the depth overlap is detected at
any point we need to make some additional
comparisons to determine whether any of the
surfaces should be reorded. If any of these
tests is true no reordering is necessary
for that surface. The tests are listed below:

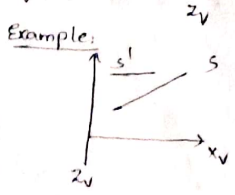


1) The bounding rectangles in the xy-plane for the two surfaces don't overlap.
2) Surface S is completely behind the overlapping surface relative to the viewing position.
3) The overlapping surface is completely infront of S relative to the viewing position.
4) The projection of two surfaces on to the view plane don't overlap.



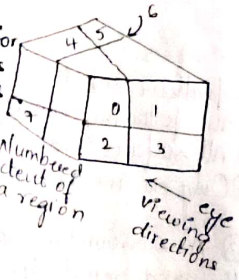
Two surfaces with depth overlap but no overlap in x-direction

Example:



OCT TREE Method:

When an OCT TREE representation is used for viewing volume hidden surface elimination is accomplished by projecting OCT TREE nodes on to the viewing surface in a front-to back order as shown in figure surfaces infront of these octants are visible to the viewer. Any surfaces in the back of octants (4,5,6,7) may be hidden by the front surfaces



Ray-casting method:

Visibility of surfaces can be determined by tracing a ray of light from centre of projection to the objects in the scene:
1) Find out which objects the ray of light intersects.
2) Determine which of these objects is closer to the viewer.
3) Then set the pixel colour to this object.
4) This method is effective for scenes fowith curved surface, polygon faces.

Unit-3

Color models:

Since no finite set of colour, light sources can be combined to display all possible colours, three standard primary's were defined in 1931 by the international commission on illusion referred as CIE

1) XYZ colour model: The set of CIE primary's is generally referred to as XYZ (or) X,Y,Z colour model where X,Y,Z represent vectors in a 3Dimensional space additive colour space. any colour C_λ can be expressed as Additional colours

$$C_\lambda = xX + yY + zZ$$

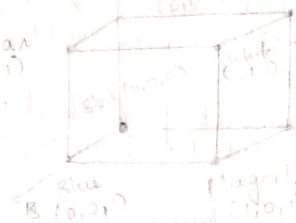
standard primary's (X,Y,Z)

* Normalized amounts are calculated as

$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad z = \frac{Z}{X+Y+Z}$$ with $x+y+z=1$

2) RGB model: As your XYZ colour system RGB colours scheme is an additive model, intensities of primary colours are added to produce other colours

$$C_\lambda = RR + GG + BB$$



3) YIQ model: A television monitor uses a single composite signal and the National Television System Committee (NTSC) colour model for forming the composite video signal is the YIQ model. Brightness information is contained in the Y parameter while chromaticity information (or) hue and purity information is contained in I & Q.

* The conversion from RGB values to YIQ values is accomplished with the transformation

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.528 & 0.371 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$