Chapter 4
3D Graphics System

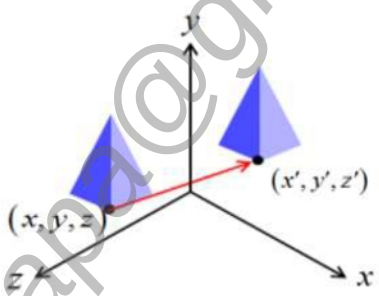
Introduction

- If we have to show or draw chart or graph then 2D geometry is useful but most of the natural objects around us need to show by three dimensions i.e. X(Width), Y(Height) and Z(depth)
- In computers graphics, 3-D (three dimensions or three-dimensional) graphics describes an image that provides the perception of width, height and depth.
- Three dimension is the extension of Two Dimensional concept which takes Z co-ordinate apart from X and Y coordinate.
- So we can perform different transformation by specifying the three dimensional transformation vector.
- However, the 3d Transformation is more complex than 2D transformation. (Now we have third co-ordinate and three planes named xy, yz and zx).

Basic Transformation Techniques

1. Translation

- A translation is applied to an object by repositioning it along a straight line path from one coordinate location to another.



- We translate a three dimensional point p(x,y,z) by adding a **translation distance** t_x, t_y and t_y, to the original coordinate position (x,y,z) to move the point to new position P'(x', y' ,z') as shown in the figure below

$$x' = x + t_x \qquad y' = y + t_y \qquad z' = z' + t_z \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots i$$

where, translation distance pair (t_x, t_y, t_z) is called **translation vector or shift vector.**

- In matrix representation, using homogeneous coordinate, this transformation can be performed through multiplication.

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

Inverse translation,

$T^{-1}(t_x, t_y, t_z) = T(-t_x, -t_y, -t_z)$

2. Scaling

- This transformation changes the size of an object such that we can magnify or reduce its size.
- In case of polygons scaling is done by multiplying coordinate values (x,y,z) of each vertex by scaling factors s_x s_y and s_z to produce the final transformed coordinates (x',y', z')

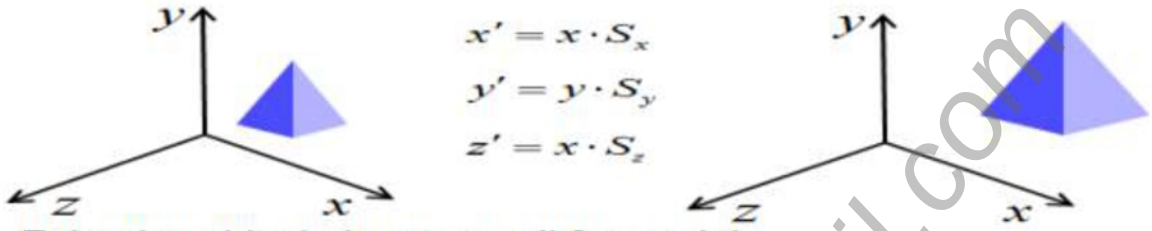
- S_x scales object in X-direction and S_y scales object in Y-direction and S_z scales object in Z direction.

- If s_x, s_y and s_z are equal then scaling is uniform such that it maintains relative object proportion otherwise scaling is called differential scaling.
- Any positive number can be assigned to the scaling factors s_x, s_y and s_z, values less than 1 reduces the size of objects while values greater than 1 produces enlargement. Value 1 for s_x ,s_y and s_z leaves the size of object unchanged.

Case 1: Scaling About Origin

- If p(x,y z) be scaled to a point p'(x', y',z') by s_x time in X-units, s_y times in y-units and s_z times in Z-unit, then the equation for scaling is given as

i.e. P' = S. P, where S is the Scaling Matrix



$$\mathbf{P'} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{S} \cdot \mathbf{P}$$

Fig: Scaling about origin and scaling matrix

Case 2: Scaling About arbitrary point (X_f, Y_f , Z_f)

- For performing scaling of any point P(X, Y, Z) by Scaling Factor (Sx, S_y, S_z) about any arbitrary point (X_f, Y_f, Z_f) , we have to perform following successive transformations
  - i. First we shift /translate the given point such that the arbitrary point is translated to the origin using translation vector ($-X_f$, $-Y_f$, $-Z_f$)
    i.e.
    $$X' = X - X_f \quad Y' = Y - Y_f \quad Z' = Z - Z_f$$
  - ii. Perform scaling operation about origin.
    i.e.
    $X' = S_x . X$ i.e. $X' = S_x . (X - X_f)$
    $Y' = S_y . Y$ i.e. $Y' = S_y . (Y - Y_f)$
    $Z' = S_z . Z$ i.e. $Z' = S_z . (Z - Z_f)$
  - iii. Finally shift the transformed point such that the arbitrary point is translated back to its original position using translation vector (X_f, Y_f, Z_f)
    i.e.
    $X' = X + X_f$ i.e. $X' = S_x . (X - X_f) + X_f$
    $Y' = Y + Y_f$ i.e. $Y' = S_y . (Y - Y_f) + Y_f$
    $Z' = Z + Z_f$ i.e. $Z' = S_z . (Z - Z_f) + Z_f$

- Hence, if p(x,y,z) be scaled to a point p'(x', y', z') by s_x time in X-units, s_y times in y-units and s_z in z-units about arbitrary point (x_f, y_f, z_f) then the equation for scaling is given as

$$x' = x_f + s_x . (x - x_f)$$
$$y' = y_f + s_y . (y - y_f)$$
$$Z' = Z_f + s_z . (z - z_f)$$

- In matrix representation, using homogeneous coordinate this transformed can be performed through multiplication.

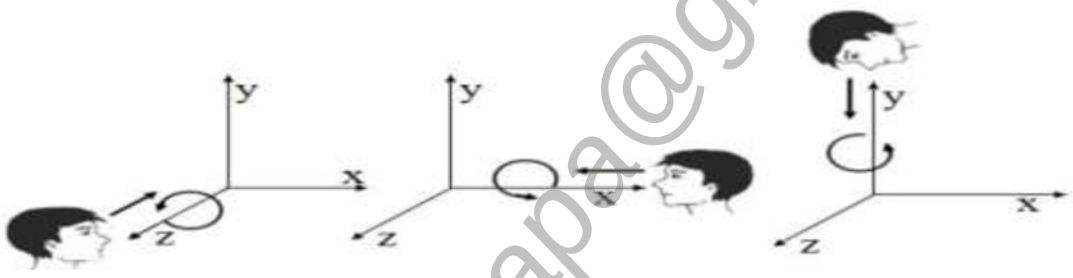
$$T=T(x_f, y_f, z_f) \quad S(S_x,S_y,S_z).T(-x_f, -y_f, -z_f)$$

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

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

3. Rotation

- To generate a rotation transformation for an object, in 3D space, we require the following.
  - i. Angle of rotation
  - ii. Pivot Point
  - iii. Direction of Rotation
  - iv. Axis of Rotation



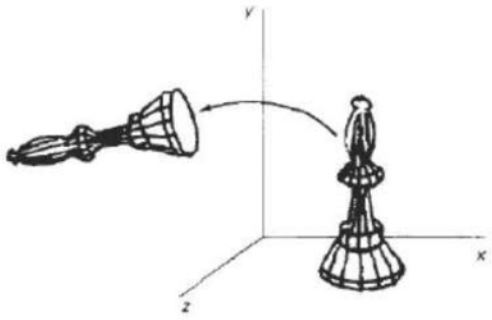
- 3D rotation is slightly different than 2d rotation because the two parameters (Rotation angle and pivot point) do not uniquely define a rotation in 3D space.
- This is because object can rotate along different circular path centering a given pivot point.
- For this, an axis of rotation is specified instead of a center of rotation.

Case 1: Rotation about Z-axis(Anticlockwise): XY plane

- When rotation is performed about Z axis, Z-component does not change but X and Y coordinate changes as in case of 2D.

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

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



Matrix representation for rotation around z-axis,

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Case 2: Rotation about X-axis(Anticlockwise) : YZ plane

Note:
Transformation equation for rotation about the other two axes can be obtained with the cyclic permutations of the coordinate x, y and z. i.e. we use the replacements

$$X \rightarrow Y \rightarrow Z \rightarrow X$$

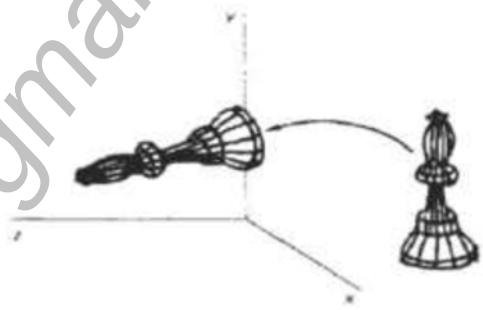
- When rotation is performed about X axis, X-component does not change and Y and Z coordinate changes.

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

Y'= ycosθ –zsinθ

Z'= ysinθ + zcosθ

X' = X



Matrix representation for rotation around x-axis

$$\therefore R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Case 3: Rotation about Y-axis(Anticlockwise): XZ plane

- When rotation is performed about Y axis, Y-component does not change and X and Z coordinate changes.

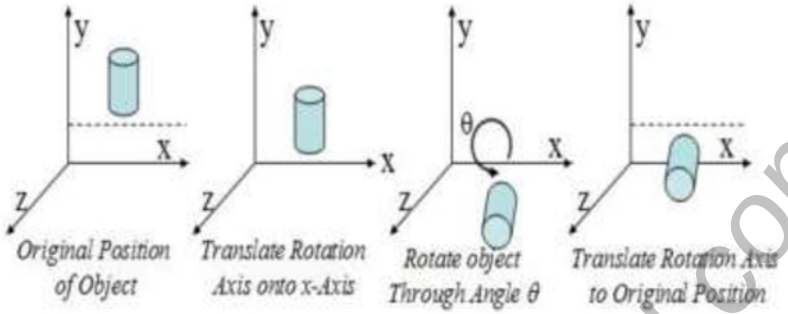
$$\mathbf{P' = R_y(\theta) \cdot P}$$

z' = zcosθ – xsinθ
x' = zsinθ + xcosθ
y' = y

- Matrix representation for rotation around y-axis,

$$R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

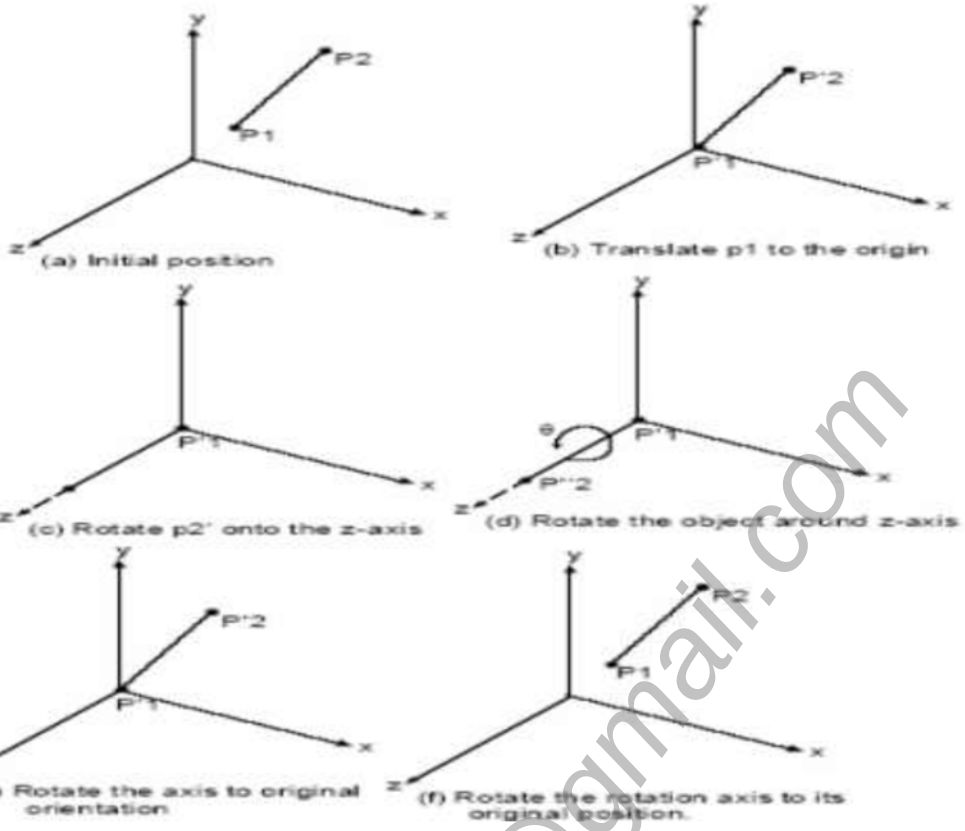
Case 4: Rotation about any axis parallel to any coordinate axis



- When an object is to be rotated about an axis parallel to one of the coordinate axis, we need to perform some series of transformations.
  - **i.** Translate the object so that the rotation axis coincides with the parallel coordinate axis.
    T(-a,-b,-c), where (a,b,c) is any point on the rotation axis
  - ii. Perform the specified rotation about that coordinate axis(X or Y Axis or Z Axis)
    R(θ)
  - iii. Translate the object so that the rotation axis is moved to its original position
    T(a, b, c)
- Hence the net transformation for rotation about any axis parallel to any coordinate axis is
  T= T(a, b, c) R(θ) T(-a, -b, -c)

Case 5: Rotation about any arbitrary axis not parallel to any coordinate axis

- When and object is to be rotated about an axis that is not parallel to any of the coordinate axis, we need to perform some series of transformations **steps** as below.
  i. Translate the object such that the rotation axis passes through the coordinate origin.
  ii. Rotate the axis such that axis of rotation coincides with one of the coordinate axis.
  iii. Perform specific rotation about that coordinate axis.
  iv. Apply inverse rotation to bring the rotation axis back to its original orientation.
  v. Apply inverse translation to bring the rotation axis back to its original position.
- The above steps can be diagrammatically illustrated as below



Composite transformation matrix is obtained as

Step-1: Translate the arbitrary axis so that it passes through the origin

Here, the translation matrix is given by:

$$T = \begin{pmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Step-2: Align the arbitrary axis on any major co-ordinate axis (z-axis)

Here, at first to find the angle of rotation, project the arbitrary axis on yz-plane so that the x-coordinate will be eliminated. Here, we can find angle 'k' for rotation about x-axis by projection & 'α' for rotation about y-axis directly. After these two rotations, the arbitrary axis will align with the z-axis.

2. a: Rotate about X-axis by angle K in anticlockwise direction so that the arbitrary axis lies on xz plane.

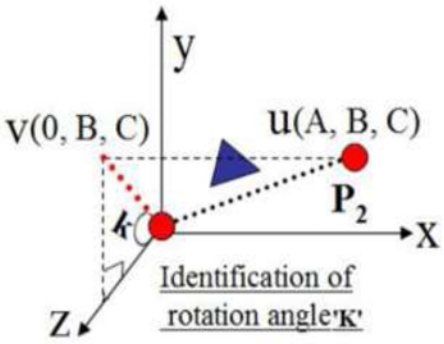
Length of the line OU= $\sqrt{(A^2 + b^2 + c^2}$ (i.e.. length)

Here,

Sin k = B/V &
Cos k = C/V

Now, the translation matrix is given by:

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \text{Cos k} & \text{-Sin k} & 0 \\ 0 & \text{Sin k} & \text{Cos k} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C/V & -B/V & 0 \\ 0 & B/V & C/V & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



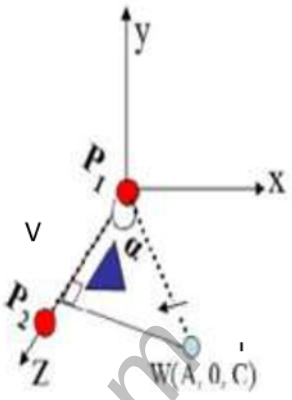
Where, V= OV = $\sqrt{(B^2 + C^2)}$

b: Rotation about y-axis by an angle α in clockwise direction so that the arbitrary axis aligns with z-axis.

Here, ow=ou= $\sqrt{(A^2+b^2+c^2}$

Sinα= A/W

Cosα=V/W, where V=$\sqrt{(W^2-A^2)}$ =$\sqrt{(A^2+B^2+C^2-A^2)}$=$\sqrt{(B^2+C^2)}$



$$R_Y=\begin{pmatrix} cos\alpha & o & sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -sin\alpha & 0 & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$=\begin{pmatrix} V/W & o & A/W & 0 \\ 0 & 1 & 0 & 0 \\ -A/W & 0 & V/W & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Since, the rotation axis is aligned to the z axis, now perform the specified rotation (anticlockwise assumed here) about angle θ about Z-axis

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Apply inverse rotation about y-axis and X-axis.
- Finally apply inverse translation.

- Therefore the final composite matrix is given as
  T = T(tx,ty,tz).R(xaxis,-k).R(yaxis, α).R(Z-axis, θ).R(yaxis, -α).R(xaxis, k). T(-tx,-ty,-tz)

4. Reflection

- In 3D reflection, the reflection can also occur through a plane xy, yz or zx in addition to x, y and z axis.
- But in case of 2D reflection take place about XY plane i.e. about Z-axis

i. Reflection through XY plane

- Here only the z coordinate Values change i.e. reversed in sign. Thus the transformation matrix is given as

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

ii. Reflection through YZ plane

- Here only the X Coordinate Values change i.e. reversed in sign. Thus the transformation matrix is given as

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

iii. Reflection through ZX plane

- Here only the Y Coordinate Values change i.e. reversed in sign. Thus the transformation matrix is given as

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

iv. Reflection through X axis

- Reflection about X axis is equivalent to 180^0 rotation about X axis.

Matrix representation for rotation around x-axis

$$\therefore R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 180 & -\sin 180 & 0 \\ 0 & \sin 180 & \cos 180 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

v. Reflection through Y axis

- Reflection about Y axis is equivalent to 180^0 rotation about Y axis.

$$R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos 180 & 0 & \sin 180 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 180 & 0 & \cos 180 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

vi. Reflection through Z axis

- Reflection about Z axis is equivalent to 180^0 rotation about Z axis.

Matrix representation for rotation around z-axis,

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos 180 & -\sin 180 & 0 & 0 \\ \sin 180 & \cos 180 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

5. Shear

- Shearing transformation alters the shape of the object.

i. X-Shear

- This transformation alters Y and Z coordinate values by an amount that is proportional to the X value while leaving the X value unchanged i.e.

$$X' = x$$
$$Y' = y + S_{hy}x$$
$$Z' = z + s_{hz}x$$

- Matrix representation for X-shear is given as

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ s_{hy} & 1 & 0 & 0 \\ s_{hz} & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ii. Y-Shear

- This transformation alters X and Z coordinate values by an amount that is proportional to the Y value while leaving the Y value unchanged i.e.

$$y' = y$$
$$x' = x + S_{hx}y$$
$$Z' = z + s_{hz}y$$

- Matrix representation for Y-shear is given as

$$\begin{bmatrix} 1 & S_{hx} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & S_{hz} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

iii. Z-Shear

- This transformation alters X and Y coordinate values by an amount that is proportional to the Z value while leaving the Z value unchanged i.e.

$$x' = x + S_{hx}.z$$
$$y' = y + S_{hy}.z$$
$$Z' = z$$

a. Matrix representation for Z-shear is given as

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & S_{hx} & 0 \\ 0 & 1 & S_{hy} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Numeric Example
***See class notes*……..'**

3D Viewing Pipeline

- The steps for computer generation of a view of 3D scene are analogous to the process of taking a photograph - by a camera. For a snapshot, we need to position the camera at a particular point in space and then need to decide camera orientation. Finally, we snap the shutter, the seen is cropped to the size of window of the camera and the light from the visible surface is **projected** into the camera film.

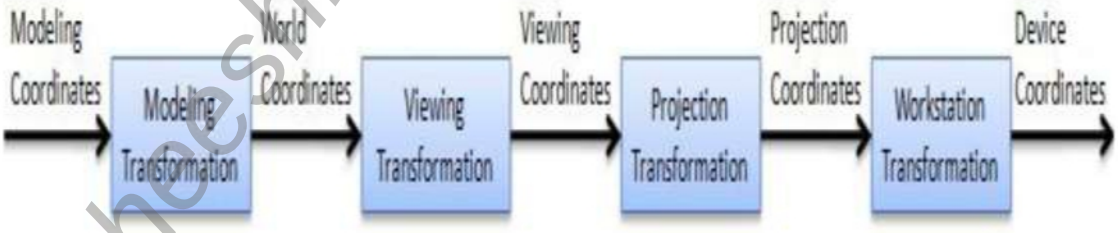


Fig: 3D viewing pipeline

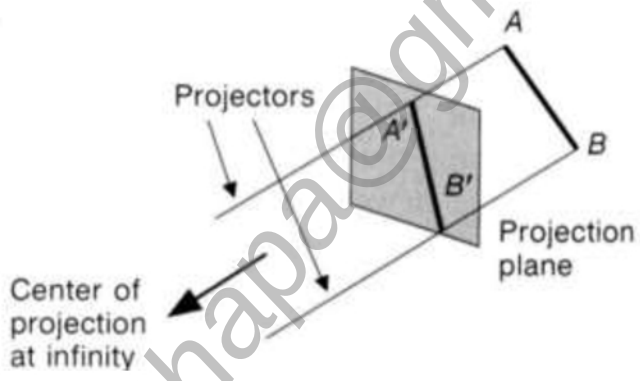
- Figure above shows the general processing steps for modelling and converting a world coordinate description of a scene to device coordinate.
- Once the scene has been modeled, world coordinate positions are converted to viewing coordinates.
- Next projection operations are performed to convert the viewing coordinates description of the scene of coordinate position on the projection plane, which will be than mapped to output device
- Object outside the specified viewing limits are clipped from further consideration and remaining objects are processed to produce the display within the device viewport.

Projection

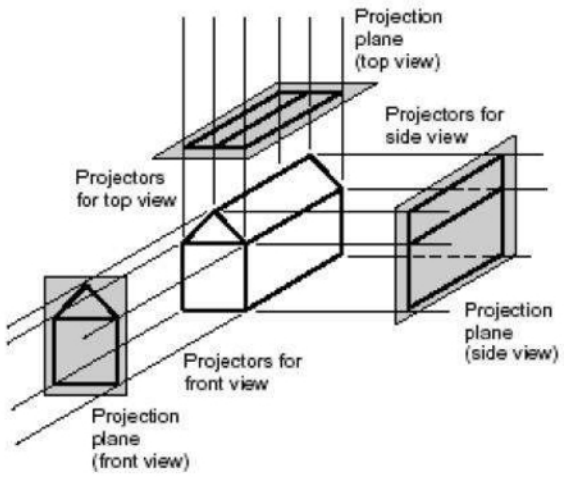
- All the object available in the nature are described in 3D World Coordinate system.
- But all graphics devices are 2D, such as monitor, printer.
- When we have to draw a 3D object on 2D output device, we have to transform the world coordinate into screen coordinate. This is achieved by projecting a 3D object on 2D plane.
- So projection is a process of representing a three dimensional object or scene into two dimensional mediums i.e. convert 3D object to 2D screen coordinate or device coordinate.
- Generally, a projection transforms N dimensions' points to N-1 dimensions.
- Projection are broadly classified into two types
  1. Parallel Projection
  2. Perspective Projection

Parallel Projection

- In parallel projection, the coordinate positions are transformed to the view plane along parallel lines i.e., all projectors are parallel to each other.
- Projectors are the line emerging from center of projection and hitting the projection plane after passing from the point in the object to be projected.
- Parallel lines are still parallel after projection.
- The center of projection (Projection Reference Point i.e. point from where projection is taken) is at infinity, shown in fig below



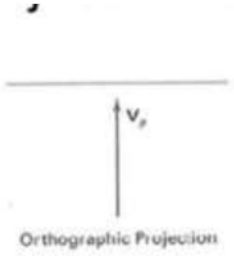
- A parallel projection preserves the relative proportions of the object but it does not give us realistic representation.
- It is used in engineering and architectural drawings to produce scale drawing of three dimensional object. This provide accurate views of various sides of an objects as shown in figure below.

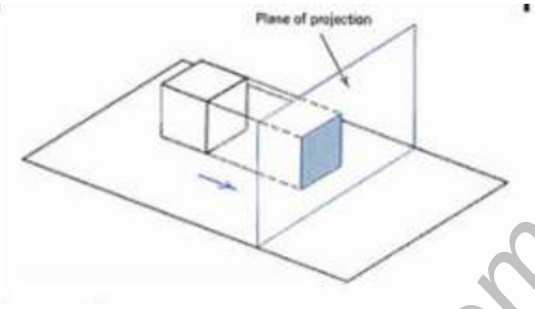


- There are two type of parallel projection
  1. Orthographic parallel projection
  2. Oblique parallel projection

Orthographic parallel Projection

- A parallel projection in which the angle between the projectors and the plane of projection is equal to 90^0 i.e. perpendicular is called orthographic parallel projection.
- Mostly used for engineering drawing.





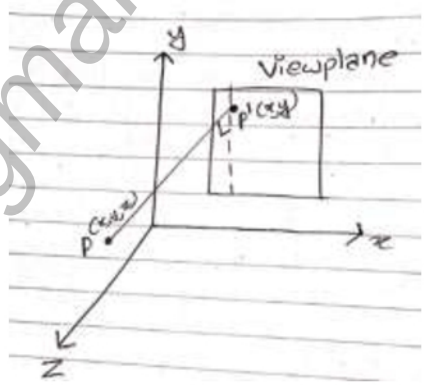
- So a true size and shape of a single face of an object is obtained.

Transformation matrix for orthographic parallel projection

- Transformation equations for an Orthographic parallel projection are straight forward.
- If the view plane is placed at the xy Plane as shown in figure aside, then any Point p(x,y,z) in viewing coordinate is transformed to projection coordinate as

  x'= x

  y'=y
- The original z-coordinate is preserved for the depth information which might be necessary for visible surface determination procedures

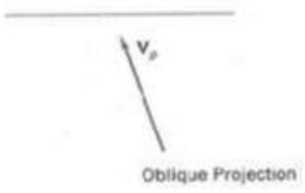


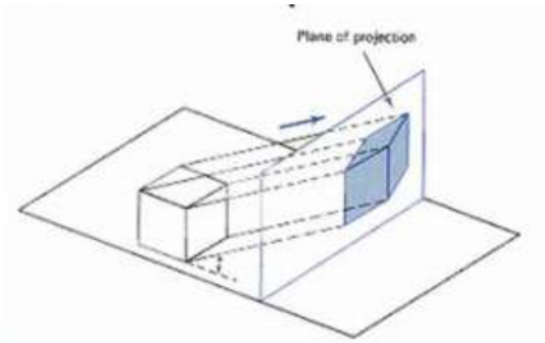
- Therefore, the transformation matrix in homogeneous coordinate form for producing any orthographic parallel projection can be written as.

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

Oblique parallel Projection

- A parallel projection in which the angle between the projectors and the plane of projection is not equal to 90 i.e. equal to oblique angle is called oblique projection.

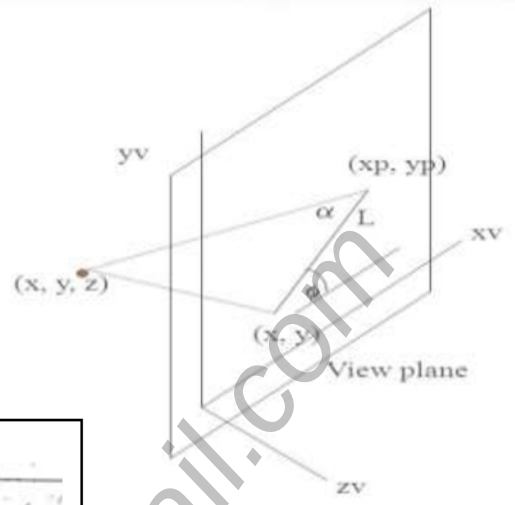




- Since the front face of the object is placed parallel to the plane of projection, hence we get true sized and shaped shadow.

Transformation Matrix for Oblique Transformation

- In the figure given aside, a Point P(x,y,z) is projected to Point P'(x_p,y_p).
- The oblique projection line from P(x,y,z) to P'(x_p,y_p) makes angle α with the line L on projection plane that joins (x,y) and (x_p,y_p)
- Line L is at angle φ with the horizontal Direction in projection plane.
- So the projection coordinates in terms of x,y,L and φ can be expressed as



From the figure aside

$\cos\phi = \frac{x_1}{L}$

$x_1 = L\cos\phi$

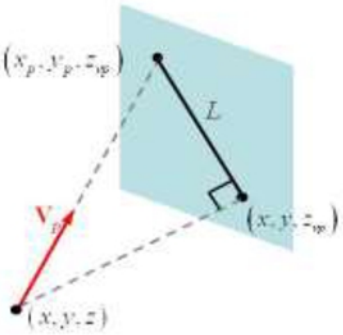
and

$\sin\phi = \frac{y_1}{L}$

$y_1 = L\sin\phi$

$\therefore x_p = x + x_1 = x + L\cos\phi$

$y_p = y + y_1 = y + L\sin\phi$ → (1)



Length L depends on angle α and z-co-ordinate of point to be projected.

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

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

$L = zL_1$, where $L_1 = 1/\tan\alpha$

From equation 1,

$x_p = x + zL_1\cos\phi$

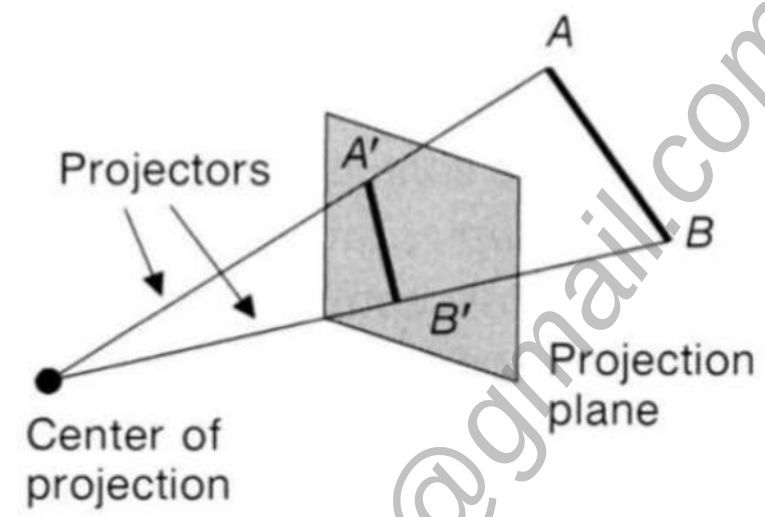
$y_p = y + zL_1\sin\phi$

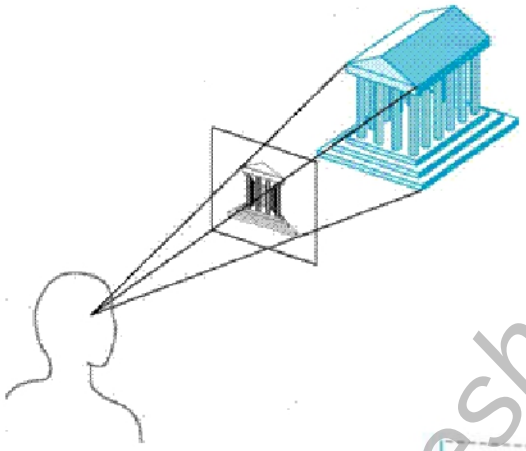
∴ The transformation matrix for producing any oblique parallel projection can be written as

$$\begin{bmatrix} x_p \\ y_p \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & L_1\cos\phi & 0 \\ 0 & 1 & L_1\sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Perspective Projection

- In perspective projection, object positions are transformed to the view plane along lines that converge to a point called as a center of projection (Convergence point/ Projection Reference Point).
- Center of projection is point from where projection is taken, for example, eye position
- Projectors are not parallel to each other.
- The center of projection is at a finite distance.
- Produces realistic views but does not preserve relative proportions.
- Projections of distance objects are smaller than the projection of the same size object that are closer to the projection plane
- Used in realistic displaying because it resembles to that our photographic systems (camera) and human eye.





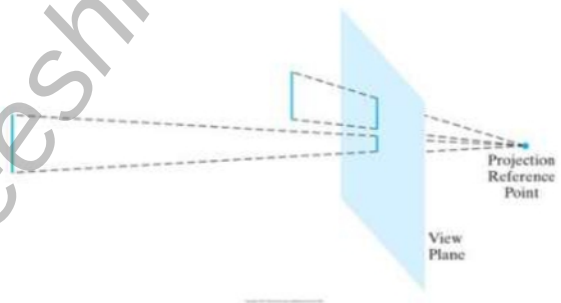
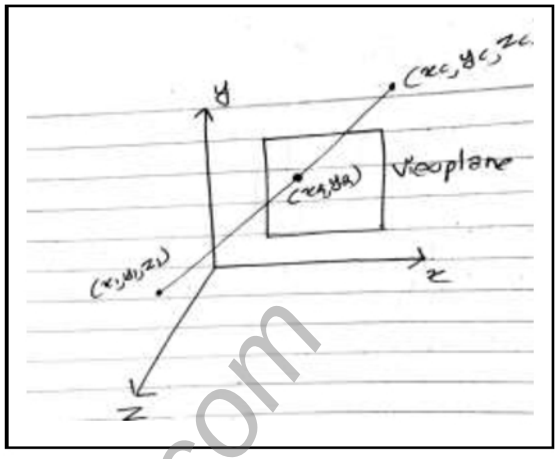


Figure 10-33 A perspective projection of two equal-length line segments at different distances from the view plane.

Transformation matrix for perspective projection

Let us assume that the center of projection is at (x_c, y_c, z_c) and the point on the object is (x1,y1,z1), then the parametric equation of line containing these points is



$X = u.x_1 + (1-u)x_c$

$= u.x_1 + x_c - ux_c$

$= x_c + u(x_1 - x_c)$……..1

similarly,

$Y = y_c + u(y_1 - y_c)$……..2

$Z = z_c + u(z_1 - z_c)$……..3

Where u is the parameter such that u = 0 to 1

- For the projection plane xy, z = 0

  Hence from equation 3,

  $0 = z_c + u(z_1 - z_c)$

  i.e. $u = -z_c / (z_1 - z_c)$

- Putting the value of U in equation 1 and 2 we get

  $X_2 = X_c - Z_c (x_1 - x_c) / (z_1 - z_c)$

  $= (x_c z_1 - x_c z_c - x_1 z_c + z_c x_c) / (z_1 - z_c)$

  $= (x_c z_1 - x_1 z_c) / ((z_1 - z_c)$

- Similarly

  $y_2 = (y_c z_1 - y_1 z_c) / (z_1 - z_c)$

- Therefore, the required transformation equation is

$$[x_2, y_2, z_2] = [(x_c z_1 - x_1 z_c)/(z_1 - z_c), (y_c z_1 - y_1 z_c)/(z_1 - z_c), 0]$$

In homogenous form (Homogenous co-ordinate system)

$$[x_2, y_2, z_2, 1] = [x_c z_1 - x_1 z_c, y_c z_1 - y_1 z_c, 0, z_1 - z_c]$$

Note: $X = X_h / h$ i.e. $X_h = X * h$ and putting $h = Z_1 - Z_c$

In Matrix form ,

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} -z_c & 0 & x_c & 0 \\ 0 & -z_c & y_c & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -z_c \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

$$\therefore T_{persp} = \begin{bmatrix} -z_c & 0 & x_c & 0 \\ 0 & -z_c & y_c & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -z_c \end{bmatrix}$$

Parallel vs Perspective Projection

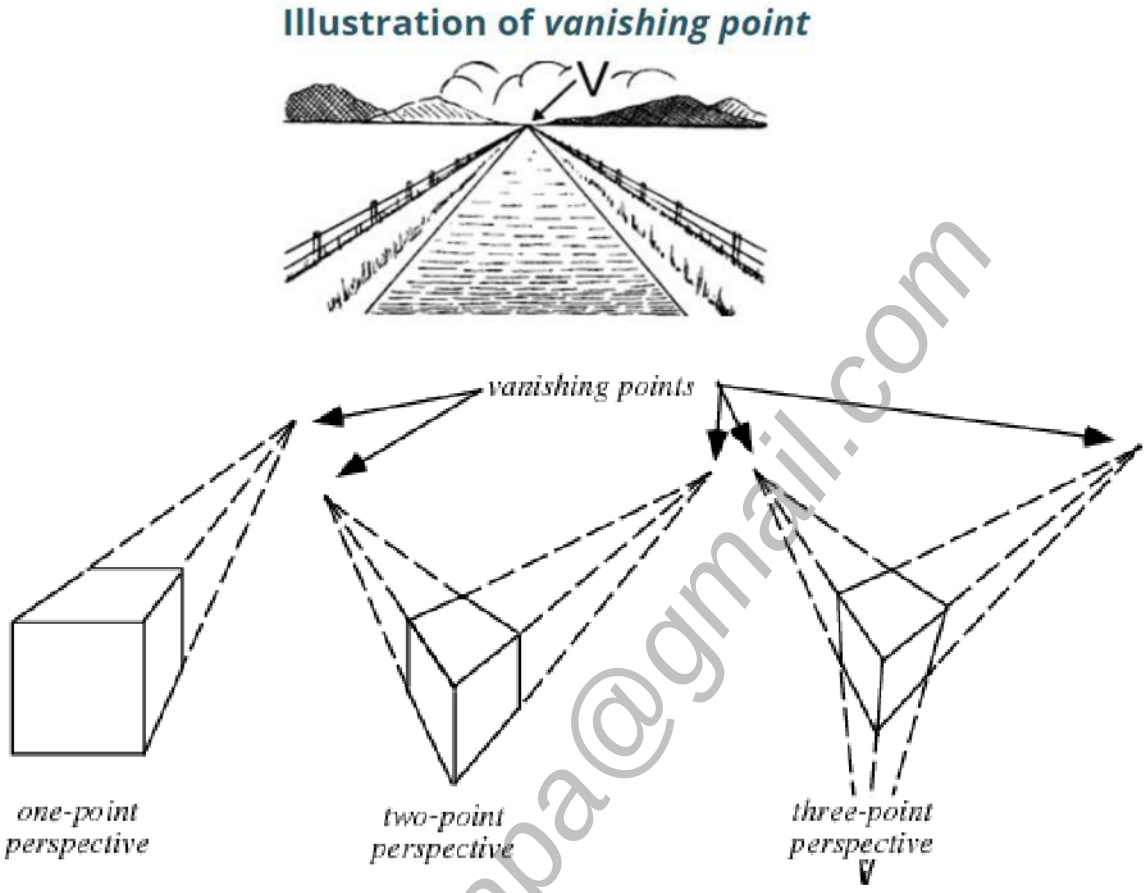
Parallel	Perspective
1. In parallel projection, the center of projection is at infinity.	1. In perspective projection, the center of projection is at finite distance
2. here, all the projectors are parallel to each other.	2. here, projectors are not parallel to each other.
3. It is a less realistic view.	3. It provides realistic view.
4. It can be used for application where exact measurement is required.	4. It resembles to that of photographic and human eye system where exact measurement is not required.
5. Useful for drawing schematic drawing.	5. Useful in architectural rendering, realistic view.
6. A parallel projection preserves the relative proportions of the object but it does not give us realistic	6. A perspective projection doesn't preserves the relative proportions of the object but it give us realistic

representation.	representation.

Assignment:

- **Define Vanishing Point. Explain 1 point, 2 point and 3-point perspective projection with suitable diagrammatic illustration.**

Vanishing Point

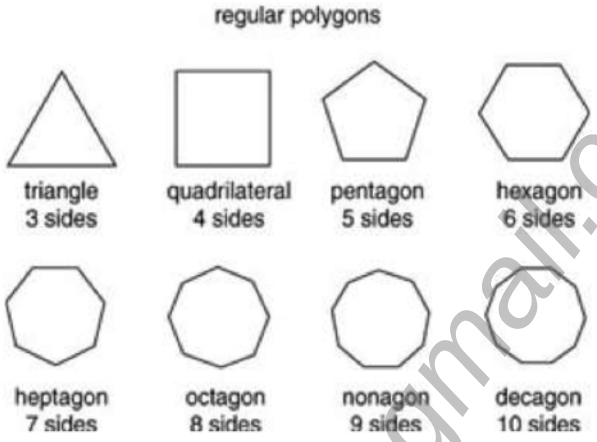


- In perspective projection farther away object from the viewer, small it appears. This property of projection gives an idea about depth.
- When a three-dimensional object is projected onto a view plane using perspective transformation equations, any set of parallel lines in the object that are not parallel to the plane, converge at vanishing point i.e. projected into converging lines
- Parallel Lines that are parallel to the view plane will be projected as parallel lines.
- The point at which a set of projected lines appears to converge is called a **vanishing point or point of convergence.**
- Each such set of projected parallel lines will have a separate vanishing point; and in general, a scene can have any number of vanishing points, depending on how many sets of parallel lines there are in the scene.
- The vanishing point for any set of lines that are parallel to one of the principal axes of an object is referred to as a **principal vanishing point**. The number of principle vanishing point is determined by the number of principle axis intersected by the view plane.
- We control the number of principal vanishing points (one, two, or three) with the orientation of the projection plane, and perspective projections are accordingly classified as one-point, two-point, or three-point projections.
- One point perspective projection occurs when only one principle axis (i.e. X or Y or Z axis) intersects the plane of projections.
- Two point or two principle vanishing point projection occurs when the plane of projection intersects exactly two of the principle axis.
- Three points perspective projection occurs when the projection plane intersects all three of the principle axis i.e. none of the principle axis is parallel to the projection plane.

3D Object Representation

Polygon

- It is often necessary to display solid objects in addition to simple 2d object like line and circle.
- To do so, the most fundamental aspects of graphics primitives called polygon, which is basic surface primitive, is used.
- Polygon is a figure with many sides and can be represented with a group of connected edges, forming a closed figure.
- Example: a triangle is a polygon with three vertices and three edges.



- A polygon is classified into two types.
  1. Concave Polygon
     - A polygon is said to be concave if the line joining any two interior points of the polygon is not going to lie completely inside the polygon.
     - For concave polygon, at least one interior angle must be greater than 180^0.

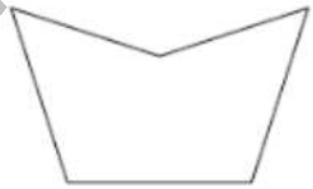


Fig: concave polygon

  2. Convex Polygon
     - A polygon is said to be convex if the line joining any two interior points of the polygon lies completely inside the polygon.
     - For convex polygon, all the interior angle must be less than 180^0
     - Example: triangle, rectangle, square etc. are some example of complex polygon.

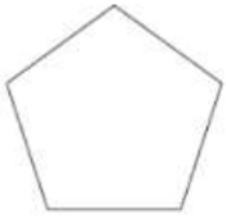


Fig: convex pentagon

Introduction to 3D object Representation

- Graphics scenes can contain many different kinds of 3D objects like trees, flowers, clouds, rocks, water etc.
- Each objects can be characterized on the basis of shape, color, thickness etc.
- There is no one method that we can use to describe objects that will include all features of those different materials. For Example,
    - Simple objects like polyhedrons (a solid figure with many plane faces) and ellipsoids (quadratic surface obtained from sphere) can be represented by **polygon and quadric surfaces.**
    - **Spline surface** are useful for designing aircraft wings, gears and other engineering objects.
    - **Octree encodings method** are used to represent internal features of objects.
- To produce realistic display of scenes, we need to use representations that accurately model object characteristics.
- Representation schemes for **solid objects** are often divided into two broad categories:
    1. Boundary representations (B-reps): Example, Polygon Surface
    2. Space-Partitioning Representations: Example, Octree Representation

Boundary Representation(B-reps)

- Boundary representations (B-reps) describe a three-dimensional object as a set of **surfaces** that separate the object interior from the environment. So it describes the shape of a 3D object using the limits.
- Typical examples of boundary representations are **polygon Surfaces and spline patches**

Polygon Surfaces

- Polygon is the basic building block of all 3d model.
- Polygon Surfaces is the most commonly used boundary representation for a 3D-object that represent the 3D-object as a set of polygons surfaces that enclose the object interior.
- Many graphics system stores all object descriptions as a set of surface polygons.
- This simplifies and speeds up the surface rendering and display of object since all surfaces can be described with simple linear equation of plane: $Ax + By + Cz + D = 0$. For this reason, polygon descriptions are often referred to as "**Standard Graphics Object**".
- The polygon surface is common in design and solid-modeling applications, since **wire frame display** can be done quickly to give general indication of surface structure. Then realistic scenes are produced by interpolating shading patterns across polygon surface to illuminate.
- **A wire frame mode/Representation** for any 3D object is created by specifying each edge of the object where two mathematically continuous smooth surfaces meet, or by connecting an object's constituent vertices using straight lines or curves.

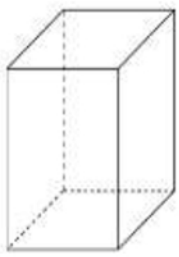


Fig: A 3D object represented by polygon surface

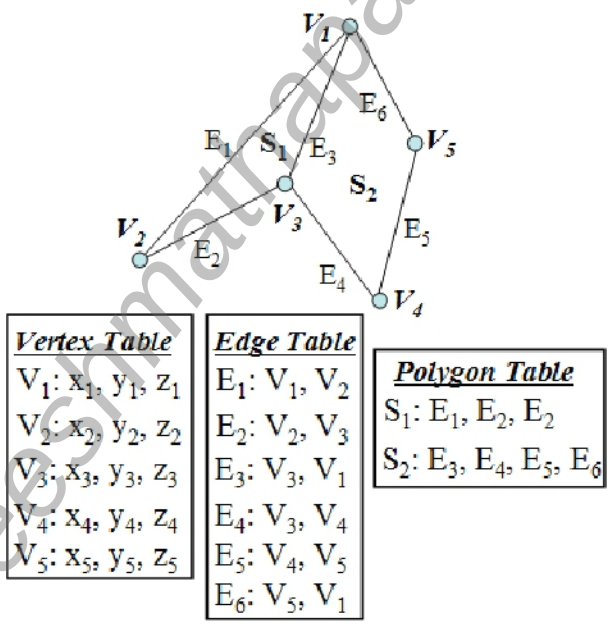
- For example, the above polyhedron can be represented as a set of six polygon surfaces and this information are stored in polygon table.

- Wire Frame does not contain surfaces, textures, or lighting. Instead, a wireframe model is a 3D image comprised of only "wires" that represent three-dimensional shapes.
- Wireframes provide the most basic representation of a three-dimensional scene or object. They are often used as the starting point in 3D modeling since they create a "frame" for 3D structures.
- If an object is in the form of cylindrical shape, we can't represent it using polygon surfaces and require some sophisticated technique called **polygon meshes** in which surfaces are **tiled.**

Polygon Table

- To specify a polygon surface, a set of vertex coordinates and associated attribute parameters are placed into tables called polygon table.
- These are used in the subsequent processing, display, error checking and manipulation of the objects in a scene.
- Polygon data table can be organized into two groups: Geometric table and Attribute Table

1. **Geometric Table**
   - Geometric data tables contain vertex coordinates and parameters to identify the spatial orientation of the polygon surfaces.
   - A convenient organization for storing geometric data is to create 3 lists as below
     - A vertex table: Table that stores coordinate of each vertex in the object.
     - An edge table: Table that stores the edge information of each polygon edge of polygon facets (a small plane surface). Edge table has pointer to vertex list.
     - A polygon surface table. Table that stores the surface information of each polygon surface. Each surface is represented by edge lists of polygon. Polygon surface table has pointer to edge list.



   - Consider the surface contains polygonal facets as shown in figure below. Here S1 and S2 are two polygon surface that represent the boundary of some 3D object. For storing geometric data, we can use following three tables.

2. **Attribute tables**
   - Attribute information for an object includes parameters specifying the degree of transparency of the object and its surface reflectivity and texture characteristics.
   - The above three table also include polygon attribute table according to their pointer information.

Plane Equation

- While producing a display of 3D object, we may need information about the spatial orientation of the individual surface components of the object.
- The information about spatial orientation of individual surface is obtained from vertex coordinate values and equation of each polygon plane.
- The equation for plane surface can be expressed in the form of:

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

Where (x, y, z) is any point on plane, & A, B, C, D are constants describing spatial properties of the plane. The values of A, B, C, D can be obtained by solving set of three plane equations using coordinate values of any three non-collinear points (Points that don't lie on same straight line) on plane. For this we select three successive polygon vertices (x_1,y_1,z_1), (x_2,y_2,z_2) , (x_3,y_3,z_3) and solve the following set of linear plane equation.

$$Ax_1 + By_1 + Cz_1 + D = 0 \quad \ldots\ldots\ldots\ldots (1)$$

$$Ax_2 + By_2 + Cz_2 + D = 0 \quad \ldots\ldots\ldots\ldots (2)$$

$$Ax_3 + By_3 + Cz_3 + D = 0 \quad \ldots\ldots\ldots\ldots (3)$$

i.e. $Ax_i + By_i + Cz = -D$

$$(A/D)x_i + (B/D)y_i + (C/D)x_i = -1, \text{ for } i=1,2 \text{ and } 3$$

The solution of the above linear equations for the ratios (A/D), (B/D) and (C/D) can be obtained in the determinant form using Cramer's rule as:

$$\begin{matrix} + & - & + \end{matrix}$$

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

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

- Expanding the determinant, the coefficient for plane coefficients can be calculated as below.

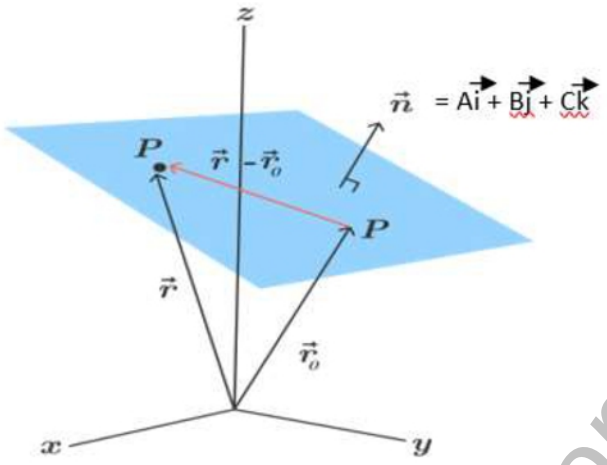
$A = -Y_1(Z_3 - Z_2) + Y_2(Z_3 - Z_1) - Y_3(Z_2 - Z_1)$ (Expanding across second column)

$B = Z_1(X_2 - X_3) - Z_2(X_1 - X_3) + Z_3(X_1 - X_2)$ (Expanding across third column)

$C = X_1(Y_2 - Y_3) - X_2(Y_1 - Y_3) + X_3(Y_1 - Y_2)$ (Expanding across first column)

$D = -[X_1(Y_2Z_3 - Y_3Z_2) - X_2(Y_1Z_3 - Y_3Z_1) + X_3(Y_1Z_2 - Y_2Z_1)]$ (Expanding across first column)

- As vertex values are already stored in vertex table, values of A, B, C and D can be computed for each polygon and stored with the other polygon data.



- Now orientation of plane surface in space can be described with the normal vector as shown in the figure below which has Cartesian component (A, B, C), where A, B and C are the plane coefficients.
- Plane equation also can be used to identify the position of spatial points relative to the plane surface of an object.
- For any point (x, y, z) and plane surface with parameter A, B, C and D

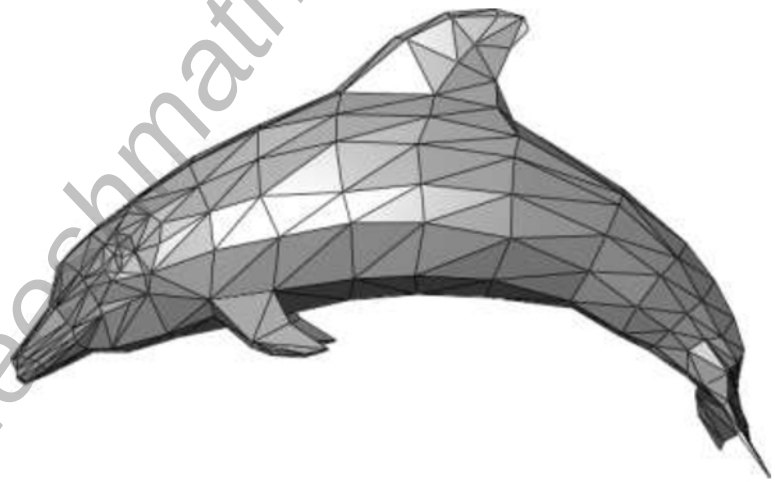
  Ax + By + Cz + D≠0 means Point (x, y, z) is not on plane

  Ax + By + Cz + D<0 means the point (x, y, z) is inside the surface.

  Ax + By + Cz + D>0, means the point (x, y, z) is outside the surface.

Polygon mesh

- A polygon mesh is a surface that is constructed out of a set of polygons that are joined together by common edges.



- A polygon mesh is a collection of edges, vertices and polygons(faces) that defines the shape of 3d object.
- When object surface is to be tiled, it is more convenient to specify the surface face with a mesh function.
- One type of polygon mesh is triangle strip/patch. This function produce n-2 connected triangles for n vertices

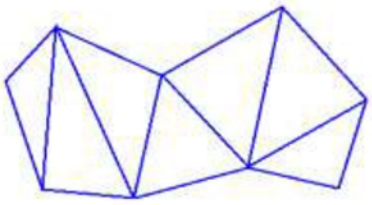


Fig: A triangle strip formed with 7 triangle connection 9 vertices

- Another similar functions is the quadrilateral mesh/patch that generates a mesh of (n-1). (m-1) quadrilaterals, given the coordinates for an n by m array of vertices.

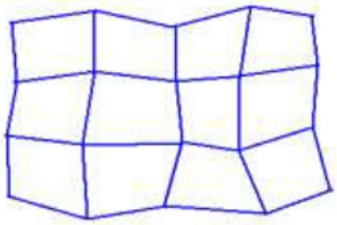


Fig: A quadrilateral meshes with 20 vertices containing 12 quadrilaterals constructed from a 5 by 4 input vertex array

- High quality graphics systems typically model objects with polygon meshes and set up a database of geometric and attribute information to facilitate processing of the polygon facets.
- Fast hardware-implemented polygon renderers are incorporated into such systems with the capability for displaying hundreds of thousands to one million or more shaded polygons per second, including the application of surface texture and special lighting effects

Representing Polygon Meshes

Technique-1

- In first technique, Polygon meshes are represented using Vertex table and polygon/Face table

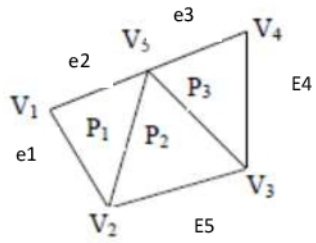


Figure above is a polygon mesh, formed with triangle strip, having three polygon surfaces/ faces named P1, P2 and P3 and 5 vertices named V1, V2, V3, V4 and V5.

- Each polygon faces are represented by a list of vertex co-ordinates in vertex table and polygon table as given below

Vertex Table
V1: (X1, Y1, Z1)
V2: (X2, Y2, Z2)
V3: (X3, Y3, Z3)
V4: (X4, Y4, Z4)
V5: (X5, Y5, Z5)

Polygon Table
P1: V1, V2, V5
P2: V2, V3, V5
P3: V3, V4, V5

- So in the above technique we see that must of the vertices in polygon table are duplicated hence it is not efficient technique.

Technique2

- In another technique of representing polygon meshes, we define polygon by pointers to an edge list. So here it makes use of vertex table, edge table and polygon/face table.
- In this method, we have vertex list V, represent the polygon as a list of pointers not to the vertex list but to an edge list.
- Each edge in edge list points to the two vertices in the vertex list. Also to one or two polygon, the edge belongs.
- Considering the same polygon mesh given above, we get the following vertex, edge and polygon table.

Vertex Table
V1: (X1, Y1, Z1)
V2: (X2, Y2, Z2)
V3: (X3, Y3, Z3)
V4: (X4, Y4, Z4)
V5: (X5, Y5, Z5)

Edge Table
E1: V1, V2
E2: V1, V5
E3: V5, V4
E4: V4, V3
E5: V3, V2
E6: V5,V2
E7: V5, V3

Polygon Table
P1: E1, E2, E6
P2: E6, E7, E5
P3: E3, E4, E7

3D Curve and Surface

- **3D curved lines** and surfaces can be generated from an input set **of mathematical functions** defining the object or **from set of user specified data points**.
- When a function is used,
  i. For curve, a graphics package can project the **defining equation for a curve** to the display system and plot pixels' position along the path of projected function.
  ii. For surface, a functional description is often represented in the form of polygon mesh (triangular polygon patches) approximation to the surface.
- When a set of discrete coordinate point is used,
  A functional description is obtained that best fits the designated points according to the constraints of the application. Example **Spline representation** are this class of **curve and surfaces Curve fitting** methods are used to display graph of data value by fitting specified curve functions to the discrete data set, using **regression technique such as least square regression**.
- Curve and surface equation can be expressed either in **parametric or nonparametric** form however many graphics **application parametric representation** are generally more convenient.

Note:

If x and y are continuous functions of t on an interval I, then the equations

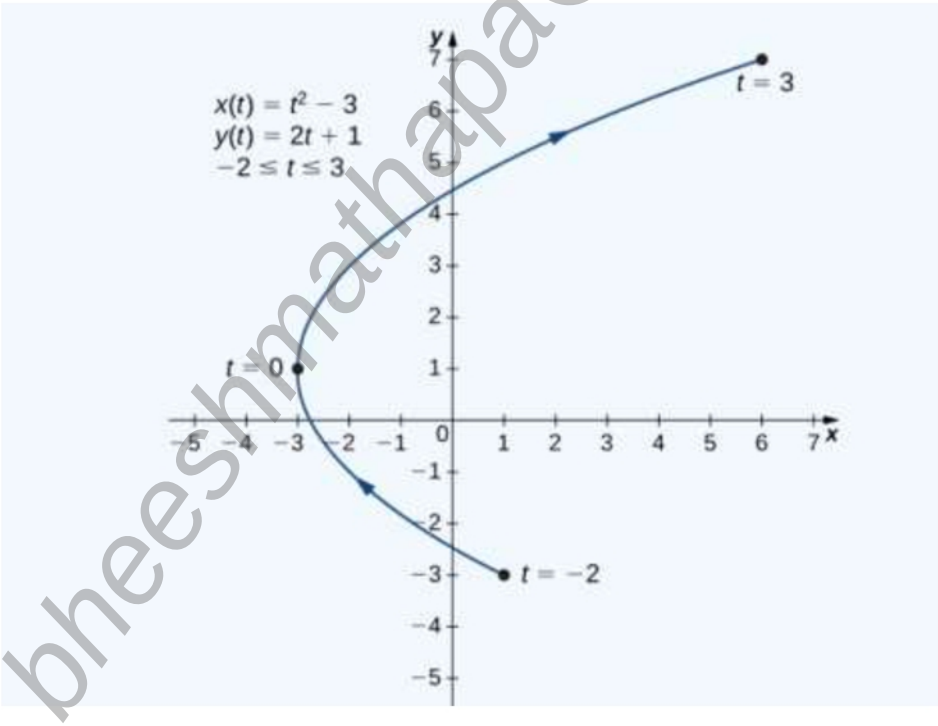
$$x=x(t)$$
and
$$y=y(t)$$

are called parametric equations and t is called the parameter. The set of points (x,y) obtained as t varies over the interval I is called the graph of the parametric equations. The graph of parametric equations is called a parametric curve or plane curve, and is denoted by CC.

For Example:

$$x(t) = t^2 - 3, \quad y(t) = 2t + 1, \quad \text{for } -2 \leq t \leq 3$$

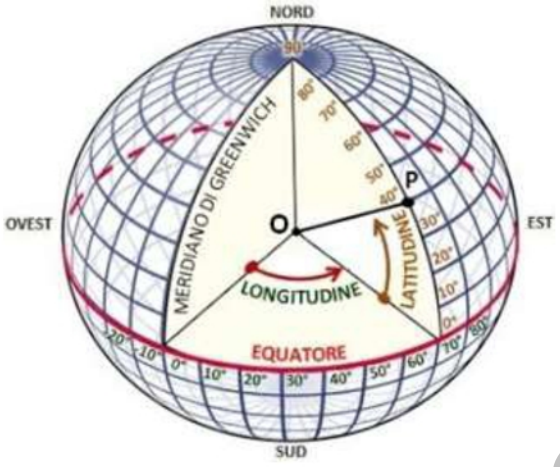
t	$x(t)$	$y(t)$
−2	1	−3
−1	−2	−1
0	−3	1
1	−2	3
2	1	5
3	6	7



Quadratic Surface

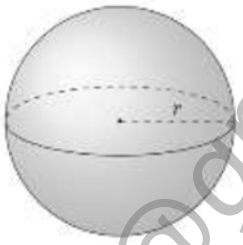
- Quadratic surface is described with second degree equation i.e. quadratic equation.
- They include sphere, ellipsoids, paraboloids, hyperboloids etc.
- Sphere and ellipsoids are common elements in graphics scene and are often available in graphics package as primitive from which more complex object can be constructed.
- A two dimensional object take one parameter as input and give values as X and Y Coordinate similarly a three dimensional object take two parameters as input and give three values as X, Y and Z coordinate.

Sphere



In Cartesian coordinates, a spherical surface with radius r centered on the coordinate origin is defined as the 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, using latitude (vertical i.e. ϕ) and longitude (horizontal: i.e. θ) angle

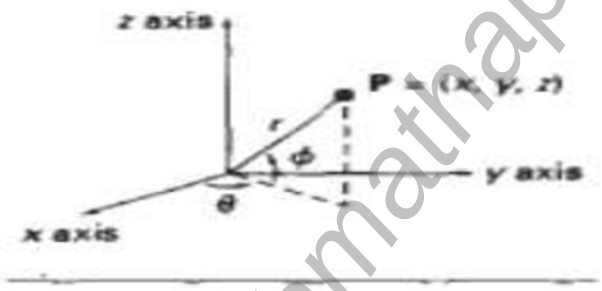


Figure 10-8
Parametric coordinate position (r, θ, ϕ) on the surface of a sphere with radius r

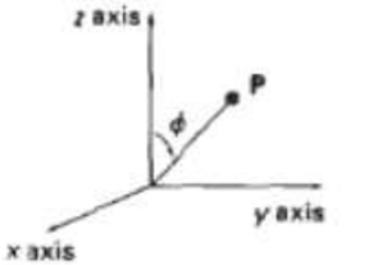


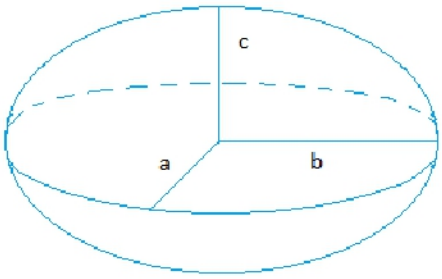
Figure 10-9
Spherical coordinate parameters (r, θ, ϕ), using colatitude for angle ϕ

i.e. $X = r\cos\phi\cos\theta$
$Y = r\cos\phi\sin\theta$
$Z = r\sin\phi$

Where ϕ is in range $-\Omega/2$ to $\Omega/2$ and θ is in range $-\Omega$ to Ω

Ellipsoid

- An ellipsoid is an extension of spherical surface where the radius in three mutually perpendicular directions can have different values.



- The Cartesian representation for points over the surface of an ellipsoid centered on the origin is

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

- And a parametric representation for the ellipsoid in terms of the latitude angle (vertical i.e. ϕ) and the longitude angle (horizontal: i.e. θ) is given as

i.e. $X = r_x \cos\phi\cos\theta$
$Y = r_y \cos\phi \sin\theta$
$Z = r_z \sin\phi$

Where ϕ is in range $-\Omega/2$ to $\Omega/2$ and θ is in range $-\Omega$ to Ω

Curved Line and Spline Representation

- 3D smooth curved line can be generated using **spline representation techniques**.
- A spline is a **flexible strip/patch** used for producing a smooth curve through a **designated set of points**.
- By varying the number and position of the lead weights, the shape of the spline is changed so that it passes through some designated set of points.
- In computer graphics, continuous curve that are formed with polynomial section with certain boundary conditions are **called spline curve**.
- In computer graphics, a spline surface can be described with two set of orthogonal spline curves.
- Splines are of two types
  i. Closed Splines: the generated curve will touch the first and last control point.
  ii. Open Splines: the generated curve will not touch the first and last control point.

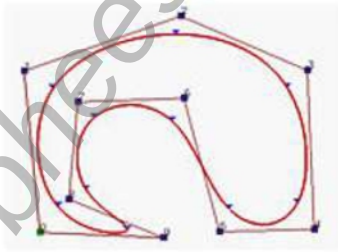


Fig: closed Spline

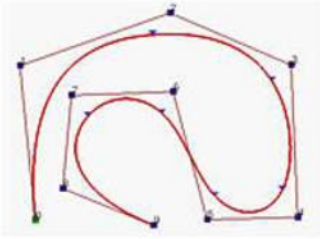


Fig: Open Spline

- 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 parametric polynomial functions in one of two ways.
  i. Approximation Curve
     - The polynomials are fitted to the general control-point path without necessarily passing through any control point.
     - Approximation curves are primarily used as design tools to structure object surfaces.

- Example: **Bezier curve**

ii. Interpolation Curve
- The polynomial sections are fitted by passing the curve through each control points.
- Interpolation curves are commonly used to digitize drawings or to specify animation paths.
- Example: **Hermite curve**

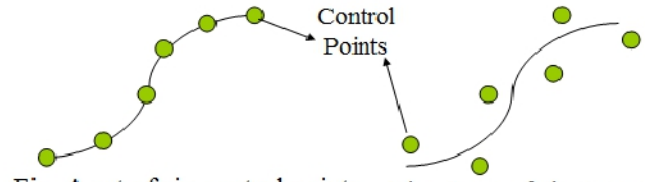


Fig. A set of six control points ***interpolated*** with piecewise continuous polynomial sections

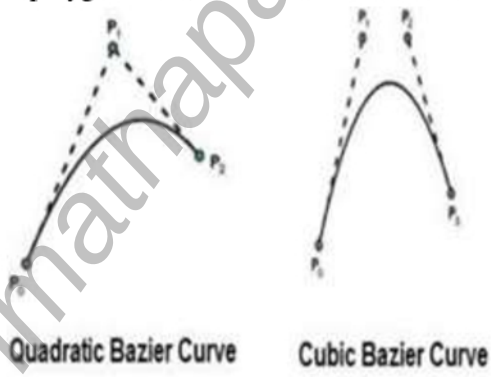
Fig. A set of six control points ***approximated*** with piecewise continuous polynomial sections

- A **cubic** spline is a spline constructed of piecewise **third-**order polynomials which pass through a set of **four control points.**
- Splines are used:
  i. To design curve and surface shapes in graphics applications,
  ii. To digitize drawings for computer storage
  iii. To specify animation paths for the objects or image.

iv. Typical CAD applications for splines include the design of automobile bodies, aircraft and spacecraft surfaces, and ship hulls.

Bezier Curve

- Bezier curve are used in computer graphics to produce **approximate spline curves** which appears reasonably smooth at all scales (as opposed to polygon lines, which will not scale nicely).



- A **quadratic Bezier curve** is defined using three control point and can be defined using two-degree polynomial equation.
- The **cubic Bezier curve** is defined by four points: the **initial position(P_0)** and the **terminating position(P_3)** (which are called "anchors") and two separate **middle points (P_1 and P_2)** (which are called "handles"). The shape of a Bezier curve can be altered by moving the handles.
- Bezier curve section can be fitted to any number of control points, the number of control points to be approximated determine the degree of Bezier polynomial.
- Bezier curve are mostly specified with blending function.
- They are easy to implement, very efficient to construct (Simple recursive process) and have a number of properties that make them highly useful and convenient for curve and surface design.
- The mathematical method for drawing curves was created by Pierre Bézier in the late 1960's for the manufacturing of automobiles.
- The Bezier curve has below listed important properties.
  i. It always passes through the first and last control points.
  ii. We get n degree polynomial for n+1 control points.
  iii. It is approximated parametric spline curve.

iv. It lies within the convex hull (convex polynomial boundary) of the control points. The convex polygon boundary that encloses the set of control points is called convex hull

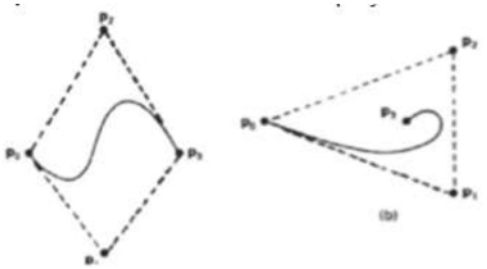


Fig. Convex hull shapes (dashed lines) for two sets of control points.

v. The slope at the beginning of the curve is along the line joining the first two points and slope at the end of curve is along the line joining last two points.
vi. The curve is a **straight line** if and only if all the control points are collinear(lie on single straight line).

Derivation for Cubic Bezier curve generation Algorithm

Given a set of n+1 control points P_0, P_1, ……P_n, a parametric bezier curve segment that will approximately fit to those points is mathematically defined as

$$P(u) = \sum_{i=0}^{n} P_i B_{i,n}(u) \quad \text{where } 0 \le u \le 1 \qquad \ldots\ldots\ldots 1)$$

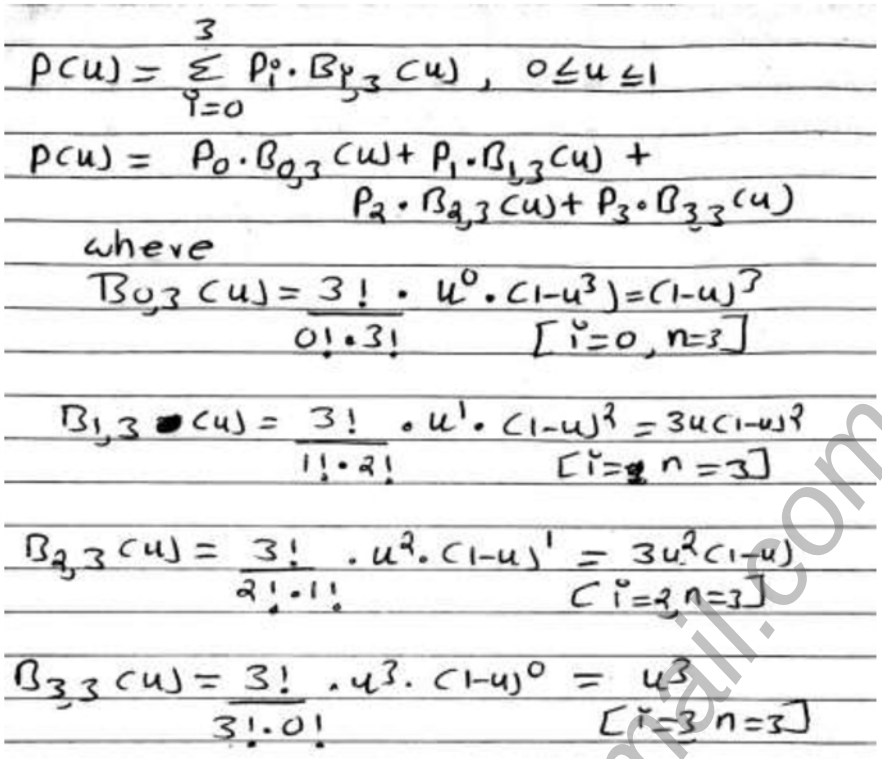
Where $B_{i,n}(U)$ is **the blending function** that determines how much the i^{th} sample point affects the positon of curve i.e. blending function determine the strength of how much a particular control point pull the curve toward it. Blending function for Bezier curve is given as

$$B_{i,n}(u) = C(n,i) \cdot u^i \cdot (1-u)^{n-i}$$

And C(n,i) is the binomial coefficient given as

$$C(n,i) = \frac{n!}{(n-i)! \cdot i!}$$

Now for Cubic Bezier curve, degree n=3 and number of control point=n+1 =4, Then from equation 1,



Substituting the calculated values, we get,

$$P(u) = P_0 (1-u)^3 + P_1 3u(1-u)^2 + P_2 3u^2(1-u) + P_3 u^3$$

where $0 \le u \le 1$

Numerical

1. **Construct the Bezier curve of order 3 with 4 polygon vertex A (1,1), B (2,3), C (4, 3) and D (6,4)**

   Or

   Find equation of Bezier curve which passes through points (1,1) and (6,4) and is controlled through points (2,3) and (4,3).

   (note: Since 4 control point is given, it means we have to create a Bezier curve of order 3)

   Solution:

   The equation for the Bezier curve is given as

   $$P(u) = P_0 (1-u)^3 + P_1 3u(1-u)^2 + P_2 3u^2(1-u) + P_3 u^3 \quad \text{.....1}$$

   where $0 \le u \le 1$ and P(u) is the points on the curve.

   Now, depending on the sequence, we get following four successive points

   $$P_0 = A(1,1),\ P_1 = B(2,3),\ P_2 = C(4,3) \text{ and } P_3 = D(6,4)$$

   let us take u = 0, ¼, ½, ¾, 1

   Hence from equation 1,

At $u=0$

$P(0) = P_0 \cdot (1-0)^3 + 0 + 0 = P_0 = (1,1)$

At $u = 1/4$

$P(1/4) = P_0 \cdot (1-1/4)^3 + P_1 \cdot 3 \cdot \frac{1}{4} \cdot \left(1-\frac{1}{4}\right)^2 +$

$P_2 \cdot 3 \cdot \left(\frac{1}{4}\right)^2 \cdot \left(1-\frac{1}{4}\right) + P_3 \cdot \left(\frac{1}{4}\right)^3$

$= \frac{27}{64}(1,1) + \frac{27}{64}(2,3) + \frac{9}{64}(4,3) + \frac{1}{64}(6,4)$

$= \left(\frac{27}{64}\times 1 + \frac{27}{64}\times 2 + \frac{9}{64}\times 4 + \frac{1}{64}\times 6, \frac{27}{64}\times 1 + \frac{27}{64}\times 3 + \frac{9}{64}\times 3 + \frac{1}{64}\times 4\right)$

$= \left(\frac{123}{64}, \frac{139}{64}\right)$

$= (1.9218, 2.1718)$

Similarly

At $u = \frac{1}{2}$, $P\left(\frac{1}{2}\right) = (3.125, 2.875)$

At $u = \frac{3}{4}$, $P\left(\frac{3}{4}\right) = (4.5156, 3.375)$

At U=1, P(1) = P_3U^3 = (6,4) * 1 = (6,4)

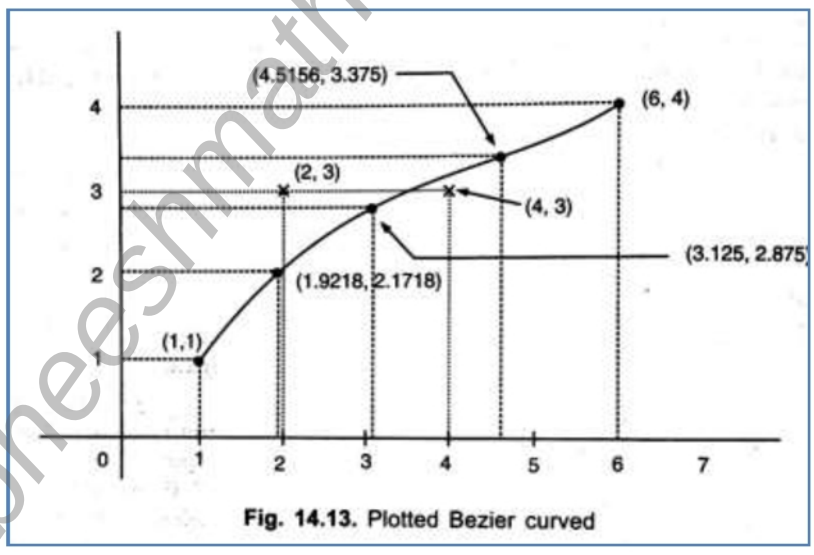


Fig. 14.13. Plotted Bezier curved

Note:

- *The above solution clearly shows that curve always pass through first and last point.*
- *More accurate and realistic curve can be obtained by increasing number of parameter u.*
- *It also clearly shows that all the curve point doesn't exactly passes through the control points thus is an approximation algorithm.*
- *It is also seen that the obtained curve lies inside the convex hull of control point polygon.*

Bezier Surface

- A given Bezier surface of degree (n, m) is defined by a set of (n + 1).(m + 1) control points ki,j
- Two sets of orthogonal Bezier curves can be used to design a **Bezier surface** by specifying by an input mesh of control points.

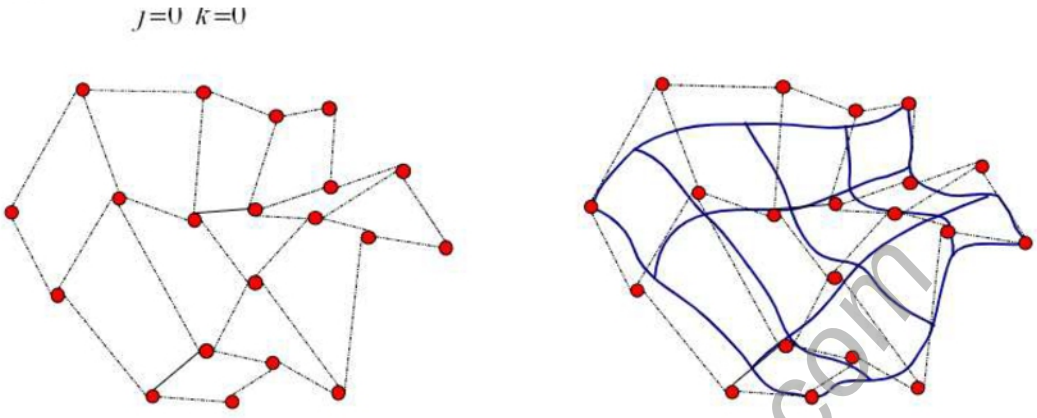


Fig: Bezier Surface

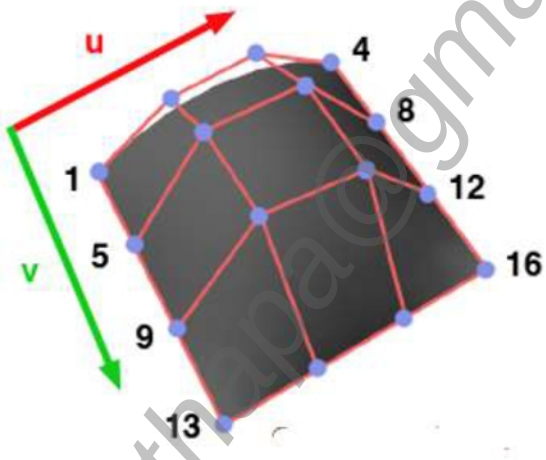


Fig: Bezier Surface

- The principle of Bezier curve extended to obtain Bezier surface.
- Rather than having 4 points (In case of cubic Bezier curve) we will define the surface with 16 points which you can see as a grid of 4x4 control points. In the case of curves, we had 1 parameter to move along the curve (t). In the case of surface, we will need two: one to move in the u direction and one to move in the v direction (see figure above). Both u and v are contained within the range [0,1]. We can see the process of going from (u, v) to 3D space as a remapping of the unit square (defined by u, v) into a smooth continuous surface within the tree dimensional space. intuitively, you can see that a Bézier surface is made of many curves going along either the u or v direction. For instance, if we move in the v direction we need to interpolate the curves defined by the points 1, 2, 3, 4 and 5, 6, 7, 8 (and so on for the rest of the surface). If we move along the u direction, the curves defined by the points 1, 5, 9, 13 and 2, 6, 10, 14 need to be interpolated instead.
- A two-dimensional Bezier surface can be defined as a parametric surface where the position of a point p as a function of the parametric coordinates u, v is given by:

$$P(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u) B_j^m(v) P_{ij}$$

where as for curves,

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}$$

is a **Bernstein polynomial (equation 2)**, and

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

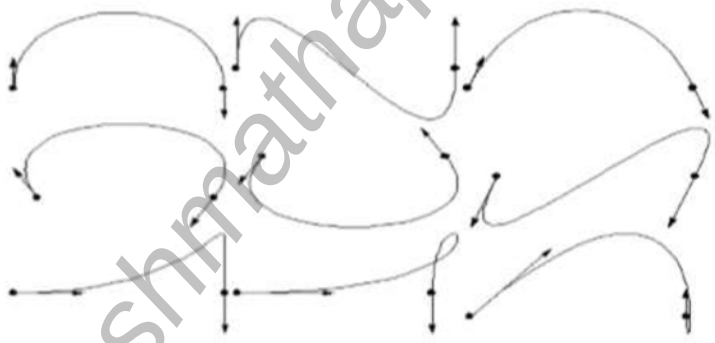
- So a point on the Bezier surface depends of the parametric values u and v, and can be defined as a double sum of control points and coefficients ("a sum of Bezier curves").

Hermite Spline

- Hermite form is defined by 2 endpoints & 2 tangent vectors (i.e. derivative(Slope) of tangent) at these endpoints.

$$\mathbf{P}(0) = \mathbf{p}_k, \quad \mathbf{P}(1) = \mathbf{p}_{k+1}, \quad \mathbf{P}'(0) = \mathbf{Dp}_k, \quad \mathbf{P}'(1) = \mathbf{Dp}_{k+1},$$

- Hermite spline is **an interpolating** piece-wise cubic polynomial with specified tangent at each control point.
- Hermite interpolation only requires 2 end points and two end point tangents, and no intermediate control point are necessary unlike Bezier curve.



- If P(u) represents a parametric cubic point function for the curve section between the control points P_k and P_{k+1}, with a value of DP_k and DP_{k+1} as parametric derivative(i.e. slope of the curve at the control points), then the parametric Hermite spline is mathematically defined as

$$P(u) = P_k(2u^3 - 3u^2 + 1) + P_{k+1}(-2u^3 + 3u^2) + DP_k(u^3 - 2u^2 + u) + DP_{k+1}(u^3 - u^2)$$

$$P(u) = P_k H_0(u) + P_{k+1} H_1(u) + DP_k H_2(u) + DP_{k+1} H_3(u)$$

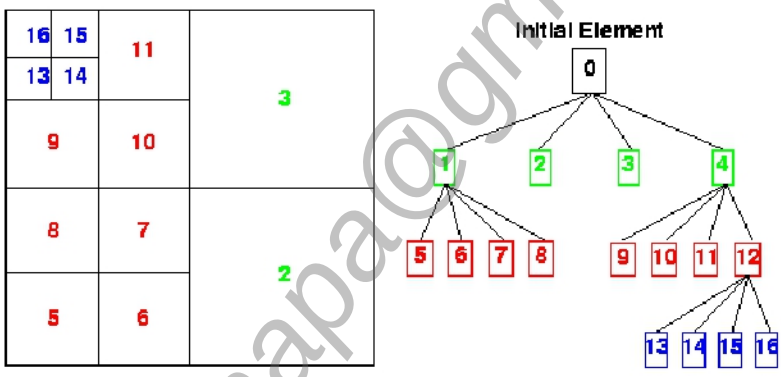
where the polynomial $H_k(u)$ for $k = 0$ to 3 are referred to as blending function.

Space Partitioning Representation: Octree Representation

- Space partitioning are often used to describe interior properties, by partitioning the spatial region containing an object into a set of small, non-overlapping, contiguous solids (usually cubes).
- Space-partitioning systems are often hierarchical, meaning that a space (or a region of space) is divided into several regions, and then the same space-partitioning system is recursively applied to each of the regions thus created.
- The regions can be organized into a tree, called a space-partitioning tree.
- Octree representation is one of the space partitioning representation.

Octree Representation

- Octree are the hierarchical tree structure that are used to represent a solid object.
- It also provides a convenient representation for storing information about object interior (Say color information).
- Application such as medical imaging that requires views of objects cross sections commonly use octree representation.
- The tree structure is organized so that each node corresponds to a region of three dimensional space.
- A quadtree is obtained from successful dividing of a 2 dimensional plane in both the dimension to generate quadrant as shown in below figure.



- So from above figure, we see that each node in qu1adtree has four data elements one for each of the quadrants in the region.
- Similar to quadtree, in an octree, its three dimensions are recursively sub divided into octants and stores eight data element in a node as shown in below figures.

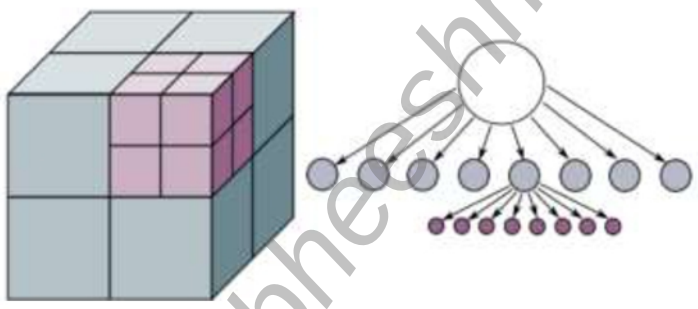


Figure: 1

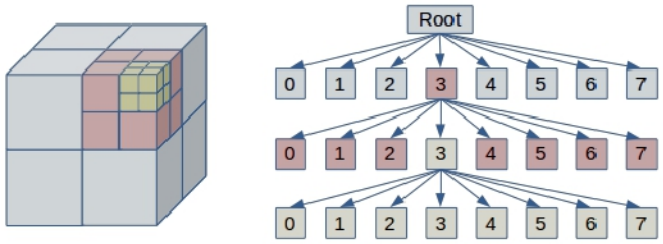
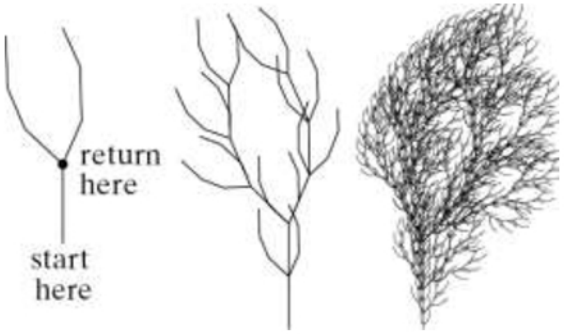


Figure: 2

- If each octant in the original space has a single color, then the octree will have only one root node. Any heterogeneous octant is further subdivided into octants and the corresponding data element in the node points to next node in the octree.

Fractal Geometry Method

- Fractals are very complex pictures generated by a computer from a single iterative or recursive formula.
- This means one formula is repeated with slightly different values over and over again, taking into account the results from the previous iteration



- Natural objects, such as mountains and clouds, have irregular or fragmented features, and these are described with fractal-geometry methods. **Natural objects can be realistically described with fractal-geometry methods, where procedures rather than equations are used to model objects.**
- For fractional-Generation procedure, If P0 = (x0 , y0 , z0) is a selected initial point, each iteration of a transformation function F generates successive levels of detail with the calculations are:

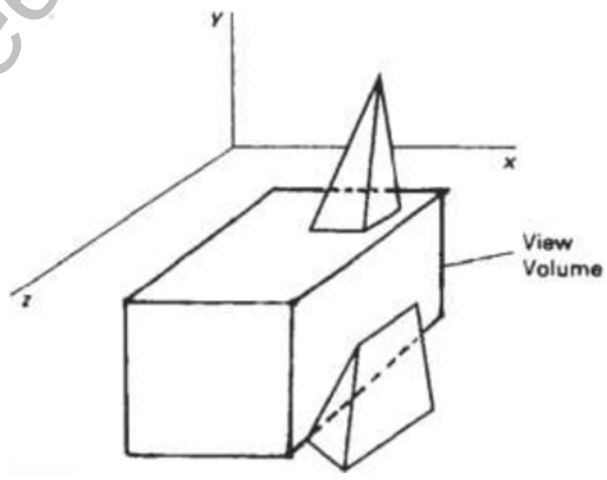
  P1 = F(P0),

  P2 = F(P2), …
- Fractals can be generated by repeating the same shape over and over again as shown in the following figure. In figure aa shows an equilateral triangle.

- In above figure b, we can see that the triangle is repeated to create a star-like shape. In figure c, we can see that the star shape in figure b is repeated again and again to create a new shape.
- We can do unlimited number of iteration or recursions to create a desired shape.

Clipping in 3D

- Clipping in three dimensions can be accomplished using extensions of two-dimensional clipping methods. Instead of clipping against straight-line window boundaries, we now clip objects against the boundary planes of the view volume.



- The view volume defines the three-dimensional volume in space that, once projected, is to fit within the view port.
- There are two parts to the view volume
    i. The view plane rectangle and
    ii. The near and far clipping planes
- **An algorithm for three-dimensional clipping identifies and saves all surface segments within the view volume for display on the output device. All parts of objects that are outside the view volume are discarded.**
- View-volume clipping boundaries are planes whose orientations depend on the type of projection, the projection window, and the position of the projection reference point.
- To clip a polygon surface, we can clip the individual polygon edges. To clip a line segment against the view volume, we would need to test the relative position of the line using the view volume's boundary plane equations.
- An endpoint (x, y, z) of a line segment
    - Is Outside a boundary plane if $Ax + By + Cz + D > 0$,
    - Is Inside the boundary if $Ax + By + Cz + D < 0$

    where A, B, C, and D are the plane parameters for that boundary.
- Lines with both endpoints outside a boundary plane are discarded, and those with both endpoints inside all boundary planes are saved.
- The intersection of a line with a boundary is found using the line equations along with the plane equation. Intersection coordinates (x_1 , y_1 , z_1) are values that are on the line and that satisfy the plane equation $Ax1 + By1 + Cz1 + D = 0$.
- To clip a polygon surface, we can clip the individual polygon edges.
- All objects within the view volume map to the interior of the specified projection window.
- Then the last step is to transform the window contents to a two-dimensional view port, which specifies the location of the display on the output device.