Chapter-5 Three Dimensional Graphics System

3D

- In computer Graphics, 3-D (three dimensions or three-dimensional) describes an image that provides the perception of depth.
- 3D Graphics 3D computer graphics or three-dimensional computer graphics are graphics that use a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering 2D images.
- 2D is "flat", using the horizontal and vertical (X and Y) dimensions, the image has only two dimensions.
- 3D adds the depth (Z) dimension. This third dimension allows for rotation and visualization from multiple perspectives. It is essentially the difference between a photo and a sculpture.

What are the issue in 3D that makes it more complex than 2D?

When we model and display a three-dimensional scene, there are many more considerations we must take into account besides just including coordinate values as 2D, some of them are:

- Relatively more co-ordinate points are necessary in comparison with 2D.
- Object boundaries can be constructed with various combinations of plane and curved surfaces.
- Consideration of projection (dimension change with distance) and transparency.
- Many considerations on visible surface detection and remove the hidden surfaces

3D Geometric Transformation

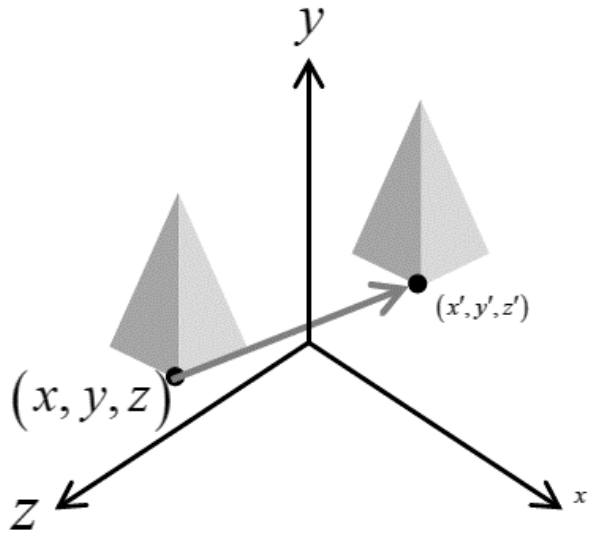
- Methods for geometric transformations and object modeling in three dimensions are extended from two-dimensional methods by including considerations for the z coordinate.
- Translation of an object in 3D is obtained by specifying a three dimensional translation vector, which determines how much the object is to be moved in each of the three coordinate directions.

Homogeneous Representation in 3D

- 2D transformations can be represented by 3 x 3 matrices using homogeneous coordinates.
- Similarly, 3D transformations can be represented by 4 x 4 matrices, by using homogeneous coordinate representations of points in 2 spaces as well.
- Thus, instead of representing a point as (x, y, z), we represent it as (x, y, z, H),
  - Where two these quadruples represent the same point it one is a non-zero multiple of the other the quadruple (0, 0, 0, 0) is not allowed.
- A standard representation of a point (x, y, z, H) with H not zero is given by (x/H, y/H, z/H, 1).
- Transforming the point to (x, y, z, 1) form is called homogenizing.

Translation

- A translation moves all points in an object along the same straight line path to new positions.
- 3DTranslation process contains the x-axis, y-axis, and z-axis.



Let's translate a point from P (x, y, z) to Q (x' y', z') and Translation distance towards x, y and z axis are t_x, t_y, t_z then the path is represented by a vector, called the translation or shift vector. We can write the components as:

$x' = x + t_x$
$y' = y + t_y$
$z' = z + t_z$
$1=1$

We can also represent the 3D Translation in matrix form

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

Example: A Point has coordinates P (1, 2, 3) in x, y, z-direction. Apply the translation with a distance of 2 towards x-axis, 3 towards y-axis, and 4 towards the z-axis. Find the new coordinates of the point?

Solution:

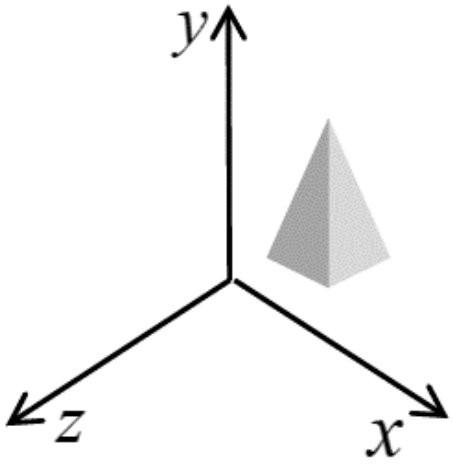
We have,
Point P = $(x_0, y_0, z_0) = (1, 2, 3)$
Shift Vector = (T_x, T_y, T_z)
Let us assume the new coordinates of P = (x_1, y_1, z_1)
Now we are going to add translation vector and given coordinates, then

$X_1 = x_0 + T_x = (1 + 2) = 3$
$Y_1 = y_0 + T_y = (2 + 3) = 5$
$Z_1 = z_0 + T_z = (3 + 4) = 7$
Thus, the new coordinates are = (3, 5, 7)

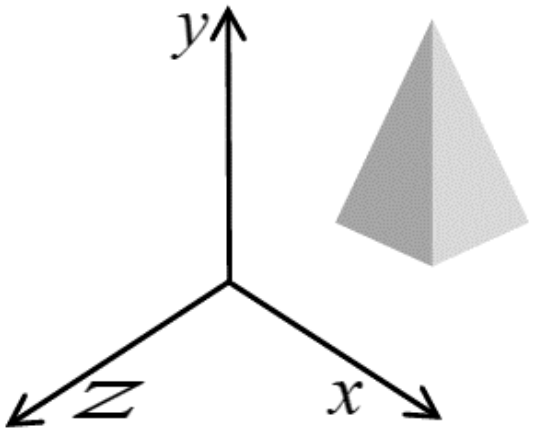
Scaling

- Scaling changes the size of an object and involves the scale factors.
- The 2D and 3D scaling are similar, but the key difference is that the 3D plane also includes the z-axis along with the x and y-axis.

- The scaling factor towards x, y and z axis is denoted by **'S_x'** **'S_y'** and **'S_z'** respectively**.**



The object before scaling



Object after scaling

The increment and decrement of an object is depends on two conditions. They are

i. If scaling factor (S_x, S_y, S_z) > 1, then the size of the object increased.
ii. If scaling factor (S_x, S_y, S_z) < 1, then the size of the object decreased.

Let us assume,
The initial coordinates of object = P (x, y, z)
Scaling factor for x-axis = S_x
Scaling factor for y-axis = S_y
Scaling factor for z-axis = S_z
The coordinates after Scaling = Q ($x_,$, $y_,$, $z_,$)

- **We can represent the 3D Scaling in the form of equation-**

$$X_1 = x. S_x$$
$$Y_1 = y. S_y$$
$$Z_1 = z. S_z$$

- **Matrix representation of 3D Scaling**

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

Therefore, P' = S.P

Example: A 3D object that have coordinates points P(1, 4, 4), Q(4, 4, 6), R(4, 1, 2), T(1, 1, 1) and the scaling parameters are 3 along with x-axis, 4 along with y-axis and 4 along with z-axis. Apply scaling to find the new coordinates of the object?

Solution: We have,
The initial coordinates of object = P (1, 4, 4), Q (4, 4, 6), R (4, 1, 2), S (1, 1, 1)
Scaling factor along with x-axis (S_x) = 3
Scaling factor along with y-axis (S_y) = 4
Scaling factor along with z-axis (S_z) = 4
Let the new coordinates after scaling = **(x', y', z')**

For coordinate P:

$X' = x . S_x = 1 \times 3 = 3$
$Y' = y . S_y = 4 \times 4 = 16$
$Z' = z . S_z = 4 \times 4 = 16$

The new coordinates = (3, 16, 16)

For coordinate Q:

$X' = x . S_x = 4 \times 3 = 12$
$Y' = y . S_y = 4 \times 4 = 16$
$Z' = z . S_z = 6 \times 4 = 24$

The new coordinates = (12, 16, 24)

For coordinate R:

$X' = x . S_x = 4 \times 3 = 12$
$Y' = y . S_y = 1 \times 4 = 4$
$Z' = z . S_z = 2 \times 4 = 8$

The new coordinates = (12, 4, 8)

For coordinate S:

$X' = x . S_x = 1 \times 3 = 3$
$Y' = y . S_y = 1 \times 4 = 4$
$Z' = z . S_z = 1 \times 4 = 4$

The new coordinates = (3, 4, 4)

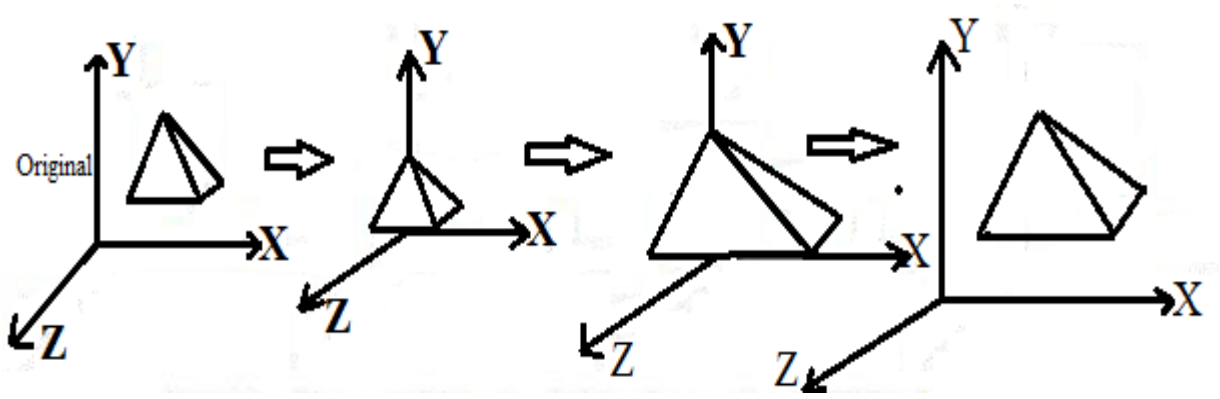
Thus, the new coordinates after scaling = P (3, 16, 16), Q (12, 16, 24), R (12, 4, 8), S (3, 4, 4).

Fixed Point Scaling in 3D

Let fixed point (x_f, y_f, z_f).

Do:

1. Translate fixed point to the origin
2. Scale object relative to the coordinate origin
3. Translate fixed point back to its original position



CM = $\mathbf{T}_{(x, y, z)}$. $\mathbf{S}_{(x, y, z)}$. $\mathbf{T}_{(-x, -y, -z)}$

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

Therefore,

P' = CM * P

Shearing

- Shearing transformations are used to modify object shapes.
- The basic difference between 2D and 3D Shearing is that the 3D plane also includes the z-axis.
- Let the point P (x, y,z) is obtained P' (x', y', z') after applying shear with shearing factor for x, y and z axis are **'Sh_x,' 'Sh_y,' and 'Sh_z.' respectively.**

Basically, Shearing in 3D Geometry are categorized into 3 different types

a) X- axis Shearing

- In 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 + Shy * x
z'= z + shz * x

3D matrix representation,

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

b) Y -axis shearing

This transformation alters Y 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 + Shx * y
z'= z +Shz * y

3D Matrix Representation,

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

a) <u>Z -axis Shearing</u>

This transformation alters x and y coordinate values by amount that is proportional to the z value while leaving z co-ordinate unchanged.

x' = x + Shx * z
y' = y + Shy * z
z' = z

3D Matrix Representation,

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

Share Parameters Shx and Shy can be assigned any real values

Example.

Given a 3D triangle with points (0, 0, 0), (1, 1, 2) and (1, 1, 3). Apply shear parameter 2 on X axis, 2 on Y axis and 3 on Z axis and find out the new coordinates of the object.

Solution

Given that,

Old corner coordinates of the triangle = A (0, 0, 0), B(1, 1, 2), C(1, 1, 3)
Shearing parameter towards X direction (Sh_x) = 2
Shearing parameter towards Y direction (Sh_y) = 2
Shearing parameter towards Y direction (Sh_z) = 3

Shearing in X Axis

For Coordinates A (0, 0, 0)

Let the new coordinates of corner A after shearing = (X', Y', Z').

Applying the shearing equations, we have-

$X' = X = 0$

$Y' = Y + Sh_y . X = 0 + 2 \times 0 = 0$

$Z' = Z + Sh_z . X = 0 + 3 \times 0 = 0$

Thus, New coordinates of corner A after shearing = (0, 0, 0).

For Coordinates B(1, 1, 2)

Let the new coordinates of corner B after shearing = (X', Y', Z').

Applying the shearing equations, we have-

$X' = X = 1$

$Y' = Y + Sh_y \times X = 1 + 2 \times 1 = 3$

$Z' = Z + Sh_z \times X = 2 + 3 \times 1 = 5$

Thus, New coordinates of corner B after shearing = (1, 3, 5).

For Coordinates C (1, 1, 3)

Let the new coordinates of corner C after shearing = (X', Y', Z').

Applying the shearing equations, we have-

$X' = X = 1$

$Y' = Y + Sh_y \times X = 1 + 2 \times 1 = 3$

$Z' = Z + Sh_z \times X = 3 + 3 \times 1 = 6$

Thus, New coordinates of corner C after shearing = (1, 3, 6).

Thus, New coordinates of the triangle after shearing in X axis = A (0, 0, 0), B(1, 3, 5), C(1, 3, 6).

Shearing in Y Axis

For Coordinates A (0, 0, 0)

Let the new coordinates of corner A after shearing = (X', Y', Z').

Applying the shearing equations, we have-

$X' = X + Sh_x \times Y = 0 + 2 \times 0 = 0$

$Y' = Y = 0$

$Z' = Z + Sh_z \times Y = 0 + 3 \times 0 = 0$

Thus, New coordinates of corner A after shearing = (0, 0, 0).

For Coordinates B (1, 1, 2)

Let the new coordinates of corner B after shearing = (X', Y', Z').

Applying the shearing equations, we have-

$X' = X + Sh_x \times Y = 1 + 2 \times 1 = 3$

$Y' = Y = 1$

$Z' = Z + Sh_z \times Y = 2 + 3 \times 1 = 5$

Thus, New coordinates of corner B after shearing = (3, 1, 5).

For Coordinates C (1, 1, 3)

Let the new coordinates of corner C after shearing = (X', Y', Z').

Applying the shearing equations, we have-

$X' = X + Sh_x \times Y = 1 + 2 \times 1 = 3$

$Y' = Y = 1$

$Z' = Z + Sh_z \times Y = 3 + 3 \times 1 = 6$

Thus, New coordinates of corner C after shearing = (3, 1, and 6).

Thus, New coordinates of the triangle after shearing in Y axis = A (0, 0, 0), B (3, 1, 5), C (3, 1, 6).

Shearing in Z Axis

For Coordinates A(0, 0, 0)

Let the new coordinates of corner A after shearing = (X', Y', Z').

Applying the shearing equations, we have-

$X' = X + Sh_x \times Z = 0 + 2 \times 0 = 0$

$Y' = Y + Sh_y \times Z = 0 + 2 \times 0 = 0$

$Z' = Z = 0$

Thus, New coordinates of corner A after shearing = (0, 0, 0).

For Coordinates B(1, 1, 2)

Let the new coordinates of corner B after shearing = (X', Y', Z').

Applying the shearing equations, we have,

$X' = X + Sh_x \times Z = 1 + 2 \times 2 = 5$

$Y' = Y + Sh_y \times Z = 1 + 2 \times 2 = 5$

$Z' = Z = 2$

Thus, New coordinates of corner B after shearing = (5, 5, 2).

For Coordinates C(1, 1, 3)

Let the new coordinates of corner C after shearing = (X', Y', Z').

Applying the shearing equations, we have-

$X' = X + Sh_x \times Z = 1 + 2 \times 3 = 7$

$Y' = Y + Sh_y \times Z = 1 + 2 \times 3 = 7$

$Z' = Z = 3$

Thus, New coordinates of corner C after shearing = (7, 7, 3).

Thus, New coordinates of the triangle after shearing in Z axis = A (0, 0, 0), B (5, 5, 2), C (7, 7, 3).

Reflection

- A Three-dimensional can be performed relative to a selected reflection axes or with respect to selected reflected plane.
- Three-dimensional reflection matrices are set up similar to those for two dimensional.
- Reflection relative to given axis are equivalent to 180^0 rotation about that axis.
- The reflection planes are either XY, XZ or YZ (i.e. 3 types)
- The reflected object is always formed on the other side of mirror.

Consider a point object O has to be reflected in a 3D plane.

Let, Initial coordinates of the object O = (X, Y, Z) and new coordinates of the reflected object O after reflection = (X', Y', Z')

Reflection Relative to XY Plane (i.e. Reflection along Z axis)

- X and y value same, z value flip

OR, this transformation changes the sign of the z coordinates, leaving the x and y coordinate values unchanged i.e.

$X' = X$

$Y' = Y$

Z’ = -Z

In Matrix form, the above reflection equations may be represented as

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

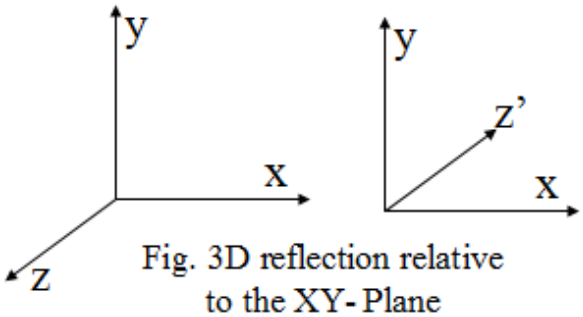


Fig. 3D reflection relative to the XY- Plane

Reflection Relative to YZ Plane (i.e. Reflection along X axis)

Y and Z value unchanged, X value flip

OR, this transformation changes the sign of the x coordinates, leaving the y and z coordinate values unchanged.

This reflection is achieved by using the following reflection equations

X’ = -X
Y’ = Y
Z’ = Z

In Matrix form, the above reflection equations may be represented as

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

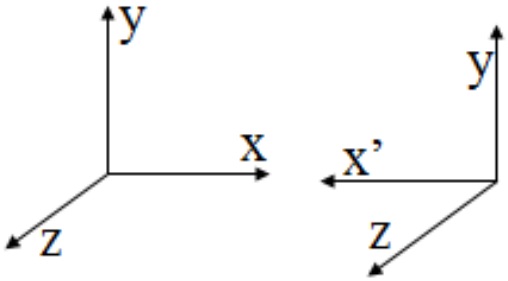


Fig. 3D reflection relative to the YZ- Plane

Reflection Relative to XZ Plane (i.e. Reflection along Y axis)

- X and Z values are unchanged while Y value flip.

OR, this transformation changes the sign of the y coordinates, leaving the x and z coordinate values unchanged.

This reflection is achieved by using the following reflection equations

X' = X
Y' = -Y
Z' = Z

In Matrix form, the above reflection equations may be represented as

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

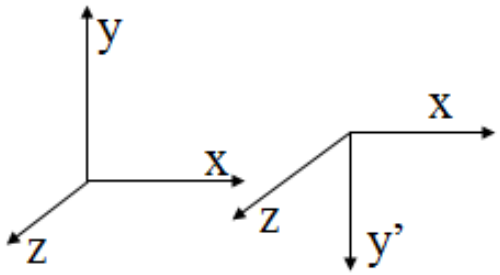


Fig. 3D reflection relative to the ZX- Plane

Example.

Given a 3D triangle with coordinate points A (3, 4, 1), B (6, 4, 2), C (5, 6, 3). Apply the reflection on the XY plane and find out the new coordinates of the object.

Solution

Given that,

Old corner coordinates of the triangle = A (3, 4, 1), B (6, 4, 2), C (5, 6, 3)
Reflection has to be taken on the XY plane

For Coordinates A (3, 4, 1)

Let the new coordinates of corner A after reflection = (X', Y', Z').

Applying the reflection equations, we have-

X' = X = 3
Y' = Y = 4
Z' = -Z= -1

Thus, New coordinates of corner A after reflection = (3, 4, -1).

For Coordinates B (6, 4, 2)

Let the new coordinates of corner B after reflection = (X', Y', Z')

Applying the reflection equations, we have-

X' = X = 6
Y' = Y = 4
Z' = -Z= -2

Thus, New coordinates of corner B after reflection = (6, 4, -2).

For Coordinates C (5, 6, 3)

Let the new coordinates of corner C after reflection = (X', Y', Z').

Applying the reflection equations, we have-

X' = X = 5
Y' = Y = 6
Z' = -Z= -3

Thus, New coordinates of corner C after reflection = (5, 6, -3).

Thus, New coordinates of the triangle after reflection = A (3, 4, -1), B (6, 4, -2), C(5, 6, -3).

Example

Given a 3D triangle with coordinate points A (3, 4, 1), B (6, 4, 2), C (5, 6, 3). Apply the reflection on the XZ plane and find out the new coordinates of the object.

Solution

Given that,

Old corner coordinates of the triangle = A (3, 4, 1), B (6, 4, 2), C(5, 6, 3)
Reflection has to be taken on the XZ plane

For Coordinates A (3, 4, 1)

Let the new coordinates of corner A after reflection = (X', Y', Z').

Applying the reflection equations, we have-

X' = X = 3
Y' = -Y = -4
Z' = Z= 1

Thus, New coordinates of corner A after reflection = (3, -4, 1).

For Coordinates B (6, 4, 2)

Let the new coordinates of corner B after reflection = (X', Y', Z').

Applying the reflection equations, we have-

X' = X = 6
Y' = -Y = -4
Z' = Z= 2

Thus, New coordinates of corner B after reflection = (6, -4, 2).

For Coordinates C (5, 6, 3)

Let the new coordinates of corner C after reflection = (X', Y', Z').

Applying the reflection equations, we have-

X' = X = 5
Y' = -Y = -6
Z' = Z= 3

Thus, New coordinates of corner C after reflection = (5, -6, 3).

Thus, New coordinates of the triangle after reflection = A (3, -4, 1), B (6, -4, 2), C (5, -6, 3).

Reflection about any axis parallel to one of the coordinate axes

Do:

1. Translate object so that reflection axis coincides with the parallel coordinate axis.
2. Perform specified reflection about that axis
3. Translate object back to its original Position

$$CM = T . R . T^{-1}$$

$$P' = CM . P$$

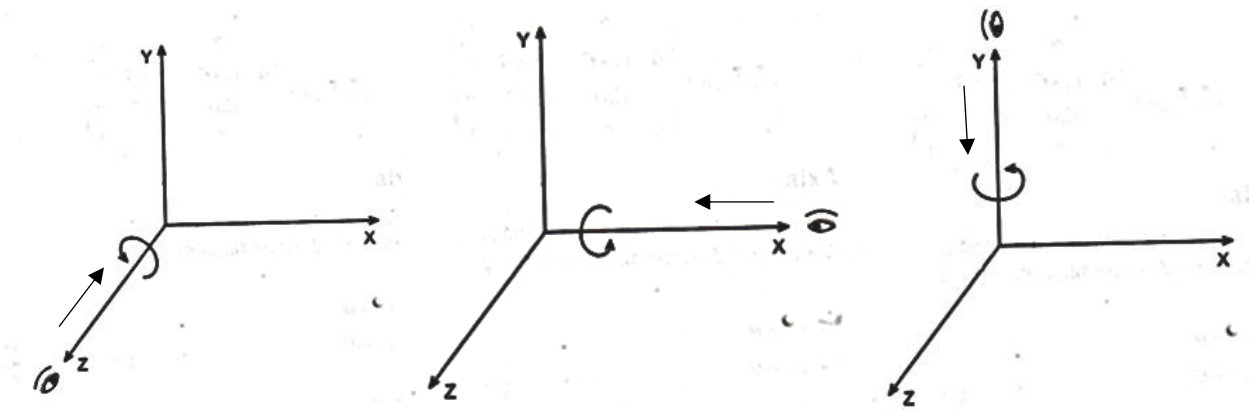
Reflection about any arbitrary plane in 3D Space

- Reflection about any arbitrary plane in 3D is similar to the reflection about any arbitrary line
- But the difference is that, we have to characterize the rotation by any normal vector 'N' in that plane.

Step 1: Translate the reflection plane to the origin of the coordinate system
Step 2: Perform appropriate rotations to make the normal vector of the reflection plane at the origin until it coincides with the z-axis.
Step 3: After that reflect the object through the z= 0 coordinate plane.
Step 4: Perform the inverse of the rotation transformation
Step 5: Perform the inverse of the translation

Rotation in 3D

- In CG, 3D rotation is a process of rotating an object to an angle in a three dimensional plane.
- Rotation is moving of an object about an angle.
- Movement can be anticlockwise or clockwise.
- 3D rotation is complex as compared to the 2D rotation. For 2D we describe the angle of rotation, but for a 3D angle of rotation and axis of rotation are required. The axis can be either x or y or z.
- To determine a rotation transformation for an object in 3D space, following information is required:
  - ✓ Angle of rotation.
  - ✓ Pivot point
  - ✓ Direction of rotation
  - ✓ Axis of rotation



Figures: Coordinate axis Rotations

Let's consider a origin as the center of rotation and a point P(x, y, z) is rotated through an angle about any one of the axes to get the transformed point P'(x', y', z'), then the equation for each rotation can be obtained as follows.

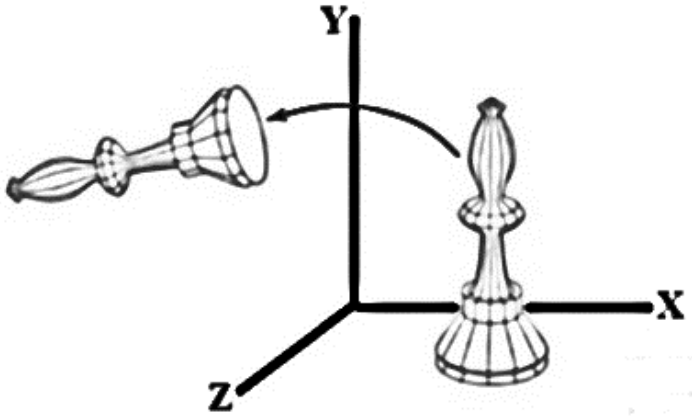
3D Z-axis Rotation

- This rotation is achieved by using the following rotation equations
- Two dimension rotation equations can be easily convert into 3D z- axis rotation equation.
- Rotation about z axis we leave z coordinate unchanged.

X' = x cos θ – y sin θ
Y' = x sin θ + y cos θ
Z' = z

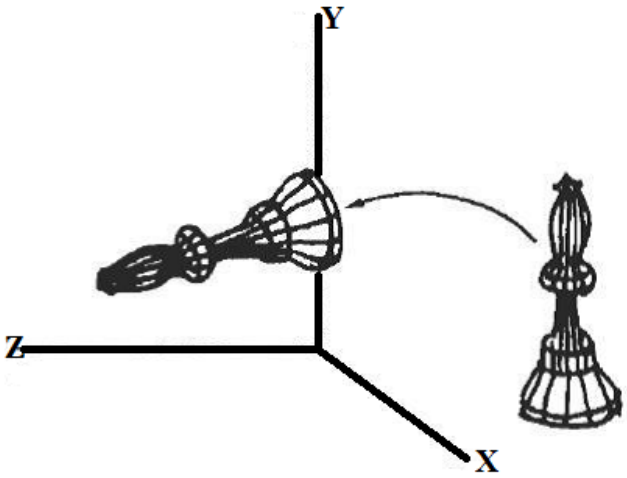
In matrix representation,

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



Therefore, **P' = $R_{z(\theta)}$. P**

3D X-axis Rotation



This rotation is achieved by using the following rotation equations:

$X' = x$

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

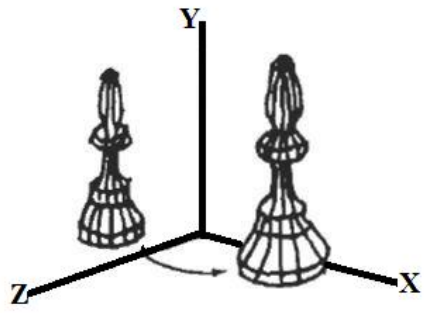
$Z' = y \sin\theta + z \cos\theta$

In matrix Representation,

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

Therefore**, P' = $R_{x(\theta)}$. P**

3D Y-axis Rotation



Equations for this rotation are as:

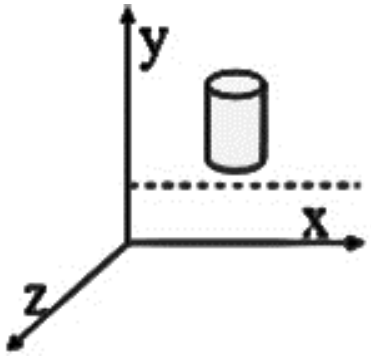
$X' = z \sin\theta + x \cos\theta$
$Y' = y$
$Z' = z \cos\theta - x \sin\theta$

Matrix representation of these equation,

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

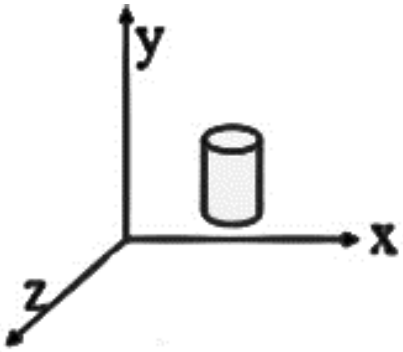
Therefore, **P' = $R_{y(\theta)}$. P**

Rotation about an axis parallel to one of the coordinate axes

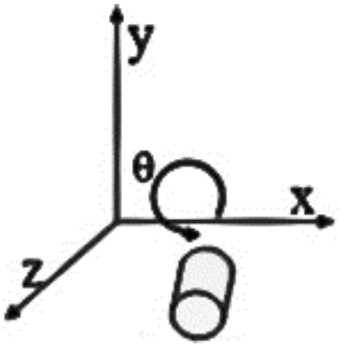


DO:

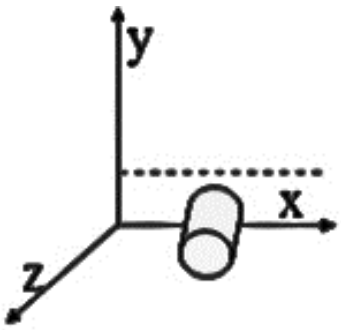
1. Translate object so that rotation axis coincides with the parallel coordinate axis.



2. Perform specified rotation about that axis



3. Translate object back to its original Position

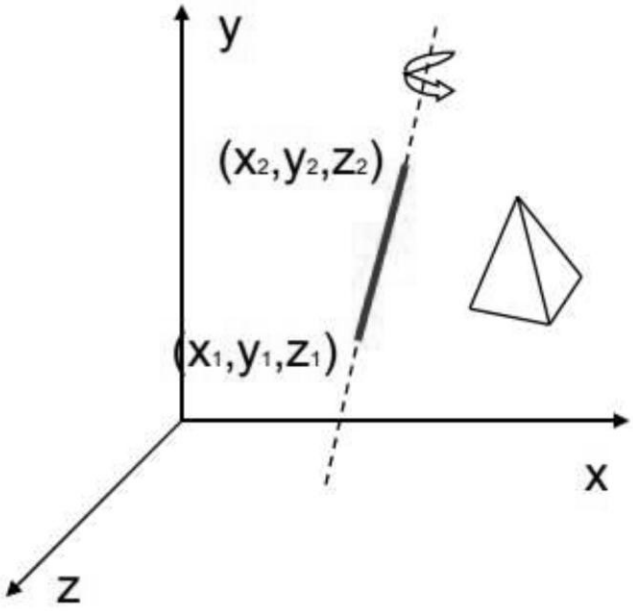


Therefore, CM= T R T^{-1}

P'= CM.P

Rotation about an arbitrary axis

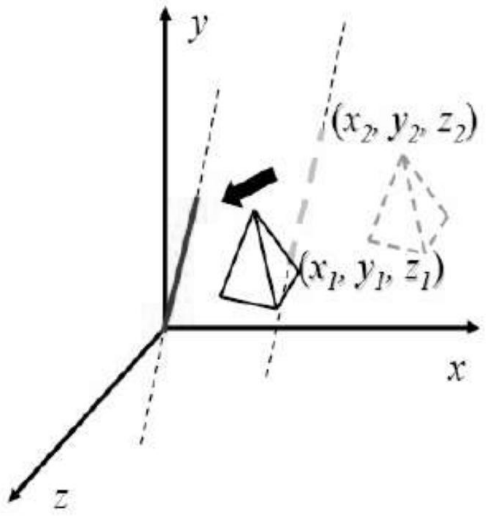
- A rotation matrix for any axis that does not coincide with a coordinate axis can be set up as a composite Transformation involving combinations of translation and the coordinate-axes rotations.



DO:

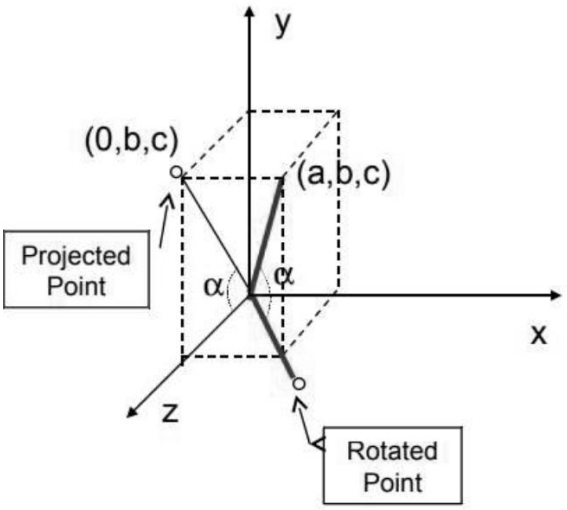
Step-1: Translate the arbitrary axis so that it passes through origin.
Step-2: Align the arbitrary axis on any major co-ordinate axis (z-axis)
Step-3: Rotate the object about yz-plane or z-axis
Step-4: Perform inverse rotation about y-axis & then x-axis
Step-5: Perform inverse translation

Step-1: Translate



$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step-2: Rotate about X-axis by an angle α

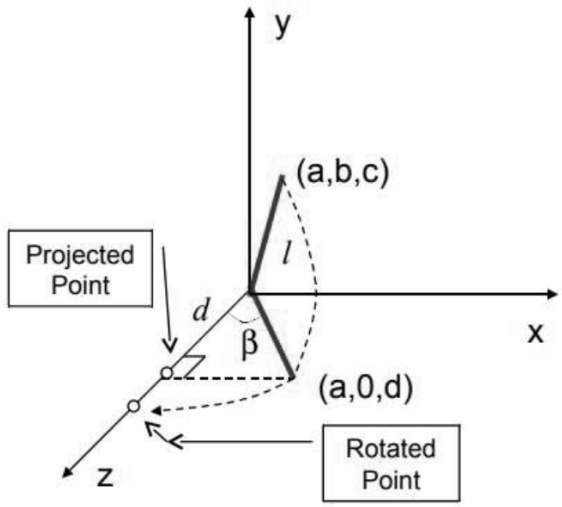


$$\sin\alpha = \frac{b}{\sqrt{b^2 + c^2}} = \frac{b}{d}$$

$$\cos\alpha = \frac{c}{\sqrt{b^2 + c^2}} = \frac{c}{d}$$

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & -b/d & 0 \\ 0 & b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step-3: Rotate about Y-axis by an angle β



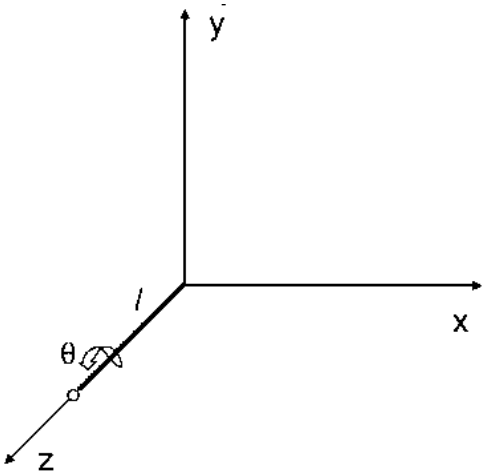
$$\sin\beta = \frac{a}{l}, \quad \cos\beta = \frac{d}{l}$$

$$l^2 = a^2 + b^2 + c^2 = a^2 + d^2$$

$$d = \sqrt{b^2 + c^2}$$

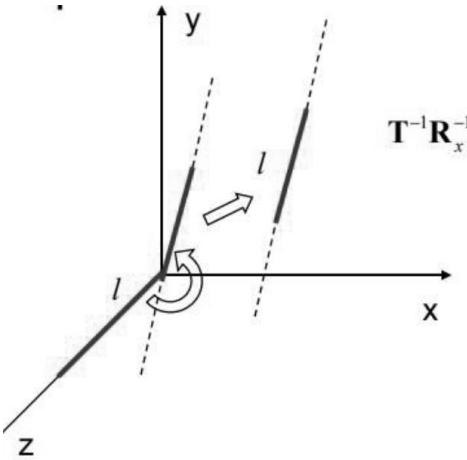
$$\mathbf{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} d/l & 0 & -a/l & 0 \\ 0 & 1 & 0 & 0 \\ a/l & 0 & d/l & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step-4: Rotate about Z-axis by an angle θ



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

Step-5 Apply the inverse transformation



$$\mathbf{T}^{-1}\mathbf{R}_x^{-1}(\alpha)\mathbf{R}_y^{-1}(\beta) = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hence,

Composite matrix (CM) = $\mathbf{T\ R_x\ R_y\ R_z\ R_y^{-1}\ R_x^{-1}\ T^{-1}}$

Numerical: Find the new co-ordinates of a unit cube 90 degree rotated about an axis defined by its end points A(2,1,0) and B(3,3,1).

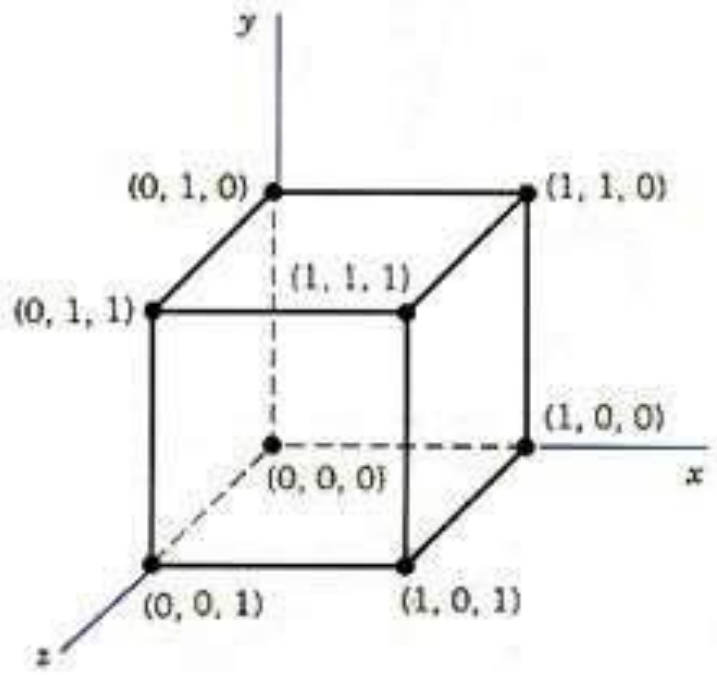
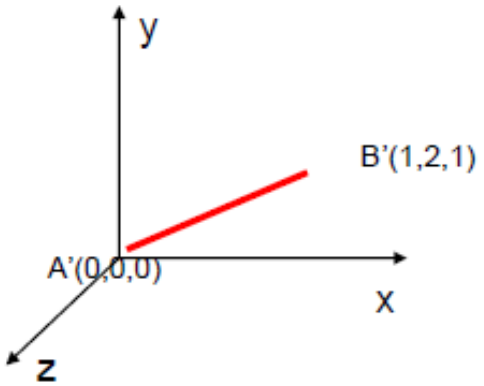


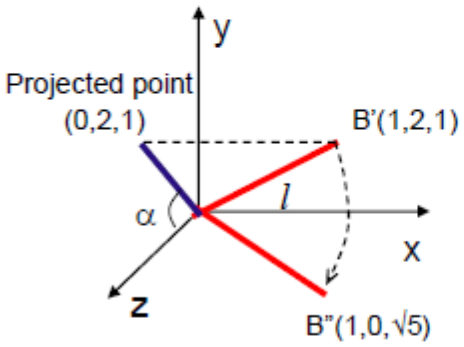
Figure: A Unit Cube

Step -1 Translate Point A to the Origin



$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step-2: Rotate axix A'B' about the X axis by and angle α, until it lies on the XZ plane.



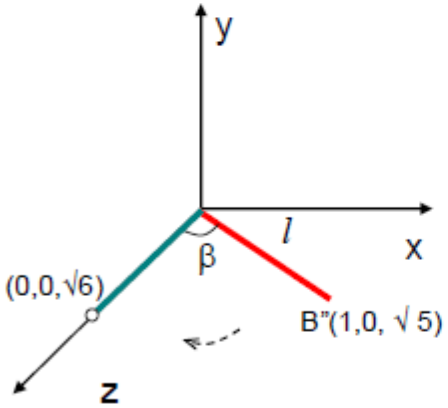
$$\sin\alpha = \frac{2}{\sqrt{2^2+1^2}} = \frac{2}{\sqrt{5}} = \frac{2\sqrt{5}}{5}$$

$$\cos\alpha = \frac{1}{\sqrt{5}} = \frac{\sqrt{5}}{5}$$

$$l = \sqrt{1^2+2^2+1^2} = \sqrt{6}$$

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} & 0 \\ 0 & \frac{2\sqrt{5}}{5} & \frac{\sqrt{5}}{5} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step-3: Rotate axis A'B'' about the Y axis by and angle β until it coincides with the Z axis.



$$\sin\beta = \frac{1}{\sqrt{6}} = \frac{\sqrt{6}}{6}$$

$$\cos\beta = \frac{\sqrt{5}}{\sqrt{6}} = \frac{\sqrt{30}}{6}$$

$$\mathbf{R}_y(\beta) = \begin{bmatrix} \frac{\sqrt{30}}{6} & 0 & -\frac{\sqrt{6}}{6} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\sqrt{6}}{6} & 0 & \frac{\sqrt{30}}{6} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step-4: Rotate the Cube 90 degree about Z axis.

$$\mathbf{R}_z(90°) = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, the Composite Matrix for the Rotation about the arbitrary axis AB becomes,

$$\mathbf{R}(\theta) = \mathbf{T}^{-1}\mathbf{R}_x^{-1}(\alpha)\mathbf{R}_y^{-1}(\beta)\mathbf{R}_z(90°)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha)\mathbf{T}$$

$$\mathbf{R}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{5}}{5} & \frac{2\sqrt{5}}{5} & 0 \\ 0 & -\frac{2\sqrt{5}}{5} & \frac{\sqrt{5}}{5} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{30}}{6} & 0 & \frac{\sqrt{6}}{6} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{\sqrt{6}}{6} & 0 & \frac{\sqrt{30}}{6} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\sqrt{30}}{6} & 0 & -\frac{\sqrt{6}}{6} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\sqrt{6}}{6} & 0 & \frac{\sqrt{30}}{6} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} & 0 \\ 0 & \frac{2\sqrt{5}}{5} & \frac{\sqrt{5}}{5} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.166 & -0.075 & 0.983 & 1.742 \\ 0.742 & 0.667 & 0.075 & -1.151 \\ -0.650 & 0.741 & 0.167 & 0.560 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hence,

Multiplying R (θ) by the point matrix of the original cube we get P'

$$[P'] = R(\theta) \cdot [P]$$

$$[P'] = \begin{bmatrix} 0.166 & -0.075 & 0.983 & 1.742 \\ 0.742 & 0.667 & 0.075 & -1.151 \\ -0.650 & 0.741 & 0.167 & 0.560 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2.650 & 1.667 & 1.834 & 2.816 & 2.725 & 1.742 & 1.909 & 2.891 \\ -0.558 & -0.484 & 0.258 & 0.184 & -1.225 & -1.151 & -0.409 & -0.483 \\ 1.467 & 1.301 & 0.650 & 0.817 & 0.726 & 0.560 & -0.091 & 0.076 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Rotation about any arbitrary plane in 3D Space

- The rotation about any arbitrary plane perform same operation as the rotation about any arbitrary line, the only difference is that we have to characterize the rotation by any normal vector 'N' in that plane.

DO:

Step 1: Translate the rotation plane to the origin of the coordinate system
Step 2: Perform appropriate rotations to make the normal vector of the rotation plane at the origin until it coincides with the z-axis.
Step 3: After that rotate the object through the z= 0 coordinate plane.
Step 4: Perform the inverse of the rotation transformation
Step 5: Perform the inverse of the translation

3D Representation

- Graphics scenes can contain many different kinds of objects and material surfaces such as Trees, flowers, clouds, rocks, water, bricks, wood paneling, rubber, paper, steel, glass, plastic and cloth….etc.
- No single method can be used to describe objects that will include all features of those different materials.

- To produce realistic display of scenes, we need to use representations that accurately model object characteristics.

Some of 3D object Representations methods are:

1. Polygon and Quadric surfaces: For simple Euclidean objects
2. Spline surfaces and construction: For curved surfaces
3. Procedural methods: Eg. Fractals, Particle systems
4. Physically based modeling methods
5. Octree Encoding
6. Solid object method
7. Volume rendering, etc.

- Simple Euclidean objects like polyhedrons and ellipsoids can be represented by polygon and quadric surfaces.
- Spline surface are useful for designing aircraft wings, gears and other engineering objects.
- Procedural methods and particle systems allow us to give accurate representation of clouds, clumps of grass, and other natural objects.
- Octree encodings are used to represent internal features of objects such as medical CT images.

3D Object Representation:

Methods for solid objects are commonly divided into two broad categories:

1. Boundary representations (B-REP)

- It describes a three dimensional object as a set of surfaces that separate the object interior from the environment. Examples are polygon facets and spline patches.
- Also called B-rep or BREP,

Boundary representation of models are composed of two parts:

i. topology and
ii. Geometry (surfaces, curves and points).

- The main topological items are: ***faces, edges*** **and** ***vertices***. A face is a bounded portion of a surface; an edge is a bounded piece of a curve and a vertex lies at a point.

- Other elements are the ***shell*** (a set of connected faces), the ***loop*** (a circuit of edges bounding a face) and ***loop-edge links*** (also known as *winged edge links* or *half-edges*) which are used to create the edge circuits. The edges are like the edges of a table, bounding a surface portion.

2. Space-partitioning representation:

- It describes the interior properties, by partitioning the spatial region containing an object into a set of small, non-overlapping, contiguous solids (usually cubes). E.g.: Octree Representation

- Objects may also associate with other properties such as mass, volume, so as to determine their response to stress and temperature etc.

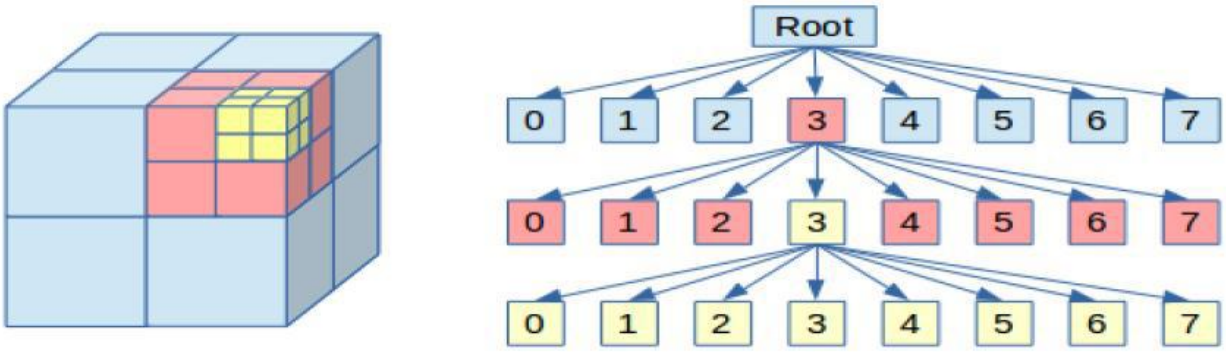


Figure: Octree Representation (Space partitioning method

3D Object Representation: Polygon Surfaces

- The most commonly used representation for a 3D graphics object.
- For other 3D objection representations, they are often converted into polygon surfaces before rendering.

- Since all surfaces can be described with linear equations. Set of polygon surfaces are used to represent object boundary that enclose the object interior in 3D graphics

Three ways to represent polygon surfaces

1. Polygon Tables

2. Plane Equations
3. Polygon Meshes

Polygon Tables:

- The polygon surface is specified with a set of vertex coordinates and associated attribute parameters.
- For each polygon input, the data are placed into tables that are to be used in the subsequent processing.
- **Polygon data tables** (This is the specification of polygon surfaces using vertex coordinates and other attributes) can be organized into two groups:

    i. **Geometric tables**
    ii. **Attribute tables** (Contain attribute information for an object such as parameters specifying the degree of transparency of the object and its surface reflectivity and texture characteristics.)

I) Geometric tables

- Geometric data tables contains vertex coordinate and the other parameter which specify geometry of polygon.
- A convenient organization for storing geometric data is to create three lists:
    - A **vertex table,**
    - **An edge table,**
    - **A polygon surface table.**

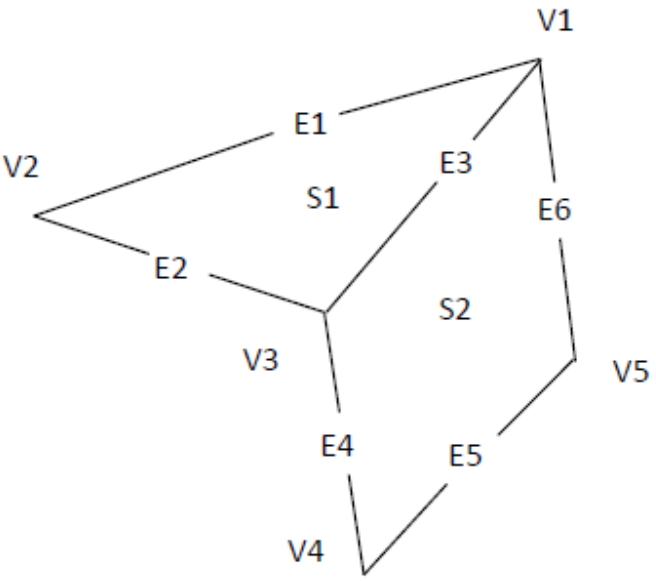
Some consistency checks of the geometric data table:

- ✓ Every vertex is listed as an endpoint for at least 2 edges
- ✓ Every edge is part of at least one polygon
- ✓ Every polygon is closed
- ✓ Each polygon has at least one shared edge
- ✓ If the edge table contains pointers to polygon every edge referenced by a polygon pointer has a reciprocal back to the polygon

Rules:

- Coordinate values for each vertex in the object are stored in the vertex table.
- The edge table contains pointers back into the vertex table to identify the vertices for each polygon edge.

- The polygon table contains pointers back into the edge table to identify the edges for each polygon



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:	V2, V3
E3:	V3, V1
E4:	V3, V4
E5:	V4, V5
E6:	V5, V1

Polygon Surface Table	
S1:	E1, E2, E3
S2:	E3, E4, E5, E6

Figure: Geometric Data Table Representation

II) **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 the polygon attribute according to their pointer information.

Extra information can be added to the data tables for faster information extraction. For instance, edge table can be expanded to include forward points into the

polygon table so that common edges between polygons can be identified more rapidly.

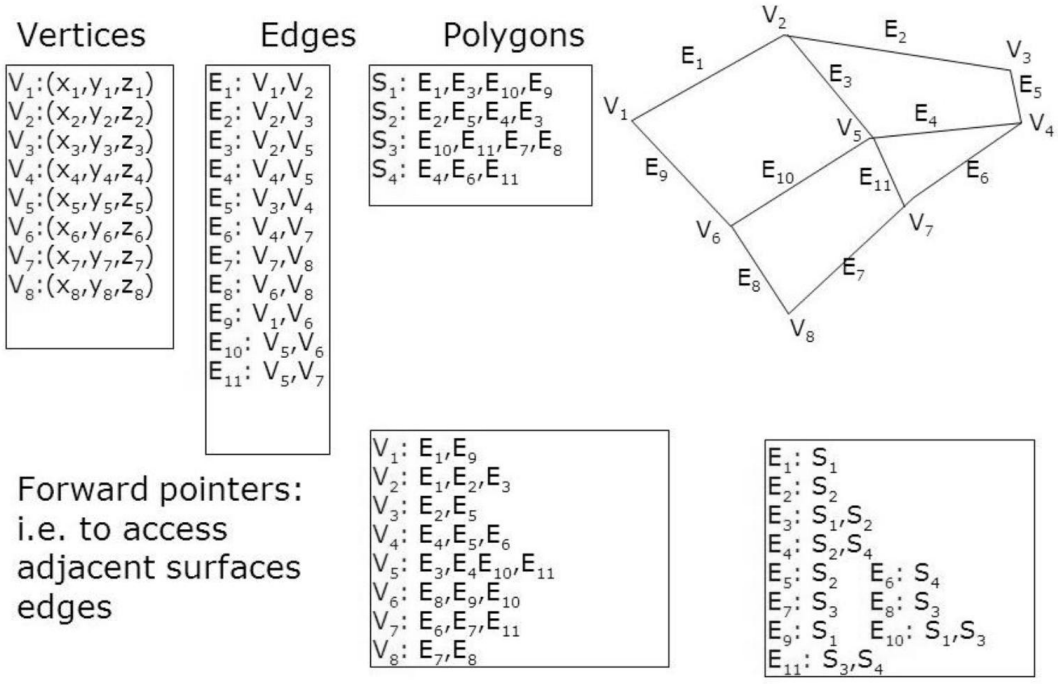
E1: V1, V2, S1
E2: V2, V3, S1
E3: V3, V1, S1, S2
E4: V3, V4, S2
E5: V4, V5, S2
E6: V5, V1, S2

This is useful for the rendering procedure that must vary surface shading smoothly across the edges from one polygon to the next. Similarly, the vertex table can be expanded so that vertices are cross-referenced to corresponding edges.

Additional geometric information that is stored in the data tables includes the slope for each edge and the coordinate extends for each polygon. As vertices are input, we can calculate edge slopes and we can scan the coordinate values to identify the minimum and maximum x, y and z values for individual polygons.

The more information included in the data tables will be easier to check for errors.

Example-2



Plane Equations:

- It is used to determine the spatial orientation of the individual surface component of the object.

The equation for a plane surface can be expressed in the form
$Ax + By + Cz + D = 0$

Where (x, y, z) is any point on the plane and the coefficient A, B, C, D are constants.

Let $(x_1\ y_1, z_1,)$, (x_2, y_2, z_2), and (x_3, y_3, z_3) be three successive polygon vertices of the polygon.

$Ax_1 + By_1 + Cz_1 + D = 0,$
$Ax_2 + By_2 + Cz_2 + D = 0,$
$Ax_3 + By_3 + Cz_3 + D = 0$

The solution of these three plane equation for three non collinear points can be obtained in determinant form, using **Cramer's rule** as:

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

Expanding the determinants, we can write the calculations for the plane coefficients in the form:

$$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$$
$$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$$
$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$
$$D = -x_1(y_2z_3 - y_3z_2) - x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1)$$

This values of A, B, C, D are then store in polygon data structure with other polygon data.

Orientation of plane is described with normal vector to the plane.

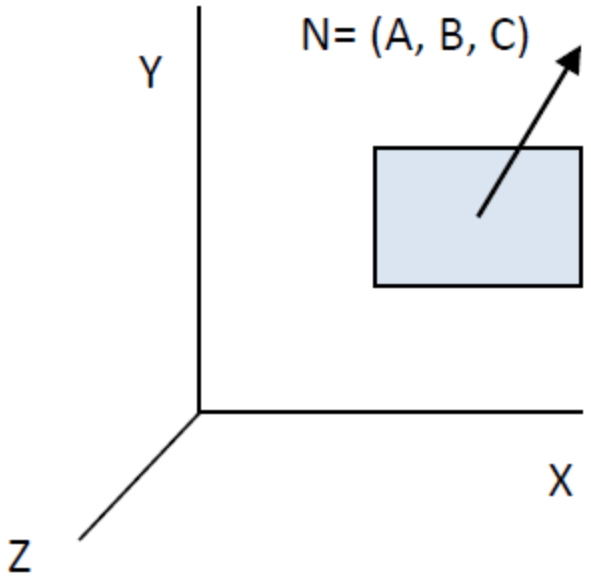


Figure: the vector N normal to the surface

Here N= (A, B, C) where A, B, C are the plane coefficient.

When we are dealing with the polygon surfaces that enclose object interior we define the side of the faces towards object interior is as inside face and outward side as outside face.

We can calculate normal vector N for any particular surface by cross product of two vectors in counter clockwise direction in right handed system then.

$$N = (v2 - v1)(v3 - v1)$$

Now N gives values A, B, C for that plane and D can be obtained by putting these values in plane equation for one of the vertices and solving for D.

Using plane equation in vector form we can obtain D as

$$N \cdot P = -D$$

Plane equation is also used to find position of any point (x, y, z) compare to plane surface as follows

If $Ax + By + Cz + D \neq 0$ the point (x, y, z) is not on that plane.
If $Ax + By + Cz + D < 0$ the point (x, y, z) is inside the surface.
If $Ax + By + Cz + D > 0$ the point (x, y, z) is outside the surface.

These equation are valid for right handed system provides plane parameter A, B, C, and D were calculated using vertices selected in a counter clockwise order when viewing the surface in an outside to inside direction.

Polygon Mesh

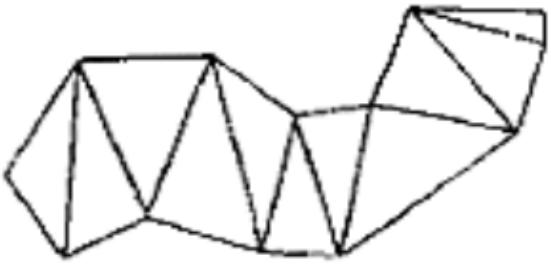


Figure: A triangle strip formed with 11 triangle mesh connecting 13 vertices

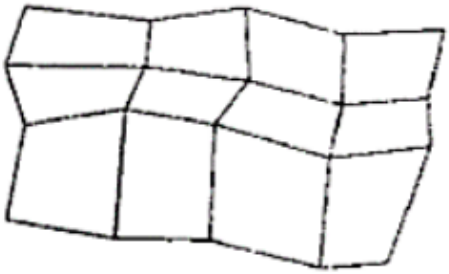


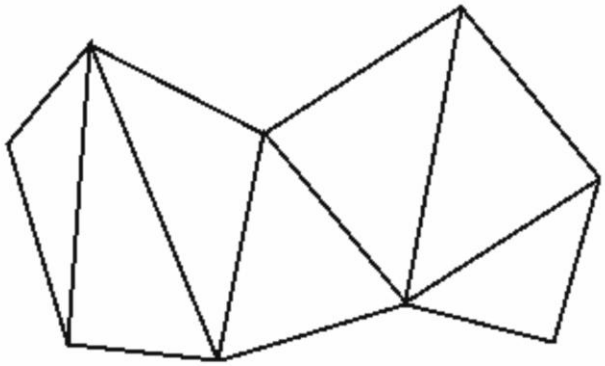
Figure: A quadrilateral mesh containing 12 quadrilaterals constructed from a 5 by 4 input vertex array.

- Polygon mesh is the **collection of vertices, edges, and faces** that make up a 3D object.
- A polygon mesh defines the shape and contour of every 3D character & object, whether it be used for 3D animated film, advertising, or video games
- An edge can be shared by two or more polygons and vertex is shared by at least two edges.
- Each vertex in the polygon mesh stores x, y, and z coordinate information.
- Then each face of that polygon contains surface information which is used by the rendering engine to calculate lightning and shadows (among other things).
- Polygon meshes can be used to model almost any object.
- It's also possible to generate polygon meshes in **real time** making them both powerful and dynamic. As such, they are used extensively throughout the world of computer graphics

Figure: Polygon Mesh Example (toy model) [Image Source: Internet]

Types:

1. **Triangular Mesh**
   It produces n – 2 connected triangles, given the coordinates for n vertices.



2. **Quadrilateral Mesh**
   Another similar functions the quadrilateral mesh that generates a mesh of (n-1) (m-1) quadrilaterals, given the coordinates for an n by m array of vertices

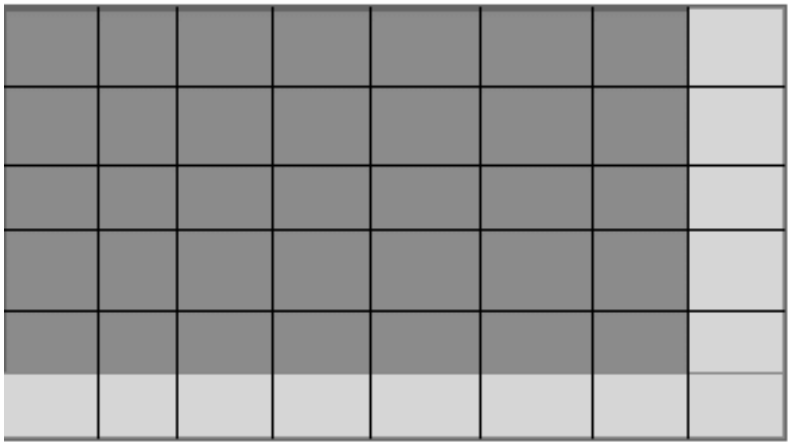


Figure: 6 by 8 vertices array, 35 element quadrilateral mesh

Polygon mesh is represented in following ways

- ✓ Explicit representation
- ✓ Pointer to vertex list
- ✓ Pointer to edge list

❖ A single plane surface can be specified with a function such as fill Area. But when object surfaces are to be tiled, it is more convenient to specify the surface facets with a mesh function.

❖ One type of polygon mesh is triangle strip. This function produce n-2 connected triangles.

❖ Another similar function is the quadrilateral mesh, which generates a mesh of (n-1) by (m-1) quadrilaterals, given the co-ordinates for an n x m array of vertices as shown in above figure.

Explicit Representation

❖ In explicit representation each polygon stores all the vertices in order in the memory as,

$$P = (((x1, y1, z1), (x2, y2, z2)), \dots , ((xm, ym, zm), (xn, yn, zn)))$$

❖ It process fast but requires more memory for storing.

Pointer to Vertex list

❖ In this method each vertex stores in vertex list and then polygon contains pointer to the required vertex.

$$V = ((x1, y1, z1), (x2, y2, z2), \dots , (xn, yn, zn))$$

And now polygon of vertices 3, 4, 5 is represented as $P = ((v3, v4), (v4, v5), (v5, v3))$.

- It is considerably space saving but common edges is difficult to find.

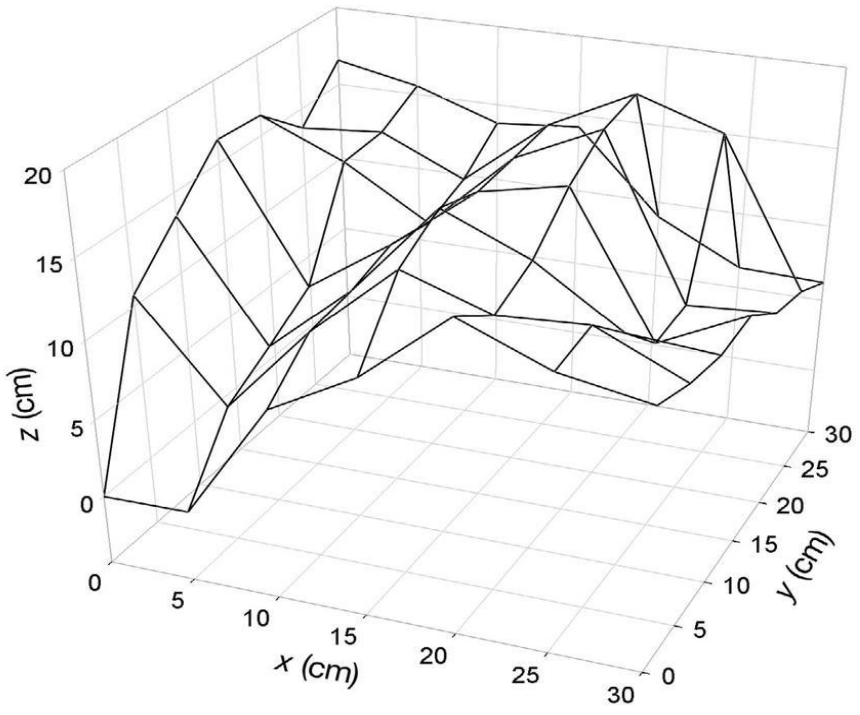
Pointer to Edge List

- In this polygon have pointers to the edge list and edge list have pointer to vertex list for each edge two vertex pointer is required which points in vertex list.
  $V = ((x1, y1, z1), (x2, y2, z2), \dots, (xn, yn, zn))$
  $E = ((v1, v2), (v2, v3), \dots, (vn, vm))$
  $P = (E1, E2, En)$
- This approach is more memory efficient and easy to find common edges.

Wireframe Model:

- A wireframe is a three-dimensional model that only includes vertices and lines. It does not contain surfaces, textures, or lighting like a 3D mesh.
- Instead, a wireframe model is a 3D image comprised of only "wires" that represent three-dimensional shapes.
- A **wire-frame model** is a visual presentation of a 3-dimensional (3D) or physical object used in 3D computer graphics.

- 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. For example, a 3D graphic designer can create a model from scratch by simply defining points (vertices) and connecting them with lines (paths).
- Once the shape is created, surfaces or textures can be added to make the model appear more realistic.



Blobby Object

- Non rigid object like cloth, rubber, liquids, water droplets, etc.
- These objects tend to exhibit a degree of fluidity
- Object that don't maintain a fixed shape but changes their surface characteristics during motion or closer to another object are called blobby object.
- E.g. molecular structure, water droplet, muscle shape in human body etc.

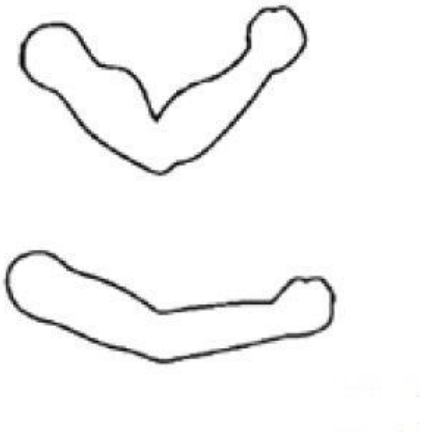


Figure: Blobby muscle in human

- Most Common function used for blobby object is Gaussian density function and a surface function is defined as

$$f(x,y,z) = \sum_k b_k e^{-a_k r_k^2} - T = 0$$

$$\text{Where, } r_k^2 = \sqrt{x_k^2 + y_k^2 + z_k^2}$$

T= some specified threshold
a, b = parameters used to adjust the amount of blobbines for individual object.

Advantages

- Can represent organic, blobby or liquid line structures.
- Suitable for modeling natural phenomena like water, human body.
- Surface properties can be easily derived from mathematical equations.

Disadvantages

- Requires expensive computation
- Requires special rendering engine
- Not supported by most graphics hardware

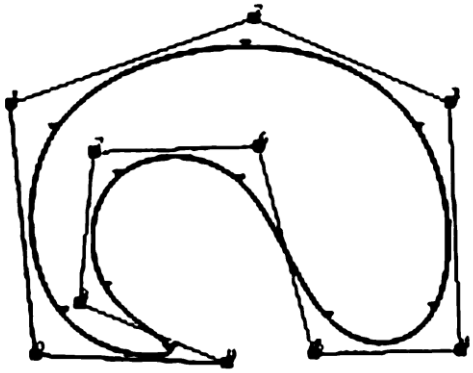
Spline Representation

- Splines are the smooth curves passing through a set of given points.
- A Spline is a flexible strips used to produce smooth curve through a designated set of points
- A curve drawn with these set of points is spline curve.
- Spline curves are used to model 3D object surface shape smoothly.
- In computer graphics, continuous curve that are formed with polynomial section with certain boundary conditions are called spline curve.
- Mathematically, spline are described as piece-wise cubic polynomial functions.

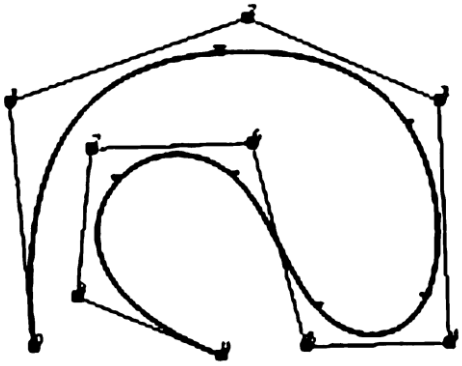
- In computer graphics, a spline surface can be described with two set of orthogonal spline curves.
- Spline is used in graphics application to design and digitize drawings for storage in computer and to specify animation path for the objects or the camera in the scene.
- CAD application for spline includes the design of automobile bodies, aircraft and spacecraft surface, and ship hulls etc.

Splines are of two types

1. **Closed Splines:** the generated curve will touch the first and last legs of the control



2. **Open Splines**: the generated curve will not touch the first and last legs of the control



Interpolation and approximation spline

We specify spline curve by giving a set of coordinate positions called **control points.** This indicates the general shape of the curve.

Interpolation Spline: When curve section passes through each control point, the curve is said to interpolate the set of control points and that spline is known as Interpolation Spline.

- Interpolation curves are used to digitize drawings or to specify animation paths.

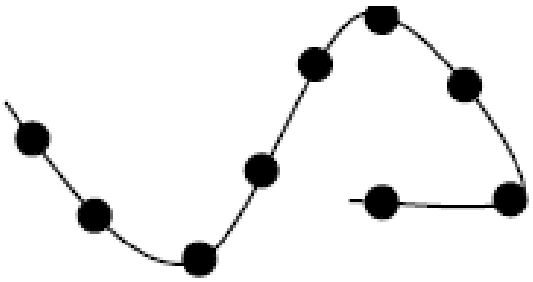


Figure: A set of Nine Control points interpolated with piecewise continuous polynomial section

Approximation Spline: When curve section follows general control point path without necessarily passing through any control point, the resulting curve is said to approximate the set of control points and that curve is known as Approximation Spline.

OR,

When the polynomials are fitted to the general control point path without necessarily passing through any control points, the resulting curve is said to approximate the set of control points.

Figure: A set of Nine Control points approximated with piecewise continuous polynomial section

- A spline curve is designed, modified and manipulated with operations on the control points. The curve can be translated, rotated or scaled with transformation applied to the control points.
- Spline curve can be modified by selecting different control point position.
- We can apply transformation on the curve according to need like translation scaling etc.

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

The shape of the convex hull is to imagine a rubber band stretched around the position of the control points so that each control point is either on the perimeter of the hull or inside it.

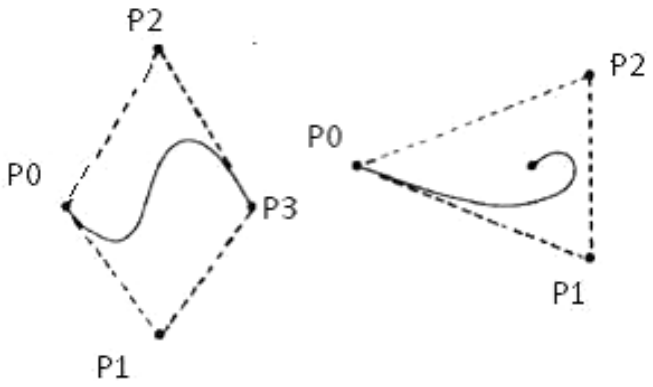


Figure: Convex hull shapes (dashed lines) for two sets of control points

Parametric continuity condition

- For smooth transition from one curve section on to next curve section we put various continuity conditions at connection points.

Let parametric coordinate functions as

$$x = x(u),\ y = y(u),\ z = z(u) \qquad \because u1 \ll u \ll u2$$

Then

Zero order parametric continuity (c^0) means simply curves meets i.e. last point of first curve section & first points of second curve section are same.

First order parametric continuity (c^1) means first parametric derivatives are same for both curve section at intersection points.

Second order parametric continuity (c^2) means both the first & second parametric derivative of two curve section are same at intersection.

- ❖ Higher order parametric continuity is can be obtain similarly.
- ❖ First order continuity is often sufficient for general application but some graphics package like CAD requires second order continuity for accuracy.

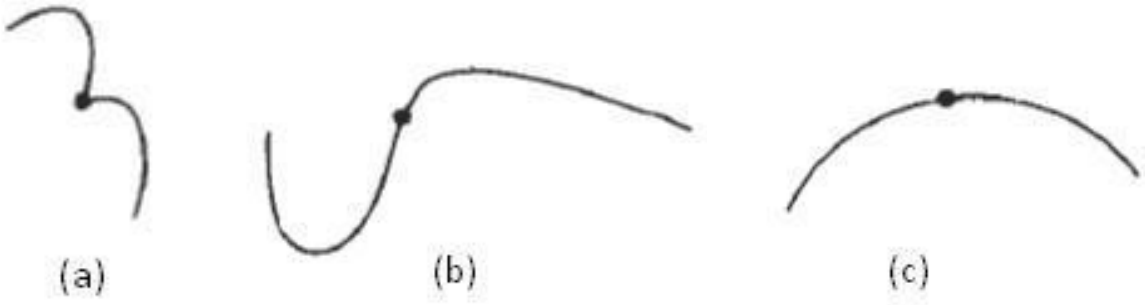


Figure: Piecewise construction of a curve by joining two curve segments uses different orders of continuity: (a) zero-order continuity only, (b) first-order continuity, and (c) second-order continuity.

Geometric continuity condition

- ❖ Another method for joining two successive curve sections is to specify condition for geometric continuity.

Zero order geometric continuity (g^0) is same as parametric zero order continuity that two curve section meets.

First order geometric continuity (g^1) means that the parametric first derivatives are proportional at the intersection of two successive sections but does not necessary Its magnitude will be equal.

Second order geometric continuity (g^2) means that the both parametric first & second derivatives are proportional at the intersection of two successive sections but does not necessarily magnitude will be equal.

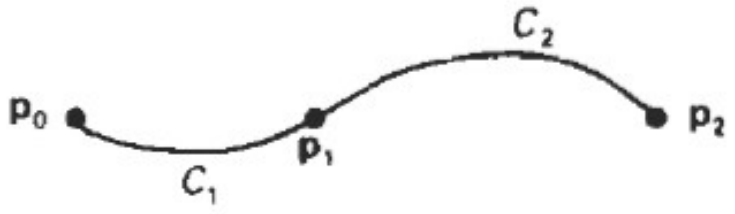


Figure: Three control points fitted with two curve sections joined with a parametric continuity

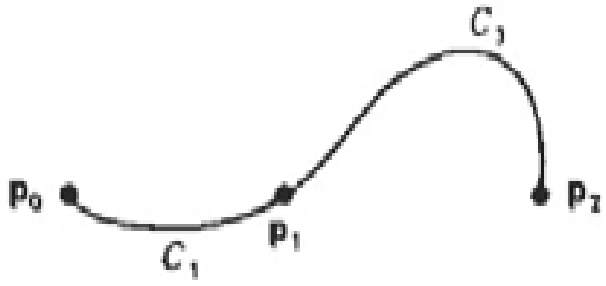


Figure: geometric continuity where the tangent vector of curve C3 at point p1 has a greater magnitude than the tangent vector of curve C1 at p1.

Spline Specifications

There are three methods to specifying a particular spline representation:

1. We can state the set of **boundary conditions** that are imposed on the spline

2. We can state the matrix that characterizes (**characterizing matrix**) the spline

3. We can state the set of **blending functions** that determine how specified geometric constraints on the curve are combined to calculate positions along the curve path.

To illustrate these three equivalent specifications, suppose we have the following parametric cubic polynomial representation for the x coordinate along the path of a spline section

$$X(u) = a_x u^3 + b_x u^2 + c_x u + d_x \quad \text{where, } 0 \leq u \leq 1 \ldots\ldots\ldots (1)$$

Boundary conditions for this curve might be set, for example, on the endpoint coordinates x (0) and x(1) and on the parametric first derivatives at the endpoints *x'(0)* and x'(1).

These four boundary conditions are sufficient to determine the values of the four coefficients a_x, b_x, c_x, and d_x.

From the boundary condition, we can obtain the **characterizing matrix** for spline That characterizes the spline curve by first rewriting the equation (1) as the matrix product

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

$$= U.C \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Where U is the row matrix of powers of parameter u**,** and C is the coefficient column matrix
From the above equation the boundary conditions in the matrix form and solve for the coefficient matrix

$$C.C = M_{spline} . M_{geom} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Where M_{geom} is a four-element column matrix containing geometric constraint values on the spline and M_{spline} is 4-by-4 matrix that performs the geometric constraint values to the polynomial coefficients and provides a characterization for spline curve.

Matrix M_{geom} contains control point coordinate values and other geometric constraints that have been specified. We can substitute the matrix representation for C into equation (2) then

$$(u) = U.\ M_{spline}.\ M_{geom} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

The matrix M_{spline}, characterizing a spline representation, sometimes called the basis matrix, is particularly useful for transforming from one spline representation to another.

Finally we can expand the equation (4) to obtain a polynomial representation for coordinate x in terms of geometric constraint parameters

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

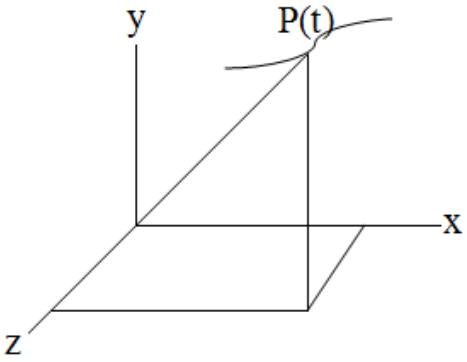
Where g_k are the constraint parameters, such as the control-point coordinates and slope of the curve at the control points, and $BF_k(u)$ are the polynomial **blending functions**

- We can state the set of blending functions (or basis functions) that determine how specified geometric constraints on the curve are combined to calculate positions along the curve path

Parametric Cubic Curve

- A parametric cubic curve is defined as:

$$P(t) = \sum_{i=0}^{3} a_i t^i \qquad 0 <= t <= 1 \quad \text{------ (i)}$$



Where, P(*t*) is a point on the curve
a= algebraic coefficients
t= tangent Vector

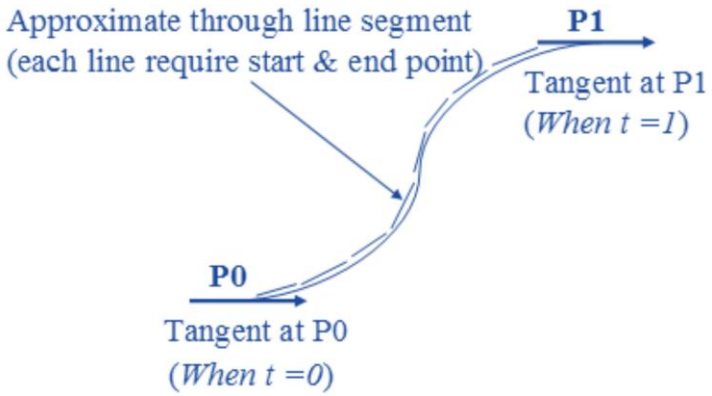
Expanding equation (i) yield

$$P(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0 \text{ ---------- (ii)}$$

This equation is separated into three components of P (t)

$$x(t) = a_{3x} t^3 + a_{2x} t^2 + a_{1x} t + a0x$$
$$y(t) = a_{3y} t^3 + a_{2y} t^2 + a_{1y} t + a0y$$
$$z(t) = a_{3z} t^3 + a_{2z} t^2 + a_{1z} t + a0z \text{ ----------(iii)}$$

Here, to solve (iii) the *twelve unknown* coefficients a_{ij} (algebraic coefficients) must be specified. From the known end point coordinates of each segment, six of the twelve needed equations are obtained. The other six are found by using tangent vectors at the two ends of each segment. The direction of the tangent vectors establishes the slopes (direction cosines) of the curve at the end point



Hermite Interpolation

- The procedure for defining a cubic curve using ***end points*** and ***tangent vector*** is one form of Hermite interpolation
- Each cubic curve segment is parameterized from 0 to 1 so that known end points correspond to the limit values of the parametric variable *t*, that is P(0) and P(1)
- Substituting *t* = 0 and *t* = 1 the relationship between two end point vectors and the algebraic coefficients are found

$$P(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0$$

$$P(0) = a_0 \text{ and } P(1) = a_3 + a_2 + a_1 + a_0 \text{ ----- ---- ---- (IV)}$$

To find the tangent vectors equation (ii) must be differentiated with respect to t

$$P(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0$$

$$P'(t) = 3 a_3 t^2 + 2 a_2 t + a_1$$

The tangent vectors at the two end points are found by substituting t = 0 and t = 1 in this equation

$$P'(0) = a_1$$

$$P'(1) = 3 a_3 + 2 a_2 + a_1 \text{---------- (V)}$$

The algebraic coefficients 'ai' in equation (ii) can now be written explicitly in terms of boundary conditions – endpoints and tangent vectors are

$a_0 = P(0)$
$a_1 = P'(0)$
$a_2 = -3 P(0) - 3 P(1) - 2 P'(0) - P'(1)$
$a_3 = 2 P(0) - 2 P(1) + P'(0) + P'(1)$

(Note: - The value of a2 & a3 can be determined by solving the equation IV & V)

Substituting these values of 'a_i' in equation (ii) and rearranging the terms yields

$$\mathbf{P(t) = (2t^3 - 3t^2 + 1) P(0) + (-2t^3 + 3t^2) P(1) + (t^3 - 2t^2 + t) P'(0) + (t^3 - t^2) P'(1)}$$

The values of P (0), P (1), P' (0), P' (1) are called *geometric coefficients* and represent the known vector quantities in the above equation

The polynomial coefficients of these vector quantities are commonly known as *blending functions* by varying parameter t in these blending function from 0 to 1 several points on curve segments can be found.

Quadric Surface

- If a surface is the graph in three-space of an equation of ***second degree***, it is called a quadric surface.
- Cross section of quadric surface are conics.
- Quadric Surface is one of the frequently used 3D objects surface representation.
- The quadric surface can be represented by a ***second degree*** polynomial.
- Quadric this includes

1. Sphere: For the set of surface points (x,y,z) the spherical surface is represented as:
   $x^2+y^2+z^2 = r^2$, with radius r and centered at co-ordinate origin.
2. Ellipsoid: $\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1$, where (x,y,z) is the surface points and a,b,c are the radii on X,Y and Z directions respectively.
3. Elliptic parboiled: $\frac{x^2}{a^2}+\frac{y^2}{b^2}=z$
4. Hyperbolic parboiled : $\frac{x^2}{a^2}-\frac{y^2}{b^2}=z$
5. Elliptic cone : $\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}=0$
6. Hyperboloid of one sheet: $\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}=1$
7. Hyperboloid of two sheet: $\frac{x^2}{a^2}-\frac{y^2}{b^2}-\frac{z^2}{c^2}=1$

Bezier Curves and surfaces

- This spline approximation method was developed by the French engineer Pierre Bezier for the use of Renault automobile bodies.
- Bezier splines have a number of properties that make them highly useful and convenient for curves surface design.
- They are also easy to implement.
- It is also widely available in various CAD systems, in general graphics packages and in assorted drawing and painting packages.

Bezier Curves

- Bezier curve section can be fitted to any number of control points.
- Number of control points and their relative position gives degree of the Bezier polynomials.
- With the interpolation spline Bezier curve can be specified with boundary condition or blending function.
- Most convenient method is to specify Bezier curve with blending function.

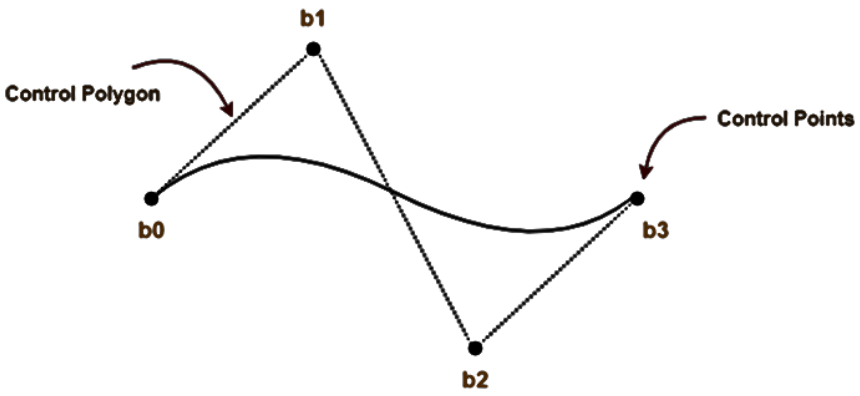


Figure: Bezier Curve

In above Figure,

- This Bezier curve is defined by a set of control points b_0, b_1, b_2 and b_3.
- Point b_0 and b_3 are ends of the curve.
- Point b_1 and b_2 determine the shape of the curve.

Derivation for Equation of Bezier Curve

A Bezier curve is parametrically represented by the following equation,

$$P(t) = \sum_{k=0}^{n} P_k \; B_{k.n}\,(t)$$

Where,

t is any parameter where 0 <= t <= 1

P (t) = any point lying on the Bezier curve

P_k = k^{th} control point of the Bezier curve

n = degree of the curve or total control point and total control point is denoted by n+1, if n control points then n-1 degree polynomial.

$B_{k,n}$ (t) = Bernstein / Bezier / Blending function = C (n,k) * t^k * $(1-t)^{n-k}$

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

Bezier Curve Equation for 4 control points (i.e. Cubic Bezier Curve)

- Cubic Bezier curve is a Bezier curve with degree 3.
- The total number of control points in a cubic Bezier curve is 4.

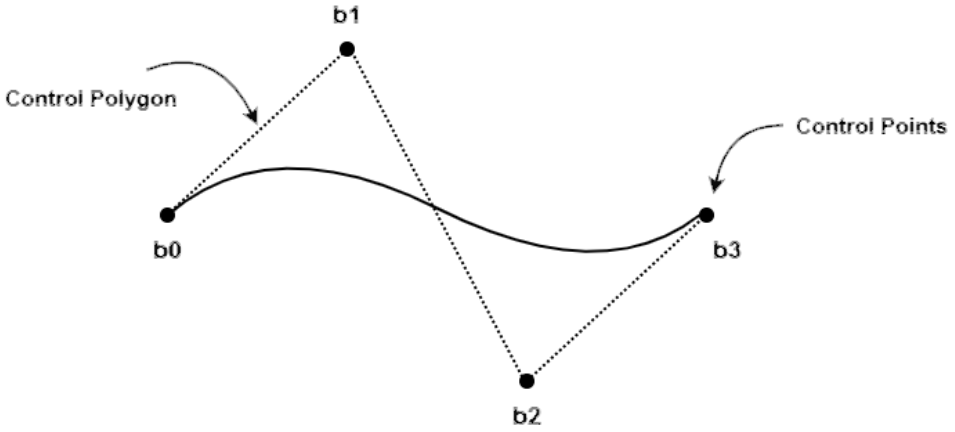


Figure: Cubic Bezier Curve

Where,

Curve is defined by 4 control points b_0, b_1, b_2 and b_3.The degree of this curve is 3. So, it is a cubic Bezier curve.

We know the parametric equation for Bezier curve is,

$$\boldsymbol{P(t) = \sum_{k=0}^{n} P_k \; B_{k.n}\,(t)}$$

Substituting n = 3 for a cubic Bezier curve, we get

$$\boldsymbol{P(t) = \sum_{k=0}^{3} P_k \; B_{k.3}\,(t)}$$

Expanding the above equation, we get,

$$P\,(t) = P_0\,B_{0,\,3}\,(t) + P_1\,B_{1,\,3}\,(t) + P_2\,B_{2,\,3}\,(t) + P_3\,B_{3,\,3}\,(t) \quad \ldots\ldots\ldots (1)$$

Now,

Calculate Blending Function $B_{0,\,3}$

$$B_{0,\,3}\,(t) = \frac{3!}{0!(3-0)!}\; t^0 * (1-t)^{3-0}$$

$$B_{0,\,3}\,(t) = (1\text{-}t)^{\,3} \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Calculate Blending Function $B_{1,\,3}$

$$B_{1,\,3}\,(t) = \frac{3!}{1!(3-1)!}\; t^1 * (1-t)^{3-1}$$

$$B_{1,\,3}\,(t) = 3t\,(1\text{-}t)^{\,2} \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Calculate Blending Function $B_{2,\,3}$

$$B_{2,\,3}\,(t) = \frac{3!}{2!(3-2)!}\; t^2 * (1-t)^{3-2}$$

$$B_{2,\,3}\,(t) = 3t^2\,(1\text{-}t) \ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Calculate Blending Function $B_{3,3}$

$$B_{3,3}(t) = \frac{3!}{3!(3-3)!}\ t^3 * (1-t)^{3-3}$$

$$B_{3,3}(t) = t^3 \text{ ……. ……. (5)}$$

Substitute (2), (3), (4) and (5) in Equation (1), we get

$$\mathbf{P(t) = B_0(1\text{-}t)^3 + B_1 3t(1\text{-}t)^2 + B_2 3t^2(1\text{-}t) + B_3 t^3}$$

Which is Required Equation for cubic Bezier curve.

Properties of Bezier curves

- It always passes through first control point i.e. $p(0) = p_0$
- It always passes through last control point i.e. $p(1) = p_n$
- Parametric first order derivatives of a Bezier curve at the endpoints can be obtain from control point coordinates as:

$$p'(0) = -np_0 + np_1$$
$$p'(1) = -np_{n-1} + np_n$$

- Parametric second order derivatives of endpoints are also obtained by control point coordinates as:

$$p''(0) = n(n-1)[(p_2 - p_1) - (p_1 - p_0)]$$
$$p''(1) = n(n-1)[(p_{n-2} - p_{n-1}) - (p_{n-1} - p_n)]$$

- Bezier curve always lies within the convex hull of the control points.

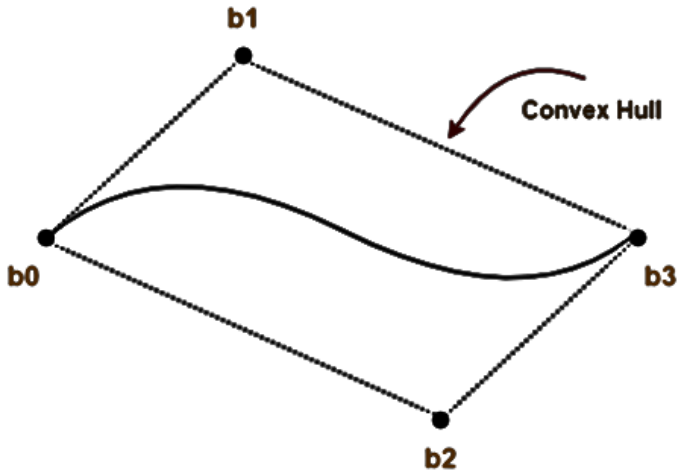


Figure: Bezier curve with convex hull

- Bezier blending function is always positive.
- Sum of all Bezier blending function is always 1.

$$\boldsymbol{P(t) = \sum_{k=0}^{n} B_{k.n}\,(t) = 1}$$

- So any curve position is simply the weighted sum of the control point positions.
- Bezier curve smoothly follows the control points without erratic oscillations.

Numerical problem:

Problme-1: Given a Bezier curve with 4 control points B_0 [1 0], B_1 [3 3], B_2 [6 3], B_3 [8 1]. Determine any 5 points lying on the curve. Also, draw a rough sketch of the curve.

Solution

Given that, the curve is defined by 4 control points. So, the given curve is a cubic Bezier curve.

The parametric equation for a cubic Bezier curve is

$$\mathbf{P\ (t) = B_0(1-t)^3 + B_1 3t(1-t)^2 + B_2 3t^2(1-t) + B_3 t^3}$$

Substituting the control points B_0, B_1, B_2 and B_3, we get,

$P(t) = [1, 0](1-t)^3 + [3 ,3]3t(1-t)^2 + [6 ,3]3t^2(1-t) + [8, 1]t^3$ ……..(1)

Now,

To get 5 points lying on the curve, assume any 5 values of t lying in the range 0 <= t <= 1.

Let 5 values of t are 0, 0.2, 0.5, 0.7, and 1.

For t = 0:

Substituting t=0 in (1), we get

$P\ (0) = [1, 0]\ (1-0)^3 + [3, 3]3(0)\ (1-t)^{\ 2} + [6, 3]3(0)^2(1-0) + [8, 1]\ (0)^3$

P (0) = [1, 0] + 0 + 0 + 0

P (0) = [1, 0]

For t = 0.2:

Substituting t=0.2 in (1), we get-

$P(0.2) = [1, 0] (1-0.2)^3 + [3, 3]3(0.2) (1-0.2)^2 + [6, 3]3(0.2)^2(1-0.2) + [8, 1] (0.2)^3$

$P(0.2) = [1, 0] (0.8)^3 + [3, 3]3(0.2) (0.8)^2 + [6, 3]3(0.2)^2(0.8) + [8, 1] (0.2)^3$

P (0.2) = [1, 0] x 0.512 + [3, 3] x 3 x 0.2 x 0.64 + [6, 3] x 3 x 0.04 x 0.8 + [8, 1] x 0.008

P (0.2) = [1, 0] x 0.512 + [3, 3] x 0.384 + [6, 3] x 0.096 + [8, 1] x 0.008

P (0.2) = [0.512, 0] + [1.152, 1.152] + [0.576, 0.288] + [0.064, 0.008]

P (0.2) = [2.304, 1.448]

For t = 0.5:

Substituting t=0.5 in (1), we get-

$P(0.5) = [1, 0] (1-0.5)^3 + [3, 3]3(0.5) (1-0.5)^2 + [6, 3]3(0.5)^2(1-0.5) + [8, 1] (0.5)^3$

$P(0.5) = [1, 0] (0.5)^3 + [3, 3]3(0.5) (0.5)^2 + [6, 3]3(0.5)^2(0.5) + [8, 1] (0.5)^3$

P (0.5) = [1, 0] x 0.125 + [3, 3] x 3 x 0.5 x 0.25 + [6, 3] x 3 x 0.25 x 0.5 + [8, 1] x 0.125

P (0.5) = [1, 0] x 0.125 + [3, 3] x 0.375 + [6, 3] x 0.375 + [8, 1] x 0.125

P (0.5) = [0.125, 0] + [1.125, 1.125] + [2.25, 1.125] + [1, 0.125]

P (0.5) = [4.5, 2.375]

For t = 0.7:

Substituting t=0.7 in (1), we get-

$P(t) = [1, 0] (1-t)^3 + [3, 3]3t (1-t)^2 + [6, 3]3t^2 (1-t) + [8, 1] t^3$

$P(0.7) = [1, 0] (1-0.7)^3 + [3, 3]3(0.7) (1-0.7)^2 + [6, 3]3(0.7)^2(1-0.7) + [8, 1],(0.7)^3$

$P(0.7) = [1, 0] (0.3)^3 + [3, 3]3(0.7) (0.3)^2 + [6, 3]3(0.7)^2(0.3) + [8, 1] (0.7)^3$

P (0.7) = [1, 0] x 0.027 + [3, 3] x 3 x 0.7 x 0.09 + [6, 3] x 3 x 0.49 x 0.3 + [8, 1] x 0.343

P (0.7) = [1, 0] x 0.027 + [3, 3] x 0.189 + [6, 3] x 0.441 + [8, 1] x 0.343

P (0.7) = [0.027, 0] + [0.567, 0.567] + [2.646, 1.323] + [2.744, 0.343]

P (0.7) = [5.984, 2.233]

For t = 1:

Substituting t=1 in (1), we get-

$P(1) = [1, 0] (1-1)^3 + [3, 3]3(1) (1-1)^2 + [6, 3]3(1)^2(1-1) + [8, 1] (1)^3$

P (1) = [1, 0] x 0 + [3, 3] x 3 x 1 x 0 + [6, 3] x 3 x 1 x 0 + [8, 1] x 1

P (1) = 0 + 0 + 0 + [8, 1]

P (1) = [8, 1]

Rough Sketch of the Curve:

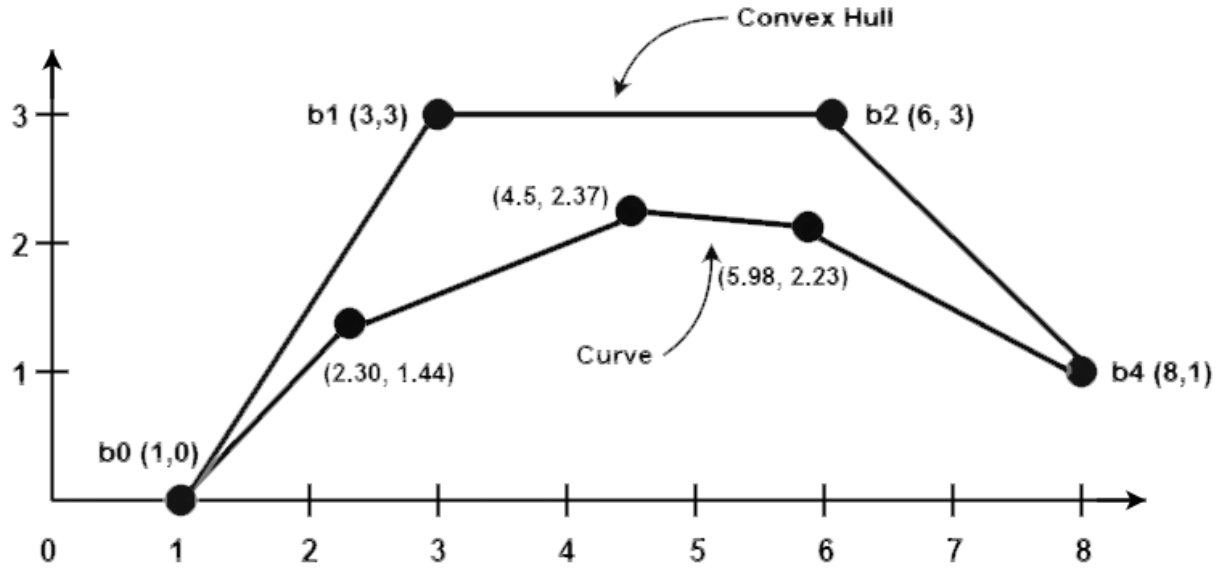


Figure: Bezier Curve and Convex Hull for Problem-1

Problem-2: Calculate the co-ordinates of Bezier curve described by 4 control points P0(0,0), P1(1,2), P2(3,3), P3(4,0)and approximate with 5 lines segments **(CLASSWORK).**

Applications of Bezier Curves

Bezier curves have their applications in the following fields

1. Computer Graphics

- Bezier curves are widely used in computer graphics to model smooth curves.
- The curve is completely contained in the convex hull of its control points.
- So, the points can be graphically displayed & used to manipulate the curve intuitively.

2. Animation

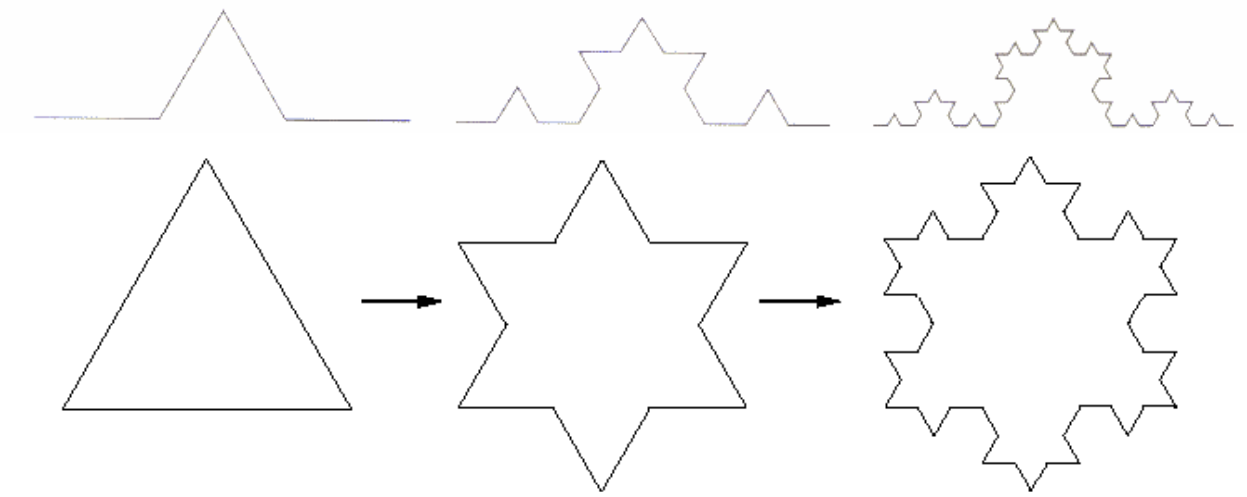
- Bezier curves are used to outline movement in animation applications such as Adobe Flash.
- Users outline the wanted path in Bezier curves.
- The application creates the needed frames for the object to move along the path.
- For 3D animation, Bezier curves are often used to define 3D paths as well as 2D curves.

3. Fonts

- True type fonts use composite Bezier curves composed of quadratic Bezier curves.
- Modern imaging systems like postscript, asymptote etc. use composite Bezier curves composed of cubic Bezier curves for drawing curved shapes.

Fractals

- ❖ Fractal objects refer to those objects which are self-similar at all resolutions.
- ❖ Most of the natural objects such as trees, mountains and coastlines are considered as fractal objects because no matter how far or how close one looks at them, they always appear to be somewhat similar. Which means it is "the same from near as from far".
- ❖ Natural objects, such as mountains and clouds, have irregular or fragmented features, and these are described with fractal geometry methods.
- ❖ Fractal objects can also be generated recursively by applying the same transformation function to an object, e.g. Scale down + rotate + translate.
- ❖ For example, a Fractal Snowflake

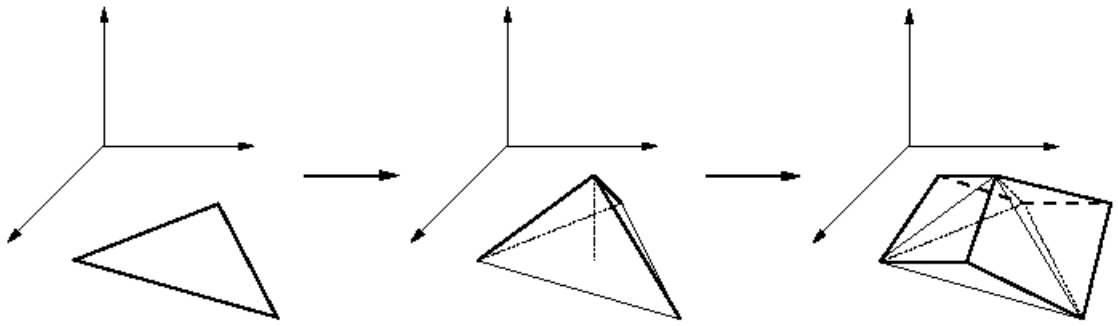


- The name "Fractal" comes from its property: fractional dimension. Unlike Euclidean dimension, the dimensions of fractal objects may not be integers, and we call these dimensions as 'fractal dimension'.

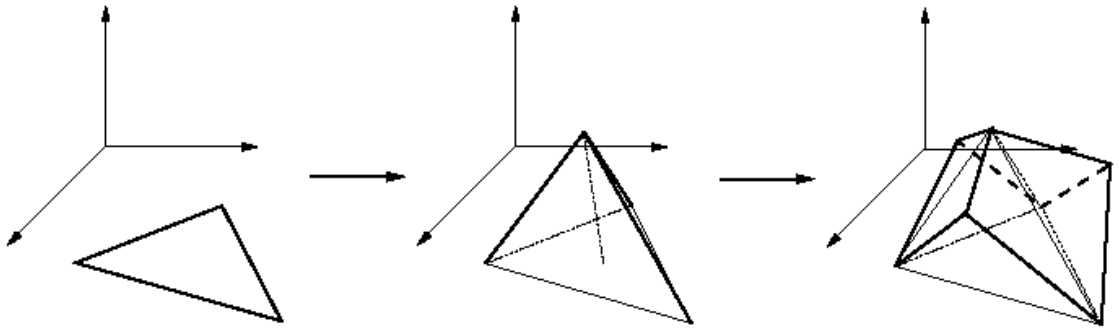
Euclidean dimensions and Fractal dimensions:

- A line segment is 1D. If we divide a line into N equal parts, the parts each look like the original line scaled down by a factor of N = N1/1.
- A square is 2D. If we divide it into N parts, each part looks like the original scaled down by a factor of N1/2.
- For the fractal snowflake, when it is divided into 4 pieces, each resulting piece looks like the original scaled down by a factor of 3, so it has the dimension ***d*** such that 41/d=3. That is, d = 1.26.

We may also create 3D objects in a similar way by adding a third dimension. The following example replaces each triangle of the object with a pyramid, by inserting a single point, at each step (except for the bottom face). This will result in a mountain like object, with a regular appearance.



To give a more natural appearance to the created object, we usually allow some limited random variations at each level of recursion:

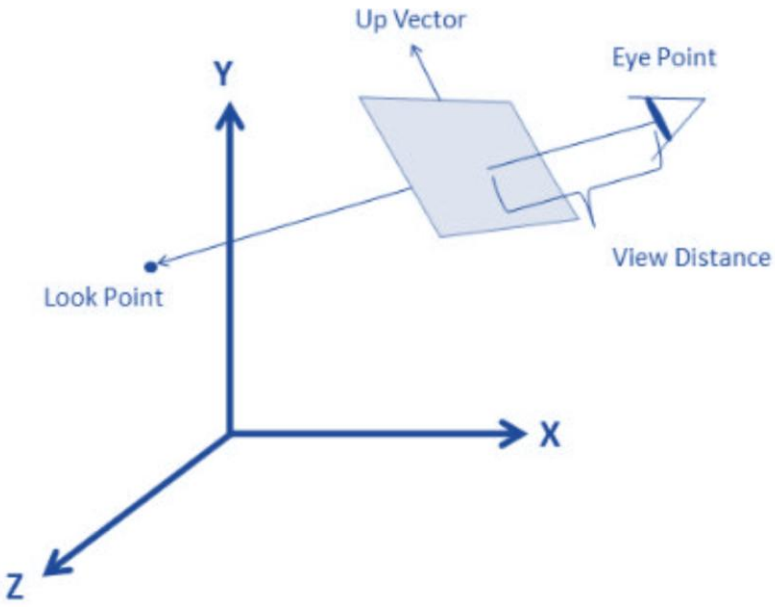


3-D Viewing

- The basic idea of the 3D viewing transformation is similar to the 2D viewing transformation.
- Viewing window is defined in world space that specifies how the viewer is viewing the scene.
- A corresponding view port is defined in screen space, and a mapping is defined to transform points from world space to screen space based on these specifications.
- The view port portion of the transformation is the same as the 2D case.
- Specification of the window, however, requires additional information and results in a more complex mapping to be defined.
- Defining a viewing window in world space coordinates is exactly like it sounds; sufficient information needs to be provided to define a rectangular window at some location and orientation.

The usual viewing parameters that are specified are:

a) **Eye Point**- The position of the viewer in world space
b) **Look Point**- The point that the eye is looking at
c) **View Distance** -The distance that the window is from the eye
d) **Window Size -** The height and width of the window in world space coordinates Up Vector which direction represents "up" to the viewer, this parameter is sometimes specified as an angle of rotation about the viewing axis



3D Viewing pipeline

The steps for computer generation of a view of 3D scene are analogous to the process of taking 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 when we snap the shutter, the scene is cropped to the size of window of the camera and the light from the visible surfaces is projected into the camera film.

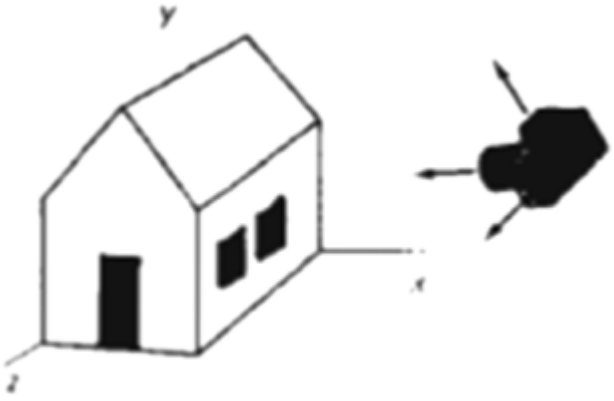


Fig: Photographing a scene involves selection of a camera position and orientation

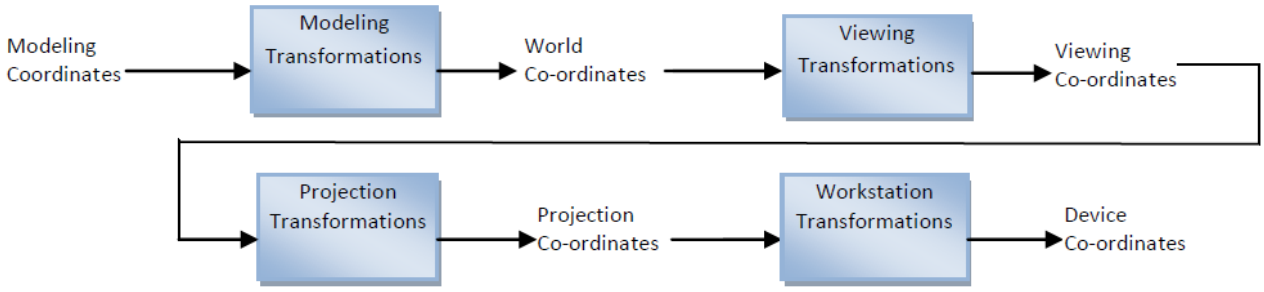


Fig: General three-dimensional transformation pipeline from modeling coordinates to final device coordinates

- 3D view pipeline is the processing steps for modeling and converting a world-coordinate description of a scene to device coordinates.
- Once the scene has been modeled, world-coordinate positions are converted to viewing coordinates.
- The viewing-coordinate system is used in graphics packages as a reference for specifying the observer viewing position and the position of the projection plane, which we can think of in analogy with the camera film plane.
- Next, projection operations are performed to convert the viewing-coordinate description of the scene to coordinate positions on the projection plane, which will then be mapped to the output device.
- Objects outside the specified viewing limits are clipped from further consideration, and the remaining objects are processed through visible-surface identification and surface rendering procedures to produce the display within the device view port.

Projections

- Theory of Projections In engineering, 3-dimensonal objects and structures are represented graphically on a 2-dimensional media.
- The act of obtaining the image of an object is termed projection.
- The image obtained by projection is known as a view.
- All projection theory are based on two variables
  - Line of sight
  - Plane of projection
- A **plane of projection** (i.e., an image or picture plane) is an imaginary flat plane upon which the image created by the line of sight is projected. The image is produced by connecting the points where the lines of sight pierce the projection plane. In effect, 3-D object is transformed into a 2-D

representation, also called projections. The paper or computer screen on which a drawing is created is a plane of projection

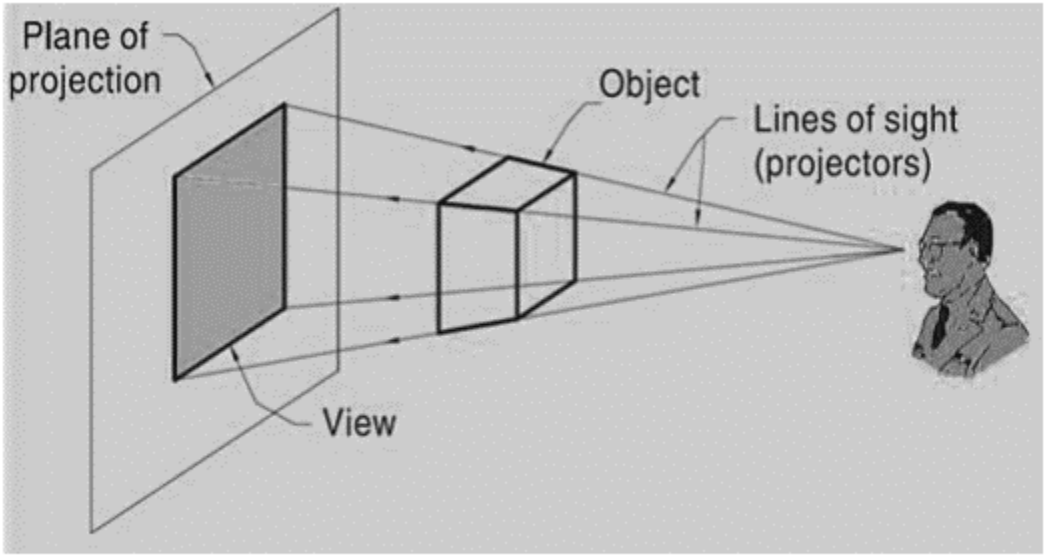


Figure : A simple Projection system

- Transform objects or points in a coordinate system from dimension m into a coordinate system of dimension n where m < n.
- Projection means the image or act of obtaining an image of the object.

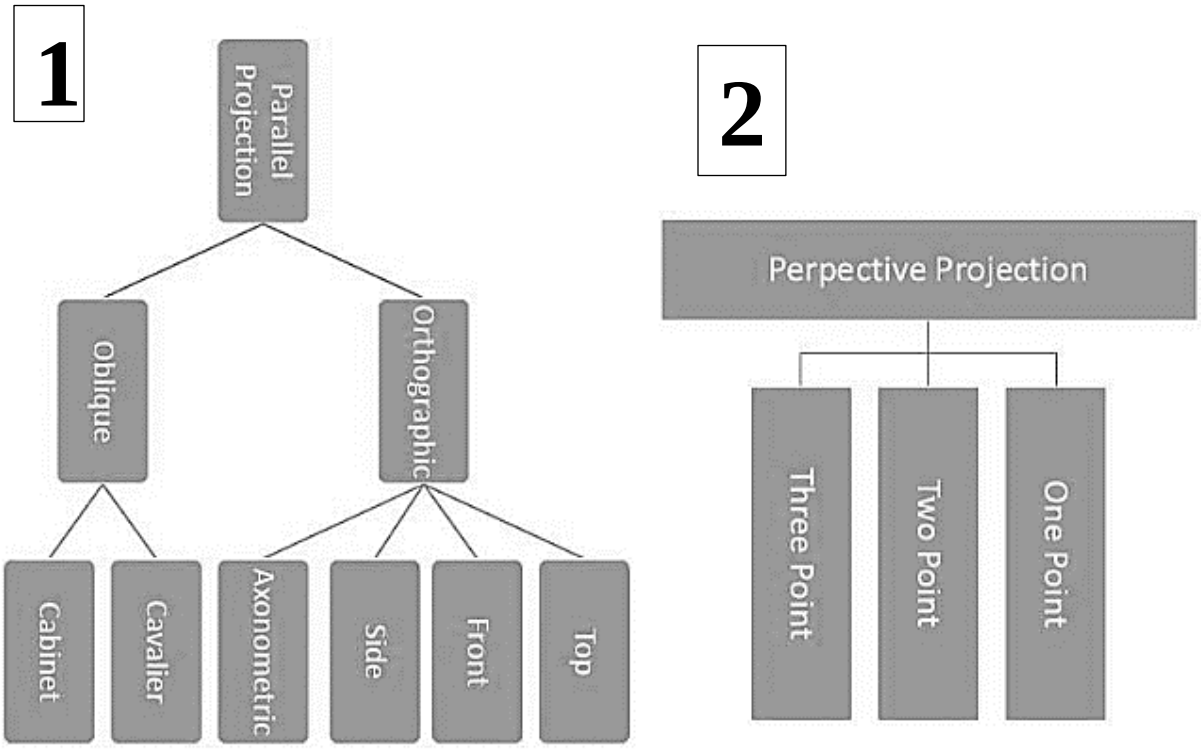
- **3D projection** is method of mapping three-dimensional points to a two-dimensional plane.
    - In general, a projection transforms an N-dimension points to N-1 dimensions.
    - Ideally, an object is projected by projecting each of its endpoints. But an object has infinite number of points. So we cannot project all those points. What we do is that we project only the corner points of an object on a 2D plane and we will join these projected points by a straight line in a 2D plane.
- Once world co-ordinate description of the objects in a scene are converted to viewing co-ordinates, we can project the three dimensional objects onto the two dimensional view plane.

Basic points in projection:

- **Center of projection**: Point from where projection is taken.
- **Projection / view plane**: the plane on which the projection of the object is formed.

- **Projectors**: Lines emerging from the center of projection and hitting the projection plane.
- **Vanishing Point :**

<u>There are two basic projection methods:</u>



1. Parallel projection

- In this projection, the co-ordinate positions are transformed to the view plane along parallel lines.
- It preserves relative proportions of objects so that accurate views of various sides of an object are obtained but doesn't give realistic representation of the 3D object.
- Can be used for exact measurements so parallel lines remain parallel

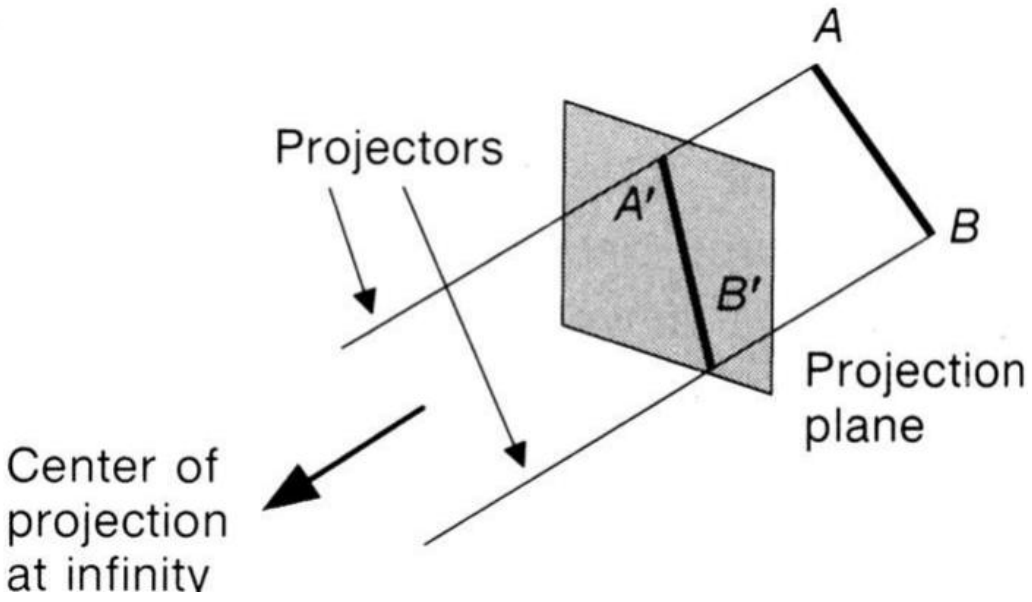
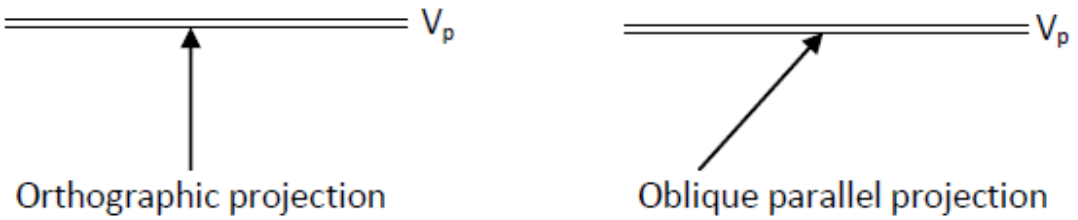


Fig: Parallel projection of an object to the view plane

There are two different types of parallel projections

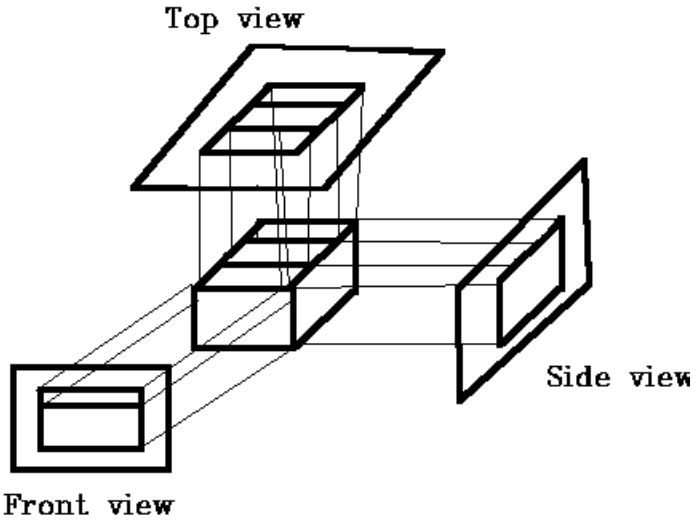
1. Orthographic parallel projection

- When the projection lines are perpendicular to view plane, the projection is **orthographic parallel projection.**
- Otherwise it is **oblique parallel projection**.

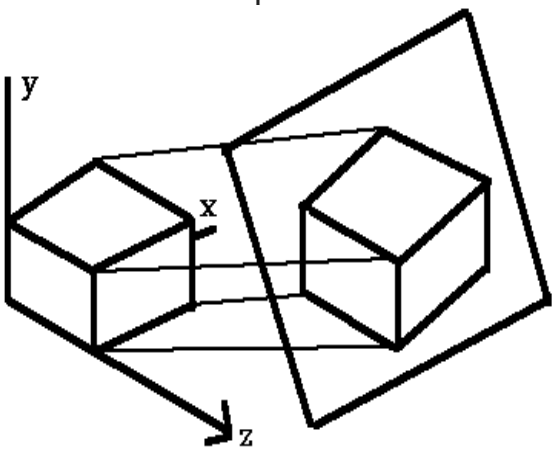


- Orthographic projections are most often used to produce the front, side, and top views of an object.
- Front, side, and rear orthographic projections of an object are called ***elevations*** and
- A top orthographic projection is called a ***plain view.***

- Engineering and Architectural drawings commonly employ these orthographic projections.



- Orthographic projections that show more than one face of an object. Such views are called **axonometric orthographic projections**.
- The most commonly used axonometric projection is the **isometric projection**. Where the projection plane intersects each coordinate axis in the model coordinate system at an equal distance.
- **Axonometric projection** is a type of **orthographic projection** used for creating a pictorial drawing of an object, where the lines of sight are perpendicular to the plane of **projection**, and the object is rotated around one or more of its axes to reveal multiple sides.

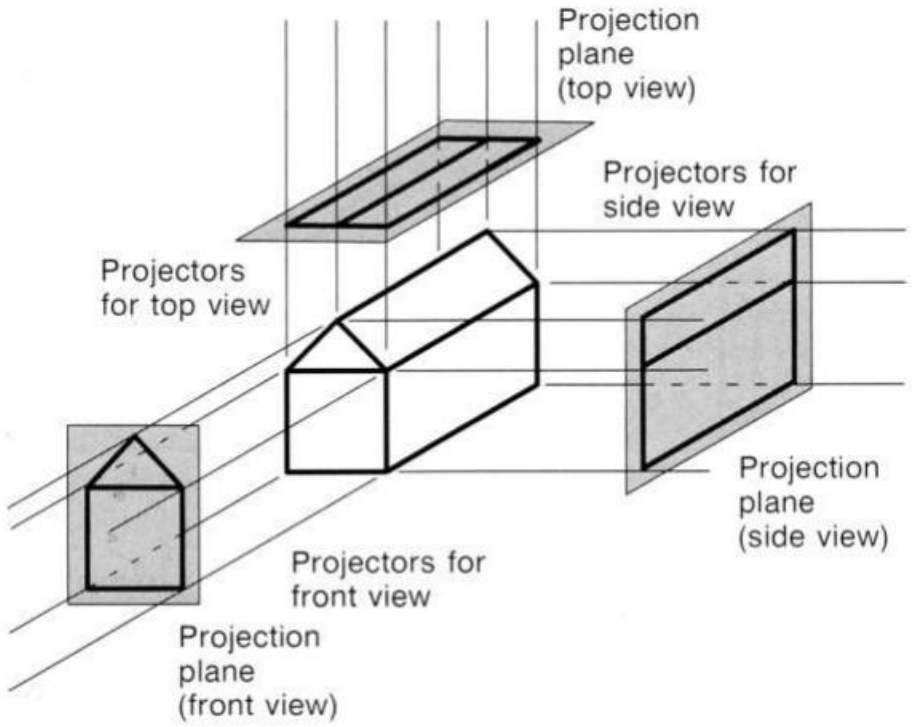


- In orthographic projection, the center of projection lies at infinity and the projections are perpendicular to the view plane. So, a true size and shape of

a single face of object is obtained. In orthographic projection the direction of projection is normal to the projection of the plane.

There are **three types of orthographic** projections

➢ Front Projection
➢ Top Projection
➢ Side Projection



2. Oblique parallel Projection

- A projection in which the angle between the projectors and the plane of plane of projection is not equal to 90 is called oblique projection.
- An oblique projection is formed by parallel projections from a center of projection at infinity intersects the plane of projection at an oblique angle.

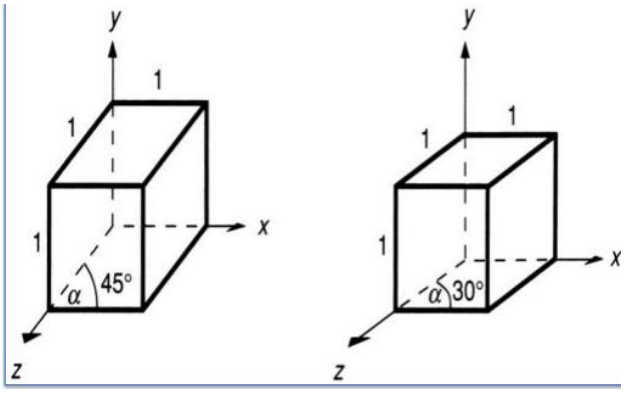
There are **two types of oblique** projections

1. **Cavalier** and
2. **Cabinet**.

- The **Cavalier projection** makes 45° angle with the projection plane. The projection of a line perpendicular to the view plane has the same length as

the line itself in Cavalier projection. In a cavalier projection, the foreshortening factors for all three principal directions are equal.

- The **Cabinet projection** makes 63.4° angle with the projection plane. In Cabinet projection, lines perpendicular to the viewing surface are projected at ½ their actual length.



Transformation Matrix for oblique parallel projection v. imp

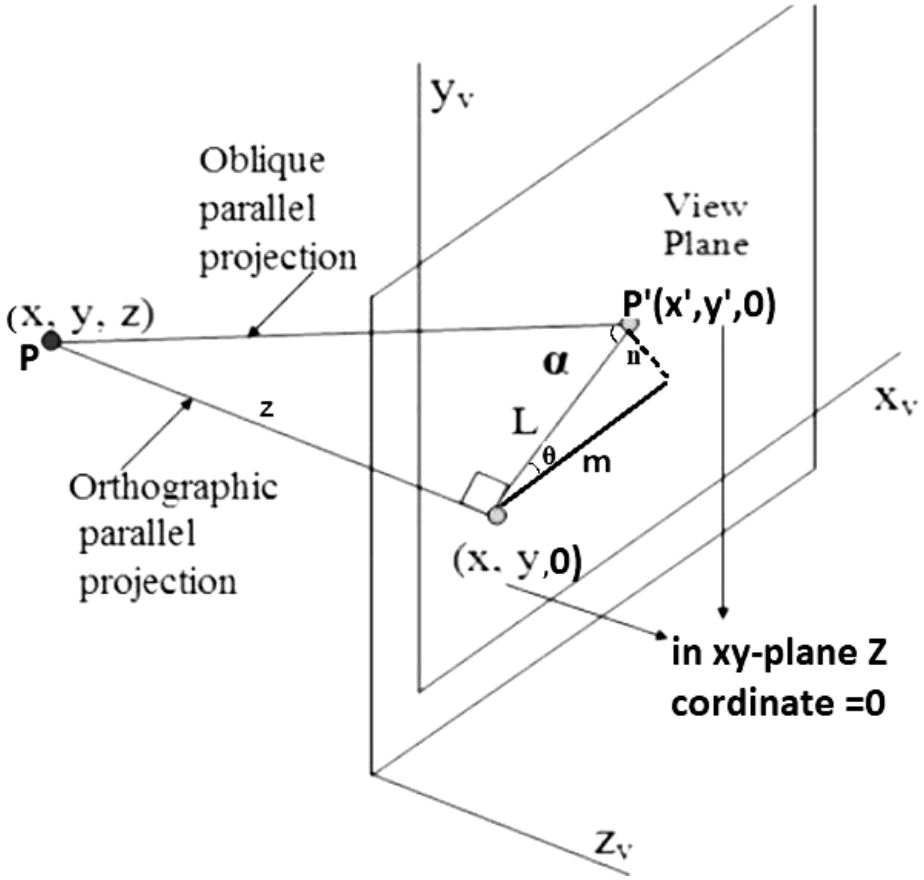


Figure: Oblique Parallel Projection

Consider a Point P (x, y, z) is projected to position P' (x', y', 0) on the view plane as parallel projection. If the P is projected orthographically then orthographic projection coordinated on the plane are (x, y, 0). Oblique projection line from (x, y, z) to (x', y', 0) makes an angle with the line on the projection plane that joins (x', y', 0) and (x, y, 0).

Consider the line of length L is at an angle θ with the horizontal direction in the projection plane.

Now, calculating projection coordinates

$$X' = x + m$$

Find value of m, $\cos \theta = m/L$ then $m = L \cos \theta$
Similarly,

$$Y' = y + n$$

For the value of n, $\sin \theta = \frac{n}{L}$ then $n = L \sin \theta$

Therefore,

$$x' = x + L \cos \theta \text{ and } y' = y + L \sin \theta$$

Since, L depends on the angle α and z coordinate of point to be projected then

$$\text{Tan } \alpha = \frac{z}{L}$$

Thus,

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

$= z\ L_1$, where L_1 is the inverse of tanα (i.e. $L_1 = \frac{1}{Tan\ \alpha} = \cot \alpha$)

So the oblique projection equations are:

$$X' = x + z\ (L_1 \cos \theta)$$
$$Y' = y + z\ (L_1 \sin \theta)$$

Finally, the homogeneous transformation matrix for producing any parallel projection onto the x_v, y_v plane can be written as:

$$M_{Parallel} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & L_1 \cos\theta & 0 \\ 0 & 1 & L_1 \sin\theta & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

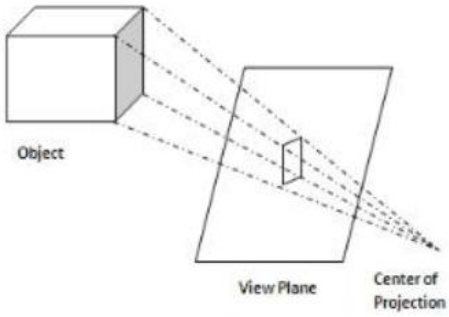
Orthographic projection is obtained when $L_1 = 0$ (occurs at projection angle α of 90 degree) Oblique projection is obtained with non-zero values for L_1.

b) Perspective Projection

- When an object is viewed from different directions and at different distances, the appearance of the object will be different. Such view is called perspective view.
- In this projection, the co-ordinate positions are transformed to the view plane along lines that converges to a point called as **center of projection**.
- The projection is perspective if the distance is finite or has fixed **vanishing point** (All lines appear to meet at some point in the view plane.) or the incoming rays converge at a point. Thus the viewing dimension of object is not exact i.e. seems large or small according as the movement of the view plane from the object.
- The perspective projection perform the operation as the scaling (i.e. zoom in & out) but here the object size is only maximize to the size of real present object i.e. only size reduces not enlarge. This operation is possible here only the movement of the view plane towards or far from the object position.
- In perspective projection, all lines of sight start at a single point. Distance from the observer to the object is finite and the object is viewed from a single point – projectors are not parallel.

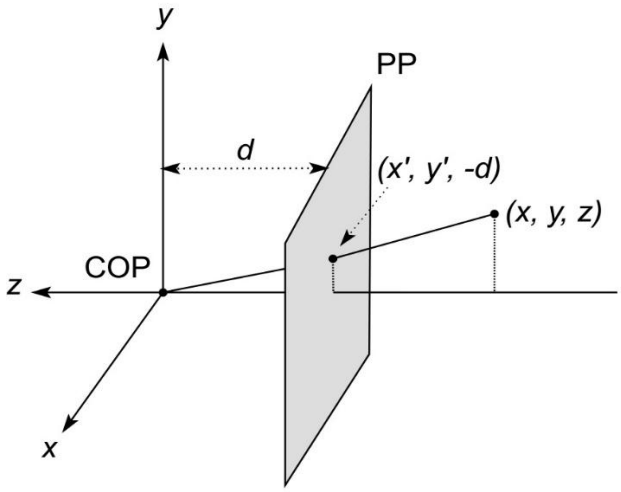
Conclusion:

A perspective projection can be described as the projector lines (lines of sight) that converge at the center of projection, which results in many visual effects of an object. Perspective projection depends on the relative position of the *eye* and the *viewplane*. A perspective projection of an object is often considered more realistic than a parallel projection, since it nearly resembles human vision and photography.



Transformation matrix for perspective projection (Derivation) V^imp

Consider the projection of a point onto the projection plane



By similar triangles, we can compute how much the *x* and *y* coordinates are scaled as

$$\frac{x'}{x} = -\frac{d}{z}$$

$$\frac{y'}{y} = -\frac{d}{z}$$

- ✓ Remember how transformations work with the last coordinate always set to one.
- ✓ What happens if the coordinate is not one?

- ✓ We divide all the coordinates by *h*
- ✓ If *h*= *1*, then nothing changes.

$$\begin{bmatrix} X/\mathbf{h} \\ Y/\mathbf{h} \\ Z/\mathbf{h} \\ \mathbf{h}/\mathbf{h} \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

These points are to recall the homogeneous coordinate system, no need to write these ticked line inside the box in exam or answer for this

- ✓ Sometimes we call this division step the "perspective divide."

Now we can re-write the perspective projection as a matrix equation

$$\begin{bmatrix} X \\ Y \\ \mathbf{h} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix}$$

After division by *h*, we get

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{x}{y}d \\ -\frac{y}{z}d \\ 1 \end{bmatrix}$$

Again, projection implies dropping the *z* coordinate to give a 2D image, but we usually keep it around a little while longer.

Difference between Perspective and Parallel Projection VVImp

BASIS OF DIFFERENCE	PERSPECTIVE PROJECTION	PARALLEL PROJECTION
Description	A perspective projection can be described as the projector lines (lines of sight) that converge at the center of projection, which results in many visual effects of an object.	A parallel projection is a projection of an object in three-dimensional space onto a fixed plane referred as the projection plane or image plane, where the rays, known as lines of sight or projection lines are parallel to each other.
Types	One-point perspective Projection. Two-point perspective projection. Three-point perspective projection.	Orthographic parallel projection Oblique parallel projection.
Accurate View Of Object	Perspective projection cannot give the accurate view of object.	Parallel projection can give the accurate view of object.
Object Representation	Perspective projection represents the object in three dimensional way.	Parallel projection represents the object in a different way like telescope.
Realistic View of Object	Perspective projection forms a realistic picture of object.	Parallel projection does not form realistic view of object.
Distance Of The Object From The Center Of Projection	The distance of the object from the center of projection is finite.	In parallel projection, the distance of the object from the center of projection is infinite.
Projector	Projector in perspective projection is not parallel.	Projector in parallel projection is parallel.
Preservation Of Relative Portion Of An Object	Perspective projection cannot preserve the relative proportion of an object.	Parallel projection can preserve the relative proportion of an object.
Lines Of Projection	The lines of perspective projection are not parallel.	The lines of parallel projection are parallel.

Clipping in 3D

- 3D Clipping, in graphics technology is used to speed up your rendering engine. The most common clipping method is the 2D clipping algorithm.
- 2D clipping is done at the last stage of rendering.
- The triangle-routine used to render the polygons onto the screen make sure that you don't draw outside the screen.
- This can be very fast, but increases the complexity of the triangle-filler.
- This clipping method works directly with two-dimensional screen-coordinates.
- In 3D clipping you do everything in 3D space.
- For example, to draw a hemisphere centered on a given point, the clipping plane passes through the center of the sphere and has a normal vector that coincides with the given point.
- 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: the view plane rectangle and 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 segments:

    1. Outside a boundary plane: $A_x + B_y + C_z + D > 0$, where A, B, C, and D are the plane parameters for that boundary.
    2. Inside the boundary if $A_x + B_y + C_z + D < 0$

- 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 $A_{x1} + B_{y1} + C_{z1} + D = 0$.
- To clip a polygon surface, we can clip the individual polygon edges.
- As in two-dimensional viewing, the projection operations can take place before the view-volume clipping or after clipping. All objects within the view volume map to the interior of the specified projection window.
- 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.

Z-clipping

Z-clipping, or depth clipping, refers to techniques that selectively render certain scene objects based on their depth relative to the screen. Most graphics toolkits allow the programmer to specify a "near" and "far" clip depth, and only portions of objects between those two planes are displayed.

Importance of clipping in video games

- Maximize the game's frame rate and visual quality
- Speed up the rendering of the current scene
- economizing the use of renderer time and memory within the hardware's capability

-End of Chapter-

****Thank You****