

UNIVERSITY of ANBAR
CS & IT COLLEGE
COMPUTER SCIENCE DEPARTMENT
3rd STAGE
Second Course



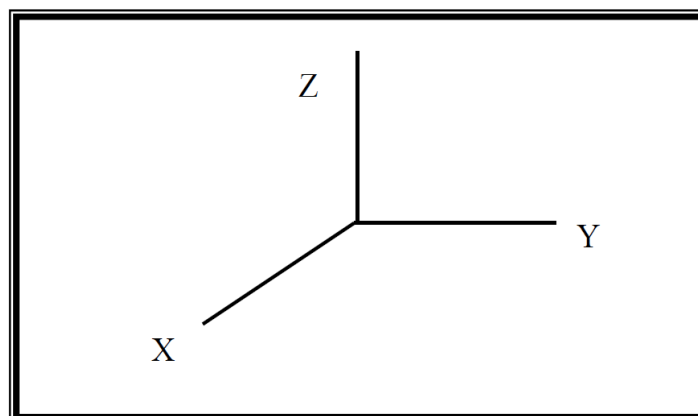
1. Computer Graphics in Three–Dimension

The world composed of three – dimensional images so the object has height, width and depth. The computer uses a mathematical model to create the image.

1.1 Coordinate system

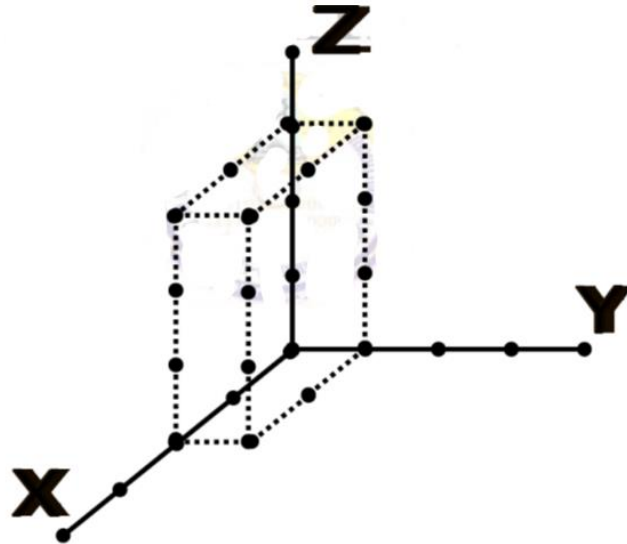
A three- dimensional coordinate system can be view as an extension of the two dimension coordinate system.

The third – dimension depth is represented by the **Z – axis** which is at right angle to the **X, Y** coordinate plane. A point can be described by triple **(X, Y, Z)** of coordinate values.



Example 1: Draw the figure: (0,0,3), (0,1,3), (2,0,3), (2,1,3), (0,1,0) (2,0,0), (2,1,0).

Solution:



1.2 The Transformations

A three-dimensional point (x, y, z) will be associated with homogeneous row vector [x, y, z, 1]. We can represent all three dimensional linear transformation by multiplication of 4*4 matrix. The new coordinate of a translate point can be calculate by using transformation.

$$T = \begin{cases} \underline{X} = X + a \\ \underline{Y} = Y + b \\ \underline{Z} = Z + c \end{cases}$$

$$(\underline{X} \ \underline{Y} \ \underline{Z} \ 1) = (X \ Y \ Z \ 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & c & 1 \end{bmatrix}$$

1.2.1 Translation

A point can be translated in 3D by adding translation coordinate (t_x, t_y, t_z) to the original coordinate (X, Y, Z) to get the new coordinate (X', Y', Z').

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

$$P' = P \cdot T$$

$$[X' \quad Y' \quad Z' \quad 1] = [X \quad Y \quad Z \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$= [X + t_x \quad Y + t_y \quad Z + t_z \quad 1]$$

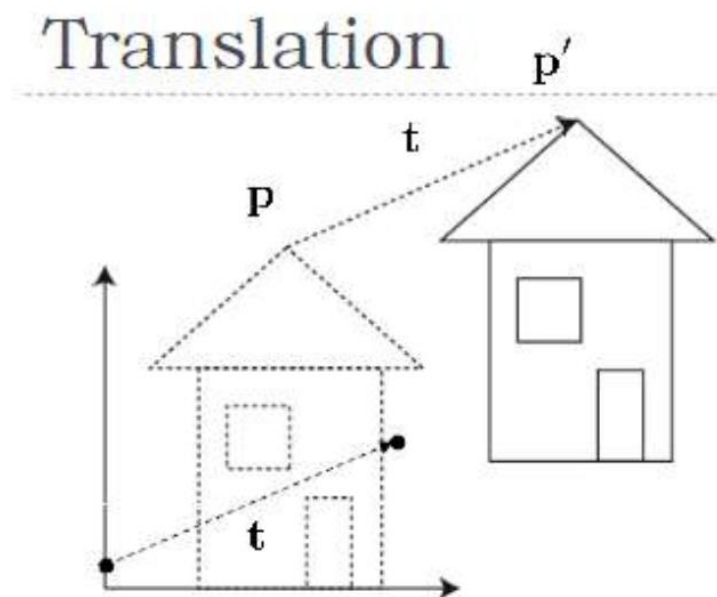


Figure: 3D Translation

Example 2: A Point has coordinates P (5, 6, 7) in x, y, z-direction. Apply the translation with a distance of 3 towards x-axis, 3 towards y-axis, and 2 towards the z-axis. Find the new coordinates of the point?

Solution:

$$(X' \ Y' \ Z' \ 1) = (X \ Y \ Z \ 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

$$(X' \ Y' \ Z' \ 1) = (5 \ 6 \ 7 \ 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 3 & 3 & 2 & 1 \end{bmatrix}$$

$$= [X+T_x \ Y+T_y \ Z+T_z \ 1] = [8 \ 9 \ 9 \ 1]$$

The new coordinates are = (8, 9, 9)

H.W 1: 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?

1.2.2 Scaling

You can change the size of an object using scaling transformation. In the scaling process, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result. The following figure shows the effect of 3D scaling:

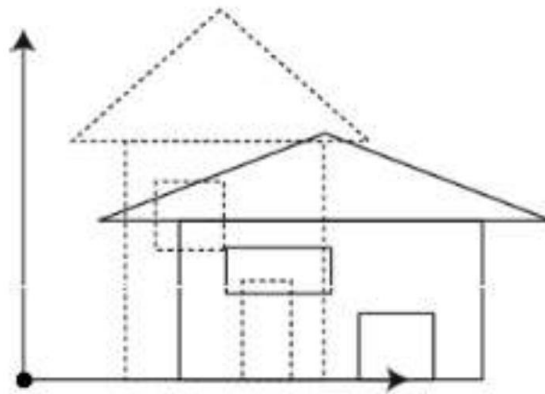


Figure: 3D Scaling

In 3D scaling operation, three coordinates are used. Let us assume that the original coordinates are (X, Y, Z) , scaling factors are (S_x, S_y, S_z) respectively, and the produced coordinates are (X', Y', Z') . This can be mathematically represented as shown below:

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = P \cdot S$$

$$\begin{aligned} [X' \quad Y' \quad Z' \quad 1] &= [X \quad Y \quad Z \quad 1] \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= [X \cdot S_x \quad Y \cdot S_y \quad Z \cdot S_z \quad 1] \end{aligned}$$

Example 3: 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:

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:

$$x' = x * S_x = 1 * 3 = 3$$

$$y' = y * S_y = 4 * 4 = 16$$

$$z' = z * S_z = 4 * 4 = 16$$

The new coordinates = (3, 16, 16)

For coordinate Q:

$$x' = x * S_x = 4 * 3 = 12$$

$$y' = y * S_y = 4 * 4 = 16$$

$$z' = z * S_z = 6 * 4 = 24$$

The new coordinates = (12, 16, 24)

For coordinate R:

$$x' = x * S_x = 4 * 3 = 12$$

$$y' = y * S_y = 1 * 4 = 4$$

$$z' = z * S_z = 2 * 4 = 8$$

The new coordinates = (12, 4, 8)

For coordinate S:

$$x' = x * S_x = 1 * 3 = 3$$

$$y' = y * S_y = 1 * 4 = 4$$

$$z' = z * S_z = 1 * 4 = 4$$

The new coordinates = (3, 4, 4)

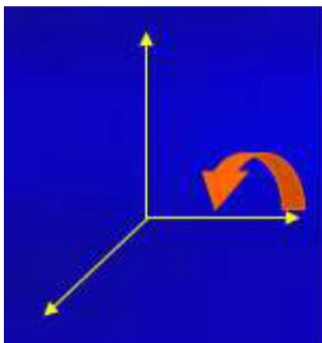
Thus, the new coordinates after scaling = **P (3, 16, 16), Q (12, 16, 24), R (12, 4, 8), S (3, 4, 4).**

1.2.3 Rotation

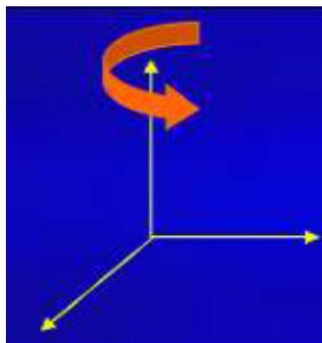
3D rotation is not same as 2D rotation. In 3D rotation, we have to specify the angle of rotation along with the axis of rotation. We can perform 3D rotation about X, Y, and Z axes. They are represented in the matrix form as below:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

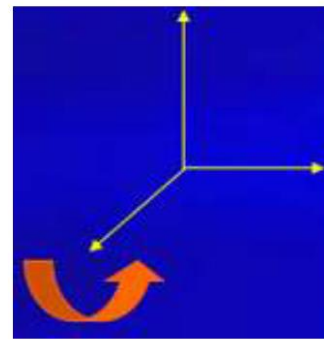
The following figure explains the rotation about various axes:



Rotation about x-axis



Rotation about y-axis



Rotation about z-axis

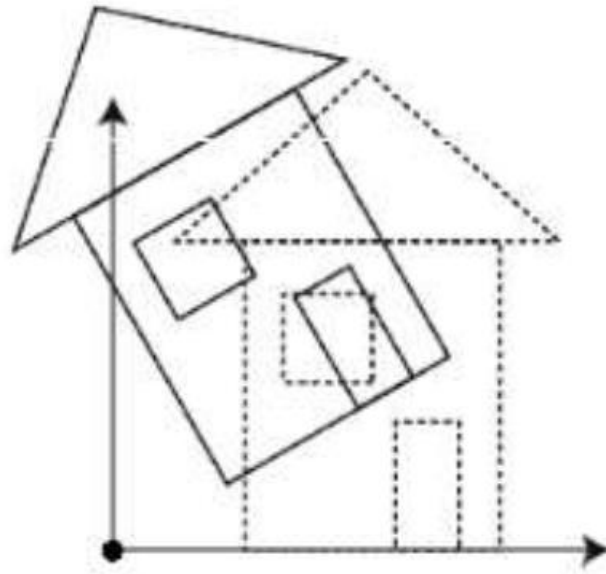


Figure: 3D Rotation

1. Rotation about X - axis:

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

2. Rotation about Y - axis:

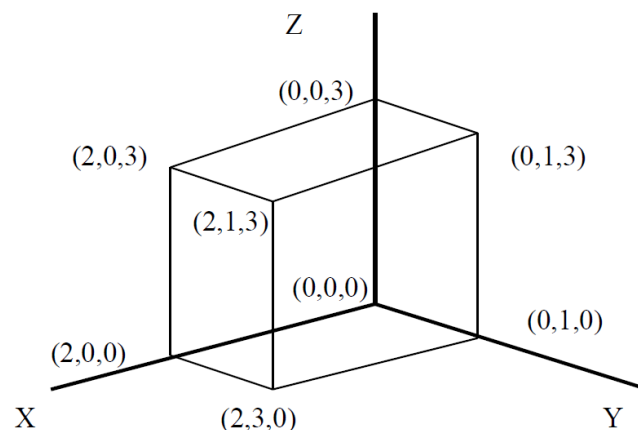
$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Rotation about Z - axis:

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

Example 4: Draw the figure $(0, 0, 0)$, $(0, 1, 0)$, $(0, 1, 3)$, $(0, 0, 3)$, $(2, 0, 0)$, $(2, 1, 0)$, $(2, 0, 3)$, $(2, 1, 3)$, and find: **Not: $\sin(90) = 1$, $\cos(90) = 0$.**

- Translate it to the point $(0, 3, 0)$.
- Scaling 4 times its size about the origin point.
- Rotate its (90°) about the Z – axis.



Solution:

a) Translate it to the point (0, 3, 0).

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 3 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 3 & 0 & 1 \\ 2 & 0 & 3 & 1 \\ 2 & 1 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 3 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 0 & 1 \\ 0 & 3 & 0 & 1 \\ 0 & 4 & 3 & 1 \\ 0 & 3 & 3 & 1 \\ 2 & 3 & 0 & 1 \\ 2 & 6 & 0 & 1 \\ 2 & 3 & 3 & 1 \\ 2 & 4 & 3 & 1 \end{bmatrix}$$

b) Scaling 4 times its size.

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 3 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 3 & 0 & 1 \\ 2 & 0 & 3 & 1 \\ 2 & 1 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 12 & 1 \\ 0 & 0 & 12 & 1 \\ 8 & 0 & 0 & 1 \\ 8 & 12 & 0 & 1 \\ 8 & 0 & 12 & 1 \\ 8 & 4 & 12 & 1 \end{bmatrix}$$

c) Rotate its (90°) about the Z – axis.

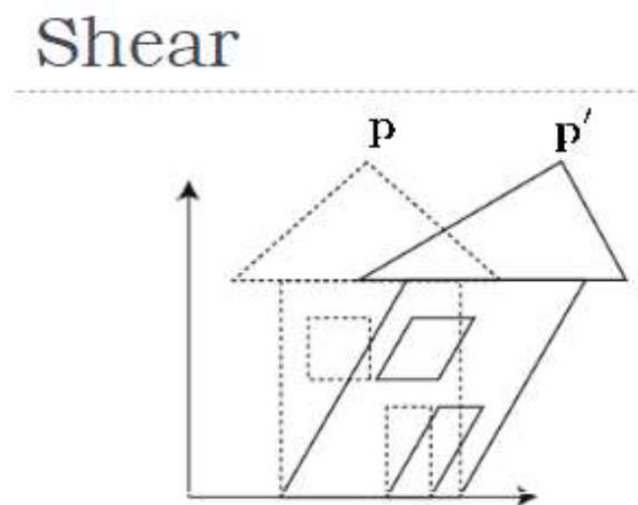
$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 3 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 3 & 0 & 1 \\ 2 & 0 & 3 & 1 \\ 2 & 1 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} \cos(90) & \sin(90) & 0 & 0 \\ -\sin(90) & \cos(90) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 1 \\ -1 & 0 & 3 & 1 \\ 0 & 0 & 3 & 1 \\ 0 & 2 & 0 & 1 \\ -3 & 2 & 0 & 1 \\ 0 & 2 & 3 & 1 \\ -1 & 2 & 3 & 1 \end{bmatrix}$$

H.W 2: Draw the figure A (4 , 4 , 0) , B (-3 , 3 , 4) , C (-2 , 3 , 3) , D (3 , -3 , 4) , E (3 , -2 , 3) and find:

- Translate above shape to point (-3, 4, 3).
- Scaling the figure, twice in X direction, three in Y direction and once in Z direction.
- Rotate the figure (180°) about X – axis.

1.2.4 Shear

We can denote shearing with ‘SH_x,’ ‘SH_y,’ and ‘SH_z.’ These ‘SH_x,’ ‘SH_y,’ ‘SH_z’ are called “**Shearing factor.**” The basic difference between 2D and 3D Shearing is that the 3D plane also includes the z-axis.



We can perform shearing on the object by following three ways-

1. **Shearing along the x-axis:** In this, we can store the x coordinate and only change the y and z coordinate.

We can represent shearing along x-axis by the following equation-

$$x_1 = x_0$$

$$y_1 = y_0 + SH_y \cdot x_0$$

$$z_1 = z_0 + SH_z \cdot x_0$$

3D Shearing Matrix:

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ SH_y & 1 & 0 & 0 \\ SH_z & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{x} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

2. **Shearing along the y-axis:** In this, we can store the y coordinate and only change the x and z coordinate.

We can represent shearing along with y-axis by the following equation-

$$x_1 = x_0 + SH_x \cdot y_0$$

$$y_1 = y_0$$

$$z_1 = z_0 + SH_z \cdot y_0$$

3D Shearing Matrix:

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = \begin{pmatrix} 1 & SH_x & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & SH_z & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{x} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

3. Shearing along with z-axis: In this, we can store the z coordinate and only change the x and y coordinate.

We can represent shearing along with z-axis by the following equation-

$$x_1 = x_0 + SH_x \cdot z_0$$

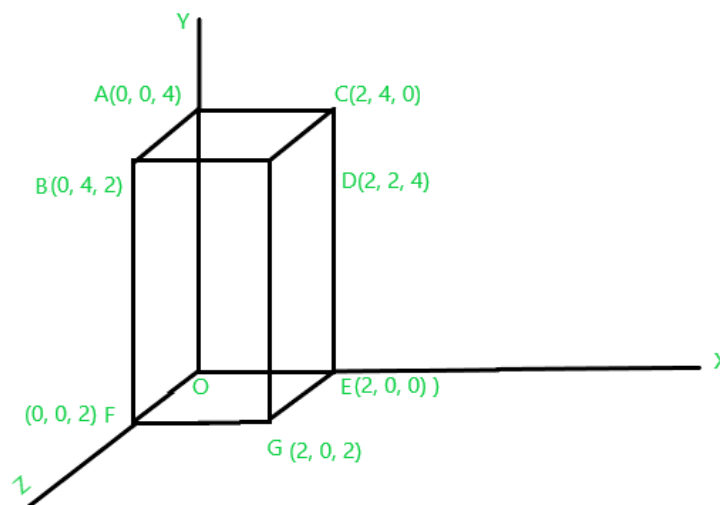
$$y_1 = y_0 + SH_y \cdot z_0$$

$$z_1 = z_0$$

3D Shearing Matrix:

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & SH_x & 0 \\ 0 & 1 & SH_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix}$$

Example 5: Perform Shearing Transformation in the given cuboid (OABCDEFG) along Z-direction if a shearing parameter is as follows $S_x=2$, $S_y=3$.



Solution:

Point O[0, 0, 0] becomes O' after performing Reflection transformation:

$$O'(X Y Z 1) = (0 \ 0 \ 0 \ 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 3 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [0 \ 0 \ 0 \ 1]$$

Point A'[0, 0, 4] becomes A' after performing Reflection transformation:

$$A'(X Y Z 1) = (0 \ 0 \ 4 \ 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 3 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [8 \ 12 \ 4 \ 1]$$

Point B'[0, 4, 2] becomes B' after performing Reflection transformation:

$$B'(X Y Z 1) = (0 \ 4 \ 2 \ 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 3 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [8 \ 10 \ 2 \ 1]$$

Point C'[2, 4, 0] becomes C' after performing Reflection transformation:

$$C'(X Y Z 1) = (2 \ 4 \ 0 \ 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 3 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [2 \ 4 \ 0 \ 1]$$

Point D'[2, 2, 4] becomes D' after performing Reflection transformation:

$$D'(X Y Z 1) = (2 \ 2 \ 4 \ 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 3 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [10 \ 14 \ 4 \ 1]$$

Point E'[2, 0, 0] becomes E' after performing Reflection transformation:

$$E'(X Y Z 1) = (2 \ 0 \ 0 \ 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 3 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [2 \ 0 \ 0 \ 1]$$

Point F'[0, 0, 2] becomes F' after performing Reflection transformation:

$$F'(X Y Z 1) = (0 \ 0 \ 2 \ 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 3 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

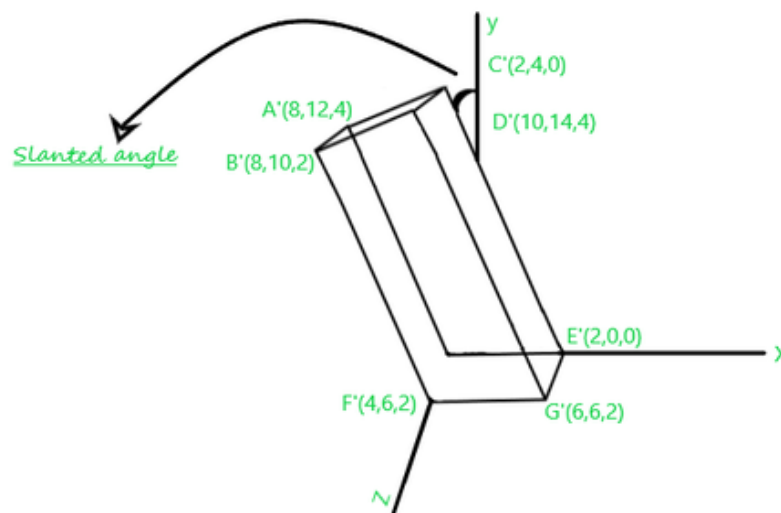
$$= [4 \ 6 \ 2 \ 1]$$

Point G'[2, 0, 2] becomes G' after performing Reflection transformation:

$$G'(X Y Z 1) = (2 \ 0 \ 2 \ 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 3 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [6 \ 6 \ 2 \ 1]$$

Finally, after performing the Shearing transformation on the given cuboid your Fig.1 will look like as below:



H.W 3: A Triangle with (2, 2), (0, 0) and (2, 0). Apply Shearing factor 2 on X-axis and 2 on Y-axis. Find out the new coordinates of the triangle?

1.2.5 Reflection

The Reflection is a mirror image of the original object. We can differentiate 2D and 3D reflection by adding Z-axis. The Z-axis shows the depth of the surface. In the Reflection process, the size of the object does not change.

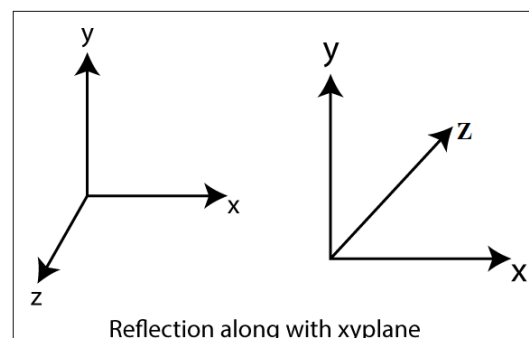
We can represent Reflection by using the following three ways:

1. **Reflection along with xy Plane:** In the xy plane reflection, the value of z is negative.

$$x_1 = x_0$$

$$y_1 = y_0$$

$$z_1 = -z_0$$



Matrix of 3D Reflection:

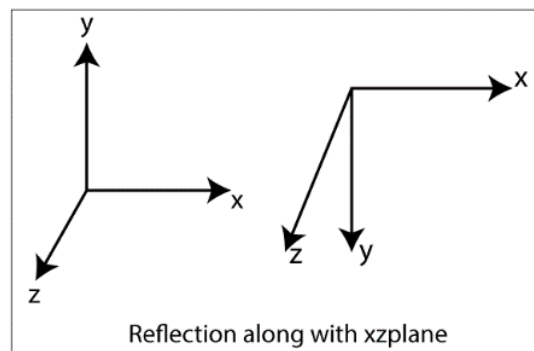
$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{x} \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix}$$

2. Reflection along with xz Plane: In the xz plane reflection the value of y is negative.

$$x_1 = x_0$$

$$y_1 = -y_0$$

$$z_1 = z_0$$

**Matrix of 3D Reflection:**

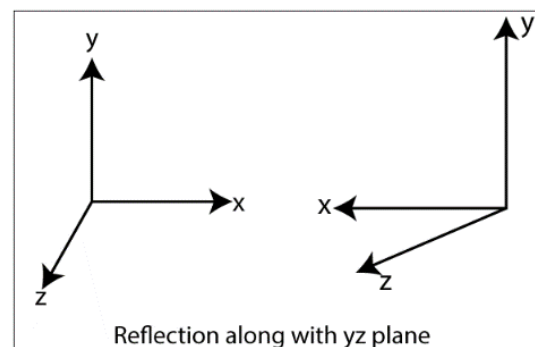
$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{x} \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix}$$

3. Reflection along with yz Plane: In the yz plane reflection the value of x is negative.

$$x_1 = -x_0$$

$$y_1 = y_0$$

$$z_1 = z_0$$

**Matrix of 3D Reflection:**

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{x} \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix}$$

Example 6: A 3D triangle with coordinates points P (4, 5, 2), Q (7, 5, 3), R (6, 7, 4). Apply reflection on xy plane and find the new coordinates of triangle?

Solution:

The initial coordinates of triangle = P (4, 5, 2), Q (7, 5, 3), R (6, 7, 4)

Reflection Plane = xy

Let the new coordinates of triangle = (x₁, y₁, z)

For Coordinate P (4, 5, 2):

$$X_1 = x_0 = 4$$

$$y_1 = y_0 = 5$$

$$z_1 = -z_0 = -2$$

The new coordinates = (4, 5, -2)

For Coordinate Q (7, 5, 3):

$$X_1 = x_0 = 7$$

$$Y_1 = y_0 = 5$$

$$Z_1 = -z_0 = -3$$

The new coordinates = (7, 5, -3)

For Coordinate P (6, 7, 4):

$$X_1 = x_0 = 6$$

$$y_1 = y_0 = 7$$

$$z_1 = -z_0 = -4$$

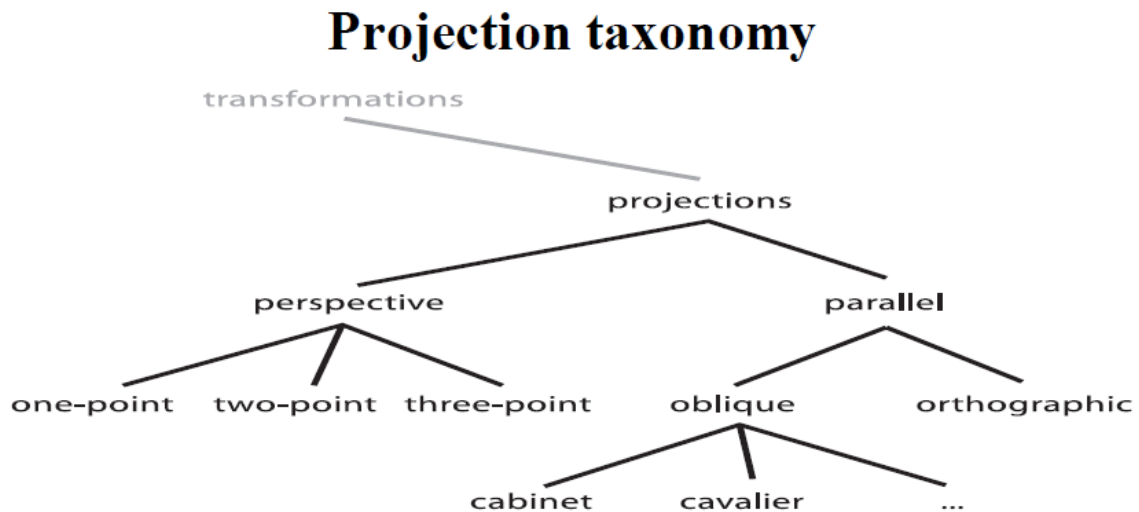
The new coordinates = (6, 7, 4)

Transformation Matrices are a basic tool for transformation. A matrix with n x m dimensions is multiplied with the coordinate of objects. Usually 3 x 3 or 4 x 4 matrices are used for transformation. For example, consider the following matrix for various operation.

$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$	$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$Sh = \begin{bmatrix} 1 & Sh_x^y & Sh_x^z & 0 \\ Sh_y^x & 1 & Sh_y^z & 0 \\ Sh_z^x & Sh_z^y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Translation Matrix	Scaling Matrix	Shear Matrix
$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Rotation Matrix		

1.3 Projection Transformation

A projection is transformations that perform a conversion from three-dimension representation to a two dimension representation. There are two basic types of projections: **Parallel and Perspective**.



Here are some properties of projective transformations:

- ❖ Lines map to lines.
- ❖ Parallel lines *don't* necessarily remain parallel.
- ❖ Ratios are *not* preserved.

1.3.1 Parallel Projections

Parallel projection discards z-coordinate and parallel lines from each vertex on the object are extended until they intersect the view plane. In parallel projection, we specify a direction of projection instead of center of projection. In parallel projection, the distance from the center of projection to project plane is infinite. In this type of projection, we connect the projected vertices by line segments which correspond to connections on the original object. Parallel projections are less realistic, but they are good for exact measurements. In this type of projections, parallel lines remain parallel and angles are not preserved. Various types of parallel projections are shown in the following hierarchy.

Parallel projection pros and cons:

- ❖ Less realistic looking.
- ❖ Good for exact measurements.
- ❖ Parallel lines remain parallel.
- ❖ Angles not (in general) preserved.

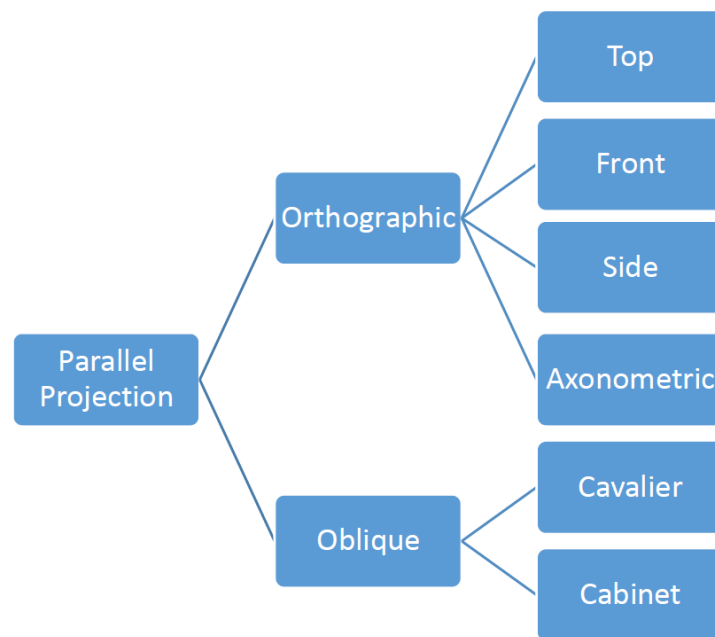


Figure: Types of Parallel Projection

There are two types of parallel projections:

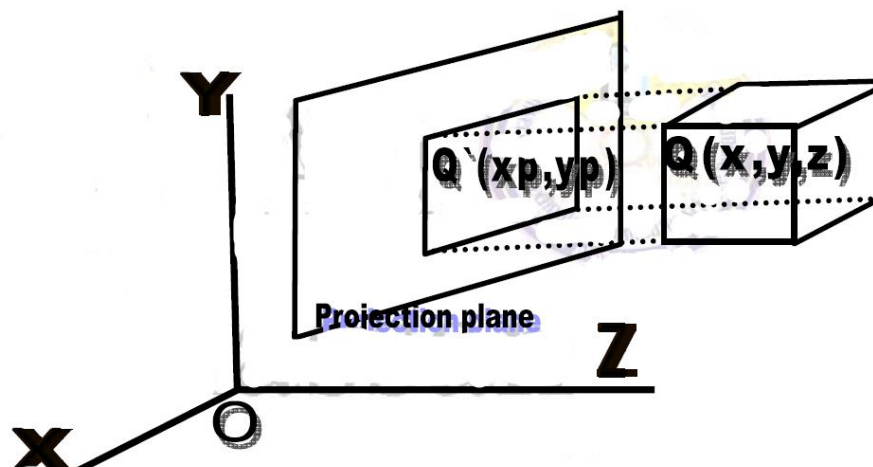
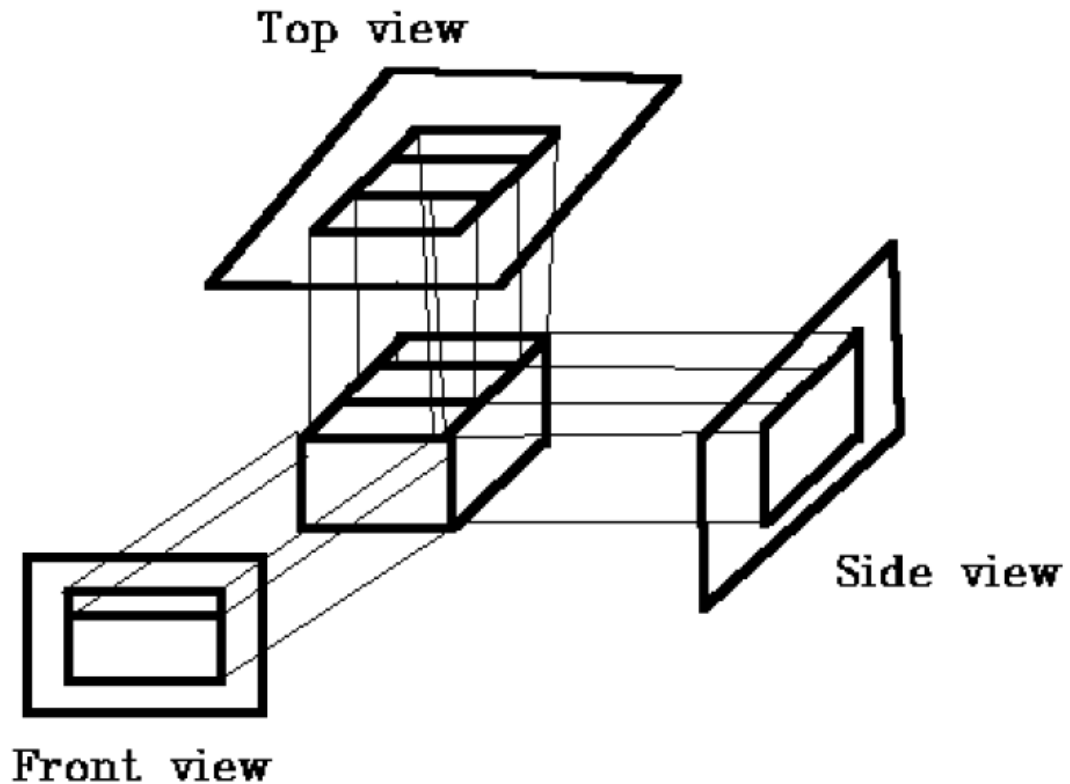
- ❖ Orthographic projection
- ❖ Oblique projection

a. Orthogonal Projection

A parallel projection is to discard one of the coordinate. Like dropping the Z coordinate and project the X, Y, Z coordinate system in to the X, Y plane.

There are three types of orthographic projections:

- Front Projection
- Top Projection
- Side Projection



The projection of a point $Q(x, y, z)$ lying on the cube is point $Q'(x_p, y_p)$ in the x, y plane where a line passing through Q and parallel to the Z -axis intersect the X, Y plane these parallel line called projectors and we get $X_p = X ; Y_p = Y$.

- Straight lines are transformed into straight lines.
- Only endpoints of a line in three-dimension are projected and then draw two-dimensional line between these projected points.
- The major disadvantages of parallel projection is its lack of depth information.

Explanation:

- Let $[x_p \ y_p \ z_p]$ is a vector of the direction of projection.
- The image is to be projected onto the $x \ y$ plane.
- If we have a point on the object at (x_1, y_1, z_1) we wish to determine where the projected point (x_2, y_2) will lie.
- The equation for a line passing through the point (x, y, z) and in the direction of projection

$$X = x_1 + x_p * u$$

$$Y = y_1 + y_p * u$$

$$Z = z_1 + z_p * u$$

If $Z=0$ then $u = -z_1/z_p$

Substituting this into the first two equation:

$$X_2 = x_1 - z_1 (x_p / z_p)$$

$$Y_2 = y_1 - z_1 (y_p / z_p)$$

Written in matrix form we set:

$$[x_2 \ y_2 \ z_2 \ 1] = [x_1 \ y_1 \ z_1 \ 1] * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -x_p/z_p & y_p/z_p & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Example: Let $P(1,0,1)$ be an object point, projected to $P'(x',y',z')$ onto the $z'=0$ plane and the vector is $(1,1,1)$. Find orthogonal projections

b. Oblique Projection

In orthographic projection, the direction of projection is not normal to the projection of plane. In oblique projection, we can view the object better than orthographic projection.

There are two types of oblique projections: Cavalier and 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 $\frac{1}{2}$ their actual length. Both the projections are shown in the following figure:

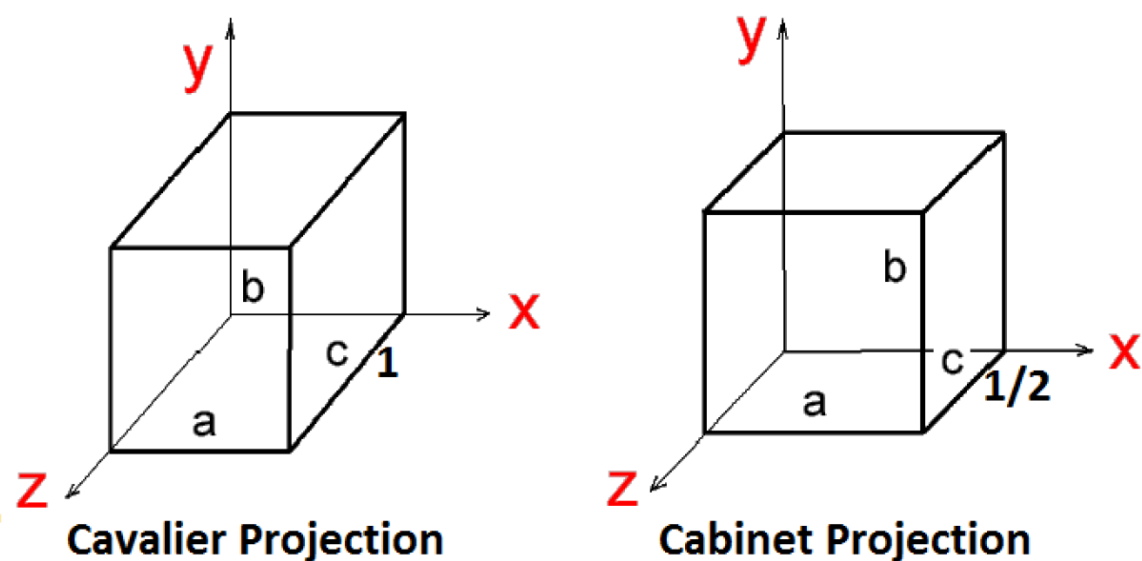


Figure: Cavalier & Cabinet Projection

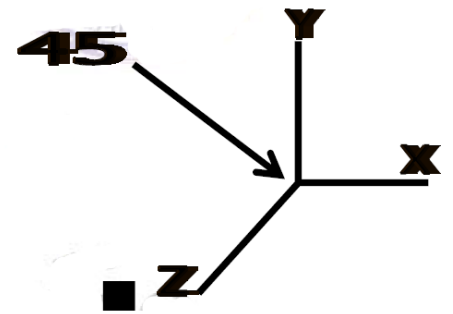
- That show 3D reality by equation:

$$X' = X + (Z * \cos Q) \text{ \& } Y' = Y + (Z * \sin Q) \text{ where } Q \text{ is slope}$$

- Z-Axis of coordinate as following:

$$X' = X + (Z * -0.7) \text{ \& } Y' = Y + (Z * -0.7) \Rightarrow Q = 45$$

$\sin 45 = \cos 45 \approx 0.7$ in three quarter are too negative



- Matrix representation

$$(X' \ Y' \ Z') = (X \ Y \ Z) * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\cos Q & -\sin Q & 1 \end{bmatrix}$$

$$= [X - Z * \cos Q \quad Y - Z * \sin Q \quad 0]$$

if need Distance only add distance as d in

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ d * -\cos Q & d * -\sin Q & 1 \end{bmatrix}$$

$$= [X - d * Z * \cos Q \quad Y - d * Z * \sin Q \quad 0]$$

1.3.2 Perspective Projection

In perspective projection, the distance from the center of projection to project plane is finite and the size of the object varies inversely with distance which looks more realistic. The distance and angles are not preserved and parallel lines do not remain parallel. Instead, they all converge at a single point called **center of projection** or **projection reference point**. There are 3 types of perspective projections which are shown in the following chart.

- **One point** perspective projection is simple to draw.
- **Two point** perspective projection gives better impression of depth.
- **Three point** perspective projection is most difficult to draw.

All three types are equally simple with computer graphics.

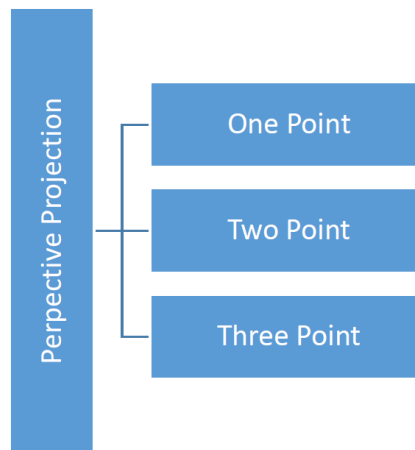


Figure: Types of Perspective Projections

The following figure shows all the three types of perspective projection:

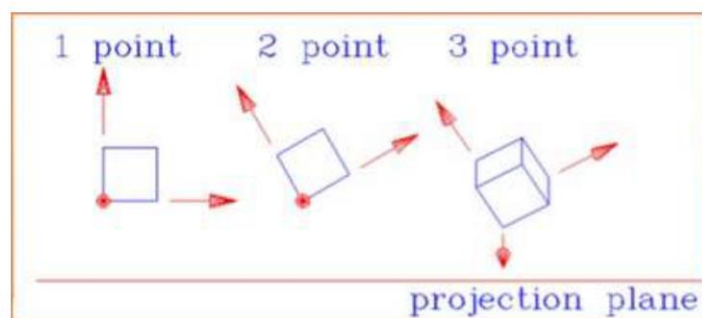


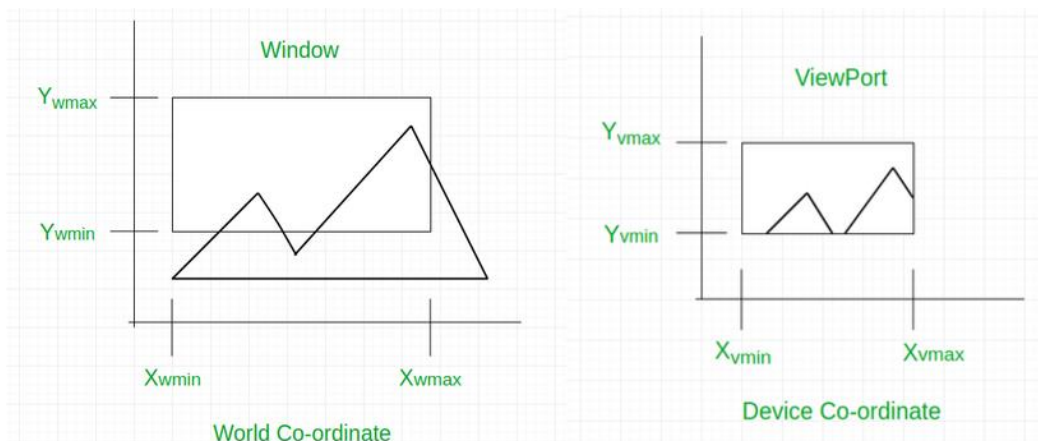
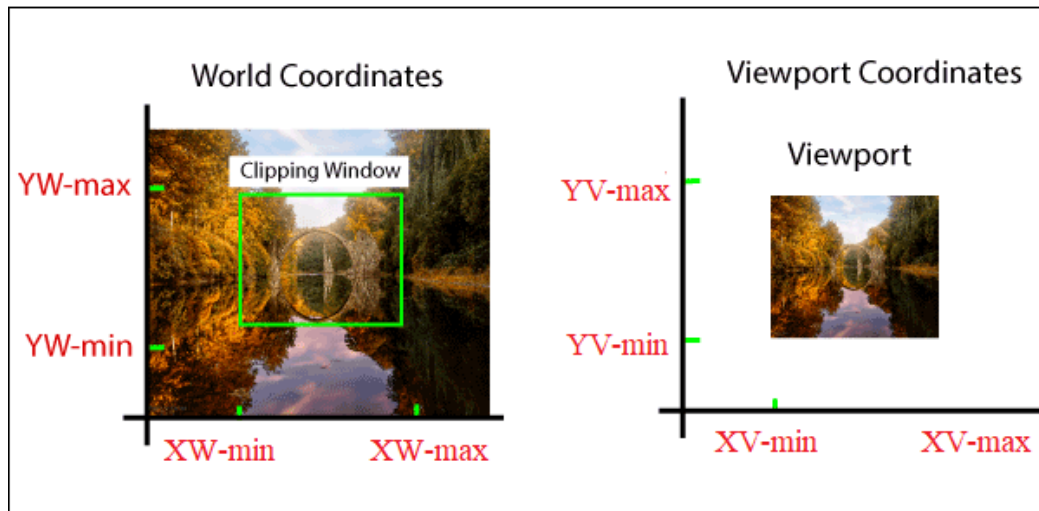
Figure: 1-point, 2-point, 3-point perspective projection

Perspective projections pros and cons:

- ❖ Size varies inversely with distance
- ❖ looks realistic
- ❖ Distance and angles are not (in general) preserved
- ❖ Parallel lines do not (in general) remain parallel

1.4 Window to Viewport Transformation

Window to Viewport Transformation is the process of transforming a 2D world-coordinate objects to device coordinates. Objects inside the world or clipping window are mapped to the viewport which is the area on the screen where world coordinates are mapped to be displayed.



General Terms

- **World coordinate** – It is the Cartesian coordinate w.r.t which we define the diagram, like X_{wmin} , X_{wmax} , Y_{wmin} , Y_{wmax}
- **Device Coordinate** – It is the screen coordinate where the objects is to be displayed, like X_{vmin} , X_{vmax} , Y_{vmin} , Y_{vmax}
- **Window** – It is the area on world coordinate selected for display.
- **ViewPort** – It is the area on device coordinate where graphics is to be displayed

Mathematical Calculation of Window to Viewport

It may be possible that the size of the Viewport is much smaller or greater than the Window. In these cases, we have to increase or decrease the size of the Window according to the Viewport and for this, we need some mathematical calculations.

(x_w, y_w) : A point on Window

(x_v, y_v) : Corresponding point on Viewport

- we have to calculate the point (x_v, y_v)

$$\text{Normalized Point on Window} \left(\frac{X_w - X_{wmin}}{X_{wmax} - X_{wmin}}, \frac{Y_w - Y_{wmin}}{Y_{wmax} - Y_{wmin}} \right)$$

$$\text{Normalized Point on Viewport} \left(\frac{X_v - X_{vmin}}{X_{vmax} - X_{vmin}}, \frac{Y_v - Y_{vmin}}{Y_{vmax} - Y_{vmin}} \right)$$

Now the relative position of the object in Window and Viewport are same.

For x coordinate,

$$\frac{X_w - X_{wmin}}{X_{wmax} - X_{wmin}} = \frac{X_v - X_{vmin}}{X_{vmax} - X_{vmin}}$$

For y coordinate,

$$\frac{Y_w - Y_{wmin}}{Y_{wmax} - Y_{wmin}} = \frac{Y_v - Y_{vmin}}{Y_{vmax} - Y_{vmin}}$$

So, after calculating for x and y coordinate, we get

$$X_v = X_{vmin} + (X_w - X_{wmin}) S_x$$

$$Y_v = Y_{vmin} + (Y_w - Y_{wmin}) S_y$$

where, s_x is scaling factor of x coordinate and s_y is scaling factor of y coordinate

$$S_x = \frac{X_{vmax} - X_{vmin}}{X_{wmax} - X_{wmin}}$$

$$S_y = \frac{Y_{vmax} - Y_{vmin}}{Y_{wmax} - Y_{wmin}}$$

Example 7:

Let's assume,

- For window, $X_{wmin} = 20$, $X_{wmax} = 80$, $Y_{wmin} = 40$, $Y_{wmax} = 80$.
- For viewport, $X_{vmin} = 30$, $X_{vmax} = 60$, $Y_{vmin} = 40$, $Y_{vmax} = 60$.
- Now a point (X_w, Y_w) be $(30, 80)$ on the window. We have to calculate that point on viewport i.e. (X_v, Y_v) .
- First of all, calculate scaling factor of x coordinate S_x and scaling factor of y coordinate S_y using above mentioned formula.
- $S_x = (60 - 30) / (80 - 20) = 30 / 60$
 $S_y = (60 - 40) / (80 - 40) = 20 / 40$

So, now calculate the point on viewport (X_v, Y_v) .

$$X_v = 30 + (30 - 20) * (30 / 60) = 35$$

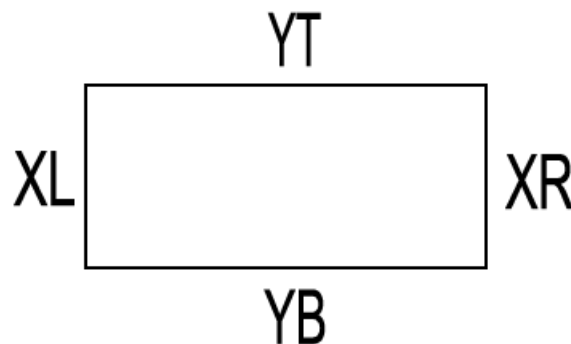
$$Y_v = 40 + (80 - 40) * (20 / 40) = 60$$

So, the point on window $(X_w, Y_w) = (30, 80)$ will be $(X_v, Y_v) = (35, 60)$ on viewport.

1.5 CLIPPING

Clipping is a process which divided each element of the picture into its risible and invisible portions allowing the invisible portion to be discarded. If we wish to display only a portion of the total picture, we use a window to select that portion of the picture which is to be viewed (like clipping or cutting out a picture from a magazine). This is known as clipping. The process of **clipping** determines which elements of the picture lie inside the window and so are visible.

The **clipping window** assumes to be rectangles whose sides are aligned with the coordinate area. The X extent is measured from X min to X max and the Y extent is measured from Y min to Y max.



There are three types of Clipping:

- Point
- Line
- Polygon

1.5.1 Point Clipping

Is to determine if a point (X, Y) is visible or not by a simple pair of inequalities:

$$X_{\text{left}} \leq X \leq X_{\text{right}}$$

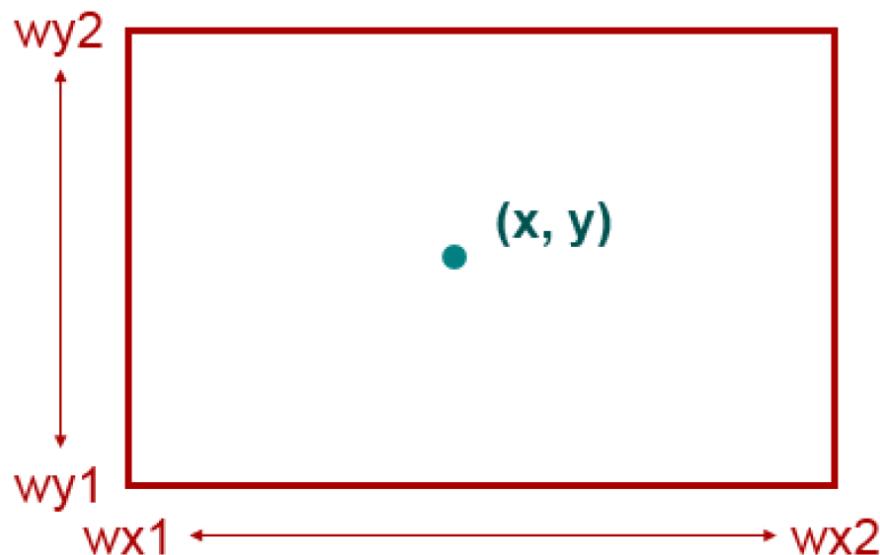
$$Y_{\text{bottom}} \leq Y \leq Y_{\text{top}}$$

Where X_{left} , X_{right} , Y_{bottom} , Y_{top} are the position of the edge of the screen.

These inequalities provides us with a very simple method of clipping pictures on a point-by-point basis.

Clipping a point from a given window is very easy. Consider the following figure, where the rectangle indicates the window. Point clipping tells us whether the given point (X, Y) is within the given window or not; and decides whether we will use the minimum and maximum coordinates of the window.

The X-coordinate of the given point is inside the window, if X lies in between $W_{x1} \leq X \leq W_{x2}$. Same way, Y coordinate of the given point is inside the window, if Y lies in between $W_{y1} \leq Y \leq W_{y2}$.



A point $p(x, y)$ is inside the rectangular window (**visible**) if all the following inequalities are true.

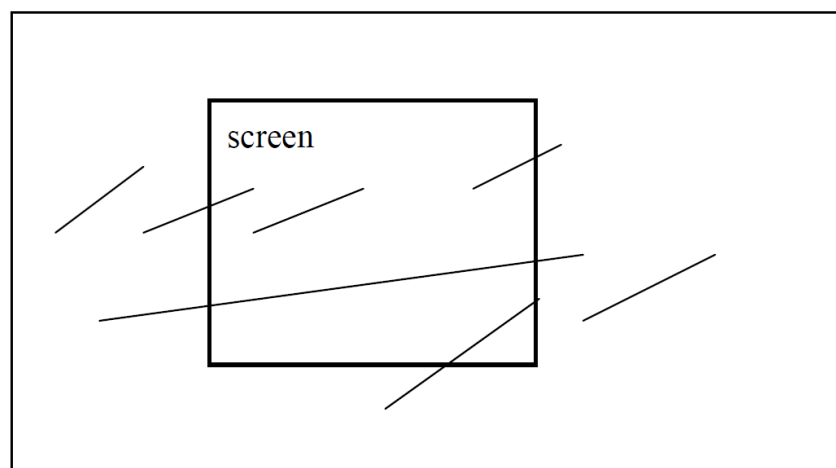
$$X_{\min} \leq X \leq X_{\max} \quad Y_{\min} \leq Y \leq Y_{\max}$$

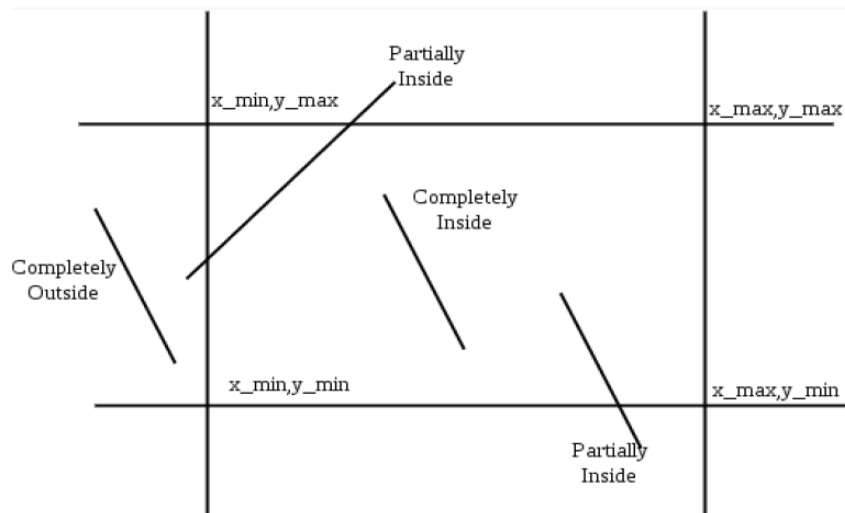
If any of these inequalities is false point P is outside the window and is not displayed (**invisible**).

1.5.2 Line Clipping

It would be quite inappropriate to clip pictures by converting all picture elements into points and using point clipping, the clipping process would take far too long. We must instead attempt to clip large elements of the picture.

The concept of line clipping is same as point clipping. In line clipping, we will cut the portion of line which is **outside of window** and keep only the portion that **is inside the window**. Following Figure shows a number of different lines with respect to the screen:

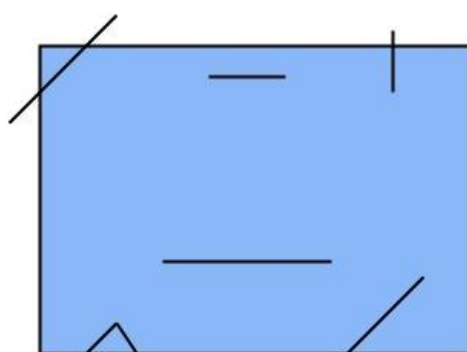
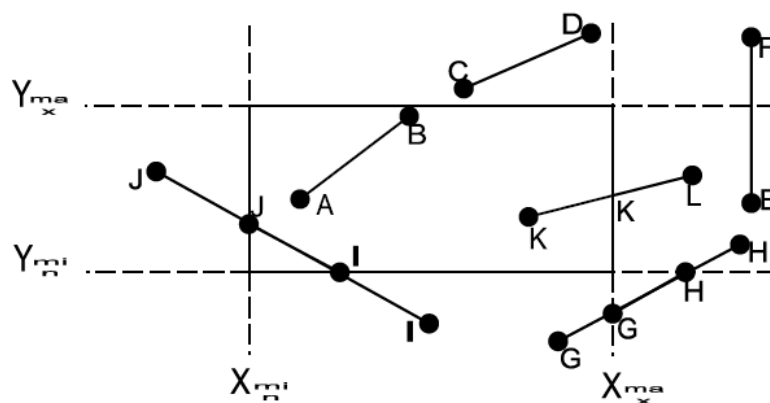




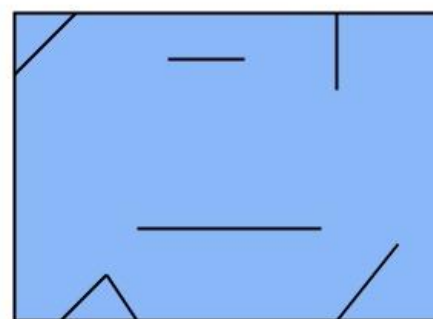
- **Line Segment Clipping**

Line clipping process is into two phases:

- 1: Identify those lines which intersect the window and so need to be clipped.
- 2: Perform the clipping.



Original Picture
or
Before Clipping



After Clipping

All line segments fall into one of the following clipping categories:

1 – Visible:

Both endpoints of the line segment lie within the window (Line AB).

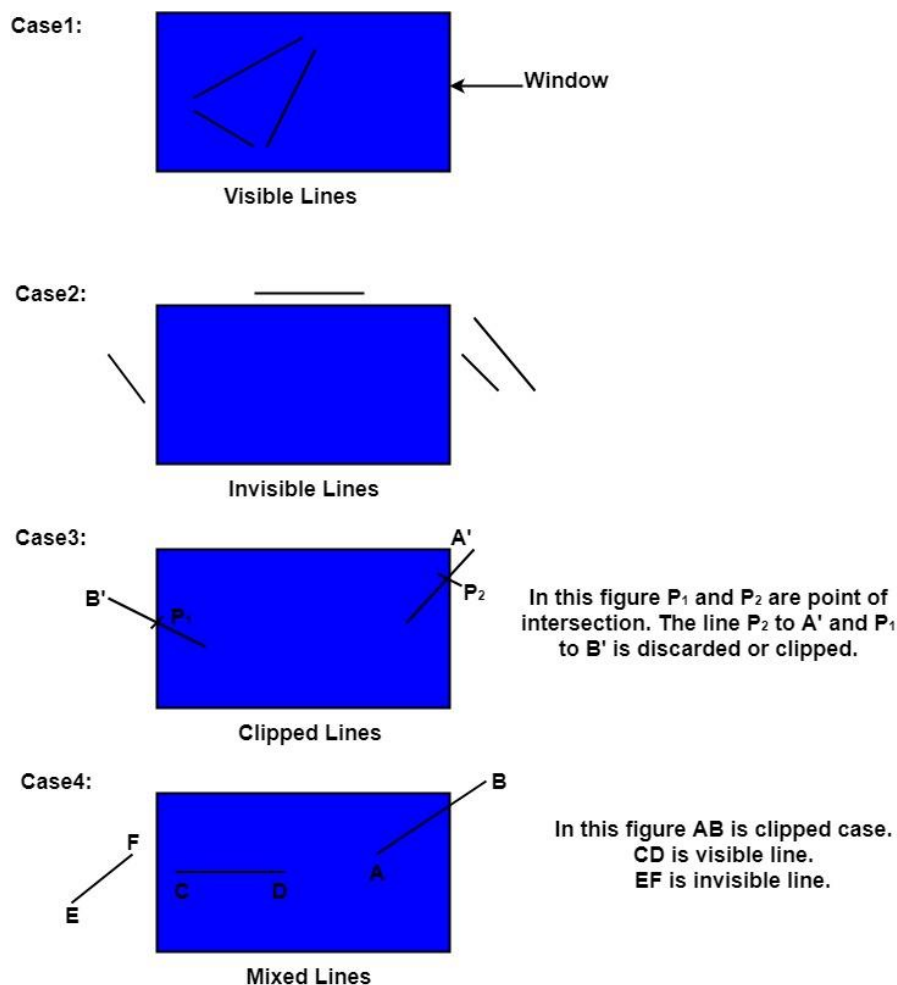
2 – Not visible:

The line segment definitely lies outside the window (Line CD and EF). This will occur if the line segment from (X1, Y1) to (X2, Y2) satisfies any one of the following four inequalities

$$X1, X2 > X_{\max} \quad Y1, Y2 > Y_{\max} \quad X1, X2 < X_{\min} \quad Y1, Y2 < Y_{\min}$$

3 – Clipping candidate:

The line is in neither category 1 nor 2 (Line GH, IJ, and KL)



Simple Visibility Algorithm:

Check for totally visible lines.

If $((x_b < x_L) \text{ OR } (x_b > x_R))$ then 1 .

If $((x_e < x_L) \text{ OR } (x_e > x_R))$ then 1 .

If $((y_b < y_B) \text{ OR } (y_b > y_T))$ then 1 .

If $((y_e < y_B) \text{ OR } (y_e > y_T))$ then 1 .

Draw line

Go to 3

1 - Check for totally invisible lines

if $((x_b < x_L) \text{ AND } (x_e < x_L))$ then 2

if $((x_b > x_L) \text{ AND } (x_e > x_L))$ then 2

if $((y_b < y_L) \text{ AND } (y_e < y_B))$ then 2

if $((y_b > y_T) \text{ AND } (y_e > y_T))$ then 2

The line is partially visibly or diagonally crosses the corner; determine the intersections go to 3

2 line is invisible

3 next line

Cohen–Sutherland Algorithm

The **Cohen–Sutherland algorithm** is a computer-graphics algorithm used for line clipping.

The Cohen–Sutherland algorithm can be used only on a rectangular clip window.

Given a set of lines and a rectangular area of interest, **the task is to remove lines which are outside the area of interest and clip the lines which are partially inside the area.**

Cohen-Sutherland algorithm divides a two-dimensional space into **9 regions** and then efficiently determines the lines and portions of lines that are inside the given rectangular area.

The algorithm can be outlined as follows:

Nine regions are created, eight "outside" regions and one "inside" region.

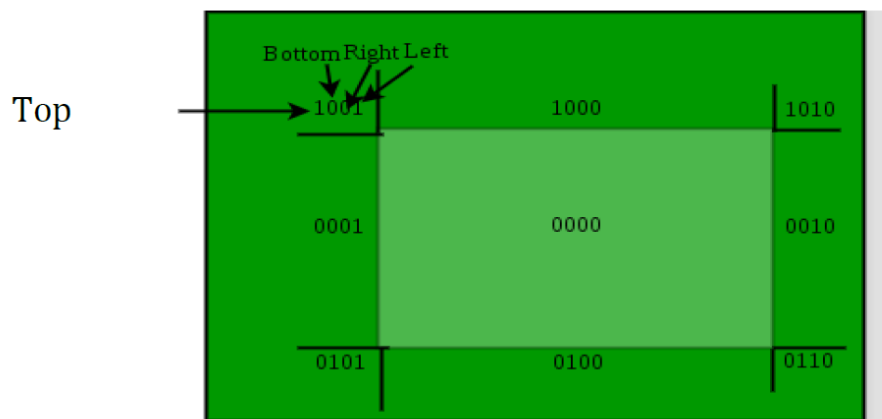
For a given line extreme point (x, y) , we can quickly find its region's four bit code. **Four bit code** can be computed by comparing x and y with four values (x_{\min} , x_{\max} , y_{\min} and y_{\max}).

If x is less than x_{\min} then bit number 1 is set.

If x is greater than x_{\max} then bit number 2 is set.

If y is less than y_{\min} then bit number 3 is set.

If y is greater than y_{\max} then bit number 4 is set



The diagram on the following page is associated with the checking order **TBRL**, corresponding to **Top, Bottom, Right, Left**. We assign a 1-bit where the region is strictly outside the boundary (in the half-plane not containing the window), and a 0-bit where the region is on the same side as the window. Thus, only the window itself is assigned all zeros. Since the high-order bit is associated with the top boundary for example, only the three regions above the window (outside the top boundary) have high-order bit equal to 1.

For example, a point that is below and to the left of the rectangle has a region code of 0101.

There are three possible cases for any given line.

1. **Completely inside the given rectangle:** Bitwise OR of region of two end points of line is 0 (Both points are inside the rectangle). **In other words**, after compute the 4-bit codes for each endpoint. If both codes are 0000, (bitwise OR of the codes yields 0000) line lies completely inside the window: pass the endpoints to the draw routine.

2. **Completely outside the given rectangle:** Both endpoints share at least one outside region which implies that the line does not cross the visible region. (bitwise AND of endpoints $\neq 0$). **In other words**, If both codes have a 1 in the same bit position (bitwise AND of the codes is not 0000), the line lies outside the window. It can be trivially rejected.

3. **Partially inside the window:** Both endpoints are in different regions. In this case, the algorithm finds one of the two points that is outside the rectangular region. The intersection of the line from outside point and rectangular window becomes new corner point and the algorithm repeats.

For Example: Point A has an outcode of 0000 and point D has an outcode of 1001. The logical AND of these outcodes is zero; therefore, the line cannot be trivially rejected. Also, the logical OR of the outcodes is not zero; therefore, the line cannot be trivially accepted. The algorithm then chooses D as the outside point (its outcode contains 1's).

Find Intersection Points

1 - Midpoint Subdivision

The line segment is divided at its midpoint into two smaller line segments. The clipping categories of the two new line segments are then determined. Each segment in category 3 is divided again into smaller segment and categorized. The bisection and categorization process continues until all segments are in category 1 (visible) or category 2 (invisible).

The midpoint coordination (**X_m , Y_m**) of a line segment joining P1 (X1 , Y1) to P2 (X2 , Y2) are given by

$$X_m = \frac{X_1 + X_2}{2} \quad Y_m = \frac{Y_1 + Y_2}{2}$$

2 - Line Intersections and Clipping

We determine the intersection points of the lines in category (3) with the boundaries of the window. The intersection points subdivided the line segment into several smaller line segments which can belong only to category 1 (visible) or category 2 (not visible). The segment in category 1 will be the clipped line segment.

Intersection Points

Intersection points with a clipping boundary can be calculated using the slope-intercept form of the line equation. For a line with endpoint coordinates (x_1, y_1) and (x_2, y_2) , the y coordinate of the **intersection point** with a **vertical boundary** can be obtained with the calculation

$$y = y_1 + m(x - x_1)$$

Where the x value is set either to **xmin** or to **xmax**, and the **slop** of the line is calculated as

$$m = (y_2 - y_1) / (x_2 - x_1).$$

Similarly, if we are looking for the intersection with a **horizontal boundary**, the x coordinate can be calculated as

$$x = x_1 + (y - y_1) / m$$

Note:

1. If the boundary line is **vertical** then

$X=x_{min}$ if the line is left

$X=x_{max}$ if the line is right

$$y = y_1 + m(x - x_1)$$

2. If the boundary line is **horizontal** then

$Y=y_{min}$ if the line is bottom

$Y=y_{max}$ if the line is top

$$x = x_1 + (y - y_1) / m$$

Example 8: Apply the Cohen Sutherland line clipping algorithm to clip the line segment with coordinates (30,60) and (60,25) against the window with $(X_{min}, Y_{min}) = (10, 10)$ and $(X_{max}, Y_{max}) = (50, 50)$.

Solution

Clip bit code

AB 1000 AND

0010

0000 (Partially inside) (clipping)

First, find the slope of line AB from the equation:

$$m = (y_2 - y_1) / (x_2 - x_1)$$

$$m = (25 - 60) / (60 - 30)$$

$$= -35 / 30$$

$$= -1.16$$

Then, we find the coordinate of intersection point from line A A-.

The boundary line A A- is **horizontal**, so $Y_{max} = y = 50$ and calculate x value from this:

$$x = x_1 + (y - y_1) / m$$

$$= 30 + (50 - 60) / -1.16$$

$$= 30 + -10 / -1.16$$

$$= 30 + 8.6$$

$$= 38.6$$

The coordinate of intersection point is **A-(38.6, 50)**.

We find the coordinate of intersection point from line **BB-**.

The boundary line **BB-** is vertical, so **x_{max}=x=50** and calculate y value from this:

$$\begin{aligned} y &= y_1 + m(x - x_1) \\ &= 25 + (-1.16)(50 - 60) \\ &= 25 + 11.6 \\ &= 36.6 \end{aligned}$$

The coordinate of intersection point is **B-(50, 36.6)**.

Example 9: Window is defined A(20,20), B(90,20), C(90,70), D(20,70). Find visible portion of

Line1: P1(10,30), P2(80,90)

Line2: Q1(20,10), Q2(70,60)

Using Cohen Sutherland line clipping algorithm.

Solution

$$X_{\min}=20, X_{\max}=90, y_{\min}=20, y_{\max}=70$$

Clip bit code

P1P2 0001 AND

1000

0000 (**Partially inside**) (**clipping**)

Q1Q2 0101 AND

0000

0000 (**Partially inside**) (**clipping**)

First find the slope of line **P1P2** from the equation:

$$m = (y_2 - y_1) / (x_2 - x_1)$$

$$=(90-30)/(80-10)$$

$$=60/70$$

$$=0.8$$

Then find the coordinate of intersection point from line P1P1-.

The boundary line P1P1- is **vertical**, so **Xmin=x=20** and calculate y value from this:

$$y = y1 + m(x - x1)$$

$$=30+0.8(20-10)$$

$$=30+8=38$$

The coordinate of intersection point **P1-(20,38)**.

Then find the coordinate of intersection point from line P2P2- .

The boundary line P1P1- is **horizontal**, so **ymax=y=70** and find x from this equation:

$$x = x1 + (y - y1) / m$$

$$=80+(70-90)/0.8$$

$$=80+(-20)/0.8$$

$$=80+(-25)=55$$

The coordinate of intersection point **P2-(55,70)**.

Find the slope of second line **Q1Q2**

$$m = (y2 - y1) / (x2 - x1)$$

$$=(60-10)/(70-20)=50/50=1$$

Then find the coordinate of intersection point from line Q1Q1-

The boundary line Q1Q1- is horizontal, so **ymin=y=20** and calculate x value from this:

$$x = x_1 + (y - y_1) / m$$

$$= 20 + (20 - 10) / 1$$

$$= 20 + 10 = 30$$

The coordinate of intersection point is Q1- (30,20)

H.W: Window is defined A(10,20), B(20,20), C(20,10), D(10,10) Find visible portion of line P(15,15), Q(5,5) using Cohen Sutherland line clipping algorithm.

1.5.3 Polygon Clipping

In a polygon, all lines are connected. Lines can be a combination of edges and vertices, which together form a polygon. A polygon refers to a two-dimensional architecture made up of a number of straight lines.

Polygon clipping is a process in which we only consider the part which is inside the view pane or window. We will remove or clip the part that is outside the window. **We will use the Sutherland-Hodgeman polygon clipping algorithm** for [polygon clipping](#). **In this algorithm**, all the vertices of the polygon are clipped against each edge of the clipping window.

First the polygon is clipped against the left edge of the polygon window to get new vertices of the polygon. These new vertices are used to clip the polygon against right edge, top edge, bottom edge, of the clipping window as shown in the following figure.

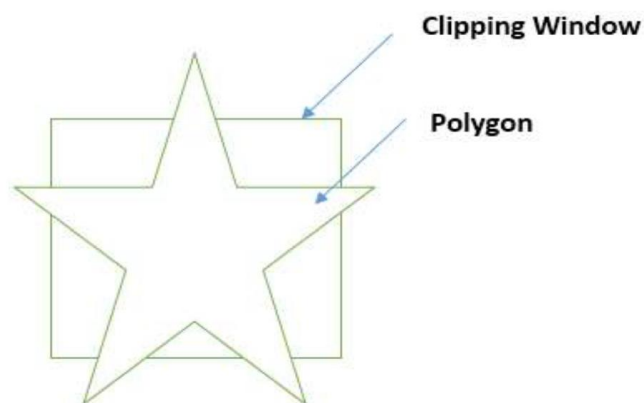


Figure: Polygon before Filling

While processing an edge of a polygon with clipping window, an intersection point is found if edge is not completely inside clipping window and the partial edge from the intersection point to the outside edge is clipped. The following figures show left, right, top and bottom edge clippings:

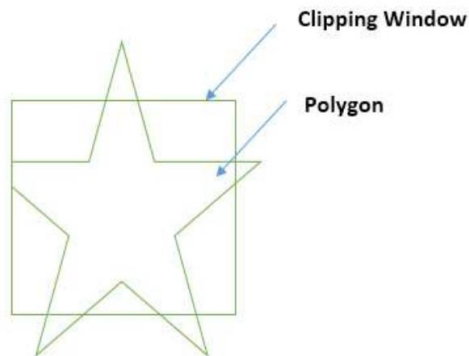


Figure: Clipping Left Edge

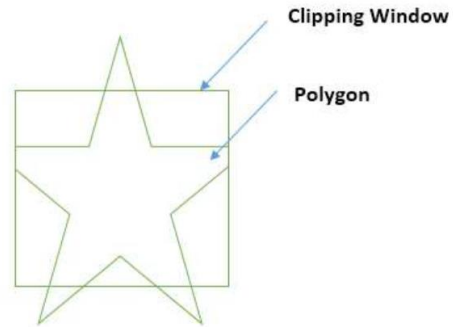


Figure: Clipping Right Edge

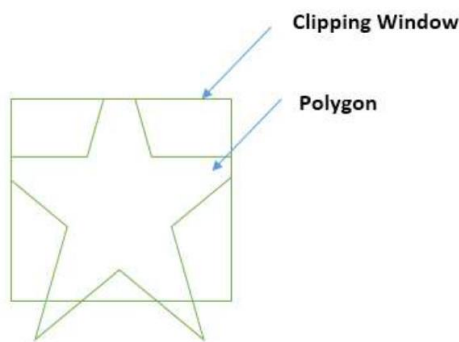


Figure: Clipping Top Edge

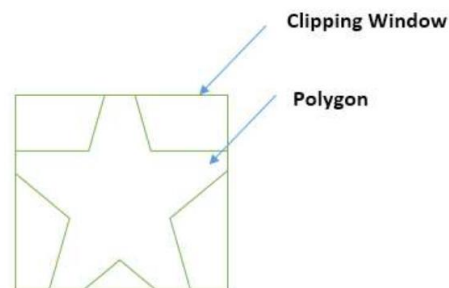


Figure: Clipping Bottom Edge



Polygon Clipping Algorithm

Case - 1:

If the first point and the second point inside the window then store second point.

Case - 2:

If the first point inside and the second point outside the window then store the intersection.

Case - 3:

If the first point and the second point outside the window then nothing.

Case - 4:

If the first point outside and the second point inside the window then store the intersection and the second point.

The input polygon is: V1. V2. V3. V4

If apply the clipping for **Left** Then the result is **V2 P1 P2 V1**

If apply the clipping for **Right** Then the result is **P2 V1 V2 P1**

If apply the clipping for **TOP** Then the result is **P1 P2 V1 V2**

If apply the clipping for **Bottom** Then the result is **V1 V2 P1 P2**

We apply the operation 4 times (Bottom, right, top, left)

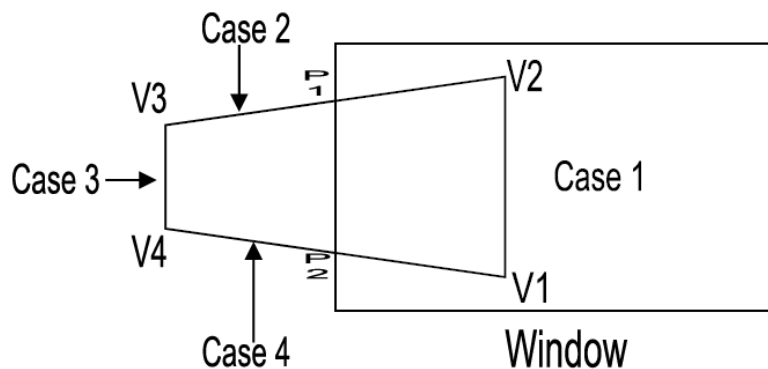
Left: V2 P1 P2 V1

Bottom: P1 P2 V1 V2

Right: P2 V1 V2 P1

Top: V1 V2 P1 P2

- Clips a polygon against each edge of the window.
- For each edge it inputs a list of vertices and outputs a new list of vertices.
- The input list is a sequence of consecutive vertices of the polygon obtained from the previous edge clipping.

Example 10:**Example :**

There are 4 possible cases:

Case 1:

First and second V1, V2 inside the window, V2 is sent to the output list.

Case 2:

First vertex V2 inside and the second vertex V3 outside the window, the intersection point (P1) of the side of the polygon joining the vertices and the edge is added to the output list.

Case 3:

Both vertices V3, V4 outside the window and no point is output.

Case 4:

First vertex V4 outside the window and the second vertex V1 inside the window, the intersection point (P2) and the second vertex V1 are added to the output list.

The result of this left clipping is the transformation of

Input list {V1, V2, V3, V4} to the

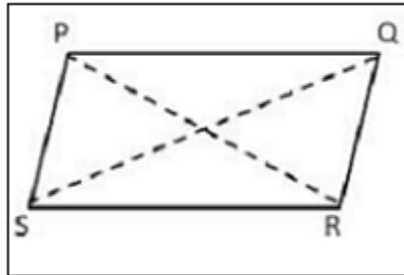
Output list {V1, V2, P1, P2}.

There are two basic types of polygon-

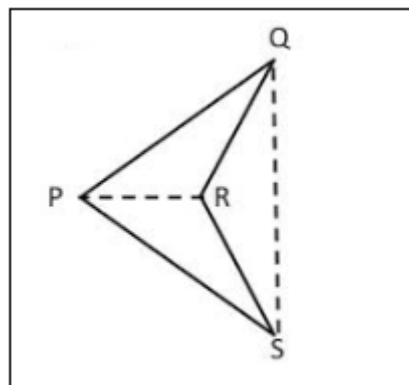
- 1- **Concave Polygon**
- 2- **Convex Polygon**

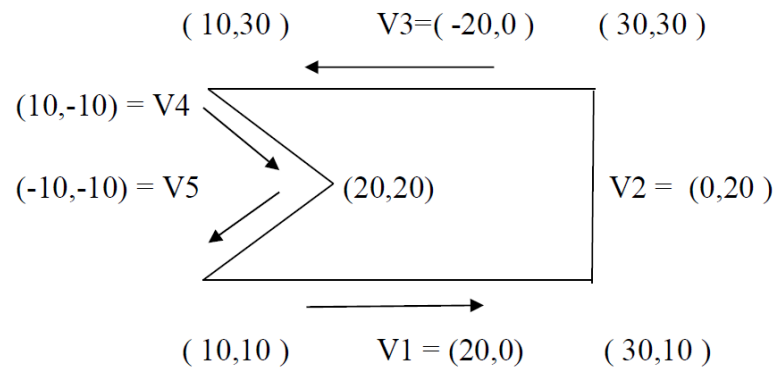
1.5.4 Convex and Concave Window

Concave Polygon: The concave polygon does not have any part of its diagonals in its exterior. In a concave polygon, at least one angle should be greater than 180° (angle $>180^\circ$).



Convex Polygon: The convex polygon has at least one part of diagonal in its exterior. In a convex polygon, all the angles should be less than 180° (angle $<180^\circ$).



Example 11:

$$V1 = (x_2 - x_1, y_2 - y_1) = (30 - 10, 10 - 10) = (20, 0)$$

$$V2 = (x_2 - x_1, y_2 - y_1) = (30 - 30, 30 - 10) = (0, 20)$$

$$V3 = (x_2 - x_1, y_2 - y_1) = (10 - 30, 30 - 30) = (-20, 0)$$

$$V4 = (x_2 - x_1, y_2 - y_1) = (20 - 10, 20 - 30) = (10, -10)$$

$$V5 = (x_2 - x_1, y_2 - y_1) = (10 - 20, 10 - 20) = (-10, -10)$$

$$V1 \cdot V2 = \begin{vmatrix} 20 & 0 \\ 0 & 20 \end{vmatrix} = 400$$

$$V2 \cdot V3 = \begin{vmatrix} 0 & 20 \\ -20 & 0 \end{vmatrix} = 400$$

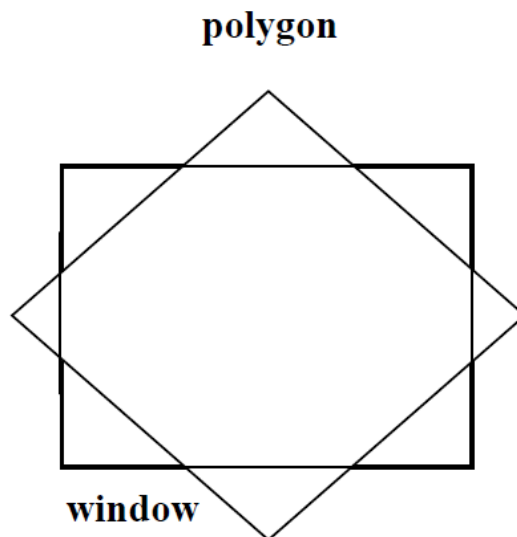
$$V3 \cdot V4 = \begin{vmatrix} -20 & 0 \\ 10 & -10 \end{vmatrix} = 200$$

$$V4 \cdot V5 = \begin{vmatrix} 10 & -10 \\ -10 & -10 \end{vmatrix} = -200$$

$$V5 \cdot V1 = \begin{vmatrix} -10 & -10 \\ 20 & 0 \end{vmatrix} = 200$$

H. W:

1. A window has been set to set-window (2, 5, 3, 7), and a view port has been set to set-viewport (0, 0.5, 0.5, 1). Find the viewing transformation.
2. We have a set window (5, 10, 5, 10), check the points:
 - a) P1 (6, 7).
 - b) P2 (8, 16), were inside the rectangular window or not (By Low)?
3. We have a set window (5, 10, 5, 10), clipping the point:
 - a) P1 (7, 7)
 - b) P2 (18, 20)
 - c) P3 (8, 16)
4. If the clipping window is $XL = 5$, $XR = 10$, $YB = 5$, $YT = 10$. Clip the lines, using simple visibility algorithm:
 - a) $AB = (6, 6) - (8, 7)$
 - b) $EF = (14, 8) - (12, 14)$
 - c) $IJ = (1, 6) - (7, 8)$
5. Clip the following figure using *polygon clipping algorithm*.



6. Determine the window of the end points P1(5,5) , P2(20,5) , P3(18,10) , P4(20,20) , P5(5,20) , P6(10,10).