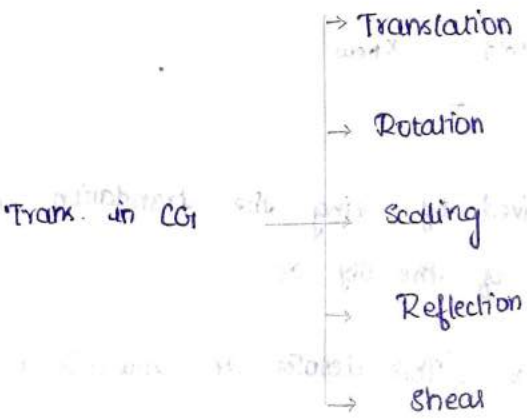
2D Transformations:

* In CG, Trans. is a process of modifying & re-positioning the existing graphics.

* 2D trans. take place in a 2D plane.

* Trans. are helpful in changing the position, size, orientation, shape etc of obj.

Trans. Techniques:



2D Translation:

* In CG, 2D Translation is a process of moving an obj. from 1 position to another in 2D plane.

Consider a pt. obj. O has to be moved from 1 position to another in a 2D plane.

Let -

→ Initial coordinates of obj. O = (x_{old}, y_{old})

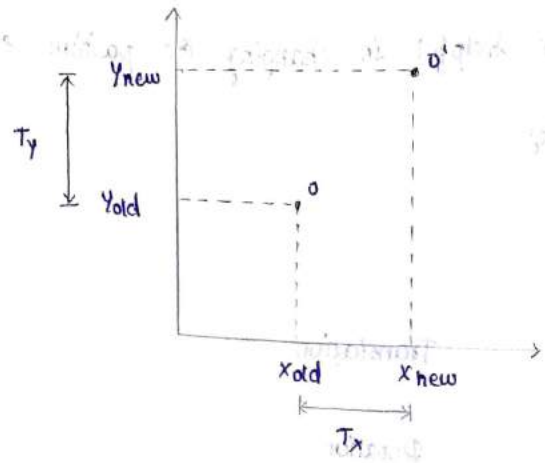
→ New " of obj. O after translation = (x_{new}, y_{new})

→ Translation vector / shift vector = (T_x, T_y)

Given a translation vector (T_x, T_y)

→ T_x defines the distance the x_{old} coordinate has to be moved

→ T_y " " " " y_{old} " " " " "



This translation is acheived by ting the translation coordinates to the old coordinates of the Obj. as -

* $x_{new} = x_{old} + T_x$ (This denotes tr. towards x axis)

* $y_{new} = y_{old} + T_y$ (" " " " y ")

In Matrix form, the above eqns. may be rep. as

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} x_{old} \\ y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

Prob:

Given a circle C with radius 10 & center coordinates (1, 4). Apply the translation with distance 5 towards X axis & 1 towards Y axis. Obtain the new coordinates of C without changing radius.

Sol:

Given:

Old center coordinates of C = $(X_{old}, Y_{old}) = (1, 4)$

Translation vector = $(T_x, T_y) = (5, 1)$

Let new center coordinates of C = (X_{new}, Y_{new})

Applying translation eqns.,

$X_{new} = X_{old} + T_x = 1 + 5 = 6$

$Y_{new} = Y_{old} + T_y = 4 + 1 = 5$

Thus, new center coordinates of C = (6, 5)

Alternatively,

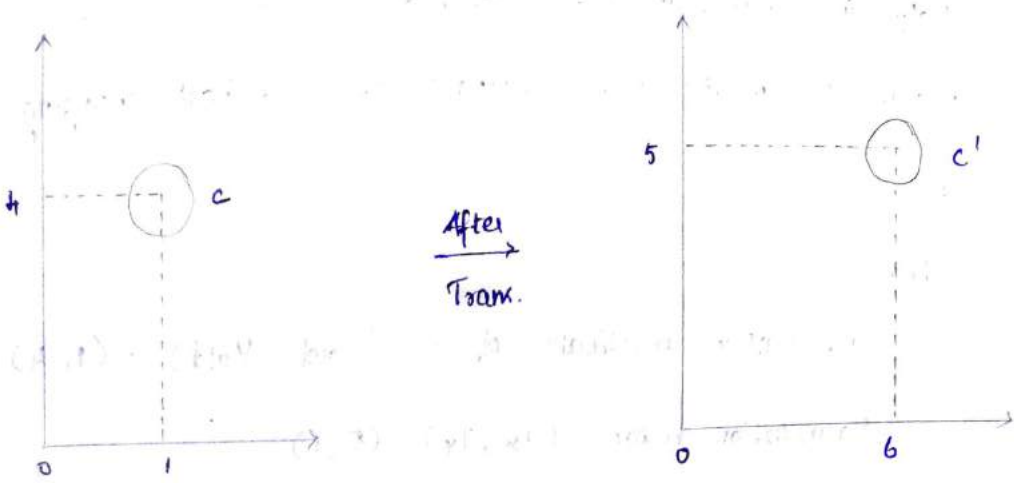
In matrix form, the new center coordinates of C after translation may be obtained as:

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 4 \end{bmatrix} + \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1+5 \\ 4+1 \end{bmatrix}$$

$$= \begin{bmatrix} 6 \\ 5 \end{bmatrix}$$


4
C
0
1
After
Trans.
5
C'
0
6

2D Rotation:

* In CG, 2D Rotation is a process of rotating an obj. with respect to an angle in a 2D plane.

Consider a pt. obj. O has to be rotated from 1 angle to another in a 2D plane.

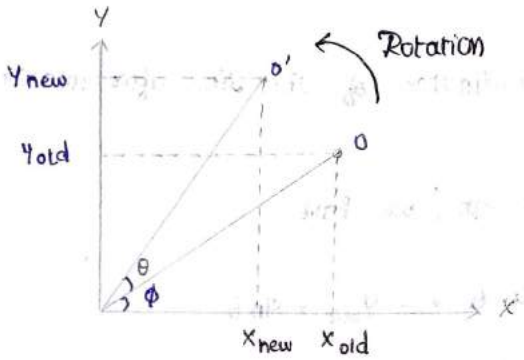
Let

→ Initial coordinates of obj. O = (x_{old}, y_{old})

→ " angle of obj. O with respect to origin = ϕ

→ Rotation angle = θ

→ New coordinates of obj. O after rotation = (x_{new}, y_{new})



This rotation is acheived by using the following rotation eqn.

$$x_{new} = x_{old} \times \cos\theta - y_{old} \times \sin\theta$$

$$y_{new} = x_{old} \times \sin\theta + y_{old} \times \cos\theta$$

In matrix form, the above rotation eqns may be rep. as -

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} x_{old} \\ y_{old} \end{bmatrix}$$

Rotation Matrix

Prob-1:

Given a line seg. with starting pt. as (0,0) & ending pt. as (4,4). Apply 30 deg. rotation anticlockwise direction on the line seg. & find out the new coordinates of the line.

Solution:

We rotate a st. line by its end pts. with same angle. Then, we re-draw a line bet. the new end pts.

Given:

→ Old ending coordinates of the line =

$$(x_{old}, y_{old}) = (4, 4)$$

→ Rotation angle $\theta = 30^\circ$

Let new ending coordinates of the line after rotation $= (x_{new}, y_{new})$

Applying the rotation eqn., we have

$$x_{new} = x_{old} \times \cos\theta - y_{old} \times \sin\theta$$

$$= 4 \times \cos 30^\circ - 4 \times \sin 30^\circ$$

$$= 4 \times (\sqrt{3}/2) - 4 \times (1/2)$$

$$= 2\sqrt{3} - 2$$

$$= 2(\sqrt{3} - 1)$$

$$= 2(1.73 - 1)$$

$$= 1.46$$

$$y_{new} = x_{old} \times \sin\theta + y_{old} \times \cos\theta$$

$$= 4 \times \sin 30^\circ + 4 \times \cos 30^\circ$$

$$= 4 \times (1/2) + 4 \times (\sqrt{3}/2)$$

$$= 2 + 2\sqrt{3}$$

$= 2(1+\sqrt{3})$

$= 2(1+1.73)$

$= 5.46$

Thus, New ending coordinates of the line after rotation = (1.46, 5.46)

Alternatively,

In matrix form, the new ending coordinates of the line after rotation may be obtained as:

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} x_{old} \\ y_{old} \end{bmatrix}$$

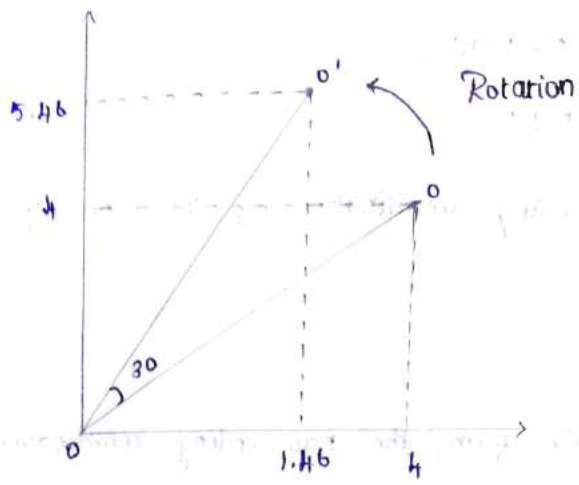
$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} \cos 30 & -\sin 30 \\ \sin 30 & \cos 30 \end{bmatrix} \times \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

$$= \begin{bmatrix} 4 \times \cos 30 - 4 \times \sin 30 \\ 4 \times \sin 30 + 4 \times \cos 30 \end{bmatrix}$$

$$= \begin{bmatrix} 4 \times \cos 30 - 4 \times \sin 30 \\ 4 \times \sin 30 + 4 \times \cos 30 \end{bmatrix}$$

$$= \begin{bmatrix} 1.46 \\ 5.46 \end{bmatrix}$$

New ending coordinates of line after rotation = (1.46, 5.46)



2D Scaling :

* In CG, Scaling is a process of modifying / altering the size of obj.

→ Scaling may be used to increase / reduce the size of obj.

→ Scaling subjects the coordinate pts. of the orginal obj. to change.

→ Scaling factor determines whether the obj. size is to be increased / reduced.

→ If scaling factor > 1, then obj. size is increased.

→ " " " < 1, " " " " reduced.

Consider a pt obj. O has to be scaled in a 2D plane.

Let -

→ Initial coordinates of obj. O - (x_{old}, y_{old})

→ Scaling factor for x-axis = S_x

→ " " " y " = S_y

→ New coordinates of obj. O after scaling = (x_{new}, y_{new})

This scaling is acheived by using following scaling eqns.

$$x_{new} = x_{old} \times S_x$$

$$y_{new} = y_{old} \times S_y$$

In Matrix form,

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} * \begin{bmatrix} x_{old} \\ y_{old} \end{bmatrix}$$

S = Scaling matrix

Prob:

Given a sq. obj. with coordinate pts. A(0, 3), B(3, 3), C(3, 0), D(0, 0). Apply the scaling parameter 2 towards X axis & 3 towards Y axis & obtain the new coordinates of obj.

Sol:

Given

→ Old corner coordinates of sq. = A(0, 3), B(3, 3), C(3, 0) D(0, 0)

→ $S_x = 2$

→ $S_y = 3$

For coordinates A(0, 3)

Let, new coordinates of corner A after scaling = (x_{new}, y_{new})

Applying scaling eqn., we have

→ $x_{new} = x_{old} \times S_y = 0 \times 2 = 0$

→ $y_{new} = y_{old} \times S_y = 3 \times 3 = 9$

New coordinates of corner A after scaling = (0, 9)

For coordinates B(3, 3)

Let new coordinates of corner B after scaling = (x_{new}, y_{new})

Applying scaling eqn.

→ $x_{new} = 3 \times 2 = 6$

→ $y_{new} = 3 \times 3 = 9$

New coordinates of corner B after scaling = (6, 9)

For coordinates C(3, 0)

Let new coordinates of corner C after scaling = (x_{new}, y_{new})

Applying scaling eqn.

→ $x_{new} = 3 \times 2 = 6$

→ $y_{new} = 0 \times 3 = 0$

New coordinates of corner C after scaling = (6, 0)

For coordinates D(0, 0)

Let new coordinates of corner D after scaling = (x_{new}, y_{new})

Applying scaling eqn.

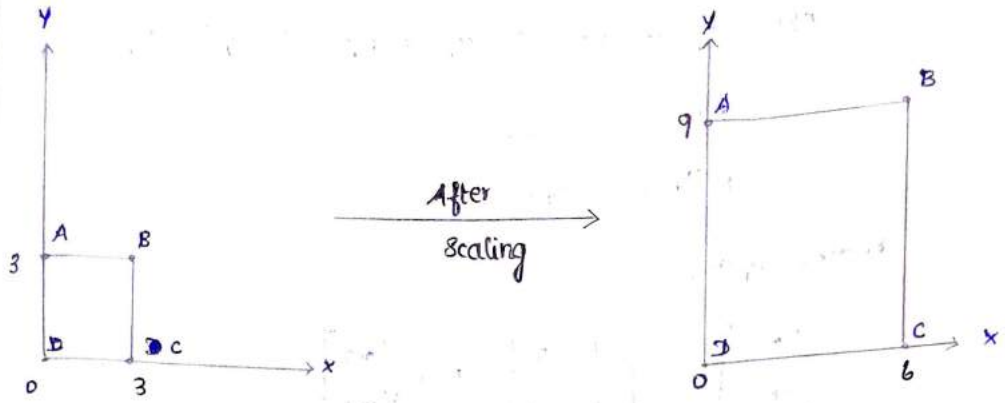
→ $x_{new} = 0 \times 2 = 0$

→ $y_{new} = 0 \times 3 = 0$

New coordinates of corner D after scaling = (0, 0)

Thus, new coordinates of sq. after scaling =

A(0, 9), B(6, 9), C(6, 0), D(0, 0)



17-11-2020 2D Reflection

* Reflection is a kind of rotation where the angle of rotation is 180 deg.

* The reflected obj. is always formed on the other side of mirror.

* The size of reflected obj. is same as the size of original object.

Consider a pt. obj. O has to be reflected in a 2D plane.

Let

→ Initial coordinates of the obj. O = (x_{old}, y_{old})

→ New " of the reflected obj. O after reflection = (x_{New}, y_{new})

Reflection on X-Axis:

This reflection is acheived by using the following ref. eqn.

* $x_{new} = x_{old}$

* $y_{new} = -y_{old}$

In Matrix form

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} * \begin{bmatrix} x_{old} \\ y_{old} \end{bmatrix}$$

Reflection Matrix

Ref. on Y-axis:

This ref. is acheived by using the following ref. eqn.

* $x_{new} = -x_{old}$
* $y_{new} = y_{old}$

Matrix form,

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_{old} \\ y_{old} \end{bmatrix}$$

Ref. Mat.

Prob. 1

Given a triangle with coordinate ~~position~~ pts. A (3, 4), B(6, 4), C(5, 6). Apply the ref. on x axis, & obtain the new coordinates of the obj.

Sol.

Given

→ old corner coordinates of the triangle = A(3, 4), B(6, 4), C(5, 6)

→ Ref. has to be taken on x axis.

For coordinates A(3, 4)

Let the new coordinates of corner A after ref. = (x_{new}, y_{new})

Applying ref. eqns.

→ $x_{new} = x_{old} = 3$

→ $y_{new} = -y_{old} = -4$

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

For coordinates B(6, 4)

Let the new coordinates of corner B after ref. = (x_{new}, y_{new})

Applying ref. eqns.

→ $x_{new} = x_{old} - 6$

→ $y_{new} = -y_{old} = -4$

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

For Coordinates C (5, 6)

Let new coordinates of corner C after ref. = (x_{new}, y_{new})

Applying ref. eqn.

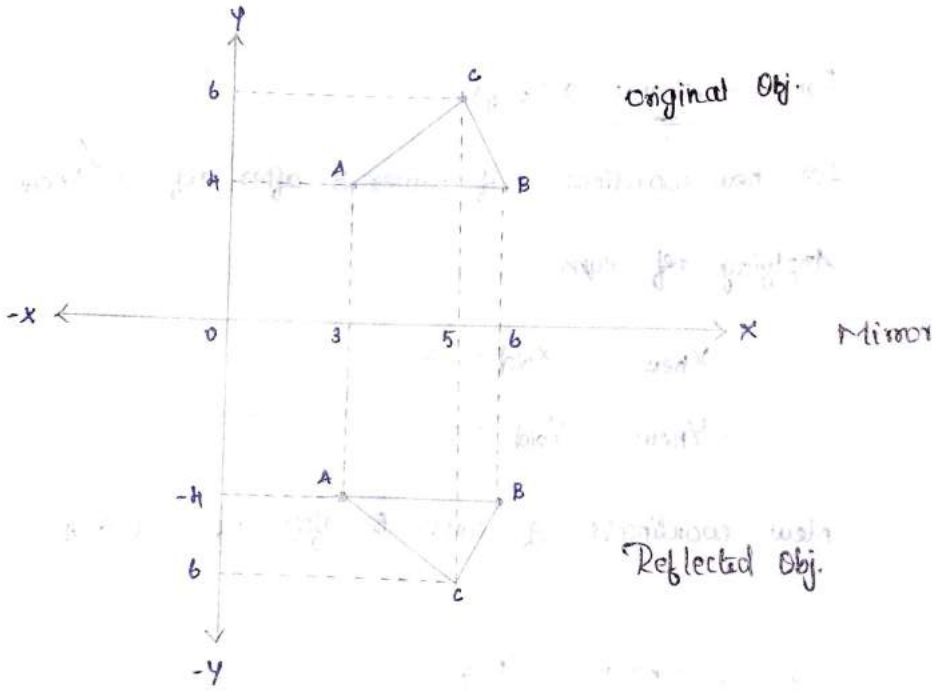
→ $x_{new} = x_{old} = 5$

→ $y_{new} = -y_{old} = -6$

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

Thus, New coordinates of the triangle after ref. =

A(3, -4), B(6, -4), C(5, -6)



Prob. 2:

Given a triangle with coordinate pts. A(3, 4), B(6, 4), C(5, 6). Apply the ref. on the Y axis & obtain the new coordinates of obj.

Sol:

Older coordinates of triangle = A(3, 4), B(6, 4), C(5, 6)

For Coordinates A(3, 4)

Let new coordinates of corner A after ref. = (x_{new}, y_{new})

Applying ref. eqns:

→ $x_{new} = -x_{old} = -3$

→ $y_{new} = y_{old} = 4$

New coordinates of corner A after ref = (-3, 4)

For coordinates B(6, 4)

Let new coordinates of corner B after ref. = (x_{new}, y_{new})

Applying ref. eqns.

→ $x_{new} = -x_{old} = -6$

→ $y_{new} = y_{old} = 4$

New coordinates of corner B after ref. = (-6, 4)

For coordinates C(5, 6)

Let new coordinates of corner C after ref. = (x_{new}, y_{new})

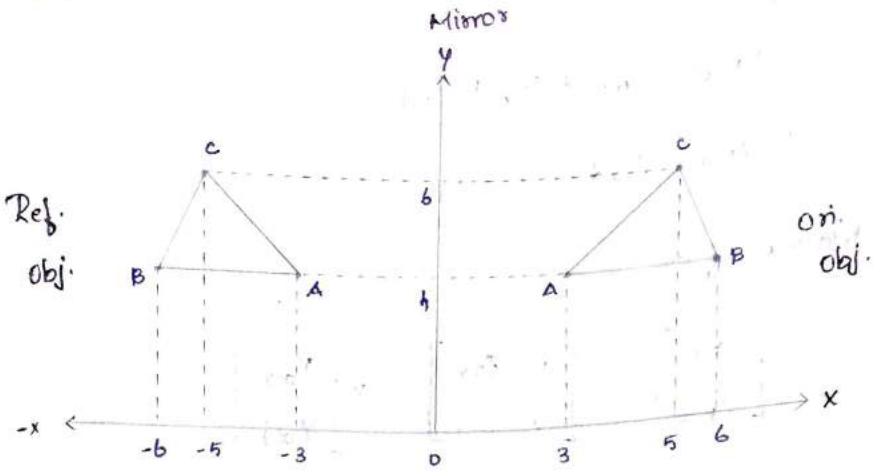
Applying ref. eqns.

→ $x_{new} = -x_{old} = -5$

→ $y_{new} = y_{old} = 6$

New coordinates of corner C after ref. = (-5, 6)

Thus, New coordinates of triangle after ref. = A(-3, 4) B(-6, 4) C(-5, 6)



2D Shearing:

* In CG, 2D shearing is an ideal technique to change the shape of an existing obj. in a 2D plane.

* In a 2D plane, the obj. size can be changed along x direction as well as y direction. So, there are two versions of shearing:

Versions of Shearing

Shearing in x direction | Shearing in y direction

Consider a pt. obj. O has to be sheared in a 2D plane.

Let

→ Initial coordinates of obj. O = (x_{old}, y_{old})

→ Shearing parameter towards x direction = Sh_x

→ " " " y " = Sh_y

→ New coordinates of obj. O after shearing = (x_{new}, y_{new})

Shearing in X Axis:

Shearing in X Axis is achieved by using the following shearing eqns.

→ $X_{new} = X_{old} + Sh_x \times Y_{old}$

→ $Y_{new} = Y_{old}$

Matrix form

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1 & Sh_x \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Shearing Mat. (in X axis)

Shearing in Y axis:

- Shearing in Y axis is ~~achei~~ achieved by using the following shearing eqns.

→ $X_{new} = X_{old}$

→ $Y_{new} = Y_{old} + Sh_y \times X_{old}$

Matrix form

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ Sh_y & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Shearing Mat. (in Y axis)

Prob 1:

Given a triangle with pts. (1,1), (0,0) & (1,0). Apply shear parameter 2 on X axis & 2 on Y axis & find out the new coordinates of the obj.

Given

→ Old corner coordinates of the triangle = A(1,1) B(0,0) C(1,0)

→ Shearing parameter towards x direction (Sh_x) = 2

→ " " " y " (Sh_y) = 2

Shearing in X Axis:

For coordinates of corner A(1,1)

Let new coordinates of corner A after shearing = (x_{new}, y_{new})

Applying shearing eqns.

→ $x_{new} = x_{old} + Sh_x \times y_{old} = 1 + 2 \times 1 = 3$

→ $y_{new} = y_{old} = 1$

New coordinates of corner A after shearing = (3, 1)

For coordinates B(0,0)

Let new coordinates of corner B after shearing = (x_{new}, y_{new})

Applying shearing eqns.

→ $x_{new} = x_{old} + Sh_y \times y_{old} = 0 + 2 \times 0 = 0$

→ $y_{new} = y_{old} = 0$

New coordinates of corner B after shearing = (0, 0)

For coordinates C(1,0)

Let new coordinates of corner C after shearing = (x_{new}, y_{new})

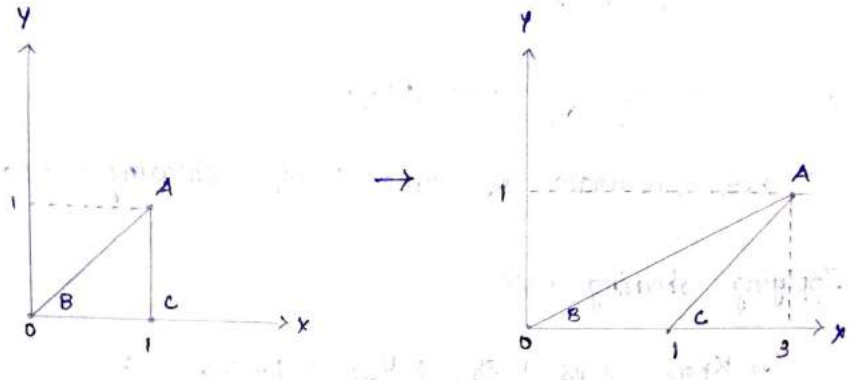
Applying shearing eqns.

$x_{new} = x_{old} + Sh_x \times y_{old} = 1 + 2 \times 0 = 1$

→ $y_{new} = y_{old} = 0$

New coordinates of coordinates of corner C after shearing = (1, 0)

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



Shearing in X Axis

Shearing in Y axis:

For Coordinates A (1, 1)

Let new coordinates of corner A after shearing = (X_{new}, Y_{new})

Applying Shearing eqns.

$\rightarrow X_{new} = X_{old} = 1$

$\rightarrow Y_{new} = Y_{old} + Sh_y \times X_{old} = 1 + 2 \times 1 = 3$

New coordinates of corner A after shearing = (1, 3)

For Coordinates B (0, 0)

Let new coordinates of corner B after shearing = (X_{new}, Y_{new})

Applying Shearing eqns.

$X_{new} = X_{old} = 0$

$Y_{new} = Y_{old} + Sh_y \times X_{old} = 0 + 2 \times 0 = 0$

New coordinates of corner B after shearing = (0, 0)

For coordinates C (1, 0)

Let new coordinates of corner C after shearing = (x_{new}, y_{new})

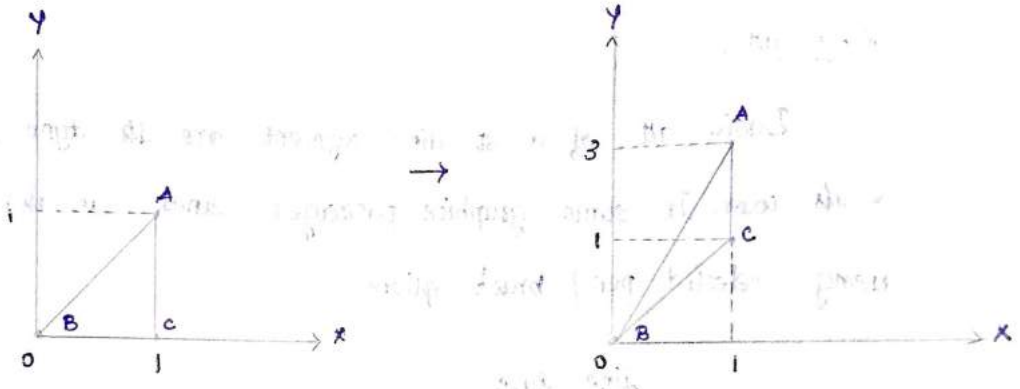
Applying shearing eqns.

→ $x_{new} = x_{old} = 1$

→ $y_{new} = y_{old} + sh_y \times x_{old} = 0 + 2 \times 1 = 2$

New coordinates of corner C after shearing : (1, 2)

Thus, new coordinates of triangle after shearing in Y axis = A (1, 3), B (0, 0), C (1, 2).



Shearing in Y axis

UNIT-2

2D Transformation

Changing Position, shape, size, or orientation of an object on display is known as transformation.

Basic Transformation

- Basic transformation includes three transformations **Translation**, **Rotation**, and **Scaling**.
- These three transformations are known as basic transformation because with combination of these three transformations we can obtain any transformation.

Translation

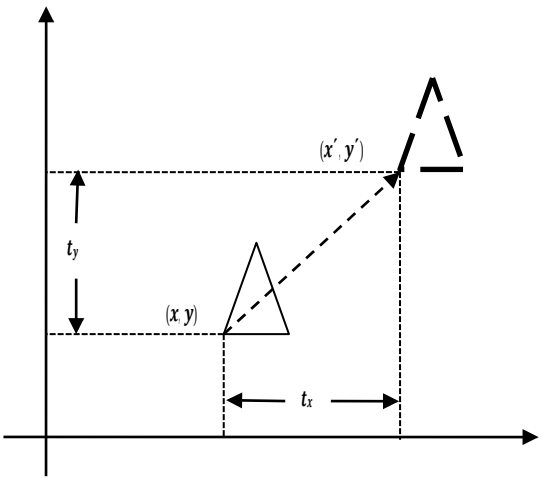


Fig. 3.1: - Translation.

- It is a transformation that used to reposition the object along the straight line path from one coordinate location to another.
- It is rigid body transformation so we need to translate whole object.
- We translate two dimensional point by adding translation distance t_x and t_y to the original coordinate position (x, y) to move at new position (x', y') as:

$x' = x + t_x$ **&** $y' = y + t_y$

- Translation distance pair $(t_x t_y)$ is called a **Translation Vector** or **Shift Vector**.
- We can represent it into single matrix equation in column vector as;

$$P' = P + T$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

- We can also represent it in row vector form as:

$$P' = P + T$$

$$[x' \quad y'] = [x \quad y] + [t_x \quad t_y]$$

- Since column vector representation is standard mathematical notation and since many graphics package like **GKS** and **PHIGS** uses column vector we will also follow column vector representation.
- **Example**: - Translate the triangle [A (10, 10), B (15, 15), C (20, 10)] 2 unit in x direction and 1 unit in y direction.

We know that

$P' = P + T$

$$P' = [P] + \quad t_x \left[t_y \right]$$

For point (10, 10)

$A' = \begin{bmatrix} 10 \\ 10 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

$A' = \begin{bmatrix} 12 \\ 11 \end{bmatrix}$

For point (15, 15)

$B' = \begin{bmatrix} 15 \\ 15 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

$B' = \begin{bmatrix} 17 \\ 16 \end{bmatrix}$

For point (10, 10)

$C' = \begin{bmatrix} 20 \\ 10 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

$C' = \begin{bmatrix} 22 \\ 11 \end{bmatrix}$

- Final coordinates after translation are [A' (12, 11), B' (17, 16), C' (22, 11)].

Rotation

- It is a transformation that used to reposition the object along the circular path in the XY - plane.
- To generate a rotation we specify a rotation angle $\boldsymbol{\theta}$ and the position of the **Rotation Point** (**Pivot Point**) (x_r, y_r) about which the object is to be rotated.
- Positive value of rotation angle defines counter clockwise rotation and negative value of rotation angle defines clockwise rotation.
- We first find the equation of rotation when pivot point is at coordinate origin $(0, 0)$.

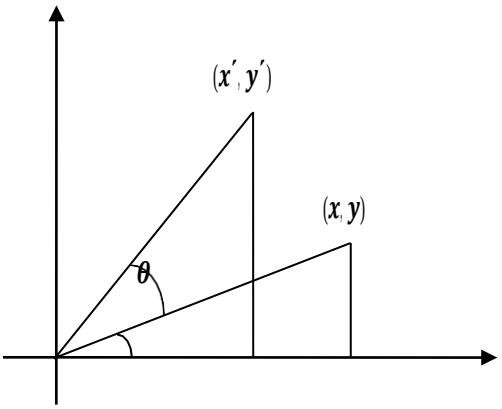


Fig. 3.2: - Rotation.

- From figure we can write.

$x = r\cos\emptyset$

$y = r\sin\emptyset$

and

$x' = r\cos(\theta + \emptyset) = r\cos\emptyset\cos\theta - r\sin\emptyset\sin\theta$

$y' = r\sin(\emptyset + \theta) = r\cos\emptyset\sin\theta + r\sin\emptyset\cos\theta$

- Now replace $r\cos\emptyset$ with x and $r\sin\emptyset$ with y in above equation.

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

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

- We can write it in the form of column vector matrix equation as;

$P' = R \cdot P$

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

- Rotation about arbitrary point is illustrated in below figure.

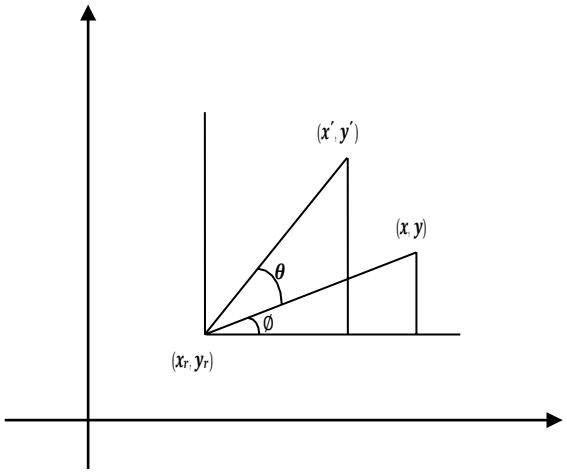


Fig. 3.3: - Rotation about pivot point.

- Transformation equation for rotation of a point about pivot point (x_r, y_r) is:

$$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta$$
$$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta$$

- These equations are differing from rotation about origin and its matrix representation is also different.
- Its matrix equation can be obtained by simple method that we will discuss later in this chapter.
- Rotation is also rigid body transformation so we need to rotate each point of object.
- **Example**: - Locate the new position of the triangle [A (5, 4), B (8, 3), C (8, 8)] after its rotation by 90° clockwise about the origin.

As rotation is clockwise we will take $\theta = -90^\circ$.

$$P' = R \cdot P$$
$$P' = \begin{bmatrix} \cos(-90) & -\sin(-90) \\ \sin(-90) & \cos(-90) \end{bmatrix} \begin{bmatrix} 5 & 8 & 8 \\ 4 & 3 & 8 \end{bmatrix}$$
$$P' = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 5 & 8 & 8 \\ 4 & 3 & 8 \end{bmatrix}$$
$$P' = \begin{bmatrix} 4 & 3 & 8 \\ -5 & -8 & -8 \end{bmatrix}$$

- Final coordinates after rotation are [A' (4, -5), B' (3, -8), C' (8, -8)].

Scaling

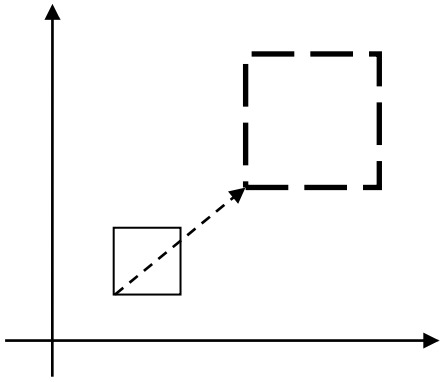


Fig. 3.4: - Scaling.

- It is a transformation that used to alter the size of an object.
- This operation is carried out by multiplying coordinate value (x, y) with scaling factor (s_x, s_y) respectively.
- So equation for scaling is given by:

$x' = x \cdot s_x$

$y' = y \cdot s_y$

- These equation can be represented in column vector matrix equation as:

$P' = S \cdot P$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

- Any positive value can be assigned to (s_x, s_y).
- Values less than 1 reduce the size while values greater than 1 enlarge the size of object, and object remains unchanged when values of both factor is 1.
- Same values of s_x and s_y will produce **Uniform Scaling**. And different values of s_x and s_y will produce **Differential Scaling**.
- Objects transformed with above equation are both scale and repositioned.
- Scaling factor with value less than 1 will move object closer to origin, while scaling factor with value greater than 1 will move object away from origin.
- We can control the position of object after scaling by keeping one position fixed called **Fix point** (x_f, y_f) that point will remain unchanged after the scaling transformation.

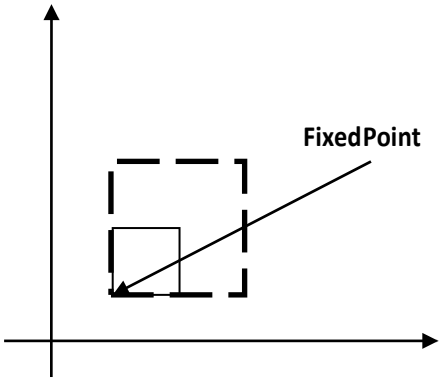


Fig. 3.5: - Fixed point scaling.

- Equation for scaling with fixed point position as (x_f, y_f) is:

$x' = x_f + (x - x_f)s_x$ $\qquad$ $y' = y_f + (y - y_f)s_y$

$x' = x_f + xs_x - x_f s_x$ $\qquad$ $y' = y_f + ys_y - y_f s_y$

$x' = xs_x + x_f(1 - s_x)$ $\qquad$ $y' = ys_y + y_f(1 - s_y)$

- Matrix equation for the same will discuss in later section.
- Polygons are scaled by applying scaling at coordinates and redrawing while other body like circle and ellipse will scale using its defining parameters. For example ellipse will scale using its semi major axis, semi minor axis and center point scaling and redrawing at that position.
- **Example**: - Consider square with left-bottom corner at (2, 2) and right-top corner at (6, 6) apply the transformation which makes its size half.

As we want size half so value of scale factor are $s_x = 0.5$, $s_y = 0.5$ and Coordinates of square are [A (2, 2), B (6, 2), C (6, 6), D (2, 6)].

$P' = S \cdot P$

$P' = [S_x \quad 0][2 \quad 6 \quad 6\ 2]$

$0 \quad s_y\ 2 \quad 2 \quad 6\ 6$

$P' = [0.5 \quad 0][2 \quad 6 \quad 6 \quad 2]$

$0 \quad 0.5\ 2 \quad 2 \quad 6\ 6$

$P' = [1 \quad 3 \quad 3 \quad 1]$

$1 \quad 1 \quad 3 \quad 3$

- Final coordinate after scaling are [A' (1, 1), B' (3, 1), C' (3, 3), D' (1, 3)].

Matrix Representation and homogeneous coordinates

- Many graphics application involves sequence of geometric transformations.
- For example in design and picture construction application we perform Translation, Rotation, and scaling to fit the picture components into their proper positions.
- For efficient processing we will reformulate transformation sequences.
- We have matrix representation of basic transformation and we can express it in the general matrix form as:

$\boldsymbol{P' = M_1 \cdot P + M_2}$

Where $\boldsymbol{P}$ and $\boldsymbol{P'}$ are initial and final point position, $\boldsymbol{M_1}$ contains rotation and scaling terms and $\boldsymbol{M_2}$ contains translation al terms associated with pivot point, fixed point and reposition.

- For efficient utilization we must calculate all sequence of transformation in one step and for that reason we reformulate above equation to eliminate the matrix addition associated with translation terms in matrix $\boldsymbol{M_2}$.
- We can combine that thing by expanding 2X2 matrix representation into 3X3 matrices.
- It will allows us to convert all transformation into matrix multiplication but we need to represent vertex position (x, y) with homogeneous coordinate triple (x_h, y_h, h) Where $x = \frac{x_h}{h}$, $y = \frac{y_h}{h}$ thus we can also write triple as $(h \cdot x, h \cdot y, h)$.
- For two dimensional geometric transformation we can take value of $\boldsymbol{h}$ is any positive number so we can get infinite homogeneous representation for coordinate value (x, y).
- But convenient choice is set $\boldsymbol{h} = 1$ as it is multiplicative identity, than (x, y) is represented as $(x, y, 1)$.
- Expressing coordinates in homogeneous coordinates form allows us to represent all geometric transformation equations as matrix multiplication.
- Let's see each representation with $\boldsymbol{h} = \mathbf{1}$

Translation

$\boldsymbol{P' = T_{(t_x, t_y)} \cdot P}$

$x \quad y$

$x' \quad 1 \quad 0 \quad t_x \quad x$

$[y'] = [0 \quad 1 \quad t_y][y]$

$1 \quad 0 \quad 0 \quad 1 \quad 1$

NOTE: - Inverse of translation matrix is obtain by putting $-\boldsymbol{t_x}$ & $-\boldsymbol{t_y}$ instead of $\boldsymbol{t_x}$ & $\boldsymbol{t_y}$.

Rotation

$\boldsymbol{P' = R_{(\theta)} \cdot P}$

$x' \quad \cos\theta \quad -\sin\theta \quad 0 \quad x$

$[y'] = [\sin\theta \quad \cos\theta \quad 0][y]$

$1 \quad 0 \quad 0 \quad 1 \quad 1$

NOTE: - Inverse of rotation matrix is obtained by replacing $\boldsymbol{\theta}$ by $-\boldsymbol{\theta}$.

Scaling

$\boldsymbol{P' = S_{(s_x s_y)} \cdot P}$

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

Composite Transformation

- We can set up a matrix for any sequence of transformations as a **composite transformation matrix** by calculating the matrix product of individual transformation.
- For column matrix representation of coordinate positions, we form composite transformations by multiplying matrices in order from right to left.

Translations

- Two successive translations are performed as:

$$P' = T(t_{x2}, t_{y2}) \cdot \{T(t_{x1}, t_{y1}) \cdot P\}$$

$$P' = \{T(t_{x2}, t_{y2}) \cdot T(t_{x1}, t_{y1})\} \cdot P$$

$$P' = \begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

$$P' = \begin{bmatrix} 1 & 0 & t_{x1} + t_{x2} \\ 0 & 1 & t_{y1} + t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

$$P' = T(t_{x1} + t_{x2}, t_{y1} + t_{y2}) \cdot P\}$$

Here P' and P are column vector of final and initial point coordinate respectively.

- This concept can be extended for any number of successive translations.

Example: Obtain the final coordinates after two translations on point $p(2,3)$ with translation vector $(4, 3)$ and $(-1, 2)$ respectively.

$$P' = T(t_{x1} + t_{x2}, t_{y1} + t_{y2}) \cdot P$$

$$P' = \begin{bmatrix} 1 & 0 & t_{x1}+t_{x2} \\ 0 & 1 & t_{y1}+t_{y2} \end{bmatrix} \cdot P = \begin{bmatrix} 1 & 0 & 4+(-1) \\ 0 & 1 & 3+2 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$\begin{matrix} 0 & 0 & 1 \end{matrix} \quad \begin{matrix} 0 & 0 & 1 \end{matrix} \quad \begin{matrix} 1 \\ 1 \end{matrix}$$

$$P' = \begin{bmatrix} 0 & 0 & 3 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \\ 1 \end{bmatrix}$$

Final Coordinates after translations are $p(5, 8)$.

Rotations

- Two successive Rotations are performed as:

$$P' = R(\theta_2) \cdot \{R(\theta_1) \cdot P\}$$

$$P' = \{R(\theta_2) \cdot R(\theta_1)\} \cdot P$$

$$P' = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

$$P' = \begin{bmatrix} \cos\theta_2\cos\theta_1 - \sin\theta_2\sin\theta_1 & -\sin\theta_1\cos\theta_2 - \sin\theta_2\cos\theta_1 & 0 \\ \sin\theta_1\cos\theta_2 + \sin\theta_2\cos\theta_1 & \cos\theta_2\cos\theta_1 - \sin\theta_2\sin\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

$$P' = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

$$P' = R(\theta_1 + \theta_2) \cdot P$$

Here P' and P are column vector of final and initial point coordinate respectively.

- This concept can be extended for any number of successive rotations.

Example: Obtain the final coordinates after two rotations on point $p(6,9)$ with rotation angles are 30^o and 60^o respectively.

$$P' = R(\theta_1 + \theta_2) \cdot P$$

$$P' = \begin{bmatrix} cos(\theta_1+\theta_2) & -sin(\theta_1+\theta_2) & 0 \\ sin(\theta_1+\theta_2) & cos(\theta_1+\theta_2) & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot P$$

$$P' = \begin{bmatrix} cos(30+60) & -sin(30+60) & 0 \\ sin(30+60) & cos(30+60) & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot P$$

$$P' = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 6 \\ 9 \\ 1 \end{bmatrix} = \begin{bmatrix} -9 \\ 6 \\ 1 \end{bmatrix}$$

Final Coordinates after rotations are $p(-9, 6)$.

Scaling

- Two successive scaling are performed as:

$$P' = S(s_{x2}, s_{y2}) \cdot \{S(s_{x1}, s_{y1}) \cdot P\}$$

$$P' = \{S(s_{x2}, s_{y2}) \cdot S(s_{x1}, s_{y1})\} \cdot P$$

$$P' = \begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

$$P' = \begin{bmatrix} s_{x1} \cdot s_{x2} & 0 & 0 \\ 0 & s_{y1} \cdot s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

$P' = S(s_{x1} \cdot s_{x2}, s_{y1} \cdot s_{y2}) \cdot P$

Here P' and P are column vector of final and initial point coordinate respectively.

- This concept can be extended for any number of successive scaling

Other Transformation

- Some package provides few additional transformations which are useful in certain applications. Two such transformations are reflection and shear.

Reflection

- A reflection is a transformation that produces a mirror image of an object.

The mirror image for a two –dimensional reflection is generated relative to an **axis of reflection** by rotating the object 180° about the reflection axis.

- Reflection gives image based on position of axis of reflection. Transformation matrix for few positions are discussed here.

Transformation matrix for reflection about the line $y = 0$, *the x axis.*

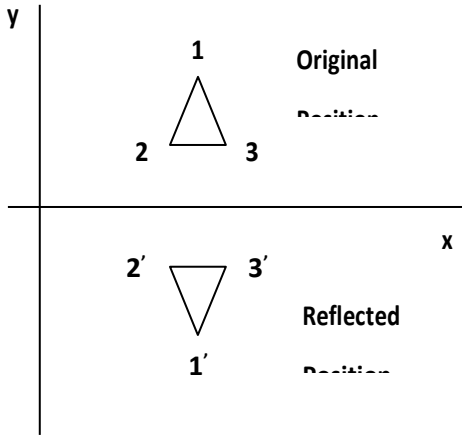


Fig. 3.9: - Reflection about x - axis.

- This transformation keeps x values are same, but flips (Change the sign) y values of coordinate positions.

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

Transformation matrix for reflection about the line $x = 0$, *the y axis.*

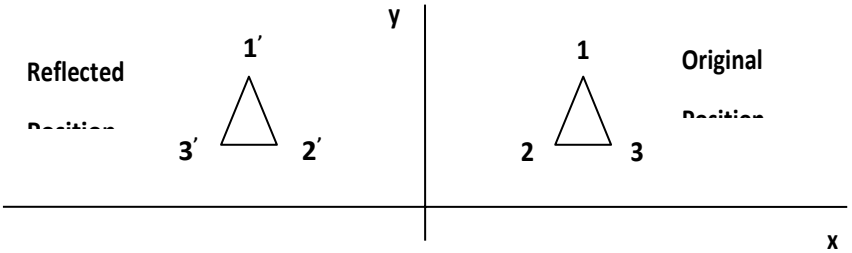


Fig. 3.10: - Reflection about y - axis.

- This transformation keeps y values are same, but flips (Change the sign) x values of coordinate positions.

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

Transformation matrix for reflection about the ***Origin***.

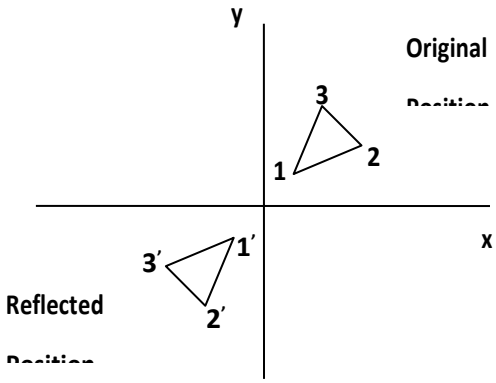


Fig. 3.11: - Reflection about origin.

- This transformation flips (Change the sign) x and y both values of coordinate positions.

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

Transformation matrix for reflection about the line $x = y$.

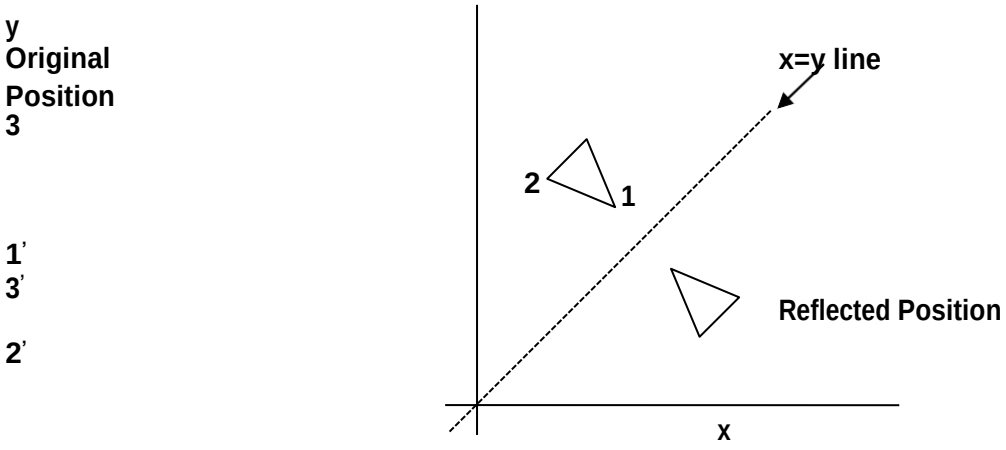
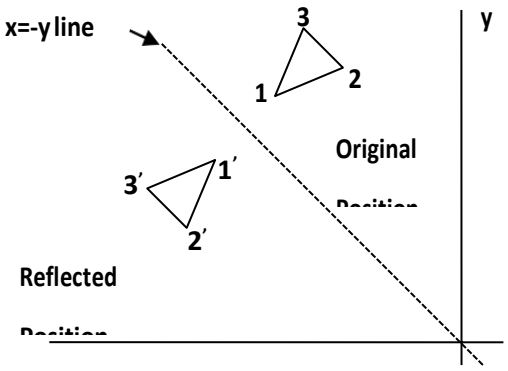


Fig. 3.12: - Reflection about x=y line.

- This transformation interchange x and y values of coordinate positions.

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

Transformation matrix for reflection about the line $x = -y$.



x

Fig. 3.12: - Reflection about x=-y line.

- This transformation interchange x and y values of coordinate positions.

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

- **Example**: - Find the coordinates after reflection of the triangle [A (10, 10), B (15, 15), C (20, 10)] about x axis.

$$P' = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 & 15 & 10 \\ 10 & 15 & 20 \\ 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} -10 & -15 & -10 \\ 10 & 15 & 20 \\ 1 & 1 & 1 \end{bmatrix}$$

- Final coordinate after reflection are [A' (10, -10), B' (15, -15), C' (20, -10)]

Shear

- A transformation that distorts the shape of an object such that the transformed shape appears as if the object were composed of internal layers that had been caused to slide over each other is called **shear**.
- Two common shearing transformations are those that shift coordinate x values and those that shift y values.

Shear in $x - direction$.

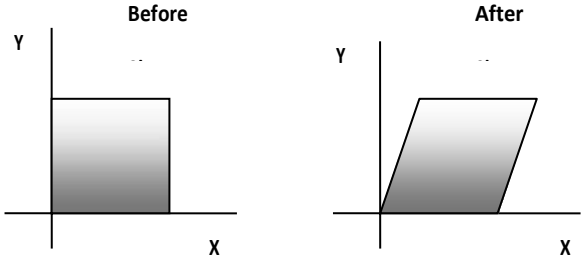


Fig. 3.13: - Shear in x-direction.

- Shear relative to $x - axis$ that is $y = 0$ line can be produced by following equation:

$x' = x + sh_x \cdot y$, $y' = y$

- Transformation matrix for that is:

$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Here sh_x is shear parameter. We can assign any real value to sh_x.

- We can generate $x - direction$ shear relative to other reference line $y = y_{ref}$ with following equation:

$x' = x + sh_x \cdot (y - y_{ref})$, $y' = y$

- Transformation matrix for that is:

$$\begin{bmatrix} 1 & sh_x & -sh_x \cdot y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

-

- Shear in $y - direction$.

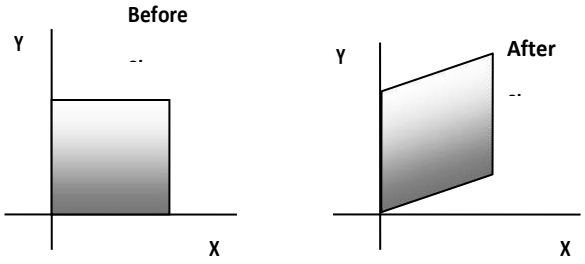


Fig. 3.14: - Shear in y-direction.

- Shear relative to $y - axis$ that is $x = 0$ line can be produced by following equation:

$x' = x$, $y' = y + sh_y \cdot x$

- Transformation matrix for that is:

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Here sh_y is shear parameter. We can assign any real value to sh_y.

- We can generate $y - direction$ shear relative to other reference line $x = x_{ref}$ with following equation:

$x' = x$, $y' = y + sh_y \cdot (x - x_{ref})$

- Transformation matrix for that is:

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y \cdot x_{ref} \\ 0 & 0 & 1 \end{bmatrix}$$

The Viewing Pipeline

- **Window:** Area selected in world-coordinate for display is called window. It defines what is to be viewed.
- **Viewport:** Area on a display device in which window image is display (mapped) is called viewport. It defines where to display.
- In many case window and viewport are rectangle, also other shape may be used as window and viewport.
- In general finding device coordinates of viewport from word coordinates of window is called as **viewing transformation.**
- Sometimes we consider this viewing transformation as window-to-viewport transformation but in general it involves more steps.

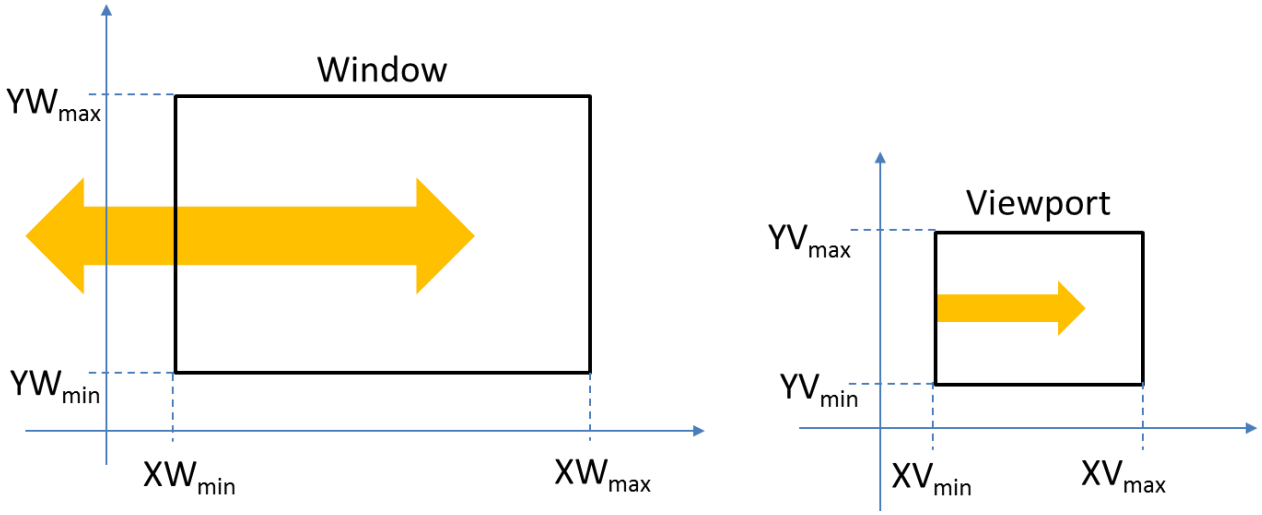


Fig. 3.1: - A viewing transformation using standard rectangles for the window and viewport.

- Now we see steps involved in viewing pipeline.

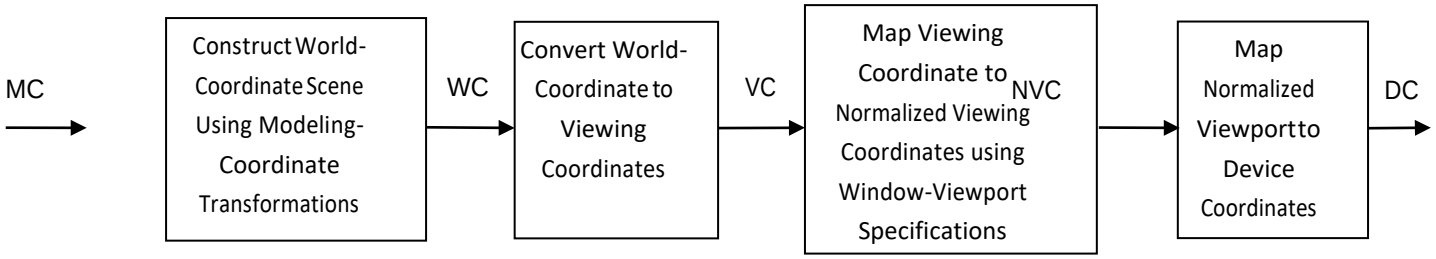


Fig. 3.2: - 2D viewing pipeline.

- As shown in figure above first of all we construct world coordinate scene using modeling coordinate transformation.
- After this we convert viewing coordinates from world coordinates using window to viewport transformation.
- Then we map viewing coordinate to normalized viewing coordinate in which we obtain values in between 0 to 1.
- At last we convert normalized viewing coordinate to device coordinate using device driver software which provide device specification.
- Finally device coordinate is used to display image on display screen.
- By changing the viewport position on screen we can see image at different place on the screen.
- By changing the size of the window and viewport we can obtain zoom in and zoom out effect as per requirement.
- Fixed size viewport and small size window gives zoom in effect, and fixed size viewport and larger window gives zoom out effect.
- View ports are generally defines with the unit square so that graphics package are more device independent which we call as normalized viewing coordinate.

Viewing Coordinate Reference Frame

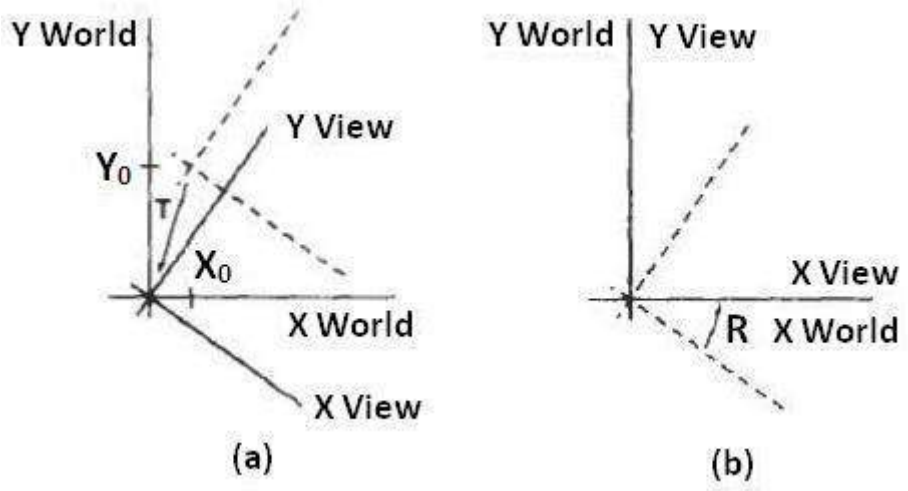


Fig. 3.3: - A viewing-coordinate frame is moved into coincidence with the world frame in two steps: (a) translate the viewing origin to the world origin, and then (b) rotate to align the axes of the two systems.

- We can obtain reference frame in any direction and at any position.
- For handling such condition first of all we translate reference frame origin to standard reference frame origin and then we rotate it to align it to standard axis.
- In this way we can adjust window in any reference frame.
- this is illustrate by following transformation matrix:

$$M_{wcvc} = RT$$

- Where T is translation matrix and R is rotation matrix.

Window-To-Viewport Coordinate Transformation

- Mapping of window coordinate to viewport is called window to viewport transformation.
- We do this using transformation that maintains relative position of window coordinate into viewport.
- That means center coordinates in window must be remains at center position in viewport.
- We find relative position by equation as follow:

$$\frac{x_v - x_{vmin}}{x_{vmax} - x_{vmin}} = \frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}}$$

$$\frac{y_v - y_{vmin}}{y_{vmax} - y_{vmin}} = \frac{y_w - y_{wmin}}{y_{wmax} - y_{wmin}}$$

- Solving by making viewport position as subject we obtain:

$$x_v = x_{vmin} + (x_w - x_{wmin})s_x$$

$$y_v = y_{vmin} + (y_w - y_{wmin})s_y$$

- Where scaling factor are :

$$s_x = \frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}$$

$$s_y = \frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}}$$

- We can also map window to viewport with the set of transformation, which include following sequence of transformations:
  1. Perform a scaling transformation using a fixed-point position of (x_{Wmin},y_{wmin}) that scales the window area to the size of the viewport.
  2. Translate the scaled window area to the position of the viewport.
- For maintaining relative proportions we take ($s_x = s_y$). in case if both are not equal then we get stretched or contracted in either the x or y direction when displayed on the output device.
- Characters are handle in two different way one way is simply maintain relative position like other primitive and other is to maintain standard character size even though viewport size is enlarged or reduce.
- Number of display device can be used in application and for each we can use different window-to-viewport transformation. This mapping is called the **workstation transformation.**

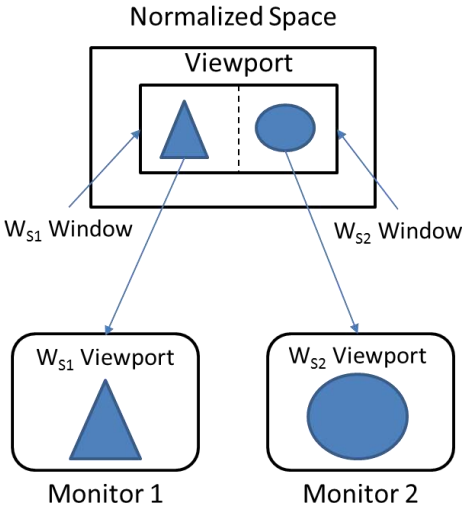


Fig. 3.4: - workstation transformation.

- As shown in figure two different displays devices are used and we map different window-to-viewport on each one.

Clipping Operations

- Generally, any procedure that identifies those portions of a picture that are either inside or outside of a specified region of space is referred to as a **clipping algorithm**, or simply **clipping**. The region against which an object is to clip is called a **clip window.**
- Clip window can be general polygon or it can be curved boundary.

Application of Clipping

- It can be used for displaying particular part of the picture on display screen.
- Identifying visible surface in 3D views.
- Antialiasing.
- Creating objects using solid-modeling procedures.
- Displaying multiple windows on same screen.
- Drawing and painting.

Point Clipping

- In point clipping we eliminate those points which are outside the clipping window and draw points which are inside the clipping window.
- Here we consider clipping window is rectangular boundary with edge (x_{wmin},x_{wmax},y_{wmin},y_{wmax}).
- So for finding wether given point is inside or outside the clipping window we use following inequality:

$x_{wmin} \leq x \leq x_{wamx}$

$y_{wmin} \leq y \leq y_{wamx}$

- If above both inequality is satisfied then the point is inside otherwise the point is outside the clipping window.

Line Clipping

- Line clipping involves several possible cases.
  1. Completely inside the clipping window.
  2. Completely outside the clipping window.
  3. Partially inside and partially outside the clipping window.

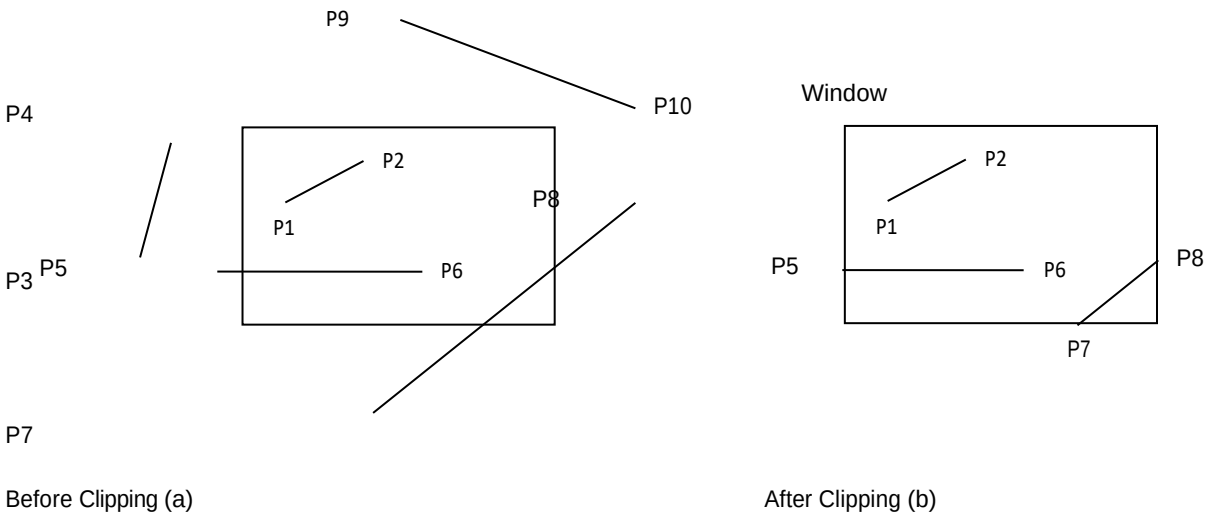


Fig. 3.5: - Line clipping against a rectangular window.

- Line which is completely inside is display completely. Line which is completely outside is eliminated from display. And for partially inside line we need to calculate intersection with window boundary and find which part is inside the clipping boundary and which part is eliminated.
- For line clipping several scientists tried different methods to solve this clipping procedure. Some of them are discuss below.

Cohen-Sutherland Line Clipping

- This is one of the oldest and most popular line-clipping procedures.

Region and Region Code

- In this we divide whole space into nine region and assign 4 bit code to each endpoint of line depending on the position where the line endpoint is located.

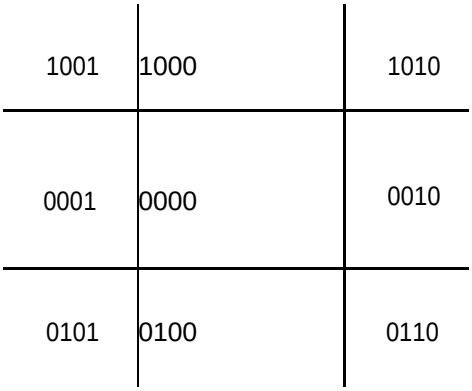


Fig. 3.6: - Workstation transformation.

- Figure 3.6 shows code for line end point which is fall within particular area.
- Code is deriving by setting particular bit according to position of area.
  Set bit 1: For left side of clipping window.

Set bit 2: For right side of clipping window. Set
bit 3: For below clipping window.
Set bit 4: For above clipping window.

- All bits as mention above are set means 1 and other are 0.

Algorithm

Step-1:

Assign region code to both endpoint of a line depending on the position where the line endpoint is located.

Step-2:

If both endpoint have code '0000'
Then line is completely inside.
Otherwise
Perform logical ending between this two codes.

If result of logical ending is non-zero
Line is completely outside the clipping window.
Otherwise
Calculate the intersection point with the boundary one by one. Divide the line
into two parts from intersection point.
Recursively call algorithm for both line segments.

Step-3:
Draw line segment which are completely inside and eliminate other line segment which found completely outside.

Intersection points calculation with clipping window boundary

- For intersection calculation we use line equation "$y = mx + b$".
- 'x' is constant for left and right boundary which is:
  - for left "$x = x_{wmin}$"
  - for right "$x = x_{wmax}$"
- So we calculate y coordinate of intersection for this boundary by putting values of x depending on boundary is left or right in below equation.

$y = y_1 + \boldsymbol{m}(x - x_1)$

- 'y' coordinate is constant for top and bottom boundary which is:
  - for top "$y = y_{wmax}$"
  - for bottom "$y = y_{wmin}$"
- So we calculate x coordinate of intersection for this boundary by putting values of y depending on boundary is top or bottom in below equation.

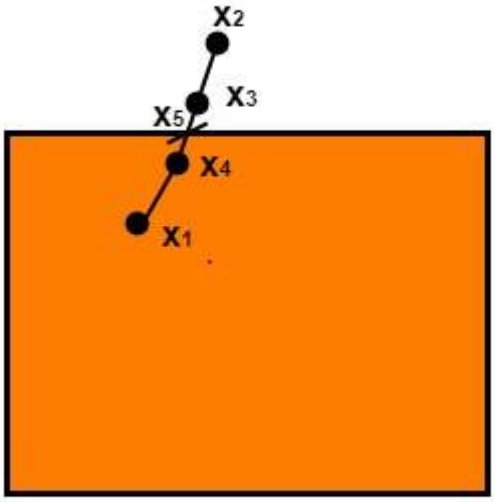
X=x1+y-y1/m

Mid Point Subdivision Line Clipping Algorithm:

It is used for clipping line. The line is divided in two parts. Mid points of line is obtained by dividing it in two short segments. Again division is done, by finding midpoint. This process is continued until line of visible and invisible category is obtained. Let (x_i,y_i) are midpoint

It is used for clipping line. The line is divided in two parts. Mid points of line is obtained by dividing it in two short segments. Again division is done, by finding midpoint. This process is continued until line of visible and invisible category is obtained. Let (x_i,y_i) are midpoint

$$x_m = \frac{x_1+x_2}{2} \quad y_m = \frac{y_1+y_2}{2}$$



Step1: Find $\frac{x_2+x_1}{2}$ i. e. $x_3 = \frac{x_2+x_1}{2}$

Step2: Find $x_4 = \frac{x_3+x_1}{2}$

Step3: Find $x_5 = \frac{x_4+x_3}{2}$

$x_5$lie on point of intersection of boundary of window.

Advantage of midpoint subdivision Line Clipping:

It is suitable for machines in which multiplication and division operation is not possible. Because it can be performed by introducing clipping divides in hardware.

Algorithm of midpoint subdivision Line Clipping:

Step1: Calculate the position of both endpoints of the line

Step2: Perform OR operation on both of these endpoints

Step3: If the OR operation gives 0000
then
Line is guaranteed to be visible
else
Perform AND operation on both endpoints.
If AND ≠ 0000
then the line is invisible
else
AND=6000
then the line is clipped case.

Step4: For the line to be clipped. Find midpoint

$X_m=(x_1+x_2)/2$

$Y_m=(y_1+y_2)/2$

$X_m$is midpoint of X coordinate.

$Y_m$is midpoint of Y coordinate.

Step5: Check each midpoint, whether it nearest to the boundary of a window or not.

Step6: If the line is totally visible or totally rejected not found then repeat step 1 to 5.

Step7: Stop algorithm.