Unit 3
Two-Dimensional Geometric Transformations

Changing co-ordinate description of an object is called transformation.
Types:
- Rigid body transformation (transformation without change in shape.)
- Non rigid body transformation (transformation with change in shape.)

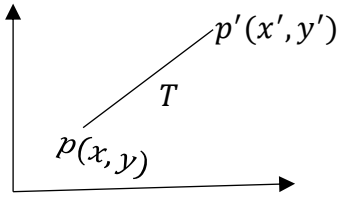
- When a transformation takes place on a 2D plane, it is called 2D transformation.
- The three basic transformations are
  - Translation
  - Rotation
  - Scaling

Other transformation includes reflection and shear.

2D Translation

Repositioning of object along a straight-line path from one coordinate location to another is called translation.
Translation is performed on a point by adding offset to its coordinate so as to generate a new coordinate position.
Let p(x, y) be translated to $p'(x', y')$ by using offset t_x and t_y in x & y direction. Then,
$x'=x + t_x$
$y'=y + t_y$
In matrix form,

i.e. $P' = P + T$ where T is transformation matrix.



2D Rotation

- Changing the co-ordinate position along a circular path is called rotation.

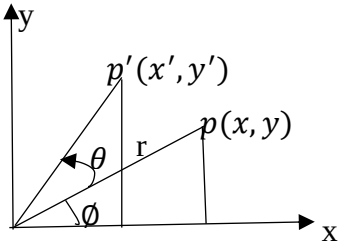
2D rotation is applied to re-position the object along a circular path in XY-plane. Rotation is generated by specifying rotation angle (θ) and pivot point (rotation point).
- The positive θ rotates object in anti-clockwise direction and the negative value of θ rotates the object in clockwise direction.

Let $p(x, y)$ be a point rotated by θ about origin to new point $p'(x', y')$.

Here,

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

But $x = rcos\emptyset$ & $y = rsin\emptyset$
$\therefore$ $\boldsymbol{x' = xcos\theta - ysin\theta}$ ……. (i)



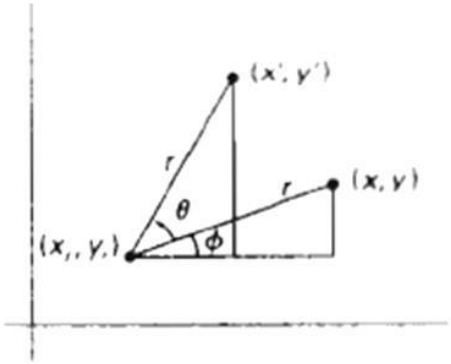
Similarly,

$\therefore$ $\boldsymbol{y' = xsin\theta + ycos\theta}$ …….. (ii)

Which are equation for rotation of (x, y) with angle θ and taking pivot as origin.

In matrix form
$P' = R.P$
$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

If the pivot point is at $(\boldsymbol{x_r, y_r})$***.***



Here,

$\cos(\emptyset + \theta) = (x' - x_r)/r$
or, $r\cos(\emptyset + \theta) = (x' - x_r)$
or, $(x' - x_r) = rcos\emptyset cos\theta - rsin\emptyset sin\theta$

Since, $rcos\emptyset = (x - x_r), rsin\emptyset = (y - y_r)$

$\therefore\ \boldsymbol{x' = x_r + (x - x_r)cos\theta - (y - y_r)sin\theta}$ …… (i)

Similarly,

$\sin(\emptyset + \theta) = (y' - y_r)/r$
or, $r\,sin(\emptyset + \theta) = (y' - y_r)$
or, $(y' - y_r) = rsin\emptyset cos\theta + rcos\emptyset sin\theta$

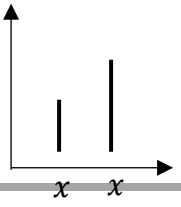
Since, $rcos\emptyset = (x - x_r), rsin\emptyset = (y - y_r)$

$\therefore\ \boldsymbol{y' = y_r + (x - x_r)sin\theta + (y - y_r)cos\theta}$ …… (ii)

These equations (i) and (ii) are the equations for rotation of a point (x, y) with angle θ taking pivot point(x_r, y_r).

➢ 2D Scaling

Scaling transformation alters the size of object. A simple two dimensional scaling operation is performed by multiplying object position (x, y) with scaling factors s_x & s_y along x & y direction to produce (x', y').
$x' = x.s_x$ & $y' = y.s_y$
In matrix form,
$P' = S.P$



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

If the scaling factor is less than 1, the size of object is decreased and if it is greater than 1 the size of object is increased. The scaling factor = 1 for both direction does not change the size of the object.

- If both scaling factors have same value then the scaling is known as **uniform scaling**.
- If the value of s_x and s_y are different, then the scaling is known as **differential scaling**. The differential scaling is mostly used in the graphical package to change the shape of the object.

➢ If fixed point is (x_f, y_f) about which rotation is made, then
$x' = x.s_x + x_f(1 - s_x)$
$y' = y.s_y + y_f(1 - s_y)$

➢ **Matrix representation & Homogenous coordinate**

The homogeneous co-ordinate system provides a uniform framework for handling different geometric transformations, simply as multiplication of matrices.
To perform more than one transformation at a time, homogeneous coordinates are used. They reduce unwanted calculations, intermediate steps, saves time and memory and produce a sequence of transformations.
We represent each Cartesian coordinate position *(x, y)* with the homogeneous coordinate triple(x_h, y_h, h), where, $x = x_h/h, y = y_h/h$. (h is 1 usually for 2D case).
Therefore, (x, y) in Cartesian system is represented as (x, y, 1) in homogeneous co-ordinate system.

For Translation: $T(t_x, t_y)$

$P' = T.P$
Where, $P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$ & $P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
By which we can get,
$x'=x + t_x$
$y'=y + t_y$

For Rotation: R(θ)

$P' = R.P$
Where, $P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, R = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ & $P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

For Scaling with scaling factors (s_x, s_y)

$P' = S.P$

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

This gives the equations,

$x' = x.s_x$ & $y' = y.s_y$

- **Composite 2D translation**

If two successive translation vector (t_{x1}, t_{y1}) & (t_{x2}, t_{y2}) is applied on position $p(x, y)$, then translated point is given by,

$P' = T_{(t_{x2},t_{y2})}.T_{(t_{x1},t_{y1})}.P$

$\therefore \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Which gives,

$x' = x + t_{x1} + t_{x2}$

$y' = y + t_{y1} + t_{y2}$

- **2D composite rotation**

$P' = \big(R(\theta_2)\big).\big(R(\theta_1)\big).P$

$= R(\theta_1 + \theta_2).P$

- **2D composite Scaling**

$P' = S_{(S_{x2},S_{y2})}.S_{(S_{x1},S_{y1})}.P$

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

#Q. Prove that two successive translations are additive.

Proof:

If two successive translation vector (t_{x1}, t_{y1}) & (t_{x2}, t_{y2}) is applied to coordinate position P, the final transformed location P' is calculated with the following composite transformation as,

$T = T_{(t_{x2},t_{y2})}.T_{(t_{x1},t_{y1})}$

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

Hence, $T_{(t_{x2},t_{y2})}.T_{(t_{x1},t_{y1})} = T_{(t_{x1}+t_{x2},t_{y1}+t_{y2})}$ which demonstrates that two successive translations are additive.

#Q. Prove that two successive rotation are additive.

Proof:

Let P be the point anticlockwise rotated by angle θ_1 to point P' and again let P' be rotated by angle θ_2 to point P", then the combined transformation can be calculated with the following composite matrix as:

$$T = R(\theta_2).R(\theta_1)$$

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

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

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

i.e. $R(\theta_2).R(\theta_1) = R(\theta_1 + \theta_2)$ which demonstrates that two successive rotations are additive.

#Q. Prove that two successive scaling are multiplicative.

Proof:

Let point P is first scaled with scaling factors s_{x1}, s_{y1} to P' and again let P' be scaled by scaling factors s_{x2}, s_{y2} to point P", then the combined transformation can be calculated with the following composite matrix

$$T = S_{(s_{x2},s_{y2})}.S_{(s_{x1},s_{y1})}$$

$$= \begin{bmatrix} \boldsymbol{s_{x2}} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} \boldsymbol{s_{x1}} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

i.e. $S_{(s_{x2},s_{y2})}.S_{(s_{x1},s_{y1})} = S_{(s_{x1}s_{x2},s_{y1}s_{y2})}$ which demonstrates that two successive scaling are multiplicative.

❖ General 2D pivot rotation

- Suppose the pivot point is located at (x_r, y_r).
- To rotate about arbitrary point, we have to perform the following transformation:
  a) Translate the object so that pivot point position is moved to coordinate origin.
  b) Rotate the object about coordinate origin.
  c) Translate the object so that pivot point is returned to original position.

Matrix representation:

$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

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

❖ General 2D fixed point scaling

- Suppose the fixed point is located at (x_f, y_f).
- To scale about arbitrary fixed point, we have to perform the following transformation:
  d) Translate the object so that fixed point coincide with the coordinate origin.
  e) Scale the object with respect to coordinate origin.
  f) Translate the object to its original position.

Matrix representation:

$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \boldsymbol{s_x} & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

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

Q. Find the scaled triangle with vertices A(0, 0), B(1, 1) & C(5, 2) after it has been magnified twice its size.

Solution:

Here, $s_x = 2$ & $s_y = 2$

Now,

$A' = S.A$

$$= \begin{bmatrix} \boldsymbol{2} & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = (0,0)$$

$B' = S.B$

$$= \begin{bmatrix} \boldsymbol{2} & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} = (2,2)$$

$C' = S.C$

$$= \begin{bmatrix} \boldsymbol{2} & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 10 \\ 4 \\ 1 \end{bmatrix} = (10,4)$$

Hence the final coordinate points are $A'(0,0)$, $B'(2,2)$, $C'(10,4)$.

Q. Rotate a triangle A(0, 0), B(2, 2), C(4, 2) about the origin by the angle of 45 degree.

Solution:

The given triangle ABC can be represented by a matrix formed from homogenous coordinates of vertices.

$$\begin{bmatrix} 0 & 2 & 4 \\ 0 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

Also, we have

$$R_{45^0} = \begin{bmatrix} cos45^0 & -sin45^0 & 0 \\ sin45^0 & cos45^0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

So the coordinates of the rotated triangle ABC are

$$R_{45^0}[ABC] = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{0} & 2 & 4 \\ 0 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & 0 & \sqrt{2} \\ 0 & 2\sqrt{2} & 3\sqrt{2} \\ 1 & 1 & 1 \end{bmatrix}$$

Hence the final coordinate points are $A'(0,0)$, $B'(0,2\sqrt{2})$, $C'(\sqrt{2},3\sqrt{2})$.

Q. Rotate a triangle (5, 5), (7, 3), (3, 3) about fixed point (5, 4) in counter clockwise by 90 degree.

Solution:

The required steps are:

1. Translate the fixed point to origin.
2. Rotate about the origin by 90 degree.
3. Reverse the translation as performed earlier.

Thus, the composite matrix is given by

$M = T_{(x_f,y_f)} R_\theta T_{(-x_f,-y_f)}$

$$= \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos90 & -sin90 & 0 \\ sin90 & cos90 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 4 \\ 1 & 0 & -5 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & -1 & 9 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

Hence the required coordinate can be calculated as:

$P' = M * P$

$$= \begin{bmatrix} 0 & -1 & 9 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 7 & 3 \\ 5 & 3 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} 4 & 6 & 6 \\ 4 & 6 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

Hence, the new required coordinates are (4, 4), (6, 6), (6, 2).

Q. Rotate a triangle A(7, 15), B(5, 8) & C(10, 10) by 45 degree clockwise about origin and scale it by (2, 3) about origin.

Solution:

The steps required are:

1. Rotate by 45 clockwise
2. Scale by $s_x = 2$ & $s_y = 3$.

Thus the composite matrix is given by;

$M = S(2,3).R_{45}$

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-45) & -\sin(-45) & 0 \\ \sin(-45) & \cos(-45) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{2}{\sqrt{2}} & \frac{2}{\sqrt{2}} & 0 \\ -\frac{3}{\sqrt{2}} & \frac{3}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The transformation points are

$A' = M.A$

$$= \begin{bmatrix} \frac{2}{\sqrt{2}} & \frac{2}{\sqrt{2}} & 0 \\ -\frac{3}{\sqrt{2}} & \frac{3}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 15 \\ 1 \end{bmatrix} =$$

$B' = M.B$

$$= \begin{bmatrix} \frac{2}{\sqrt{2}} & \frac{2}{\sqrt{2}} & 0 \\ -\frac{3}{\sqrt{2}} & \frac{3}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 8 \\ 1 \end{bmatrix} =$$

$C' = M.C$

$$= \begin{bmatrix} \frac{2}{\sqrt{2}} & \frac{2}{\sqrt{2}} & 0 \\ -\frac{3}{\sqrt{2}} & \frac{3}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} =$$

Q. A square with vertices A(0, 0), B(2, 0), C(2, 2) & D(0, 2) is scaled 2 units in x & y direction about the fixed point (1, 1). Find the coordinates of the vertices of new square.

Solution:

Here,

$s_x = 2$ & $s_y = 2$

$x_f = 1$ & $y_f = 1$

Composite matrix is,

$M = T_{(x_f,y_f)}.S.T_{(-x_f,-y_f)}$

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

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

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

Now, the transformation points are

$A' = M.A$

$$= \begin{bmatrix}2 & 0 & -1\\ 0 & 2 & -1\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0\\ 0\\ 1\end{bmatrix} = \begin{bmatrix}-1\\ -1\\ 1\end{bmatrix} = (-1, -1)$$

$B' = M.B$

$$= \begin{bmatrix}2 & 0 & -1\\ 0 & 2 & -1\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}2\\ 0\\ 1\end{bmatrix} = \begin{bmatrix}3\\ -1\\ 1\end{bmatrix} = (3, -1)$$

$C' = M.C$

$$= \begin{bmatrix}2 & 0 & -1\\ 0 & 2 & -1\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}2\\ 2\\ 1\end{bmatrix} = \begin{bmatrix}3\\ 3\\ 1\end{bmatrix} = (3, 3)$$

$D' = M.D$

$$= \begin{bmatrix}2 & 0 & -1\\ 0 & 2 & -1\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0\\ 2\\ 1\end{bmatrix} = \begin{bmatrix}-1\\ 3\\ 1\end{bmatrix} = (-1, 3)$$

Q. A triangle having vertices A(3, 3), B(8, 5) & C(5, 8) is first translated by 2 units about fixed point (5, 6) & finally rotated 90 degree anticlockwise about pivot point (2, 5). Find the final position of triangle.

Solution:

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

$$S = \begin{bmatrix}1 & 0 & 5\\ 0 & 1 & 6\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}2 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & -5\\ 0 & 1 & -6\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}2 & 0 & -5\\ 0 & 2 & -6\\ 0 & 0 & 1\end{bmatrix}$$

$$R = \begin{bmatrix}1 & 0 & 2\\ 0 & 1 & 5\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & -2\\ 0 & 1 & -5\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}0 & -1 & 7\\ 1 & 0 & 3\\ 0 & 0 & 1\end{bmatrix}$$

Composite matrix,

$M = R.S.T$

$$= \begin{bmatrix}0 & -1 & 7\\ 1 & 0 & 3\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}2 & 0 & -5\\ 0 & 2 & -6\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 2\\ 0 & 1 & 2\\ 0 & 0 & 1\end{bmatrix}$$

$$= \begin{bmatrix}0 & -2 & 13\\ 2 & 0 & 2\\ 0 & 0 & 1\end{bmatrix}$$

Now, the transformation points are;

$A' = M.A$

$$= \begin{bmatrix}0 & -2 & 13\\ 2 & 0 & 2\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}3\\ 3\\ 1\end{bmatrix} = \begin{bmatrix}7\\ 8\\ 1\end{bmatrix} = (7, 8)$$

$B' = M.B$

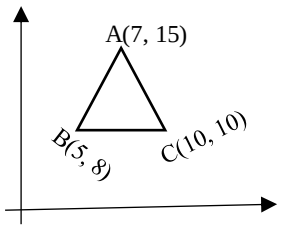
$= \begin{bmatrix} 0 & -2 & 13 \\ 2 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 18 \\ 1 \end{bmatrix}$ = (3, 18)

$C' = M.C$

$= \begin{bmatrix} 0 & -2 & 13 \\ 2 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 8 \\ 1 \end{bmatrix} = \begin{bmatrix} -3 \\ 12 \\ 1 \end{bmatrix}$ = (-3, 12)

Q. Rotate the ⊿ABC by 90° anti-clock wise about (5, 8) and scale it by (2, 2) about (10, 10).

Solution:



Step 1:
T(-5, -8)
Step 2:
R(90°)
Step 3:
T(5, 8)
Step 4:
T(-10, -10)
Step 5:
S(2, 2)
Step 6:
T(10, 10)
The composite matrix is given by
M= T(10,10).S(2,2).T(-10,-10).T(5,8).R(90) .T(-5,-8)
Complete urself.

❖ Reflection

Providing a mirror image about an axis of an object is called reflection.

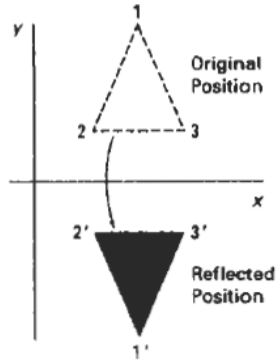
Reflection about x-axis (y=0)

The reflection of a point P(x, y) on x-axis, changes the y-coordinate sign i.e. P(x, y) changes to P'(x, -y).
In matrix form,

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

$P' = R_{fx}.P$

R_{fx}= Reflection matrix about x-axis.



Reflection about y-axis (x=0)

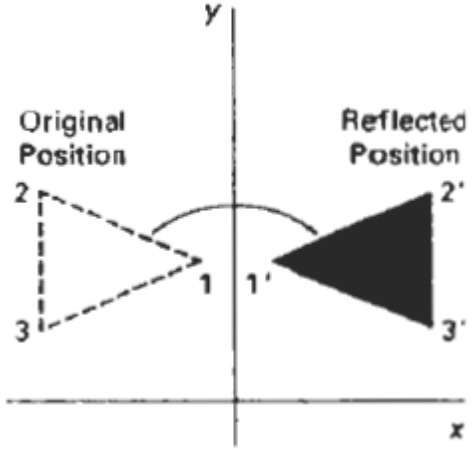
The reflection of a point P(x, y) on y-axis changes the sign of x-coordinate i.e. P(x, y) changes to P'(-x, y).

In matrix form,

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

$P' = R_{fy}.P$

R_{fy}= Reflection matrix about y-axis.



Reflection about origin

Filp both x & y coordintae of a point.

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

Reflection about y=mx+c

Perform the following transformation:

a) Translate the line so that it passes through origin.
b) Rotate the line so that it coincides with any coordinate axis.
c) Reflect object about that axis.
d) Perform reverse rotation.
e) Perform reverse translation so that line is placed to its original position.

Q. What is the basic purpose of composite transformation?

→ *The basic purpose of composing transformation is to gain efficiency by applying a single composed transformation to a point, rather than applying a series of transformation, one after another.*

Q. A triangle having vertices A(2, 3), B(6, 3) & C(4,8) is reflected about y=3x+4. Find the final position of triangle.

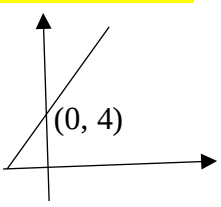
Solution:

Here,

y=3x+4

m=3, c=4

$(t_x, t_y) = (0, c) = (0, 4)$



$\theta = tan^{-1}(m) = tan^{-1}(3) = 71.56$

Now,

$$T(-t_x, -t_y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R(-\theta) = \begin{bmatrix} \cos(-71.56) & -\sin(-71.56) & 0 \\ \sin(-71.56) & \cos(-71.56) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.32 & 0.95 & 0 \\ -0.95 & 0.32 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$R(\theta) = \begin{bmatrix} \cos(71.56) & -\sin(71.56) & 0 \\ \sin(71.56) & \cos(71.56) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.32 & -0.95 & 0 \\ 0.95 & 0.32 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T(t_x, t_y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

Now, composite transformation matrix is;

$M = T(t_x, t_y).R(\theta).R_{fx}.R(-\theta).T(-t_x, -t_y)$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.32 & -0.95 & 0 \\ 0.95 & 0.32 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.32 & 0.95 & 0 \\ -0.95 & 0.32 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.32 & -0.95 & 0 \\ 0.95 & 0.32 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.32 & 0.95 & -3.8 \\ -0.95 & 0.32 & -1.28 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.32 & -0.95 & 0 \\ 0.95 & 0.32 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.32 & 0.95 & -3.8 \\ 0.95 & -0.32 & 1.28 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -0.8001 & 0.608 & -2.432 \\ 0.608 & 0.8001 & -3.2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -0.8001 & 0.608 & -2.432 \\ 0.608 & 0.8001 & 0.7996 \\ 0 & 0 & 1 \end{bmatrix}$$

Now, the transformation points are;

$A' = M.A$

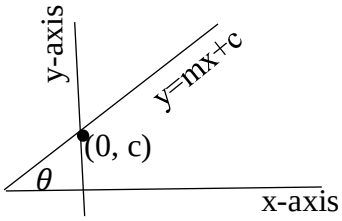
$$= \begin{bmatrix} -0.8001 & 0.608 & -2.432 \\ 0.608 & 0.8001 & 0.7996 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} -2.21 \\ 4.42 \\ 1 \end{bmatrix} = (-2.21, 4.42)$$

$B' = M.B$

$$= \begin{bmatrix} -0.8001 & 0.608 & -2.432 \\ 0.608 & 0.8001 & 0.7996 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} -5.41 \\ 6.85 \\ 1 \end{bmatrix} = (-5.41, 6.85)$$

$$C' = M.C$$
$$= \begin{bmatrix} -0.8001 & 0.608 & -2.432 \\ 0.608 & 0.8001 & 0.7996 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 8 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.77 \\ 9.63 \\ 1 \end{bmatrix} = (-0.77, 9.63)$$

Q. Derive the composite matrix for reflecting an object about any arbitary line y=mx+c.
Solution:



In order to reflect an object about any line y=mx+c, we need to perform Composite transformation as below

$T = T_{(0,c)}.R_{(\theta)}.R_{fx}.R_{(-\theta)}.T_{(0,-c)}$

And

Slope m=$tan\theta$

Also we have,

$$cos^2\theta = \frac{1}{tan^2\theta+1} = \frac{1}{m^2+1}$$

$$\therefore cos\theta = \frac{1}{\sqrt{m^2+1}}$$

Also we have,

$$sin^2\theta + cos^2\theta = 1$$

$$sin^2\theta = 1 - cos^2\theta = 1 - \frac{1}{m^2+1} = \frac{m^2}{m^2+1}$$

$$\therefore sin\theta = \frac{m}{\sqrt{m^2+1}}$$

So,

$T = T_{(0,c)}.R_{(\theta)}.R_{fx}.R_{(-\theta)}.T_{(0,-c)}$

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

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & sin\theta & -csin\theta \\ -sin\theta & cos\theta & -ccos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & sin\theta & -csin\theta \\ sin\theta & -cos\theta & ccos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{m^2+1}} & \frac{-m}{\sqrt{m^2+1}} & 0 \\ \frac{m}{\sqrt{m^2+1}} & \frac{1}{\sqrt{m^2+1}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{m^2+1}} & \frac{m}{\sqrt{m^2+1}} & \frac{-cm}{\sqrt{m^2+1}} \\ \frac{m}{\sqrt{m^2+1}} & \frac{-1}{\sqrt{m^2+1}} & \frac{c}{\sqrt{m^2+1}} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1-m^2}{m^2+1} & \frac{2m}{m^2+1} & \frac{-2cm}{m^2+1} \\ \frac{2m}{m^2+1} & \frac{m^2-1}{m^2+1} & \frac{c-cm^2}{m^2+1} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1-m^2}{m^2+1} & \frac{2m}{m^2+1} & \frac{-2mc}{m^2+1} \\ \frac{2m}{m^2+1} & \frac{m^2-1}{m^2+1} & \frac{2c}{m^2+1} \\ 0 & 0 & 1 \end{bmatrix}$$

❖ Shearing

A transformation that distorts the shape of an object such that the transformed shape appears as if the object is composed of internal layers and these layers are caused to slide over is shearing.

X-direction Shear:

An X-direction shear relative to x-axis is produced with transformation matrix equation.

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

Which transforms,

$x' = x + sh_x . y$

$y' = y$

Y-direction shear:

A Y-direction shear relative to y-axis is produced by following transformation equations.

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

Which transforms,

$x' = x$

$y' = x . sh_y + y$

X-direction shear relative to $y = y_{ref}$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & -sh_x . y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

$y' = y$

Y-direction shear relative to $x = x_{ref}$

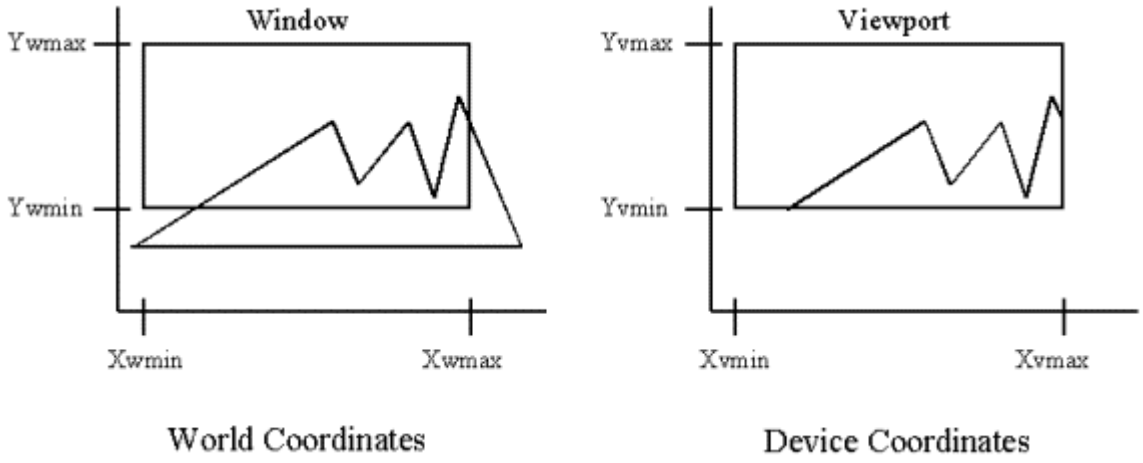
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y . x_{ref} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$x' = x$

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

❖ 2D Viewing

The process of mapping the world coordinate scene to device coordinate is called viewing transformation or windows to view port transformation.



- A world co-ordinate area selected for display is called a window and an area on the display device to which a window is mapped is called a view port.
- The window defines what is to be viewed and the viewport defines where it is to be displayed.
- Window deals with object space whereas viewport deals with image space.

Transformations from world to device coordinate involves translation, rotation and scaling operations, as well as procedure for deleting those parts of the picture that are outside the limits of selected display area i.e. clipping.
To make the viewing process independent of the requirements of any output device, graphics systems convert object description to normalized coordinates.

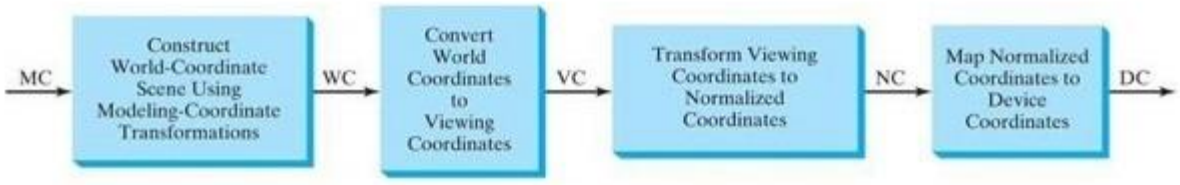


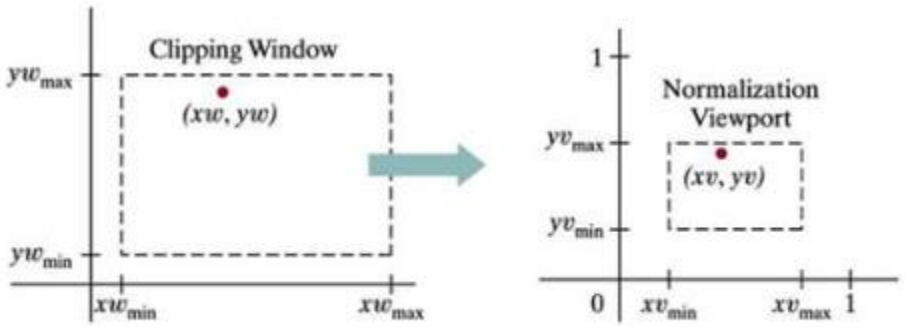
Fig. (c) Two-dimensional viewing transformation pipeline

Applications

- By changing the position of the view port, we can view objects at different positions on the display area of an output device.
- By varying the size of view ports, we can change size of displayed objects.
- Zooming effects can be obtained by successively mapping different-sized windows on a fixed-sized view port.
- Panning effects (Horizontal scrolling) are produced by moving a fixed-sized window across the various objects in a scene.

Windows to Viewport Transformation

It is the mechanism for displaying view of a picture on an output device. The world co-ordinate selected for display is called window. The area on the display device to which window is mapped is called viewport. So, window defines what is to be viewed and viewport defines where it is to be displayed. The mapping of part of world co-ordinate scene to device co-ordinate is called viewing transformation or window-to-viewport transformation.



A point (xw, yw) in a world-coordinate clipping window is mapped to viewport coordinates (xv, yv), within a unit square, so that the relative positions of the two points in their respective rectangles are the same.

To maintain the same relative placement in the viewport as in the window, we require that:

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}}$$

By solving these equations for the unknown viewport position (xv, yv) the following becomes true:

$$xv = s_x xw + t_x$$
$$yv = s_y yw + t_y$$

The scale factors (s_x, s_y) would be,

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

$$s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

And the translation factors (t_x, t_y) would be,

$$t_x = \frac{xw_{max}.xv_{min} - xw_{min}.xv_{max}}{xw_{max} - xw_{min}}$$

$$t_y = \frac{yw_{max}.yv_{min} - yw_{min}.yv_{max}}{yw_{max} - yw_{min}}$$

Transforming world coordinate to view coordinate can be obtained with the following sequence:

a) Scale (S) the clipping window to the size of viewport using fixed point position of (xw_{min}, yw_{min}).
b) Translate (T) (xw_{min}, yw_{min}) to (xv_{min}, yv_{min}).

Here,

$$S = \begin{bmatrix} s_x & 0 & xw_{min}(1-s_x) \\ 0 & s_y & yw_{min}(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} 1 & 0 & xv_{min} - xw_{min} \\ 0 & 1 & yv_{min} - yw_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

Now, window- to – viewport transformation matrix is;

$$\boldsymbol{T_{wv} = T.S = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}}$$

Examples:

Q. Window port is given by (100, 100, 300, 300) and viewport is given by (50, 50, 150, 150). Convert the window port coordinate (200, 200) to the view port coordinate.

Solution:

Here,

$(xw_{min}, yw_{min}) = (100,100)$
$(xw_{max}, yw_{max}) = (300,300)$
$(xv_{min}, yv_{min}) = (50,50)$
$(xv_{max}, yv_{max}) = (150,150)$
$(xw, yw) = (200,200)$

Then we have

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}} = \frac{150-50}{300-100} = 0.5$$

$$s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}} = \frac{150-50}{300-100} = 0.5$$

$$t_x = \frac{xw_{max}.xv_{min} - xw_{min}.xv_{max}}{xw_{max} - xw_{min}} = \frac{300\times50 - 100\times150}{300-100} = 0$$

$$t_y = \frac{yw_{max}.yv_{min} - yw_{min}.yv_{max}}{yw_{max} - yw_{min}} = \frac{300\times50 - 100\times150}{300-100} = 0$$

The equation for mapping window coordinate to view port coordinate is given by,

$$xv = s_x xw + t_x$$
$$yv = s_y yw + t_y$$

Hence,

$$xv = 0.5 \times 200 + 0 = 100$$
$$yv = 0.5 \times 200 + 0 = 100$$

The transformed viewport coordinate is (100, 100).

Q. Find the normalization transformation matrix for window to viewport which uses the rectangle whose lower left corner is at (2, 2) and upper right corner is at (6, 10) as a window and the viewport that has lower left corner at (0, 0) and upper right corner at (1, 1).

Solution:

We have

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}} = \frac{1-0}{6-2} = 0.25$$

$$s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}} = \frac{1-0}{10-2} = 0.125$$

$$t_x = \frac{xw_{max}.xv_{min} - xw_{min}.xv_{max}}{xw_{max} - xw_{min}} = \frac{6\times0 - 2\times1}{6-2} = -0.5$$

$$t_y = \frac{yw_{max}.yv_{min} - yw_{min}.yv_{max}}{yw_{max} - yw_{min}} = \frac{10\times0 - 2\times1}{10-2} = -0.25$$

The composite transformation matrix for transforming the window coordinate to viewport coordinate is given as

$$T = T_{(t_x,t_y)} S_{(s_x,s_y)}$$

$$= \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.25 & 0 & -0.5 \\ 0 & 0.125 & -0.25 \\ 0 & 0 & 1 \end{bmatrix}$$

Q. A world coordinate & viewport have the following geometry:

Window (left, right, bottom, top) = (200, 600, 100, 400)

Viewport (left, bottom, width, height) = (0, 0, 800, 600)

The following vertices are drawn in the world:- P1: (356, 125), P2: (200, 354), P3: (230, 400), P4: (564, 200). What coordinate will each occupy in viewport.

Solution:

Here,

$(xw_{min}, yw_{min}) = (200,100)$

$(xw_{max}, yw_{max}) = (600,400)$

$(xv_{min}, yv_{min}) = (0,0)$

$(xv_{max}, yv_{max}) = (800,600)$

Then we have

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}} = \frac{800-0}{600-200} = 2$$

$$s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}} = \frac{600-0}{400-100} = 2$$

$$t_x = \frac{xw_{max}.xv_{min} - xw_{min}.xv_{max}}{xw_{max} - xw_{min}} = \frac{600\times0 - 200\times800}{600-200} = -400$$

$$t_y = \frac{yw_{max}.yv_{min} - yw_{min}.yv_{max}}{yw_{max} - yw_{min}} = \frac{400\times0 - 100\times600}{400-100} = -200$$

The composite transformation matrix for transforming the window coordinate to viewport coordinate is given as

$$M = T_{(t_x,t_y)} S_{(s_x,s_y)}$$

$$= \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & -400 \\ 0 & 2 & -200 \\ 0 & 0 & 1 \end{bmatrix}$$

Now,

$$P1' = M.P1 = \begin{bmatrix} 2 & 0 & -400 \\ 0 & 2 & -200 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 356 \\ 125 \\ 1 \end{bmatrix} = \begin{bmatrix} 312 \\ 50 \\ 1 \end{bmatrix} = (312, 50)$$

$$P2' = M.P2 = \begin{bmatrix} 2 & 0 & -400 \\ 0 & 2 & -200 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 200 \\ 354 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 508 \\ 1 \end{bmatrix} = (0, 508)$$

$$P3' = M.P3 = \begin{bmatrix} 2 & 0 & -400 \\ 0 & 2 & -200 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 230 \\ 400 \\ 1 \end{bmatrix} = \begin{bmatrix} 60 \\ 600 \\ 1 \end{bmatrix} = (60, 600)$$

$$P4' = M.P4 = \begin{bmatrix} 2 & 0 & -400 \\ 0 & 2 & -200 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 564 \\ 200 \\ 1 \end{bmatrix} = \begin{bmatrix} 728 \\ 200 \\ 1 \end{bmatrix} = (728, 200)$$

❖ Clipping

The process of discarding those parts of a picture which are outside of a specified region or window is called **clipping**. The procedure using which we can identify whether the portions of the graphics object is within or outside a specified region or space is called **clipping algorithm.**

The region or space which is used to see the object is called window and the region on which the object is shown is called view port.

Clipping is necessary to remove those portions of the object which are not necessary for further operations. It excludes unwanted graphics from the screen. So, there are three cases:

1. The object may be completely outside the viewing area defined by the window port.
2. The object may be seen partially in the window port.
3. The object may be seen completely in the window port.

For case 1 & 2, clipping operation is necessary but not for case 3.

Applications of clipping:

- Extracting part of a defined scene for viewing
- Identifying visible surfaces in three-dimensional views
- Antialiasing line segments or object boundaries
- Creating objects using solid-modeling procedures
- Displaying a multi-window environment
- Drawing and painting operations that allow parts of a picture to be selected for copying, moving, erasing, or duplicating.

➢ Point Clipping

Assuming that the clip window is a rectangle in standard position, we save a point P=(x, y) for display if the following inequalities are satisfied:

$$xw_{min} \leq x \leq xw_{max}$$

$$yw_{min} \leq y \leq yw_{max}$$

Where the edges of the clip window $(xw_{min}, xw_{max}, yw_{min}, yw_{max})$ can be either the world-coordinate window boundaries or viewport boundaries. If any one of these four inequalities is not satisfied, the point is clipped.

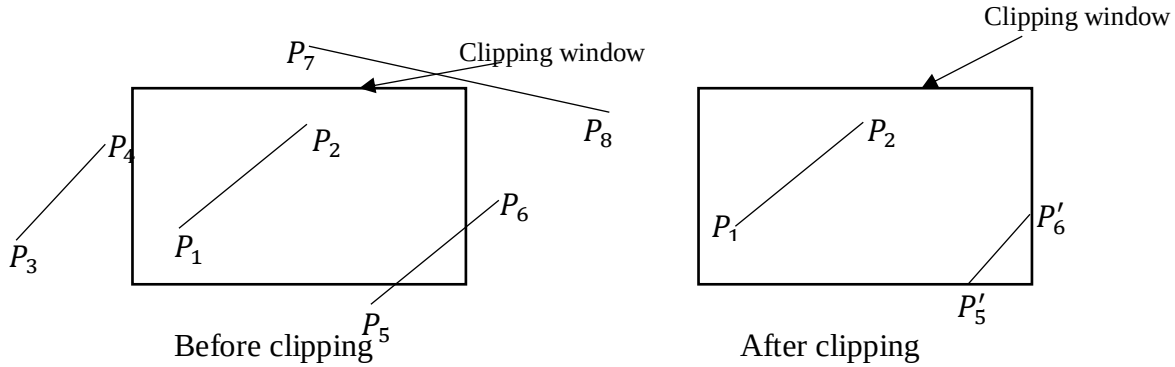
Algorithm:

1. Get the minimum and maximum coordinates of the viewing plane.
2. Get the coordinates for a point.
3. Check whether given input lies between minimum and maximum coordinates of viewing plane.
4. If yes display the point which lies inside the region otherwise discard it.

➢ Line Clipping

In line clipping, a line or part of line is clipped if it is outside the window port. There are three possibilities for the line:

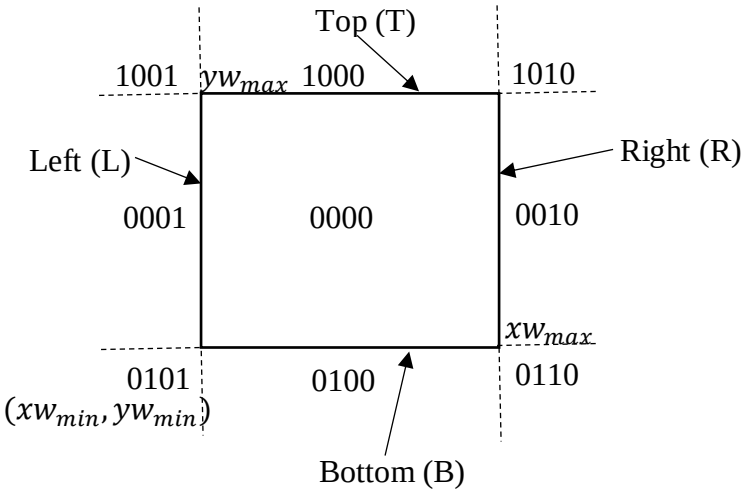
a. Line can be completely inside the window (This line should be accepted).
b. Line can be completely outside of the window (This line will be completely removed from the region).
c. Line can be partially inside the window (We will find intersection point and draw only that portion of line that is inside region).



Before clipping After clipping

Cohen-Sutherland Line Clipping Algorithm

1. Assign the four digit binary value called region code to each end point of a line.

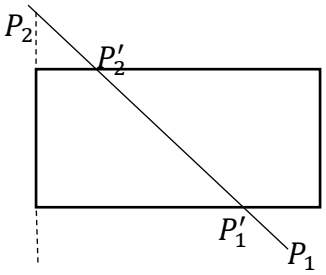


- A region code is represented as TBRL with 0000 inside clipping window.
- To calculate region code, perform following steps:
    a) Calculate the difference between endpoint coordinates and clipping boundary i.e. $x - xw_{min}, xw_{max} - x, y - yw_{min}$ & $yw_{max} - y$.
    b) Use '1' when resultant sign is $-ve$ otherwise use '0'.

2. Determine which lines are completely inside the clipping window & which lines are completely outside.

- Perform OR operation on line endpoint region code, if we get 0000, the line is completely inside clipping window & save these points.
- Perform AND operation on line endpoint region code, if we get value not equal to 0000, the line is completely outside & discard these points.

3. If condition 2 fails, the line crosses the clipping window boundary & find the point of intersection.



- Windows edges are processed in left, right, top and bottom. Here, the region code for point P_1 is 0100 & P_2 is 1001.
- To decide which boundary edges the line crosses, check for the bit position in line endpoint.
- Line crosses these window boundary edges for which bit position value are opposite.

Here,

	T	B	R	L
P1:	0	1	0	0
P2:	1	0	0	1

Since, the value at T, B & L are opposite, the line P1P2 crosses the clipping window at top, bottom & left edge.

- Now find the point of intersection with the clipping window edge.
- For calculation of intersection point, first find the slope,

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

For bottom edge intersection point; $y = yw_{min}$

$$x = x_1 + \frac{y - y_1}{m}$$

For top edge intersection point; $y = yw_{max}$

$$x = x_1 + \frac{y - y_1}{m}$$

For left edge intersection point; $x = xw_{min}$

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

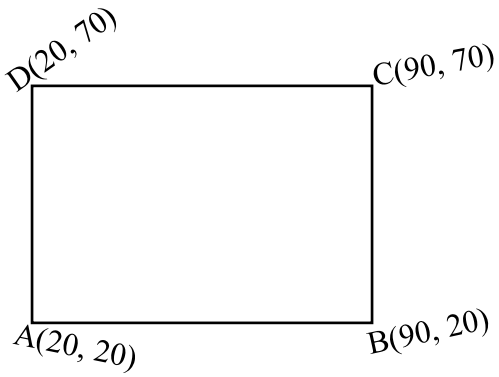
For right edge intersection point; $x = xw_{max}$

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

Examples:

Q. Consider a rectangle clipping window with A(20, 20), B(90, 20), C(90, 70) & D(20, 70). Clip the line P1P2 with P1(10, 30) & P2(80, 90) using Cohen-Sutherland line clipping algorithm.

Solution:



Here,

$xw_{min} = 20, xw_{max} = 90$

$yw_{min} = 20, yw_{max} = 70$

Region code for $P1(10, 30)$:

$x - xw_{min} = 10 - 20 = -10$	1	L
$xw_{max} - x = 90 - 10 = 80$	0	R
$y - yw_{min} = 30 - 20 = 10$	0	B
$yw_{max} - y = 70 - 30 = 40$	0	T

Region code for $P2(80, 90)$:

$x - xw_{min} = 80 - 20 = 60$ 0 L

$xw_{max} - x = 90 - 80 = 10$ 0 R

$y - yw_{min} = 90 - 20 = 70$ 0 B

$yw_{max} - y = 70 - 90 = -20$ 0 T

∴ Region code for P1= 1000

∴ Region code for P2= 0001

- Check if line P1P2 is completely inside or outside.

Take OR of 0001 & 1000

$$\begin{array}{r} 0001 \\ \underline{1000} \\ 1001 \end{array}$$ → Which is not equal to 0000 i.e. line P1P1 is not completely inside.

Take AND

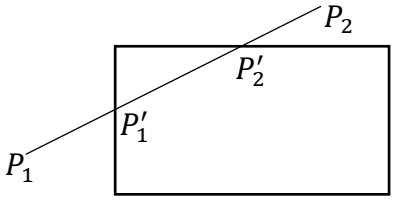
$$\begin{array}{r} 0001 \\ \underline{1000} \\ 0000 \end{array}$$ → Which means P1P2 is not completely outside.

i.e. line P1P2 crosses the clipping window.

Now,

	T	B	R	L
P1:	0	0	0	1
P2:	1	0	0	0

The line crosses the top and left edge of the clipping window.



Now find P_1' & P_2'.

Slope $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{90-30}{80-10} = \frac{6}{7}$

So for P_1'

$x = 20$

$y = 30 + \frac{6}{7}(20 - 10) = 38.5$

∴ $P_1' = (20, 38.5)$

So for P_2'

$$y = 70$$

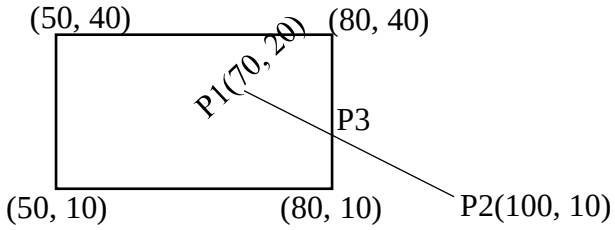
$$x = 10(90 - 60) \times \frac{7}{6} = 45$$

$\therefore\ P_2' = (45, 70)$

Thus the intersection point $P_1' = (20, 38.5)$ & $P_2' = (45, 70)$. So discarding the line segment that lie outside the boundary i.e. P_1P_1' & P_2P_2' , we get new line $P_1'P_2'$ with coordinate $P_1' = (20, 38.5)$ & $P_2' = (45, 70)$.

Q. Use the Cohen-Sutherland algorithm to clip the line P1(70, 20) and P2(100, 10) against a window lower left hand corner (50, 10) and upper right hand corner (80, 40).

Solution:



Assign 4 bit binary code to the two end point

P1=0000

P2=0010

Finding bitwise OR:

P1|P2=0000|0010=0010

Since P1|P2 !=0000, hence the two point doesn't lie completely inside the window.

Finding bitwise AND:

P1&P2=0000 & 0010=0000

Since P1&P2=0000, hence line is partially visible.

Now, finding the intersection of P1 and P2 with the boundary of window.

$p1(x_1, y_1) = (70, 20)$

$p2(x_2, y_2) = (100, 10)$

Slope $m = (10 - 20)/(100 - 70) = -1/3$

We have to find the intersection with right edge of window.

Here,

$x = 80$

$y = y_2 + m(x - x_2) = 10 + (-1/3)(80 - 100) = 10 + 6.67 = 16.67$

Thus the intersection point P3 = (80, 16.67). So, discarding the line segment that lie outside the boundary i.e. P3P2, we get new line P1P3 with coordinate P1(70, 20) and P3(80, 16.67).

Liang-Barsky Line Clipping Algorithm

This algorithm is considered to be the faster parametric line-clipping algorithm. The following concepts are used in this clipping:

1. The parametric equation of the line.
2. The inequalities describing the range of the clipping window which is used to determine the intersection between the line and the clip window.

The parametric equation of a line can be given by,

$x = x_1 + u\Delta x$

$y = y_1 + u\Delta y, \qquad 0 \leq u \leq 1$

Where, $\Delta x = x_2 - x_1$ & $\Delta y = y_2 - y_1$

Then, writing the point-clipping conditions in the parametric form:

$$xw_{min} \leq x_1 + u\Delta x \leq xw_{max}$$

$$yw_{min} \leq y_1 + u\Delta y \leq yw_{max}$$

The above four inequalities can be expressed as,

$$up_k \leq q_k$$

Where, k = 1, 2, 3, 4 (corresponds to the left, right, bottom, and top boundaries, respectively).

The parameters p & q are defined as,

$p_1 = -\Delta x$, $q_1 = x_1 - xw_{min}$ (Left Boundary)

$p_2 = \Delta x$, $q_2 = xw_{max} - x_1$ (Right Boundary)

$p_3 = -\Delta y$, $q_3 = y_1 - yw_{min}$ (Bottom Boundary)

$p_4 = \Delta y$, $q_4 = yw_{max} - y_1$ (Top Boundary)

When the line is parallel to a view window boundary, the p value for the boundary is zero.

When $p_k < 0$, as u increase line goes from the outside to inside (entering).

When $p_k > 0$, line goes from the inside to outside (existing).

When $p_k = 0$ & $q_k < 0$ then line is trivially invisible because it is outside view window.

When $p_k = 0$ & $q_k > 0$ then line is inside the corresponding window boundary.

Using the following conditions, the position of line can be determined:

Condition	Position of line
$p_k = 0$	Parallel to the clipping boundaries.
$p_k = 0$ & $q_k < 0$	Completely outside the boundary.
$p_k = 0$ & $q_k \geq 0$	Inside the parallel clipping boundary.
$p_k < 0$	Line proceeds from outside to inside.
$p_k > 0$	Line proceeds from inside to outside.

Parameters u_1 & u_2 can be calculated that define the part of line that lies within the clip rectangle. When,

1. $p_k < 0$, $maximum(0, \frac{q_k}{p_k})$ is taken.
2. $p_k > 0$, $minimum(0, \frac{q_k}{p_k})$ is taken.

If $u_1 > u_2$, the line is completely outside the clip window and it can be rejected. Otherwise, the endpoints of the clipped line are calculated from the two values of parameter u.

Algorithm:

1. Set $u_{min} = 0, u_{max} = 1$
2. Calculate the values of u $(u_{left}, u_{right}, u_{top}, u_{bottom})$
   i. If $u < u_{min}$ or $u > u_{max}$ ignore that and move to the next edge.
   ii. Else separate the u values as entering or existing values using the inner product.
   iii. If u is entering value, set $u_{min} = u$; if u is existing value, set $u_{max} = u$.
3. If $u_{min} < u_{max}$, draw a line from $(x_1 + u_{min}(x_2 - x_1), y_1 + u_{min}(y_2 - y_1))$ to $(x_1 + u_{max}(x_2 - x_1), y_1 + u_{max}(y_2 - y_1))$.
4. If the line crosses over the window, $(x_1 + u_{min}(x_2 - x_1), y_1 + u_{min}(y_2 - y_1))$ and $(x_1 + u_{max}(x_2 - x_1), y_1 + u_{max}(y_2 - y_1))$ are the intersection point of line and edge.

Example

Q. Apply Liang Barsky Line Clipping algorithm to the line with coordinates (30, 60) and (60, 25) against the window $(xw_{min}, yw_{min}) = (10, 10)$ and $(xw_{max}, yw_{max}) = (50, 50)$.

Solution:

Given,

$(xw_{min}, yw_{min}) = (10, 10)$ and

$(xw_{max}, yw_{max}) = (50, 50)$

Set $u_{min} = 0, u_{max} = 1$

$u_{left} = q_1/p_1$

$= x_1 - xw_{min}/-\Delta x$

$= 30 - \frac{10}{-(60-30)}$

= -0.67

$u_{right} = q_2/p_2$

$= xw_{max} - x_1/\Delta x$

= 50-30/(60-30)

= 0.67

$u_{bottom} = q_3/p_3$

$= y_1 - yw_{min}/-\Delta y$

$= 60 - 10/-(25 - 60)$

= 1.43

$u_{top} = q_4/p_4$

$= yw_{max} - y_1/\Delta y$

= 50-60/(25-60)

= 0.29

Since $u_{left} = -0.67$ which is less than u_{min}. Therefore we ignore it.

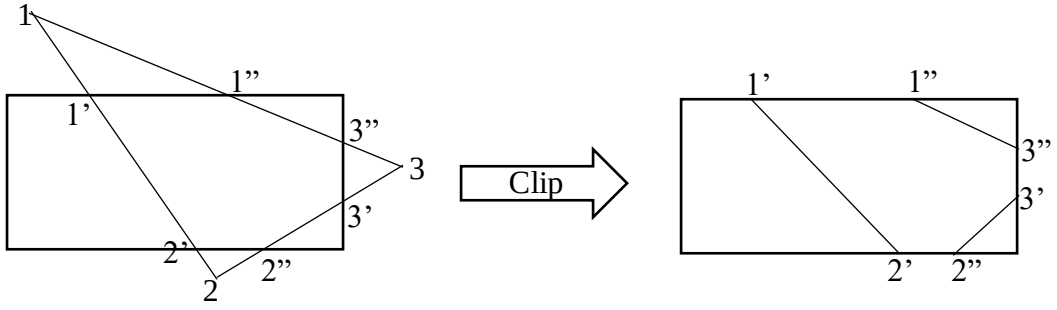
Similarly, $u_{bottom} = 1.43$ which is greater than u_{max}. So we ignore it,

$u_{right} = u_{min} = 0.67$ (Entering)

$u_{top} = u_{max} = 0.29$ (Exiting)

Since $u_{min} > u_{max}$, there is no line segment to draw .

- **Polygon Clipping**



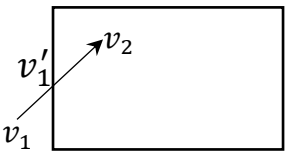
Sutherland-Hodgman Polygon Clipping Algorithm:

The Sutherland Hodgman algorithm is used for clipping polygons. 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 clipping 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.

To find new sequence of vertices four cases are considered.

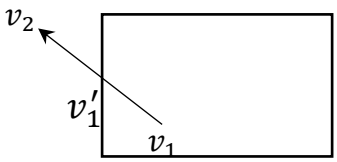
Case I



Movement: $out \rightarrow in$

Output: *intersection point, destination vertices i.e.*v_1' , v_2

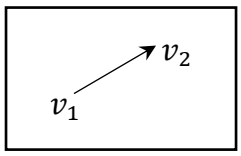
Case II



Movement: $in \rightarrow out$

Output: *intersection point i.e.* v_1'

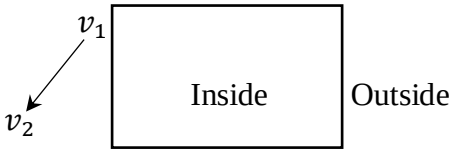
Case III



Movement: $in \rightarrow in$

Output: *destination vertices i.e.* v_2

Case IV



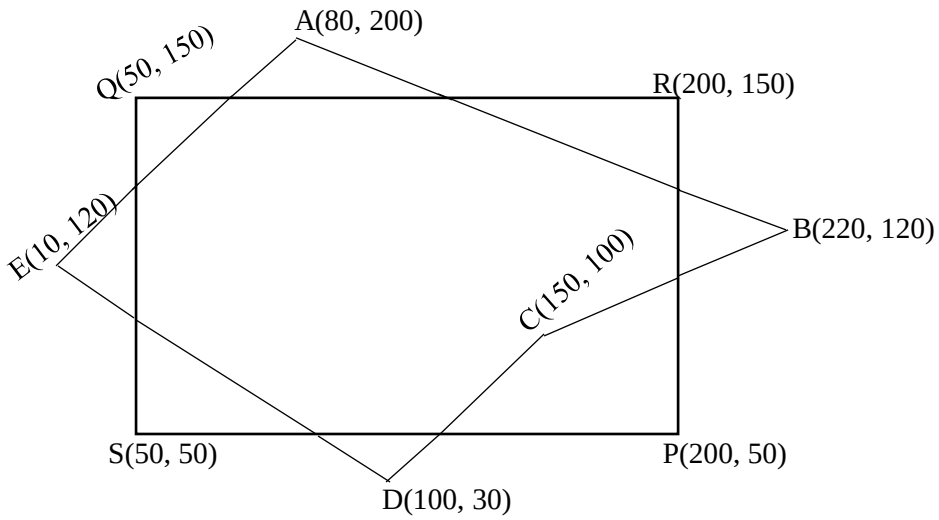
Movement: $out \rightarrow out$

Output: *none*

Example:

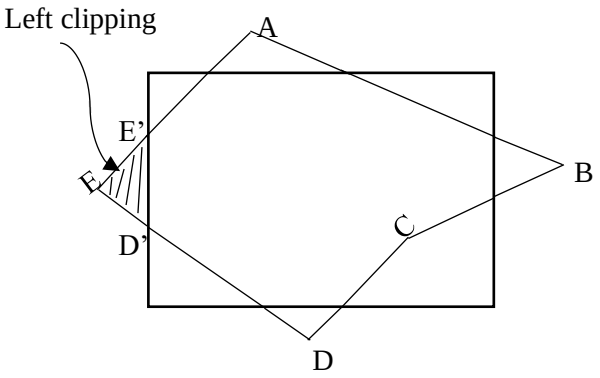
Q. Clip polygon against clipping window PQRS. The coordinates of polygon are A(80, 200), B(220, 120), C(150, 100), D(100, 30), E(10, 120). Coordinates of clipping window P(200, 50), Q(50, 150), R(200, 150) & S(50, 50).

Solution:



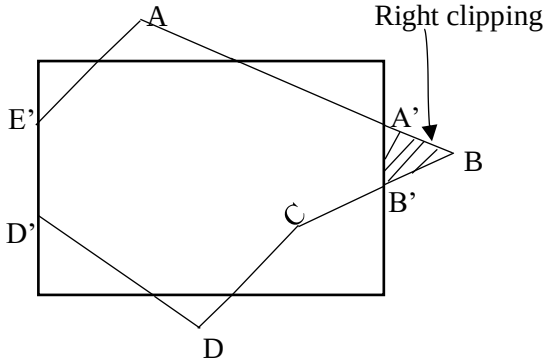
Left Clipping

vertex	case	O/P	
AB	$in \rightarrow in$	B	
BC	$in \rightarrow in$	C	
CD	$in \rightarrow in$	D	
DE	$in \rightarrow out$	D’	New vertex
EA	$out \rightarrow in$	E’, A	



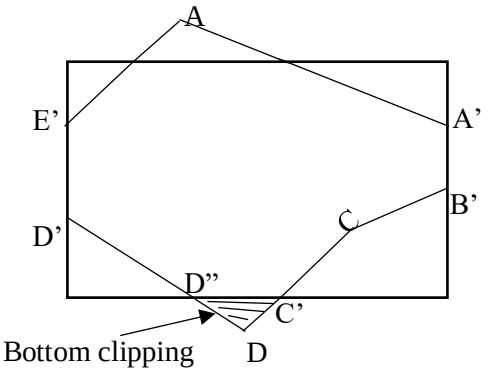
Right Clipping

vertex	case	O/P	
AB	$in \rightarrow out$	A’	New vertex
BC	$out \rightarrow in$	B’, C	
CD	$in \rightarrow in$	D	
DD’	$in \rightarrow in$	D’	
D’E’	$in \rightarrow in$	E’	
E’A	$in \rightarrow in$	A	



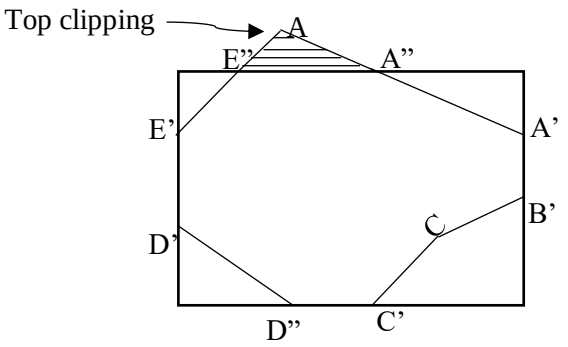
Bottom Clipping

vertex	case	O/P	
AA’	$in \rightarrow in$	A’	
A’B’	$in \rightarrow in$	B’	
B’C	$in \rightarrow in$	C	
CD	$in \rightarrow out$	C’	New vertex
DD’	$out \rightarrow in$	D”, D’	
D’E’	$in \rightarrow in$	E’	
E’A	$in \rightarrow in$	A	

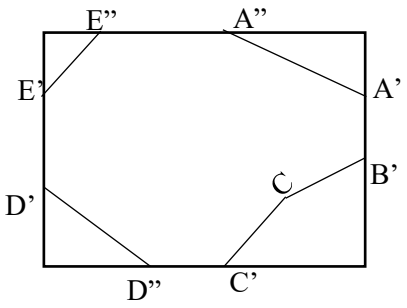


Top Clipping

vertex	case	O/P	
AA’	$out \rightarrow in$	A”, A	New vertex
A’B’	$in \rightarrow in$	B’	
B’C	$in \rightarrow in$	C	
CC’	$in \rightarrow in$	C’	
C’D”	$in \rightarrow in$	D”	
D”D’	$in \rightarrow in$	D’	
D’E’	$in \rightarrow in$	E’	New vertex
E’A	$in \rightarrow out$	E”	



Hence, final polygon after clipping:



References

- **Donald Hearne and M.Pauline Baker**, “Computer Graphics, C Versions.” Prentice Hall