Scanned by TapScanner

Unit-3

Two-Dimensional Geometric Transformations:

The operations that are applied to geometrical description of an object to change its position, orientation or size are called geometric transformations. Following are the types of transformations:

i) Translation
ii) Rotation } (transformations without change in shape) i.e, Rigid body transformation.
iii) Reflection

iv) Scaling } (transformations with change in shape) i.e, Non-rigid body transformation
v) Shearing

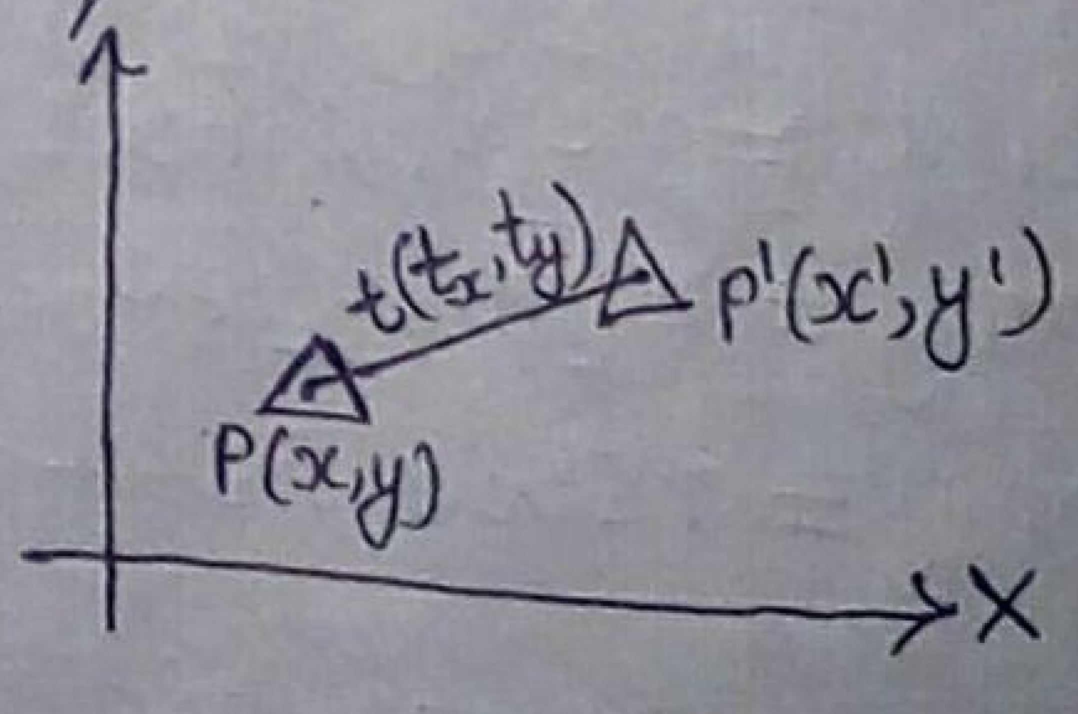
⊛. Why geometric transformations?

Ans:- In computer graphics, transformations of 2D objects are essential to many graphics applications like as a viewing aid, as a modeling tool, as an image manipulation tool etc. Transformations are needed to manipulate the initially created object and to display the modified object without redrawing it.

Rotation, Translation and scaling are major three transformations that are extensively used by most of all graphical packages. Other than these reflection and shearing transformations are also used by some graphical packages.

1) Translation:

The change in position of any object is called translation. Let $P(x,y)$ be any object translated by $t(t_x, t_y)$ to point $P'(x', y')$ as shown in the figure.



Here, $t(t_x, t_y)$ is the translation distance, t_x is translation in x-direction & t_y is translation in y-direction.

We can calculate translated points x' & y' as follows:-

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

$$\boxed{\text{i.e, } P' = T + P}$$

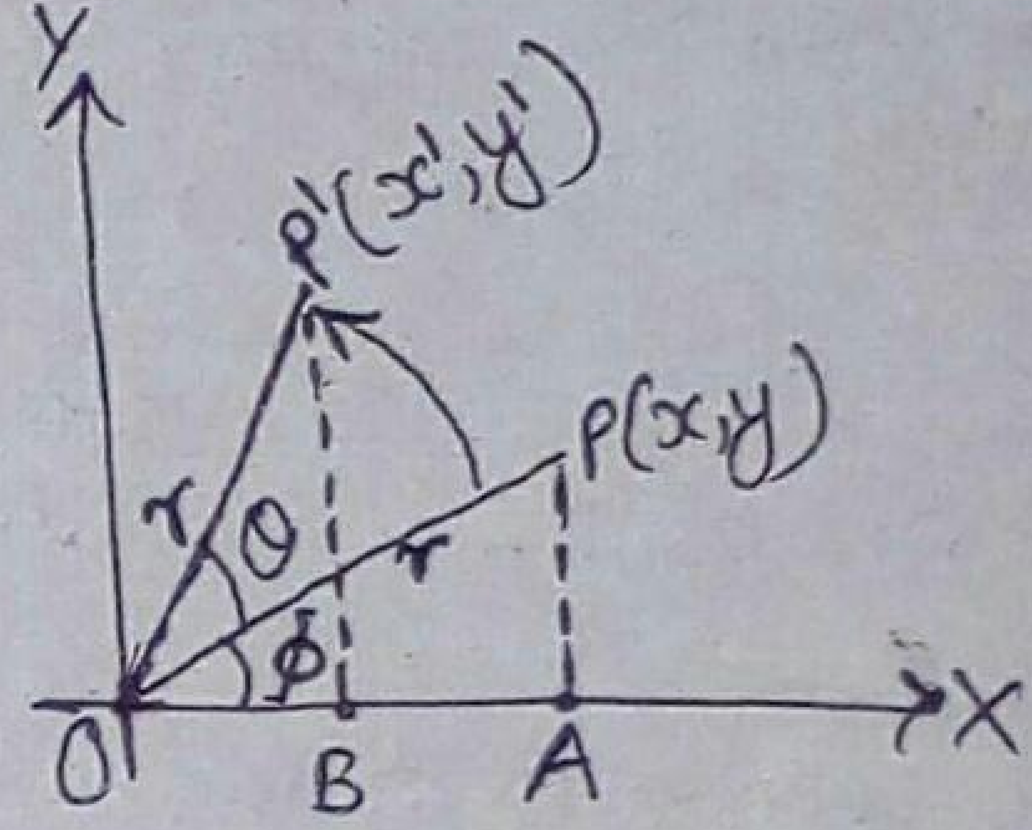
Now this can be written in matrix form as follows:-

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \quad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \quad P = \begin{bmatrix} x \\ y \end{bmatrix}$$

2) Rotation:

It is the process of changing the angle of the object. Rotation can be done by clockwise or anti-clockwise.

Let point $P(x,y)$ is at distance 'r' (i.e, radius) from origion making an angle ϕ with x-axis. Now if we rotate it with an angle θ we get new points as $P'(x',y')$ as shown in figure. New rotated points $P'(x',y')$ can be calculated as follows:-



$$x' = r\cos(\phi+\theta) \text{ — (i)}$$

or, $\cos(\phi+\theta) = \dfrac{x'}{r}$ $\quad \left(\because \cos\theta = \dfrac{\text{base}}{\text{hypotenuse}}\right)$

& $y' = r\sin(\phi+\theta)$ — (ii)

or, $\sin(\phi+\theta) = \dfrac{y'}{r}$ $\quad \left(\because \sin\theta = \dfrac{\text{perpendicular}}{\text{hypotenuse}}\right)$

Now taking eqn (i) and (ii)

$$x' = r\cos(\phi+\theta)$$

or, $x' = r[\cos\phi.\cos\theta - \sin\phi.\sin\theta]$ $\quad [\because \cos(A+B) = \cos A.\cos B - \sin A.\sin B]$.

or, $x' = r\cos\phi.\cos\theta - r.\sin\phi.\sin\theta$ — (iii)

& $y' = r\sin(\phi+\theta)$

or, $y' = r[\sin\phi.\cos\theta + \cos\phi.\sin\theta]$ $\quad [\because \sin(A+B) = \sin A.\cos B + \cos A.\sin B]$

or, $y' = r\sin\phi.\cos\theta + r\cos\phi.\sin\theta$ — (iv).

Now from triangle OAP

$\cos\phi = \dfrac{x}{r}$ $\quad \left(\because \cos\phi = \dfrac{\text{base}}{\text{hypotenuse}}\right)$

or, $x = r\cos\phi$ — (v).

& $\sin\phi = \dfrac{y}{r}$ $\quad \left(\because \sin\phi = \dfrac{\text{perpendicular}}{\text{hypotenuse}}\right)$

or, $y = r.\sin\phi$ — (vi).

Scanned by TapScanner

Now using value of (v) and (vi) in eqn (iii) and (iv).

$$x' = r\cos\phi.\cos\theta - r\sin\phi.\sin\theta$$

$$\text{or, } x' = x.\cos\theta - y\sin\theta \quad \text{— (vii)}$$

& Similarly $y' = r\sin\phi.\cos\theta + r.\cos\phi.\sin\theta$

$$\text{or, } y' = y.\cos\theta + x.\sin\theta \quad \text{— (viii)}$$

Hence we get final result as:

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

$$\& \; y' = x.\sin\theta + y.\cos\theta$$

This can be represented in matrix form as follows:-

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$P = \begin{bmatrix} x \\ y \end{bmatrix}$$

$P' = R(\theta).P$ where, $R(\theta)$ is rotation matrix with angle θ about origin.

$$\text{i.e. } \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix}$$

But when rotation is in clockwise direction then,

$$P' = R(-\theta).P$$

$$\text{i.e. } \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

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

$$\& \; y' = x\sin(-\theta) + y\cos(-\theta) = -x\sin\theta + y\cos\theta$$

3) Scaling:

Scaling is the process that alters size of an object. The size may increase or decrease on the basis of scaling factor.

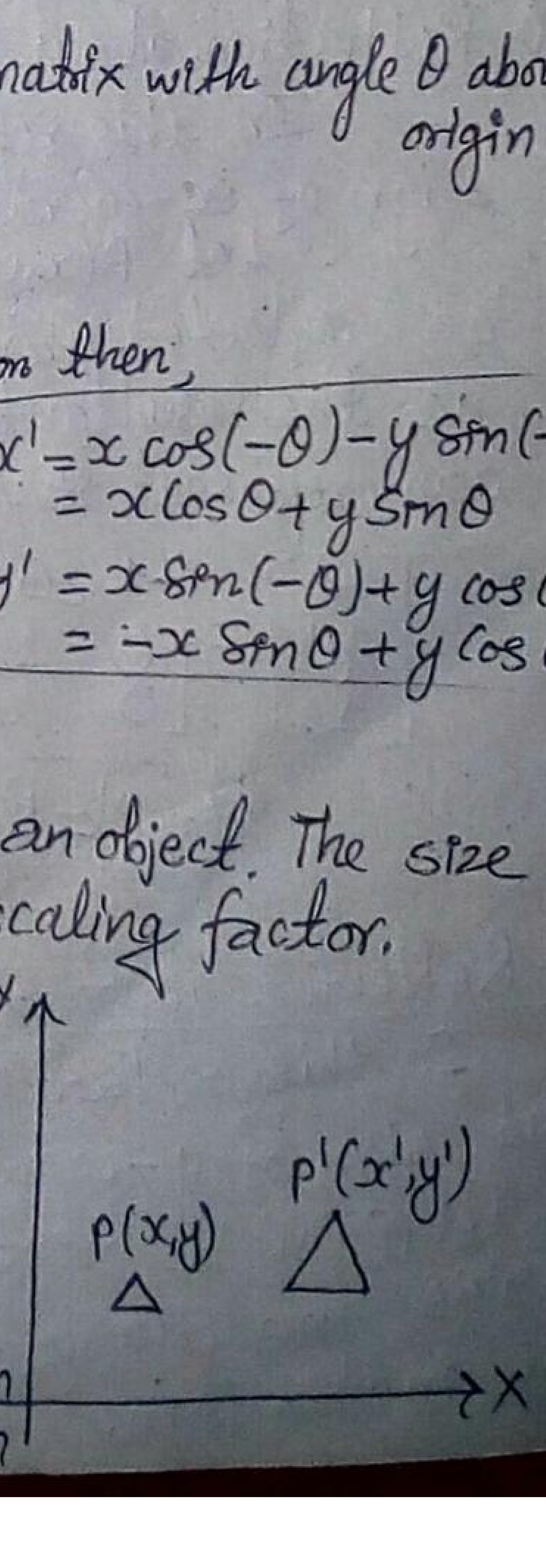
Let $P(x,y)$ be the object initially and $P'(x',y')$ be the object after scaling. So, final results x' and y' can be calculated as;

$$x' = x.S_x$$

$$y' = y.S_y$$

where, S_x = Scaling factor in x-direction

& S_y = Scaling factor in y-direction



Now the General form for scaling is; $\boxed{P' = S_n \cdot P}$

Now, $P' = S_n \cdot P$ can be written in matrix form as;

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

or, $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x.S_x + 0 \\ 0 + y.S_y \end{bmatrix}$$

or, $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x.S_x \\ y.S_y \end{bmatrix}$$ //

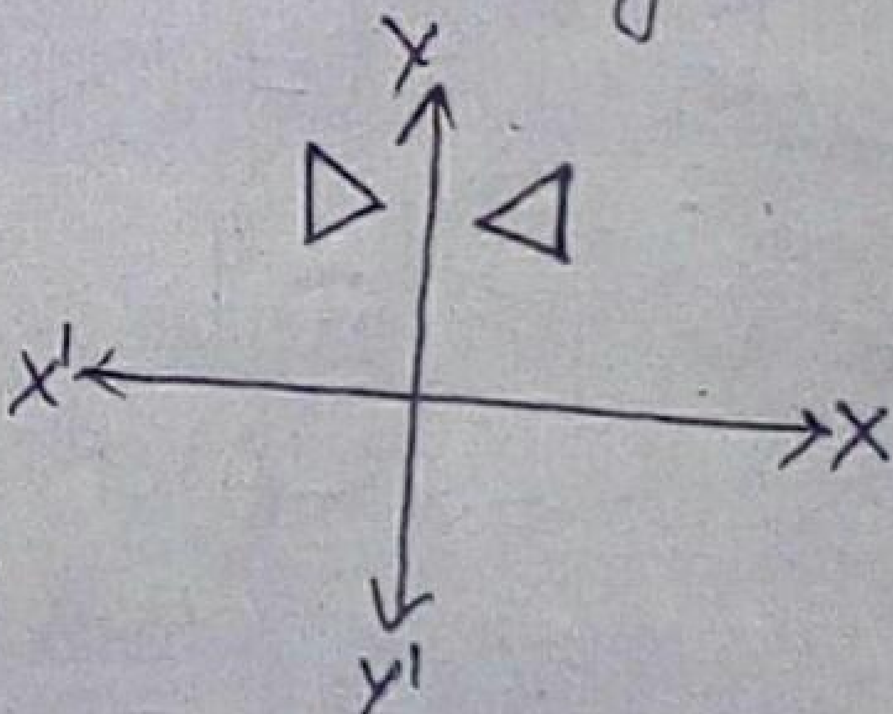
4) Reflection:

It is a transformation which produces a mirror image of an object. The mirror image can be either about x-axis or y-axis. The object is rotated by 180°. In 2D following types of reflections are performed.

i) Reflection about a line where x=0 in y-axis:-

In this reflection y-coordinate remains unchanged and sign of x-coordinate is altered.

Let $P(x,y)$ be point that is to be reflected about y-axis and $P'(x',y')$ be the resultant point then we can represent it in matrix form as follows;



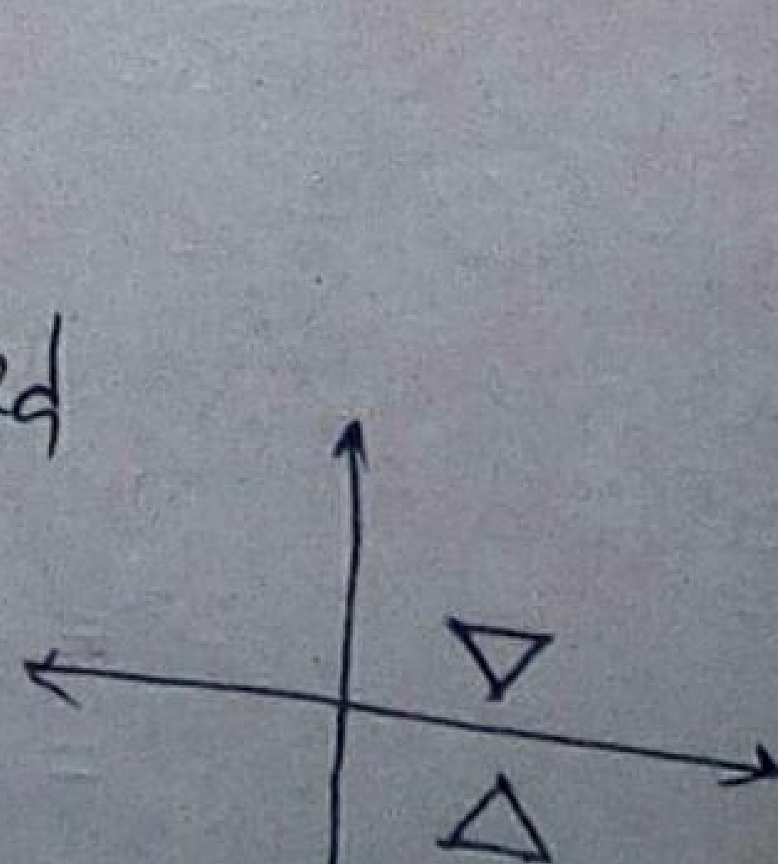
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

i.e, $\boxed{P' = R_y \cdot P}$

ii) Reflection about a line y=0 in x-axis:-

In this reflection x-coordinate remains unchanged and sign of y-coordinate is altered.

Let $P(x,y)$ be point that is to be reflected about x-axis and $P'(x',y')$ be the result point. Now in matrix form;



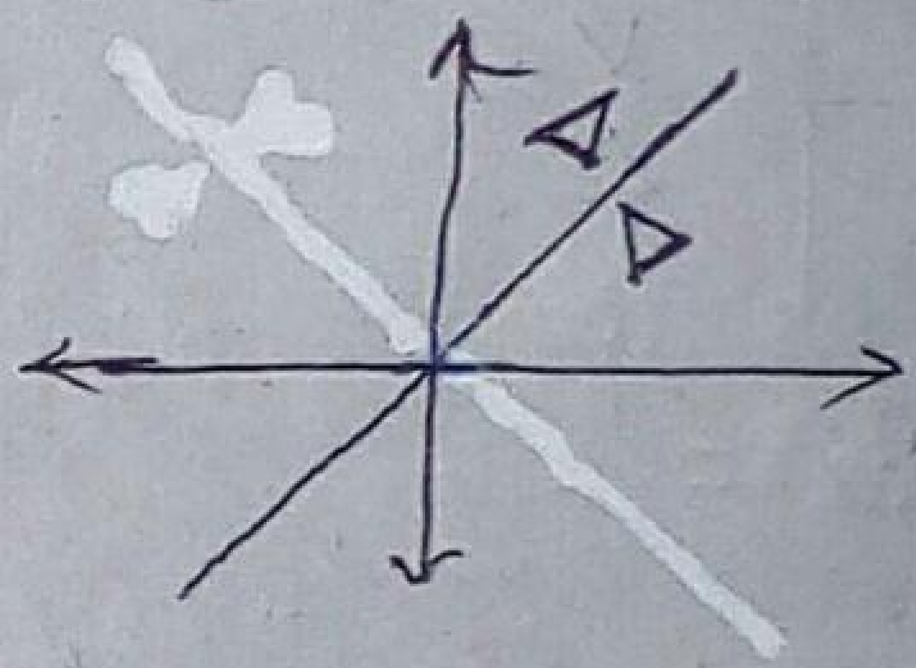
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

i.e, $\boxed{P' = R_x \cdot P}$

iii) Reflection about a line $y=x$

In this reflection the value of x and y-coordinate are swapped.

Let $P(x,y)$ be the point that is to be reflected about $y = x$ & $P'(x', y')$ be the resultant point.



In Matrix form:

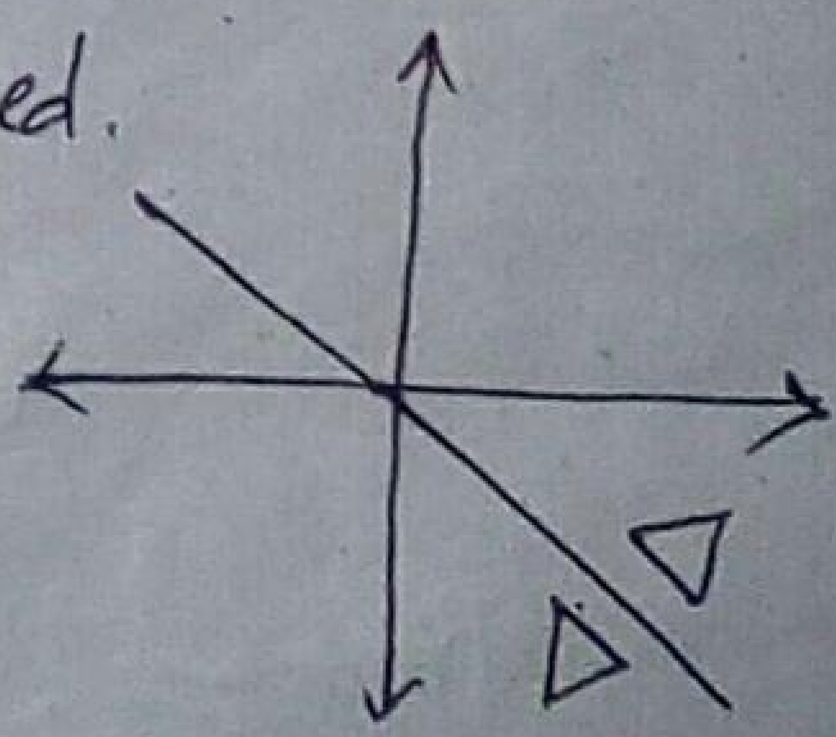
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

iv) Reflection about a line $y=-x$:

In this reflection the value of x and y-coordinate are swapped as well as sign of x & y are changed.

i.e, $x = -y$

$y = -x$



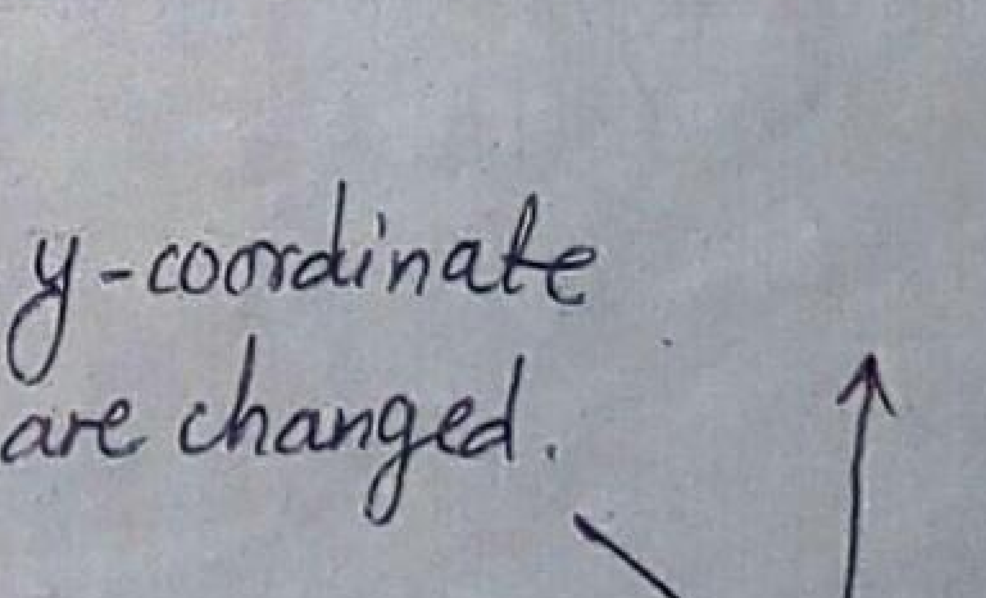
Equivalent matrix form is;

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

v) Reflection about a line perpendicular to origin

In Matrix Form

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



Signs of x and y coordinates are changed.

Scanned by TapScanner

5) Shearing:

A shear is a transformation that distorts the shape of an object along coordinate axes. It disturb the shape of an object such that the transformed shape appears as if the object were composed of interval layers that has been caused to slide over each other. Two common shearing transformations are those that shift coordinate x-values and those that shift y-values.

X-direction Shear:

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

and $y' = y$.

Y-direction Shear:

A y-direction shear relative to y-axis is produced by following transformation matrix equation.

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

which transforms $x' = x$

and $y' = sh_y \cdot x + y$.

Homogenous co-ordinate representation of 2D Transformations:

The homogenous co-ordinate system provides a uniform frame-work for handling different geometric transformation, simply as multiplication of matrices. In homogenous coordinate representation each 2D point (x, y) is represented as homogenous coordinate triple (x_h, y_h, h)

where, $x = \frac{x_h}{h}$, $y = \frac{y_h}{h}$

Note: h is usually 1 for 2D case.

It is called homogenous because it is possible to transform functions such as $f(x,y)$ into the form of $f(x/h, y/h)$ without disturbing degree of curve.

Q. Why homogenous coordinates?

Ans:- It is needed for following reasons;

i) It provides a uniform frame-work for handling different geometric transformations, simply as multiplication of matrices.

ii) To express any two-dimensional transformation as matrix multiplication.

iii) To represent all the transformation equations as matrix multiplications.

iv) To perform more than one transformation at a time.

v) To reduce unwanted calculations of intermediate steps, to save time and memory and produce a sequence of transformations.

Homogenous co-ordinate representation for translation:

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

$$\boxed{\therefore P' = T_H(t_x, t_y) \cdot P}$$

Image = transformation matrix x object.

where, t_x = translation along x-axis

& t_y = translation along y-axis.

Q. Translate a point P(7, 8) by translation vector T(-2,4).

Soln

Here, $P(7,8) = P(x,y)$ & $T(t_x, t_y) = (-2,4)$

$$x' = x + t_x = (7-2) = 5$$

$$y' = y + t_y = (8+4) = 12$$

$$P'(x', y') = (5, 12)$$

This is a method without homogenous coordinate system, now we solve using homogenous coordinate system as follows:-

$$P(x,y) \xrightarrow{H} P(x,y,1)$$

$$P'(x',y') \xrightarrow{H} P'(x',y',1)$$

So, $$T_H(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$\therefore P' = T_h(t_x, t_y) \cdot P$

$$= \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 5 \\ 12 \\ 1 \end{bmatrix}$$

Homogenous coordinate representation for rotation:

Let $P(x, y)$ be the point that is rotated about origin with rotation angle θ and $P'(x', y')$ be the resultant point.

$$P(x, y) \xrightarrow{H} P(x, y, 1)$$
$$P(x', y') \xrightarrow{H} P'(x', y', 1)$$

we know that,

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

Now in homogenous coordinate matrix form:

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

$$\boxed{\therefore P' = R_H(\theta) \cdot P}$$

For inverse rotation $\boxed{P' = R_H(-\theta) \cdot P}$

Homogenous coordinate representation for scaling:

Let $P(x, y)$ be the point that is to be scaled w.r.t. origin with scaling factor $S(S_x, S_y)$ and $P'(x', y')$ be the resultant point.

$$P(x, y) \xrightarrow{H} P(x, y, 1)$$
$$P'(x', y') \xrightarrow{H} P'(x', y', 1)$$

we know that,

$$x' = x \cdot S_x$$
$$y' = y \cdot S_y$$

Scanned by TapScanner

Now in homogenous coordinate matrix form:

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

$$\boxed{\therefore P' = S_H(S_x, S_y) \cdot P}$$

For inverse scaling

$$\boxed{P' = S_H\left(\frac{1}{S_x}, \frac{1}{S_y}\right) \cdot P}$$

⊛ Numerical Questions:

Q1. A point (4,3) is rotated clockwise by an angle of 45°. Find the rotation matrix and the resultant point.

Solution:

Clockwise rotation matrix with angle θ (i.e. Transformation matrix)

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

$$= \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ \\ \sin 45^\circ & \cos 45^\circ \end{bmatrix}$$

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

we have, $P' = R(\theta).P$

i.e. Image = Transformation matrix × Object.

or, $$P' = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} \frac{4}{\sqrt{2}} - \frac{3}{\sqrt{2}} \\ \frac{4}{\sqrt{2}} + \frac{3}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{7}{\sqrt{2}} \end{bmatrix}$$

Q2. Find the transformation of triangle A(1,0), B(0,1) and C(1,1) by rotating 90° about the origin and then translating one unit in x and y direction.

Solution:

$$\text{Object} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Scanned by TapScanner

Transformation Matrix = T(1, 1) · R(90°)

translation in homogenous coordinate system
$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation in homogenous coordinate system

$$= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ & 0 \\ \sin 90^\circ & \cos 90^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

Q3. Perform a counter clockwise 45° rotation of a triangle A(2,3), B(5,5) & C(4,3) about point (1,1).

Solution:

$$\text{Object} = \begin{bmatrix} 2 & 5 & 4 \\ 3 & 5 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$

translation by (1,1)

rotation by 45°

for counter clockwise (translation with the -ve

Composite Transformation Matrix = T(1,1) × R(45°) × T(−1,−1)

$$= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

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

We have, Image = Transformation matrix × Object.

$$\text{or, Image} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\sqrt{2}+1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 5 & 4 \\ 3 & 5 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$

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

Scanned by TapScanner

⊛ Fixed point Scaling:

To control the location of scaled object we can choose the position called fixed point. Let coordinates of fixed point = (x_f, y_f). For a vertex with coordinate (x, y) the scaled co-ordinates (x', y') are calculated as;

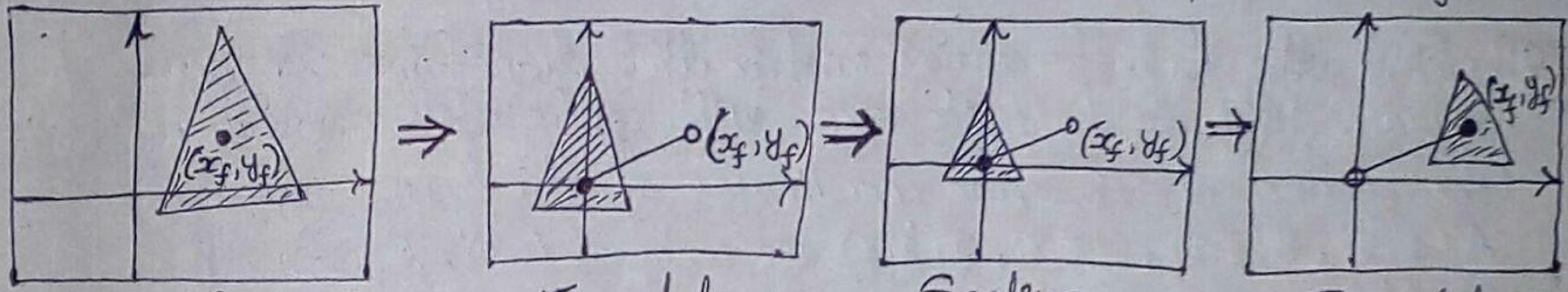
$$x' = x_f + (x - x_f) \cdot S_x$$

or $$x' = x \cdot S_x + (1 - S_x) x_f$$

& $$y' = y_f + (y - y_f) \cdot S_y$$

or $$y' = y \cdot S_y + (1 - S_y) \cdot y_f$$

where, $(1 - S_x) \cdot x_f$ and $(1 - S_y) \cdot y_f$ are constant for all points in object.



Original (Initial) ⇒ Translate ⇒ Scaling ⇒ Translate Again to initial position

figure:- Process of Scaling of a triangle about a fixed point (x_f, y_f).

translate — Scaling — again translate to get original — Gives scaling (S_x, S_y) at fixed point (x_f, y_f)

$$T(x_f, y_f) \cdot S(S_x, S_y) \cdot T(-x_f, -y_f) = S(x_f, y_f, S_x, S_y)$$

Now, in matrix form

$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & x_f(1 - S_x) \\ 0 & S_y & y_f(1 - S_y) \\ 0 & 0 & 1 \end{bmatrix}$$

Hence the homogenous matrix equation for fixed point scaling is;

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & x_f(1 - S_x) \\ 0 & S_y & y_f(1 - S_y) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Q1. Scale the polygon with coordinates A(2,5), B(7,10) and C(10,2) by two units in x-direction and two units in y-direction.

Solution

Here, $S_x = 2$ and $S_y = 2$

Therefore transformation matrix is given as $S = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

Scanned by TapScanner

The object matrix is; $\text{Object} = \begin{bmatrix} 2 & 7 & 10 \\ 5 & 10 & 2 \end{bmatrix}$

We have,

Image = Transformation Matrix × Object.

$$= \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \times \begin{bmatrix} 2 & 7 & 10 \\ 5 & 10 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 14 & 20 \\ 10 & 20 & 4 \end{bmatrix}.$$

∴ The coordinates of the image after scaling are $A'(4,10)$, $B'(14,20)$ and $C'(20,4)$.

Q2. Find the transformation matrix that transforms the given square ABCD to half its size with centre still remaining at the same position. The coordinates of the square are; $A(1,1)$, $B(3,1)$, $C(3,3)$, $D(1,3)$ and centre at $(2,2)$. Also find the resultant coordinates of square.

Solution:-

This transformation can be carried out in the following steps.

i) Translate the square so that its centre coincides with origin.
ii) Scale the square with respect to the origin.
iii) Translate the square back to its original position.

Here,

$$\text{Object} = \begin{bmatrix} 1 & 3 & 3 & 1 \\ 1 & 1 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

दिएको point हरु matrix form मा।

base मा सँधै 1 homogenous बनाको object लाई

Now the transformation matrix is obtained by multiplication of three matrices as;

Composite Transformation matrix $= T(x_f, y_f) \times S(S_x, S_y) \times T(-x_f, -y_f)$

Simply Transformation matrix

Half it's size given in question

$$= T(2,2) \times S(0.5, 0.5) \times T(-2,-2)$$

$$= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$x_f(1-S_x) = 2(1-0.5) = 1$

$y_f(1-S_y) = 2(1-0.5) = 1$

यो दुई वाहेक अरु $S(0.5, 0.5)$ को सँधै जस्तकि त्यस्तै

Scanned by TapScanner

We have,

Image = Transformation Matrix X Object.

$$\text{Image} = \begin{bmatrix} 0.5 & 0 & 1 \\ 0 & 0.5 & 1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 3 & 3 & 1 \\ 1 & 1 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.5 & 2.5 & 2.5 & 1.5 \\ 1.5 & 1.5 & 2.5 & 2.5 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Hence coordinates of resultant image are; $A'(1.5, 1.5)$, $B'(2.5, 1.5)$, $C'(2.5, 2.5)$ and $D'(1.5, 2.5)$.

Q3. Find out the final coordinates of a figure bounded by the coordinates (1,1), (3,4), (5,7), (10,3) when rotated about a point (8,8) by 30° in clockwise direction and scaled by two unit in x-direction and three unit in y-direction.

Solution:

$$\text{Object} = \begin{bmatrix} 1 & 3 & 5 & 10 \\ 1 & 4 & 7 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Now, the transformation matrix is obtained by multiplication of four matrices as;

$$\text{Transformation Matrix} = T(x_f, y_f) \times S(S_x, S_y) \times R(-\theta) \times T(-x_f, -y_f).$$

$$= T(8,8) \times S(2,3) \times R(-30°) \times T(-8,-8).$$

$$= \begin{bmatrix} 1 & 0 & 8 \\ 0 & 1 & 8 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos 30° & \sin 30° & 0 \\ -\sin 30° & \cos 30° & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -8 \\ 0 & 1 & -8 \\ 0 & 0 & 1 \end{bmatrix}$$

On multiplying T(8,8) and T(-8,8) we get Same matrix S(2,3) only change is done at points $x_f(1-S_x) = 8(1-2) = -8$ & $y_f(1-S_y) = 8(1-3) = -16$ & Matrix $R(-30°)$ is as it is written. This is same that we did in previous question difference is additional $R(-30°)$ only.

$$= \begin{bmatrix} 2 & 0 & -8 \\ 0 & 3 & -16 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.866 & 0.5 & 0 \\ -0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.732 & 1 & -8 \\ -1.5 & 2.598 & -16 \\ 0 & 0 & 1 \end{bmatrix}$$

Now, We have,

Image = Transformation Matrix X Object.

i.e, Image = $\begin{bmatrix} 1.732 & 1 & -8 \\ -1.5 & 2.598 & -16 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 3 & 5 & 10 \\ 1 & 4 & 7 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

$$= \begin{bmatrix} 1.732+1-8 & 5.196+4-8 & 8.66+7-8 & 17.32+3-8 \\ -1.5+2.598-6 & -4.5+10.392-16 & -7.5+18.186-16 & -15+7.794-16 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -5.268 & 1.196 & 7.66 & 12.32 \\ -14.902 & -10.108 & -5.314 & -23.206 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Hence the final coordinates after transformation are; (-5.268,-14.902), (1.196,-10.108), (7.66,-5.314) and (12.32,-23.206).

Two Dimensional Viewing:

2-D viewing is the process of producing different views of a 2D secence. For e.g. translated view, rotated view etc. Graphical package allows us to define part of the scene to be displayed and part of screen where defined view is displayed. In computer graphics several coordinate systems are used to construct and display a 2D scene. Some of which are as follows:-

i) Modeling coordinate system → It defines shape and size of individual objects to create a graphical scene. Arbitrary units are used to specify modeling coordinates (MC's).

ii) World coordinate system → It is used to organize the individual objects (i.e, points, lines, circles etc) into a scene. A scene is made up of collection of objects. The objects make up the world (or scene) that we want to view, and the coordinates that we use to define the scene are called world coordinates (WC's).

iii) Viewing coordinate system → It is used to define particular view of a 2D scene. For a 2-D picture, a view is selected by specifying a subarea of the total picture area. In short it is called VC's.

iv) Normalized viewing coordinate system → Normalized viewing coordinates (NVC's) are the viewing coordinates between 0 and 1. They are used to make the viewing process independent of the output device.

v) Device coordinate or Screen coordinate system → Device coordinate system (DC's) are used to define coordinates in an output device. Device coordinates are integers within the range (0,0) to (x_{max}, y_{max}) for a particular output device.

⇒ Note: Above these 5 points (i) to (v) are viewing pipeline

⊛. What is 2D viewing transformation? (Please refer video link provided to drive before starting this for better understanding)

Ans: The process of mapping of a part of world coordinate scene to device coordinate is called the 2D viewing transformation. Transformations from world to device coordinates involves translation, rotation and scaling operations, as well as procedures for deleting those parts of picture that are outside the limits of a selected display area (clipping).

→ Clipping is necessary to discard that part of a 2D scene that lie outside of window.

→ Clipping can be performed in world co-ordinate or device coordinate.

Window: A world coordinate area selected for display is called a window. It is the section of 2D scene selected for viewing.

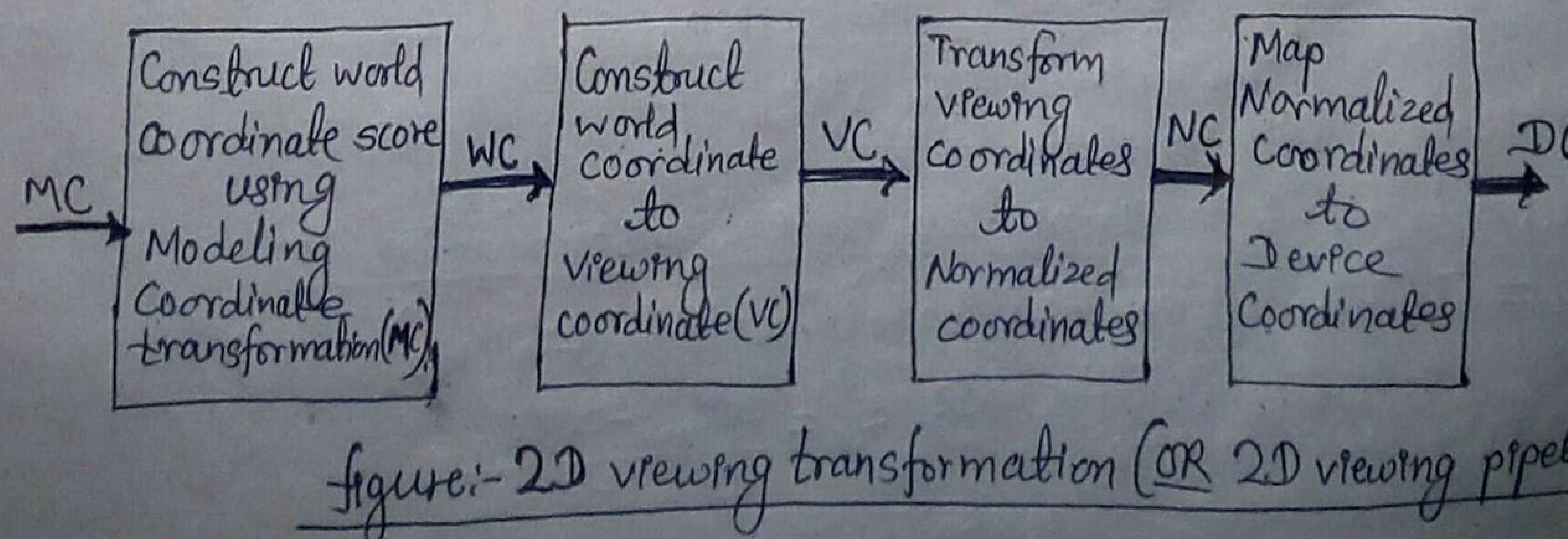


figure:- 2D viewing transformation (OR 2D viewing pipeline)

Viewport:- A area on a display device to which a window is mapped is called a viewport. The viewport indicates where selected part will be displayed on an output device.

Window and viewport are often rectangular in standard positions, because it simplifies transformation and clipping process.

Window to viewport coordinate transformation:

→ A window can be specified by four world coordinates; xw_{min}, xw_{max}, yw_{min} and yw_{max}.

→ Similarly a viewport can be described by four coordinates; xv_{min}, xv_{max}, yv_{min} and yv_{max}.

The window-to-viewport transformation can be explained in three steps as below;

Step1: Translate object along with window such that the lower left corner of the window is at origin.

$$\boxed{\text{i.e, apply } T(-xw_{min}, -yw_{min})}$$

Step2: Scale the object and the window such that window has the same dimension as that of a viewport. Simply we can say that converting the object into image and window into viewport.

$$\boxed{\text{i.e, apply } S(S_x, S_y)}$$

Step3: Finally apply another translation to move the viewport to its original position on the screen.

$$\boxed{\text{i.e, apply } T(xv_{min}, yv_{min})}$$

Therefore, net transformation (composite transformation) is;

$$T_{WV} = T(xv_{min}, yv_{min}) \cdot S(S_x, S_y) \cdot T(-xw_{min}, -yw_{min}) \quad \text{--- (i)}$$

where, S_x and S_y are scaling factors: $S_x = \dfrac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$

& $S_y = \dfrac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$

Scanned by TapScanner

Now, writing in matrix form:

$$T_{WV} = \begin{bmatrix} 1 & 0 & xv_{min} \\ 0 & 1 & yv_{min} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -yw_{min} \\ 0 & 1 & -yw_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

This is required window-to-viewport transformation. Let $P(x,y)$ be the world coordinate point that is mapped onto the viewport point $P'(u,v)$, then we must have,

$$\boxed{P' = T_{WV} \cdot P}$$

Example: Find the normalization transformation window to viewport, with window, lower left corner at (1,1) and upper right corner at (3,5) onto a viewport with lower left corner at (0,0) and upper right corner at (0.5, 0.5).

Solution:

Coordinates for window:

$xw_{min} = 1$ $\quad yw_{min} = 1$

$xw_{max} = 3$ $\quad yw_{max} = 5$

Coordinates for viewport:

$xv_{min} = 0$ $\quad yv_{min} = 0$

$xv_{max} = 0.5$ $\quad yv_{max} = 0.5$

We know that,

$$S_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}} = \frac{0.5-0}{3-1} = 0.25$$

& $$S_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}} = \frac{0.5-0}{5-1} = 0.125$$

Now, Composite Transformation matrix is; $T_{WV} = T(xv_{min}, yv_{min}) \cdot S(S_x, S_y) \cdot T(-xw_{min}, -yw_{min})$

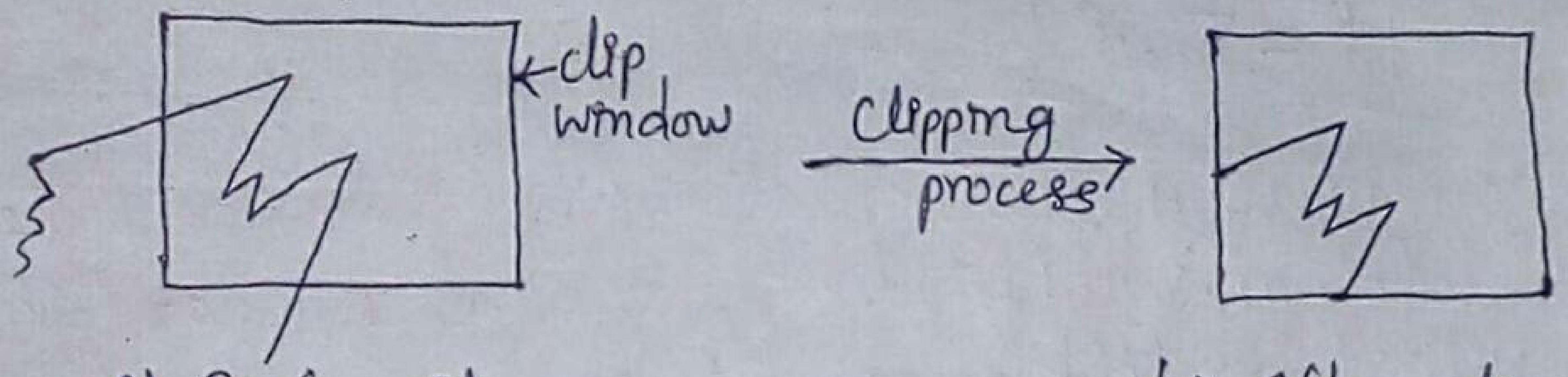
$$\text{or, } T_{WV} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 0.125 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.25 & 0 & -0.250 \\ 0 & 0.125 & -0.125 \\ 0 & 0 & 1 \end{bmatrix} \begin{array}{l} \rightarrow xv_{min}(1-S_x) \\ \rightarrow yv_{min}(1-S_y) \\ \text{others same as it is in S} \end{array}$$

Scanned by TapScanner

Clipping:

Clipping is the process of discarding (cutting off) those parts of an object which are outside of specified region (windows) is called clipping. Any procedure that identifies those parts of a picture that are either inside or outside of the specified region is called a clipping algorithm. The region against which the clipping operation is performed is called a clip window.



a) Before Clipping b) After clipping.

fig. Clipping process:

Applications of Clipping:

i) Extracting part of a defined scene for viewing.
ii) Identifying visible surfaces in three-dimensional views.
iii) Anti-aliasing line segments or object boundaries.
iv) Creating objects using solid-modeling procedures.
v) Displaying a multi-window environment.

Types of clipping:

i) Point Clipping:

→ Given a clipping window $x_{min}, x_{max}, y_{min}$ & y_{max} and coordinate point $P(x,y)$ as shown in figure:

→ $P(x,y)$ is accepted for display iff

$$x_{min} \le x \le x_{max}$$

$$y_{min} \le y \le y_{max}$$

→ Otherwise point is clipped. i.e, $P(x,y)$ is discarded.

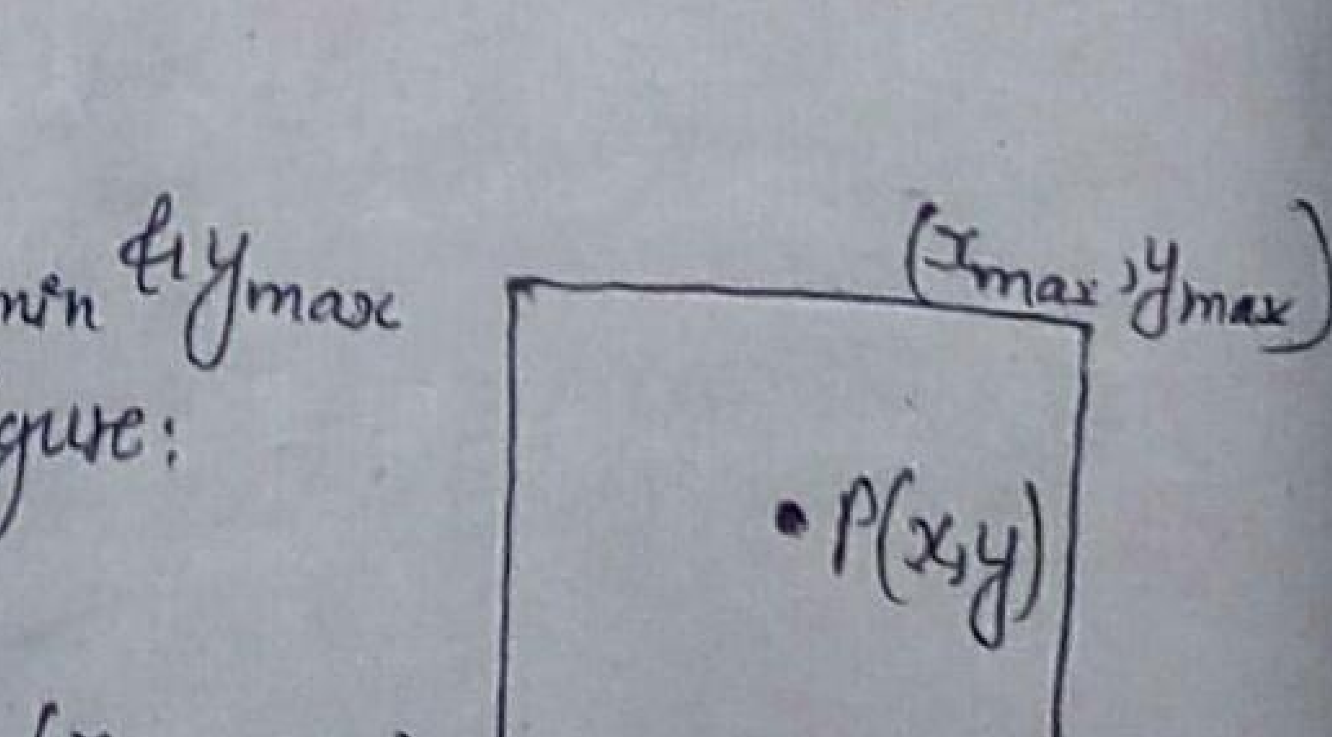


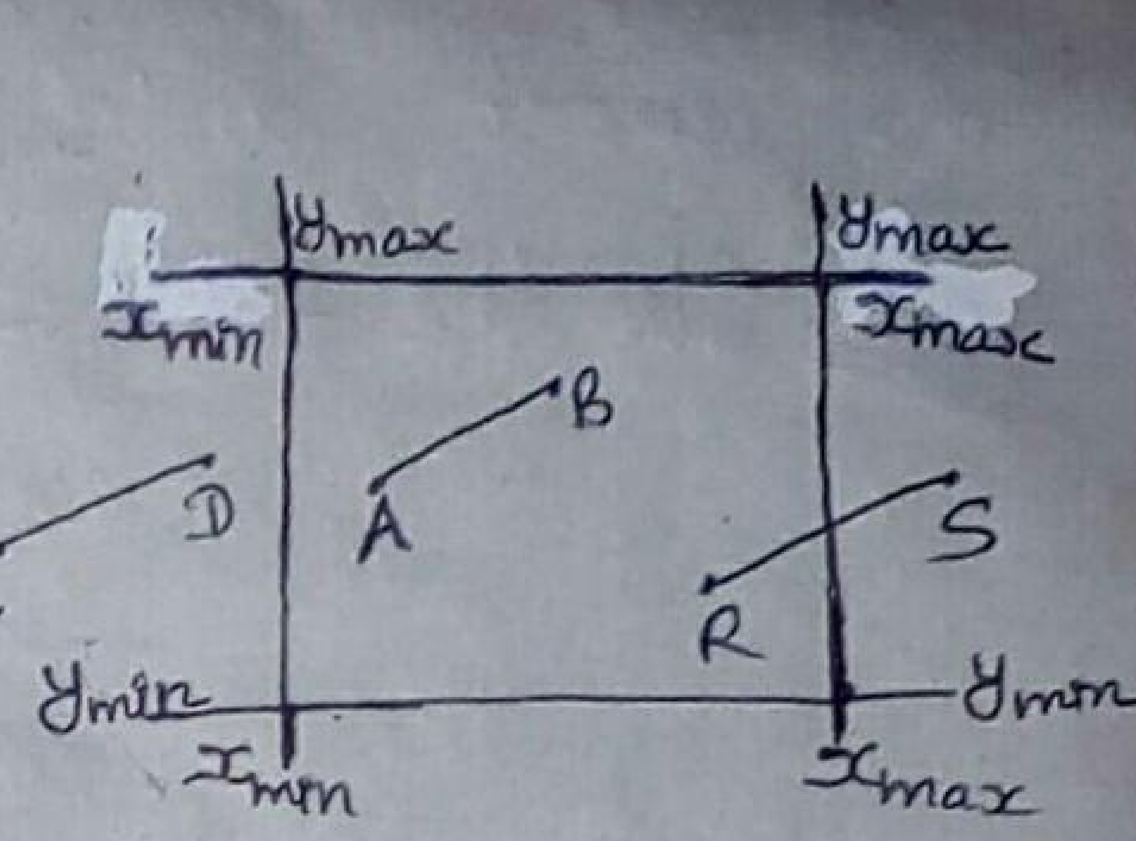
fig. Point Clipping

ii) Line clipping:

→ It is the process of discarding those parts of a line that lie outside of a specified region. (i.e, clipping window).

→ Given a clipping window ($x_{min}, x_{max}, y_{min}, y_{max}$) then there are three possible cases:-

i) Visible: → Line lies completely inside the window.
→ Accept and display the line.
→ E.g. AB line in the figure.



ii) Completely invisible:
→ Line lies completely outside of the clipping window.
→ Discard the line segment
→ E.g. CD line in the figure.

iii) Partially visible:
→ Part of line segment lies inside of the clipping window
→ In this case we need to compute intersection points and clip at the intersection points.
→ E.g. RS line in the figure.

1) Cohen Sutherland line clipping:

This is one of the oldest and most popular line-clipping algorithm. In this method, coordinate system is divided into nine regions. All regions have their associated region codes ~~Every line endpoint is assigned a four digit binary code.~~ Each bit in the code is set to either 1 (true) or 0 (false). Each region is assigned a four bit pattern as shown in figure.

1001	1000 Top	1010
0001 Left	0000 Window	0010 Right
0101	0100 Bottom	0110

fig. Region codes according to TBRL

Any point inside the clipping window has a region code 0000. For any endpoint (x,y) of a line, the code can be determined ~~ac~~ that identifies in which region the endpoint lies. The region code bits are set according to following conditions:

i) First bit is set to 1 if point lies towards left of window.
ii) Second bit is set to 1 if point lies towards right of window.
iii) Third bit is set to 1 if point lies at bottom of window.
iv) Fourth bit is set to 1 if point lies at top of the window.

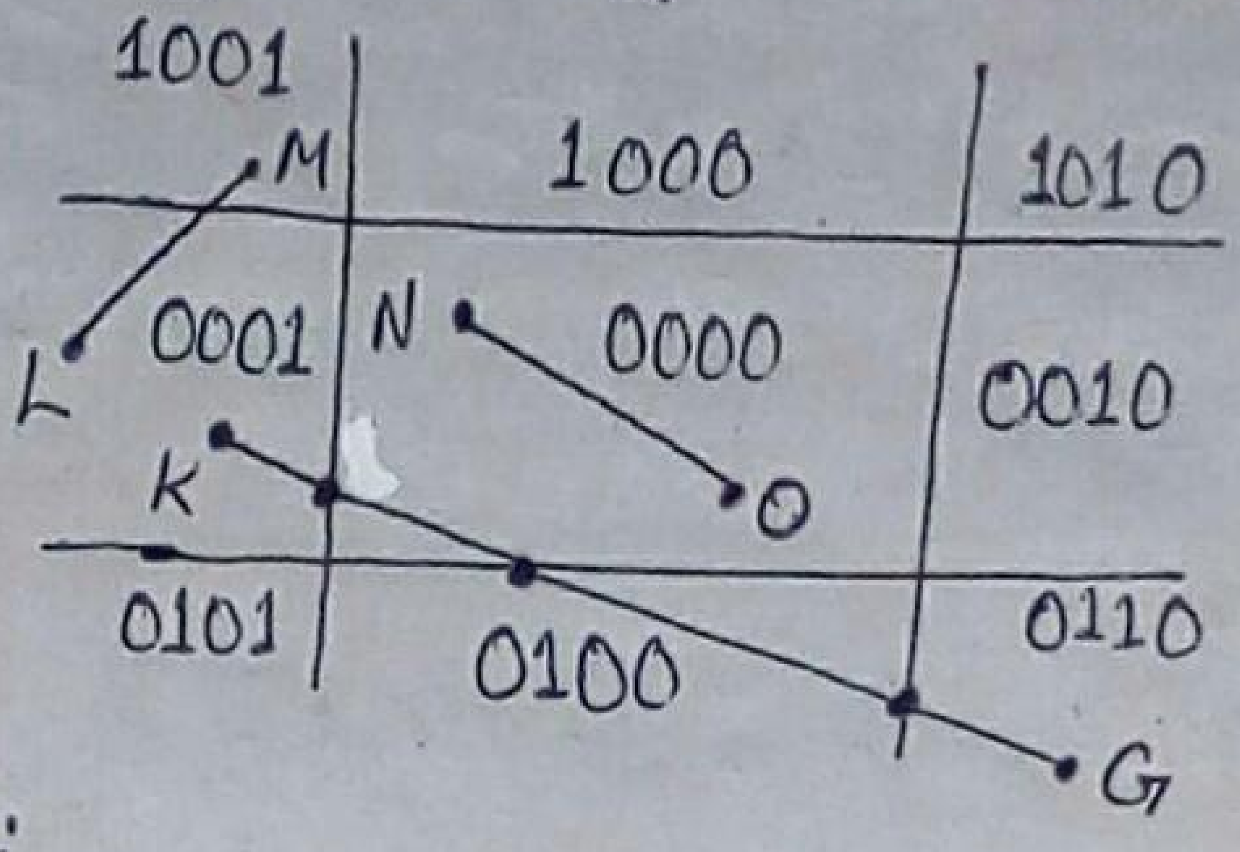
Algorithm:

Step1: Given a line segment with endpoints $P_1(x_1,y_1)$ and $P_2(x_2,y_2)$.
Step2: Compute the 4-bit region codes (outcodes) for two endpoints of line segment.
Step3: If outcodes of P_1 = outcode of P_2 = 0000 (i.e. $P_1 \| P_2 = 0000$) then, Line lies completely inside the window, so draw the line segment.
Step4: else if outcode of P_1 and outcode of $P_2 \neq 0000$ (i.e. $P_1 \&\& P_2 \neq 0000$) then, Line lies completely outside the window, so discard the line segment.
Step5: else if outcode of P_1 and outcode of $P_2 = 0000$ (i.e. $P_1 \&\& P_2 = 0000$) then Compute the intersection of line segment with window boundry and discard the portion of line segment that falls completely outside

Scanned by TapScanner

of the window. Assign a new four-bit code to the intersection and repeat until either 3 or 4 steps are satisfied.

Q1. Clip the following lines using Cohen Sutherland line clipping algorithm.



Solution:

i) For line LM:

The outcode for L is 0001
and outcode for M is 1001
The logical AND for LM = 0001 AND 1001 = 0001
which is non zero hence it is rejected.

window भित्र दुवै point भए मात्र logical OR नत्र logical AND

ii) Also for line ON:

Logical AND = 0000 accepted
& Logical AND ≠ 0000 rejected

The outcode for O is 0000
& the outcode for N is 0000
The logical OR for ON = 0000 OR 0000 = 0000
which is zero and inside the window so, it is accepted.

iii) Again for line KG:

The outcode for K is 0001
and outcode for G is 0110
The logical AND for KG = 0001 AND 0110 = 0000
This is zero. This means that this line crosses more lines which contain the clipping boundry.
Now, the line segment KG is broken into KJ, JI, IH and HG and again each segment is tested as shown below:-

For line segment KJ:

The outcode for K is 0001
and outcode for J is 0001
The logical AND for KJ = 0001 AND 0001 = 0001.
Which is non-zero hence rejected.

For line segment JI:

The outcode for J is 0000
and outcode for I is 0000
the logical AND for JI = 0000 AND 0000 = 0000
Which is zero, hence both end points are inside the window.
So, it is accepted.

Scanned by TapScanner

for line segment IH:

The outcode for I is 0100
and outcode for H is 0100
The logical AND for IH = 0100 AND 0100 = 0100
which is non-zero hence rejected.

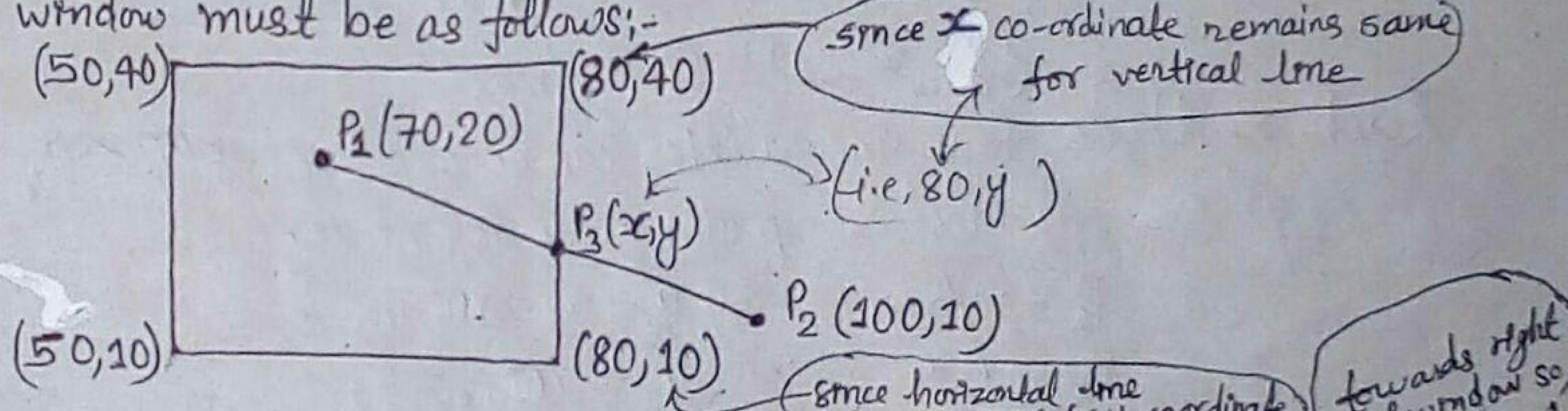
for line segment HG:

The outcode for H is 0110
and outcode for G is 0110
the logical AND for HG = 0110 AND 0110 = 0110
which is non-zero hence rejected.

Q2. Use Cohen-Sutherland algorithm to clip the line $P_1(70,20)$ and $P_2(100,10)$ against a window lower left hand corner (50,10) and upper right hand corner (80,40).

Solution:

The window must be as follows:-



Since the point P_1 lies inside the window hence its outcode is 0000.
Similarly the outcode of P_2 = 0010

The logical AND for P_1P_2 = 0000 AND 0010 = 0000
which is zero hence line is partially visible.

The slope of line $P_1P_2 = \frac{y_2-y_1}{x_2-x_1} = \frac{10-20}{100-70} = \frac{-10}{30} = \frac{-1}{3}$.

From the above figure $P_3(x,y) = P_3(80,y)$ here we need to find the value of y.

Now slope of line $P_2P_3 = \frac{y-y_2}{x-x_2} = \frac{y-10}{80-100} = \frac{y-10}{-20}$

Since slope of P_1P_2 = slope of P_2P_3

Hence, $-\frac{1}{3} = \frac{y-10}{-20}$

or, $y = 10 + \frac{20}{3} = 16.667$

Now the parts P_2P_3 of line P_1P_2 is clipped (cut off) as it is outside the window.

Hence, $P_3(x,y) = P_3(80, 16.667)$.

Scanned by TapScanner

2) Liang–Barsky Line Clipping:

→ It is an efficient line clipping algorithm which is based on parametric form of a line.

→ It reduces no. of intersections to be calculated hence performs much better than Cohen-Sutherland algorithm.

The parametric equations of line segment can be written in the form; $x = x_1 + u.\Delta x$.

$y = y_1 + u.\Delta y$ where, $0 \le u \le 1$.

$\Delta x = x_2 - x_1$

& $\Delta y = y_2 - y_1$.

Now, Following the Liang-Barsky approach, we first write the point clipping in parametric way;

$$x_{min} \le x_1 + u\,\Delta x \le x_{max}$$

$$y_{min} \le y_1 + u.\Delta y \le y_{max}.$$

Each of these four inequalities can be expressed as $up_k \le q_k$ for $k=1,2,3,4$. where parameters p and q are defined as;

$P_1 = -\Delta x$ $q_1 = x_1 - xw_{min}$ ⟷ or simply we can write $x_1 - x_{min}$ neglecting w

$P_2 = \Delta x$ $q_2 = xw_{max} - x_1$

$P_3 = -\Delta y$ $q_3 = y_1 - yw_{min}$

$P_4 = \Delta y$ $q_4 = yw_{max} - y_1$.

Now,

if $P_1 = 0 \Rightarrow -\Delta x = 0 \Rightarrow \Delta x = 0 \Rightarrow x_2 - x_1 = 0 \Rightarrow x_2 = x_1$

Similarly

if $P_2 = 0 \Rightarrow x_2 = x_1$

if $P_3 = 0 \Rightarrow y_2 = y_1$

& if $P_4 = 0 \Rightarrow y_2 = y_1$.

Conditions:

i) If $(P_i = 0)$ then line segment is parallel to window boundary. for all $i = 1, 2, 3, 4$.

ii) If $(P_1 = 0)$ & if $(q_1 < 0)$

$\Rightarrow x_1 - x_{min} < 0 \Rightarrow x_1 < x_{min}$

Then line lies outside of x_{min} boundary.

iii) If $(P_2 = 0)$ & if $(q_2 < 0)$

$\Rightarrow x_{max} - x_1 < 0 \Rightarrow x_{max} < x_1$

Then line lies outside x_{max} boundary.

Scanned by TapScanner

iv) If ($P_3 = 0$) and if ($q_3 < 0$)
$\Rightarrow y_1 - y_{min} < 0 \Rightarrow y_1 < y_{min}$
Then line lies outside of y_{min} boundry.

v) If ($P_4 = 0$) and if ($q_4 < 0$)
$\Rightarrow y_{max} - y_1 < 0 \Rightarrow y_{max} < y_1$
Then line lies outside of y_{max} boundry.

$\Rightarrow$ Hence, if ($P_i = 0$) and if ($q_i < 0$) then the line lies completely outside of window.

if $P_i = 0$ and $q_i \geq 0$ then line lies inside of ith boundry.

if $P_i \neq 0$ then we need to compute intersection point.

To calculate intersection point the below algorithm determines two value of t & based on those values, line is clipped.

Algorithm:

1.) Input $P_1(x_1, y_1)$ & $P_2(x_2, y_2)$.

2). Input clipping window boundaries ($x_{min}, x_{max}, y_{min}, y_{max}$).

3). Determine p_i and q_i as follows:

$P_1 = -\Delta x$	$q_1 = x_1 - x_{min}$	(Left boundry)
$P_2 = \Delta x$	$q_2 = x_{max} - x_1$	(right ")
$P_3 = -\Delta y$	$q_3 = y_1 - y_{min}$	(bottom ")
$P_4 = \Delta y$	$q_4 = y_{max} - y_1$	(top ").

4). If ($P_i = 0$) and $q_i < 0$ for any;
→ Line is parallel to window boundry and is outside of window so discard line segment.

5). Set two parameters t_1 & t_2 as;
$t_1 = 0$
$t_2 = 1$

6) Determine r_i as follows;
$r_i = \frac{q_i}{P_i}$ where, $i = 1, 2, 3, 4$.

7) Find all r_i for which $P_i < 0$
Set $t_1 = max(0, r_i)$.

8) Find all r_i for which $P_i > 0$.
Set $t_2 = min(1, r_i)$.

Scanned by TapScanner

9). If $(t_1 > t_2)$

→ line is completely outside window so discard line segment.

else {

If $(t_1 = 0$ & $t_2 = 1$

→ line is completely inside window, so display the line segment.

else {

clip the line segment and determine (x_1', y_1') and (x_2', y_2') as;

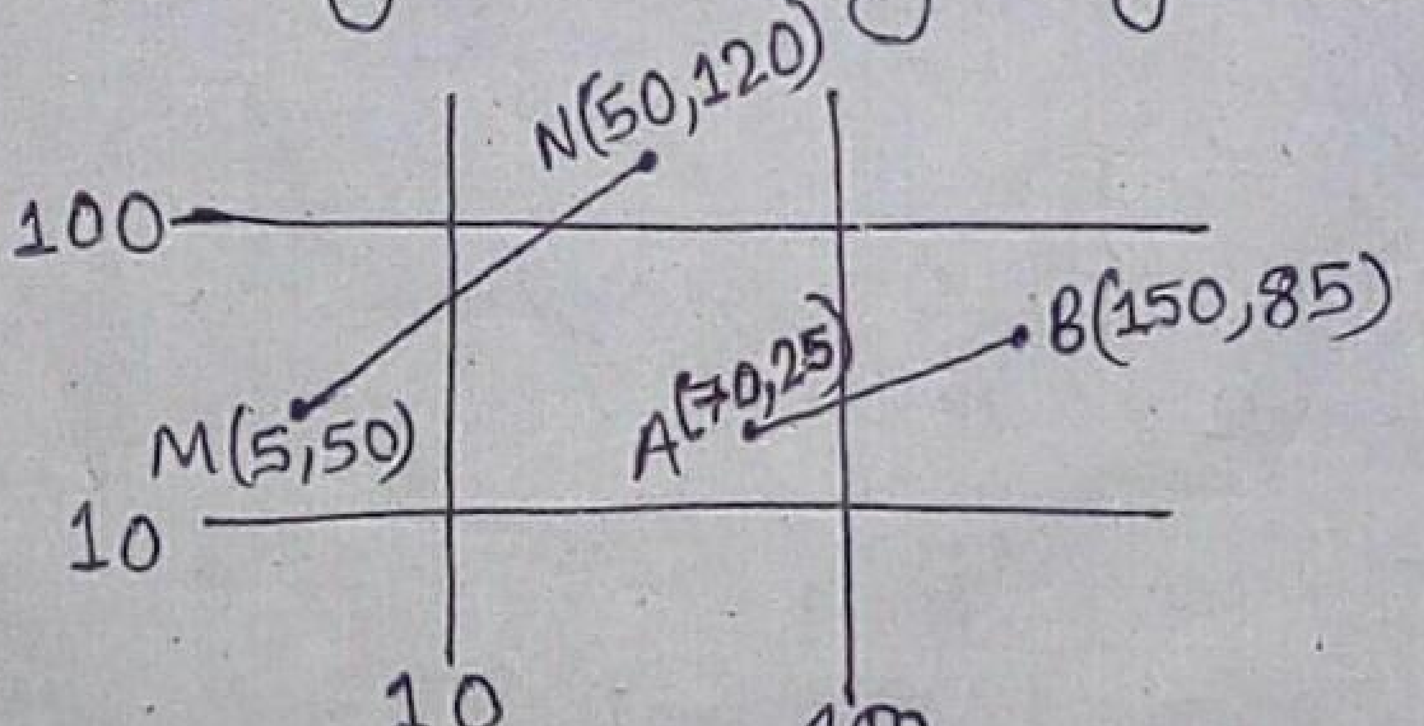
$$x_1' = x_1 + t_1 \Delta x$$
$$x_2' = x_1 + t_2 \Delta x$$
$$y_1' = y_1 + t_1 \Delta y$$
$$y_2' = y_1 + t_2 \Delta y$$

}

}

10). End.

Q1. Clip the following lines using Liang - Barsky Line Clipping algorithm;



solution:

i) For line AB:

Here, $A(70,25) = A(x_1, y_1)$

$B(150,85) = B(x_2, y_2)$

$\Delta x = x_2 - x_1 = 150 - 70 = 80$

$\Delta y = y_2 - y_1 = 85 - 25 = 60$

$x_{min} = 10$, $x_{max} = 100$

$y_{min} = 10$, $y_{max} = 100$

Now, $P_1 = -\Delta x = -80$, $q_1 = x_1 - x_{min} = 70 - 10 = 60$, $r_1 = \frac{q_1}{P_1} = \frac{60}{-80} = \frac{-3}{4}$

$P_2 = \Delta x = 80$, $q_2 = x_{max} - x_1 = 100 - 70 = 30$, $r_2 = \frac{q_2}{P_2} = \frac{30}{80} = \frac{3}{8}$

$P_3 = -\Delta y = -60$, $q_3 = y_1 - y_{min} = 25 - 10 = 15$, $r_3 = \frac{q_3}{P_3} = \frac{15}{-60} = \frac{-1}{4}$

$P_4 = \Delta y = 60$, $q_4 = y_{max} - y_1 = 100 - 25 = 75$, $r_4 = \frac{q_4}{P_4} = \frac{75}{60} = \frac{5}{4}$

Now, $t_1 = 0,\ t_2 = 1$

Again, $t_1 = \max\left(0, -\frac{3}{4}, -\frac{1}{4}\right) = 0$

$t_2 = \min\left(1, \frac{3}{8}, \frac{5}{4}\right) = 3/8$

Here, $t_1 < t_2$ so we need to find intersection point. Let (xx_1, yy_1) and (xx_2, yy_2) be intersection point.

So, $xx_1 = x_1 + t_1 \Delta x = 70 + 0 \times 80 = 70$

$xx_2 = x_1 + t_2 \Delta x = 70 + \frac{3}{8} \times 80 = 100$

$yy_1 = y_1 + t_1 \Delta y = 25 + 0 \times 60 = 25$

$yy_2 = y_1 + t_2 \Delta y = 25 + \frac{3}{8} \times 60 = 47.5$

Hence AB line is clipped at (xx_1, yy_1) & (xx_2, yy_2)

i.e, $(70, 25)$ & $(100, 47.5)$

ii) <u>For line MN:</u>

Here, $M(5,50) = (x_1, y_1)$

$N(50,120) = (x_2, y_2)$

$\Delta x = x_2 - x_1 = 50 - 5 = 45$

$\Delta y = y_2 - y_1 = 120 - 50 = 70.$

Now, $P_1 = -\Delta x = -45$, $q_1 = x_1 - x_{min} = 5 - 10 = -5$, $r_1 = \frac{q_1}{P_1} = \frac{-5}{-45} = \frac{1}{9}$

$P_2 = \Delta x = 45$, $q_2 = x_{max} - x_1 = 100 - 5 = 95$, $r_2 = \frac{q_2}{P_2} = \frac{95}{45} = \frac{19}{9}$

$P_3 = -\Delta y = -70$, $q_3 = y_1 - y_{min} = 50 - 10 = 40$, $r_3 = \frac{q_3}{P_3} = \frac{40}{-70} = -\frac{4}{7}$

$P_4 = \Delta y = 70$, $q_4 = y_{max} - y_1 = 100 - 50 = 50$, $r_4 = \frac{q_4}{P_4} = \frac{50}{70} = \frac{5}{7}$

Now, $t_1 = \max\left(0, \frac{1}{9}, -\frac{4}{7}\right) = \frac{1}{9}$

$t_2 = \min\left(1, \frac{19}{9}, \frac{5}{7}\right) = 5/7$

Here, $t_1 < t_2$, So we need to find intersection point. Let (x_1', y_1') & (x_2', y_2') be intersection points then;

$x_1' = x_1 + t_1 \Delta x = 5 + \frac{1}{9} \times 45 = 10$

$x_2' = x_1 + t_2 \Delta x = 5 + \frac{5}{7} \times 45 = 37$

$y_1' = y_1 + t_1 \Delta x = 50 + \frac{1}{9} \times 70 = 58$

$y_2' = y_1 + t_2 \Delta x = 50 + \frac{5}{7} \times 70 = 100$

Hence MN line is clipped at (x_1', y_1') and (x_2', y_2') i.e, $(10, 37)$ & $(58, 100)$.

Scanned by TapScanner

Q.2. Find the clipping coordinates for a line AB where $A=(10,10)$ and $B(60,30)$, against window with $(x_{min}, y_{min}) = (15,15)$ and $(x_{max}, y_{max}) = (25,25)$ using Liang-Barsky line clipping algorithm.

Solution:

Here, $x_1 = 10$, $x_{min} = 15$
$y_1 = 10$, $y_{min} = 15$
$x_2 = 60$, $x_{max} = 25$
$y_2 = 30$, $y_{max} = 25$

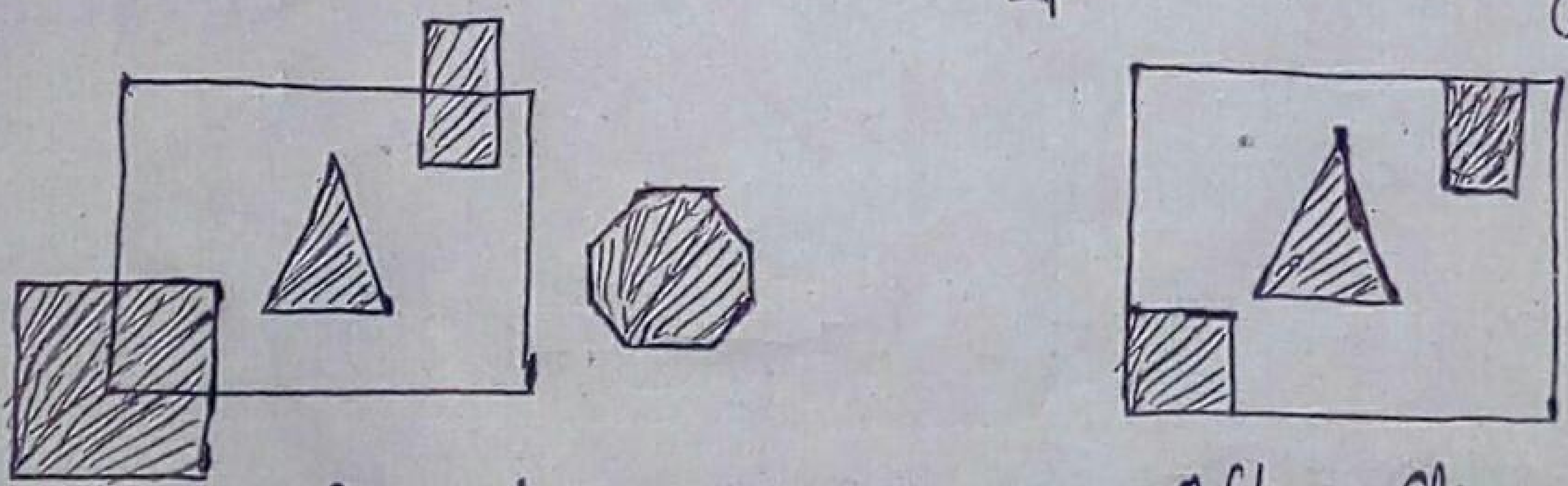
(We solve it similar way as before ✓)

$P_1 = -\Delta x = -(x_2 - x_1) = -(60-10) = -50$, $q_1 = -5$, $r_1 = \frac{q_1}{P_1} = 0.1$

⇒ Same method/process that we proceed in before question. Easy solve yourself.

⊛. Polygon Clipping:

Polygon clipping is defined as the process of removing those parts of polygon that lie outside of a clipping window. A polygon can be defined as a geometric object consisting of number of vertices and an equal number of line segments called edges.



Before clipping　　　　After Clipping

fig. polygon clipping

⊛. Sutherland-Hodgeman Polygon Clipping Algorithm:

Algorithm:

1). Construct an input vertex list $(v_0, v_1, \ldots, v_n)$ where $v_0 = v_n$.
2). Create empty output vertex list.
3). For each pair of adjcent vertices v_i and v_{i+1} perform the following inside-outside test:
   a) If out-in (v_i is outside the window boundry, v_{i+1} is inside), add insertion point v_i' and v_{i+1} to the output vertex list.
   b) If in-in (both v_i, v_{i+1} are inside the window boundry), add v_{i+1} to output vertex list.

new vertex

c). If in-out (v_i is inside the window boundry and v_{i+1} is outside), add insertion point v_i' to output vertex list.

d). If out-out (both v_i, v_{i+1} are outside the window boundry), add nothing to output vertex list.

4). Stop.

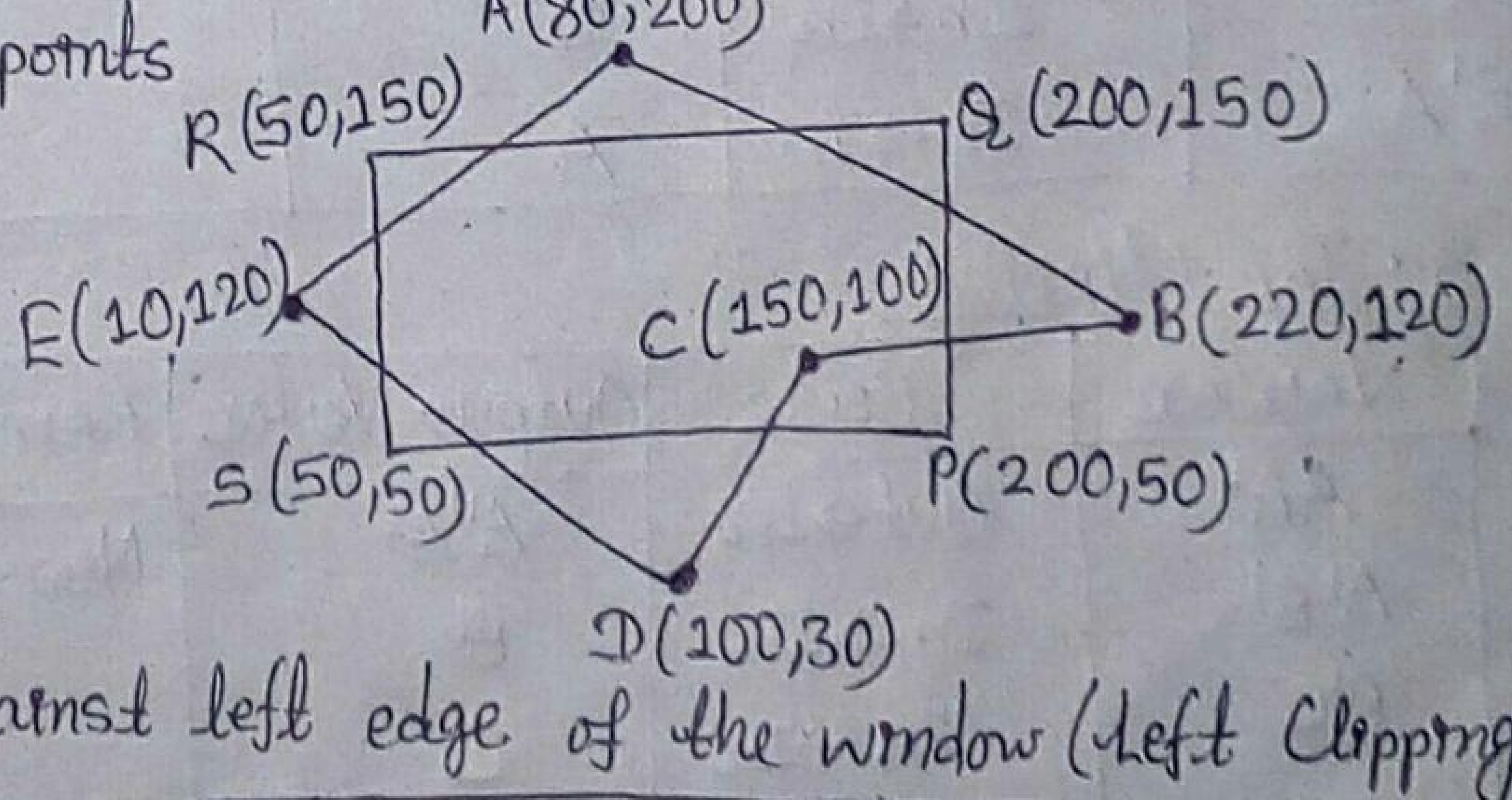
Four cases of polygon clipping against one edge:

S. No.	Case	Output
1.	Wholly inside visible region (in-in).	save endpoint.
2.	Exist visible region (in-out)	save the intersection.
3.	Wholly outside visible region (out-out)	Save nothing.
4.	Enter visible region (out-in)	Save intersection and endpoint.

Q. Clip polygon ABCD against window PQRS. The co-ordinates of the polygon are A(80, 200), B(220, 120), C(150, 100), D(100, 30), E(10, 120). Co-ordinates of the window are P(200, 50), Q(200, 150), R(50, 150), S(50, 50).

Solution:

Step1: Plot the points



Step2: Clipping against left edge of the window (Left Clipping)

Vertex	Case	Output Vertex	Remarks
AB	In → In	B	
BC	In → In	C	
CD	In → In	D	
DE	In → Out	D'	New vertex
EA	Out → In	E'A	New vertex

Scanned by TapScanner

Step: 3: Right Clipping

Vertex	Case	Output Vertex	Remarks.
AB	In→Out	A'	New vertex
BC	Out→In	B'C	New vertex
CD	In→In	D	
DD'	In→In	D'	
D'E'	In→In	E'	
E'A	In→In	A	

Step4: Bottom Clipping

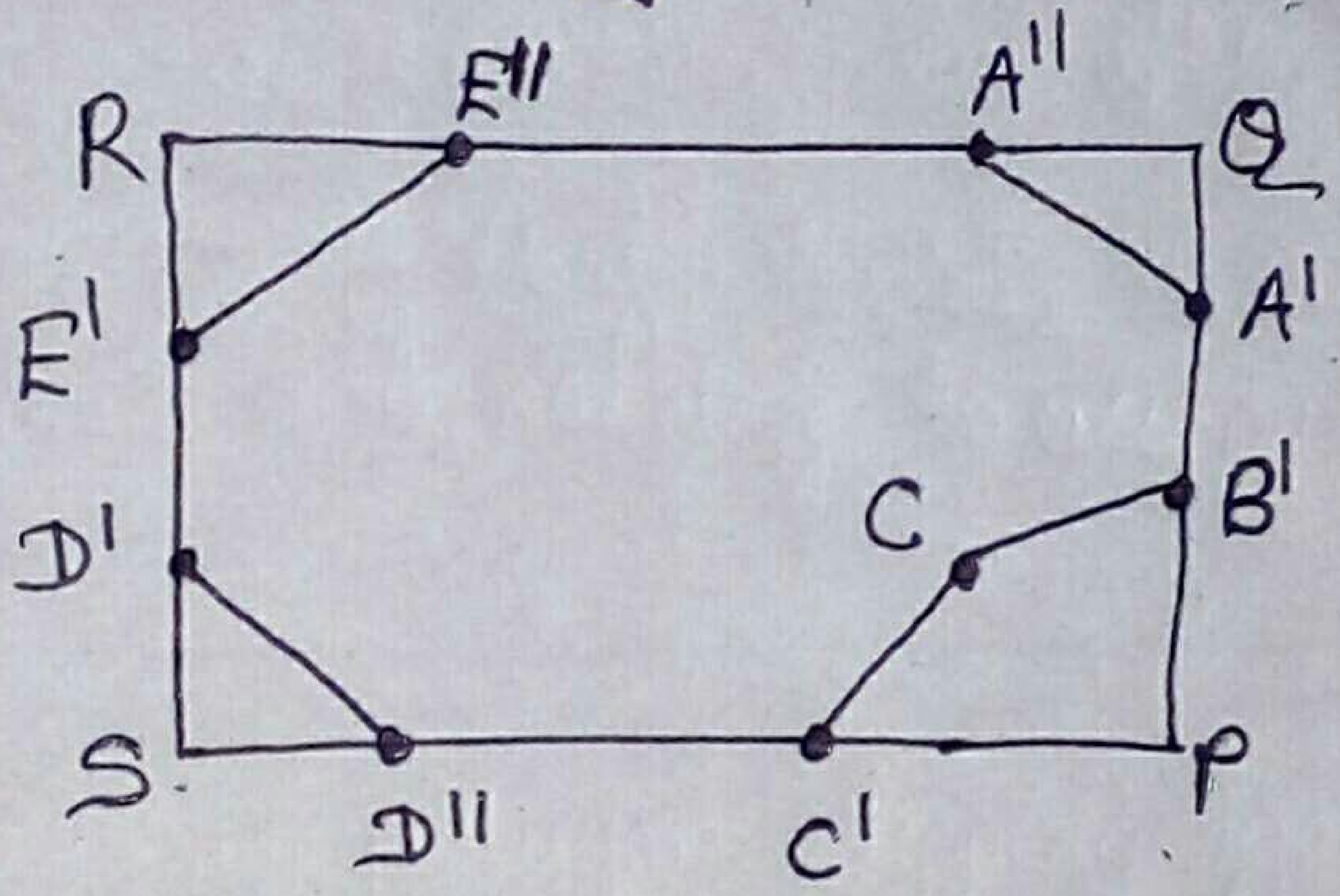
Vertex	Case	Output Vertex	Remarks
AA'	In-In	A'	
A'B'	In-In	B'	
B'C	In→In	C	
CD	In→Out	C'	New Vertex
DD'	Out→In	D''D'	New Vertex
D'E'	In→In	E'	
E'A	In→In	A	

Step5: Top Clipping

Vertex	Case	Output Vertex	Remarks
AA'	Out→In	A''A'	New vertex
A'B'	In→In	B'	
B'C	In→In	C	
CC'	In→In	C'	
CD''	In→In	D''	
D''D'	In→In	D'	
D'E'	In→In	E'	
E'A	In→Out	E''	New Vertex

Scanned by TapScanner

Step6: After Clipping



Scanned by TapScanner