Chapter 6.5 Two-dimensional transformation

- Two-dimensional translation,
- Rotation,
- Scaling,
- Reflection,
- Shear transformation,
- 2D composite transformation,
- 2D viewing pipeline,
- World to screen viewing transformation and
- Clipping (Cohen Sutherland line clipping, Liang-Barsky Line Clipping)

Transformation:

- Changing position, shape, size or orientation of an object on display is called transformation.
- In order to reposition the graphics on the screen and change the size or orientation, Transformations play a crucial role in computer graphics.
- They important to graphics for:
  - moving objects on screen / in space
  - specifying the camera's view of a 3D scene
  - mapping from model space to world space to camera space to screen space

2D (Two Dimensional) Transformation:

- When a transformation takes place on a 2D plane, it is called 2D transformation.
- Given a 2D object, transformation is to change the object's position (translation), Size (scaling), Orientation (rotation), Shapes (shear).

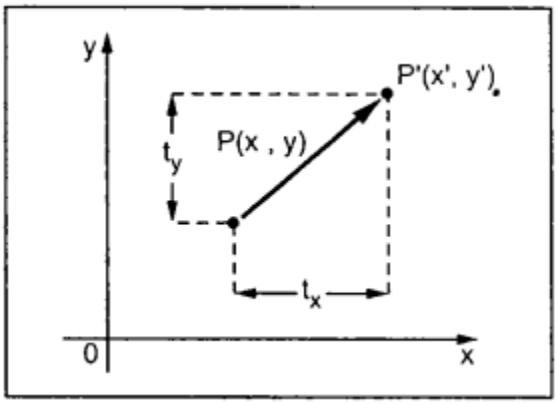
Translation:

- It is the process of changing the position of an object in a straight line path from one coordinate location to another.
- We can translate a two dimensional point by adding translation distances t_x and t_y to the original coordinate position (x, y) to move the point to a new position (x',y') as shown in figure.

$\mathbf{x' = x + t_x}$

$\mathbf{y' = y + t_y}$

- The translation distance pair ($\mathbf{t_x}$, $\mathbf{t_y}$) is called translation vector or shift vector.



- We can represent it into single matrix equation in column vector as:

$$P' = P + T$$

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

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

$$P' = P + T$$

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

Example:

Translate the triangle A(10,10), B(15,15), C(20,10) 2 unit in x direction and one unit in y direction.

We know that

$$P' = P + T$$

$$P' = [P] + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

For point (10, 10)

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

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

For point (15, 15)

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

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

For point (10, 10)

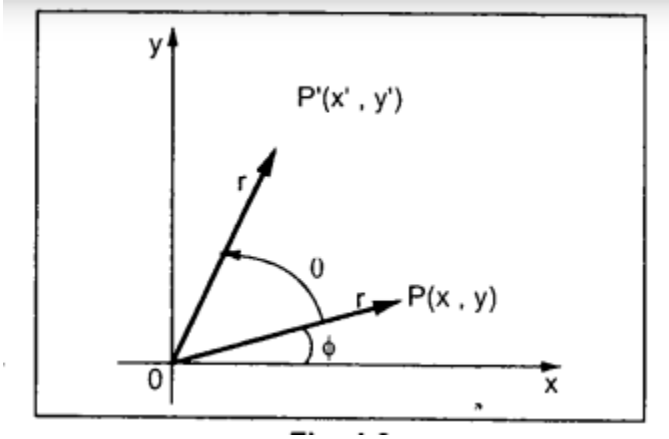
$$C' = \begin{bmatrix}20\\10\end{bmatrix} + \begin{bmatrix}2\\1\end{bmatrix}$$

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

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

Rotation:

- A two dimensional rotation is applied to an object by repositioning it along a circular path in xy plane.
- To generate a rotation, we specify a rotation angle θ and the position of the rotation point about which the object is to be rotated.
- Suppose you want to rotate it at the angle θ. After rotating it to the new location, you will get a new point P' A′, B′.



Using standard trigonometric, the original coordinate of points P x and y can be represented as −

$x = r\cos\phi$ … (i)

$y = r\sin\phi$ … (ii)

Similarly we can represent point P' (x′, y′) as −

$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta \ldots (iii)$

$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta \ldots (iv)$

Substituting equations (i) & (ii) in equations (iii) & (iv), respectively, we will get

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

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

Representing the above equation in matrix form,

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

$$P' = P \cdot R$$

Where R is the rotation matrix

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

- The rotation angle will be positive or negative. For the positive rotation angle, we may use the above rotation matrix. However, for negative angle rotation, the matrix will change as shown below:

It is important to note that positive values for the rotation angle define counterclockwise rotations about the rotation point and negative values rotate objects in the clockwise sense.

For negative values of θ i.e., for clockwise rotation, the rotation matrix becomes

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

$$= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad \because \cos(-\theta) = \cos\theta \quad \text{and} \quad \sin(-\theta) = -\sin\theta$$

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

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

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

$$\therefore \quad P^1 = [4\ 3] \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

$$= \left[4/\sqrt{2} - 3/\sqrt{2} \quad 4/\sqrt{2} + 3/\sqrt{2}\right]$$

$$= \left[1/\sqrt{2} \quad 7/\sqrt{2}\right]$$

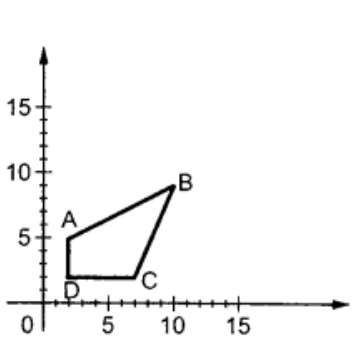
Scaling:

- Scaling is the transformation that is used to change the object's size. The operation is carried out by multiplying the coordinate value(X, Y) with Sx and Sy scaling factors.
- Scaling is performed about the origin as
  - If scale > 1 then enlarges the object and moves it away from the origin.
  - If scale = 1, then leaves the object the same.
  - If scale < 1, then shrink/reduce the object and move it towards its origin.
- Therefore the equation for scaling is given by:
- $X' = X.\ S_X$ and $Y' = Y.\ S_Y$
- The scaling factor S_X and S_Y scale the object in the X and Y direction, respectively. The above equations may also be represented in matrix form as below:

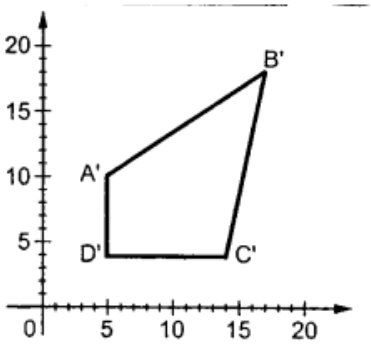
$$[x'\ y'] = [x\ y] \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

$$= [x \cdot S_x \quad y \cdot Sy]$$

$$= P \cdot S$$



(a)



(b)

Example: 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.

Here $S_x = 2$ and $S_y = 2$. Therefore, transformation matrix is given as

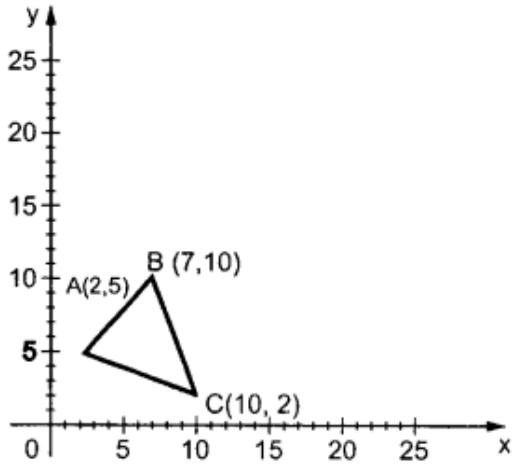
$$S = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

The object matrix is :

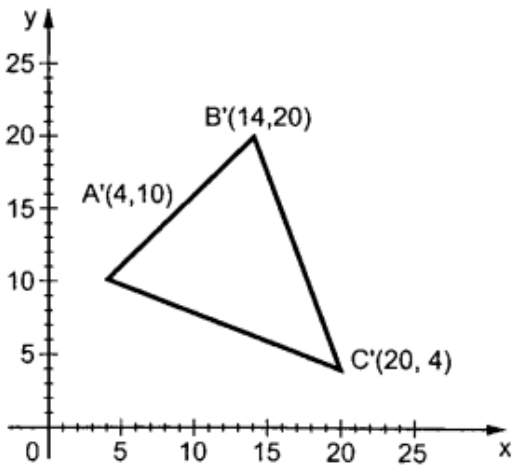
$$\begin{matrix} & x & y \\ A & 2 & 5 \\ B & 7 & 10 \\ C & 10 & 2 \end{matrix}$$

$$\therefore \quad \begin{matrix} A' \\ B' \\ C' \end{matrix} \begin{bmatrix} x'_1 & y'_1 \\ x'_2 & y'_2 \\ x'_3 & y'_3 \end{bmatrix} = \begin{bmatrix} 2 & 5 \\ 7 & 10 \\ 10 & 2 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

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



(a) Original object



(b) Scaled object

Reflection:

- A reflection is a transformation that produces a mirror image of an object relative to an axis of reflection.
- We can choose an axis of reflection in the xy plane or perpendicular to the xy plane.

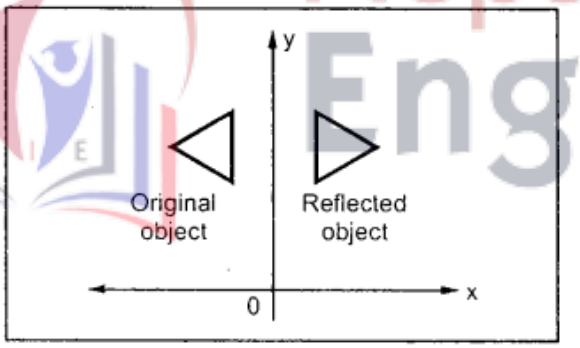
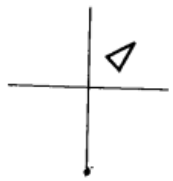
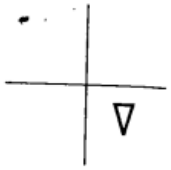
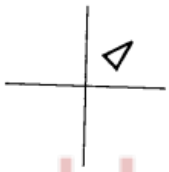
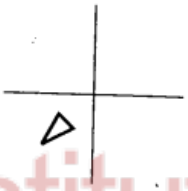
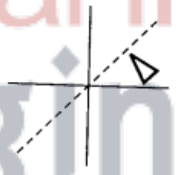
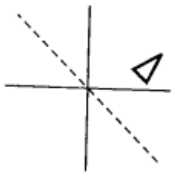
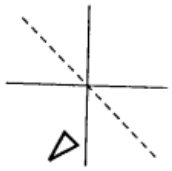


Fig: Reflection about y axis

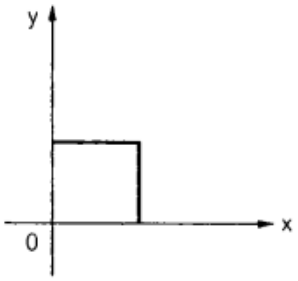
Reflection	Transformation matrix	Original image	Reflected image
Reflection about Y-axis	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$		
Reflection about X axis	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$		
Reflection about origin	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$		
Reflection about line $y = x$	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$		
Reflection about line $y = -x$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$		

Shear

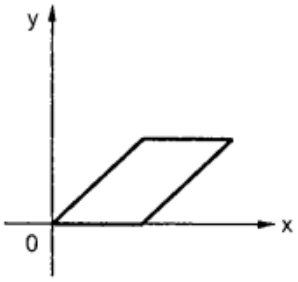
- A transformation that slants the shape of an object is called shear transformation.
- Two common shearing transformations are used. One shifts x coordinate values and shifts y coordinate values. However in both the cases only one coordinate (x or y) changes its coordinates and other preserves its values.

X Shear

- The x shear preserves the y coordinates but changes x values which causes vertical lines to tilt right or left as shown in figure.



(a) Original object



(b) Object after x shear

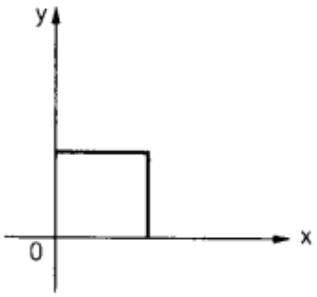
$$X\_sh = \begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x' = x + Sh_x \cdot y \quad \text{and}$$

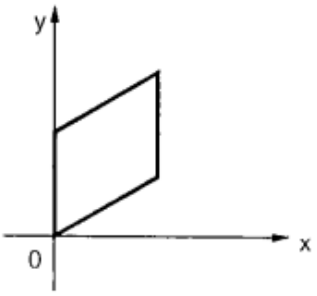
$$y' = y$$

Y Shear

- The y shear preserves x coordinates but changes y values which causes horizontal lines to transform into lines which slope up or down.



(a) Original object



(b) Object after y shear

The transformation matrix for y shear is given as

$$Y\_sh = \begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\therefore \quad x' = x \text{ and } \quad y' = y + Sh \cdot x$$

Shearing Relative to Other Reference Line

- We can apply x shear and y shear transformations relative to other reference lines.
- In x shear transformation we can use y reference line and in y shear we can use x reference line.
- The transformation matrices for both are given below:

$$\text{x shear with y reference line} : \begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ -Sh_x \cdot y_{ref} & 0 & 1 \end{bmatrix}$$

$$\text{y shear with x reference line} : \begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & -Sh_y \cdot x_{ref} & 0 \end{bmatrix}$$

Two-dimensional viewing:

- Two-dimensional viewing is the mechanism for displaying views of a picture on an output device.
- Much like what we see in real life through a small window on the wall or the viewfinder of a camera, a computer-generated image often depicts a partial view of a large scene.
- For a 2-D picture, a view is selected by specifying a subarea of the total picture area.

1. Modeling Coordinates:

- Modeling coordinates are used to construct shape of individual parts (objects or structures) of a 2D scene. For example, generating a circle at the origin with a "radius" of 2 units.
- Here, origin (0, 0), and radius 2 units are modeling coordinates.
- Modeling coordinates define object shape.
- Can be floating-point, integers and can represent units like km, m, miles, feet etc.
- The coordinates are only transferred to world coordinates where the model is actually placed in the world. This allows us to use object in many places instead of having to make a new one every time.

2. World Coordinates:

- World coordinates are used to organize the individual parts into a scene.
- World coordinates units define overall scene to be modeled.
- World coordinates represent relative positions of objects.
- Can be floating-point, integers and can represent units like km, m, miles etc.

3. Viewing Coordinates:

- Viewing coordinates are used to define particular view of the user. Viewer's position and view angle i.e. rotated/translated.
- Viewing coordinates specify the portion of the output device that is to be used to present the view.

4. Device Coordinates or Screen Coordinates:

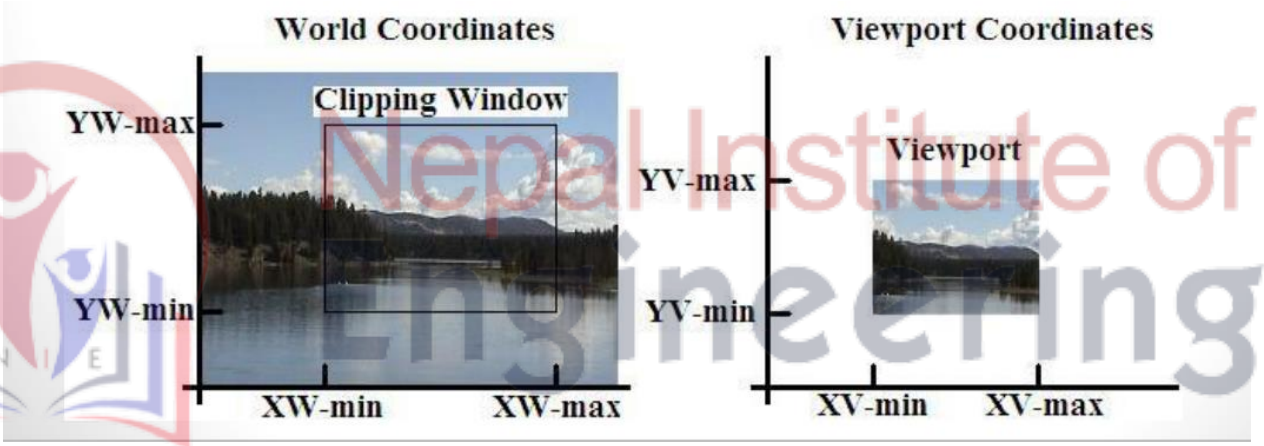
- The display coordinate system is called device coordinate system. Device coordinates are specific to output device.
- Device coordinates are integers within the range (0, 0) to (x_{max}, y_{max}) for a particular output device.

Window:

- A world-coordinate area selected for display is called a window or clipping window. That is, window is the section of the 2D scene that is selected for viewing.
- The window defines what is to be viewed

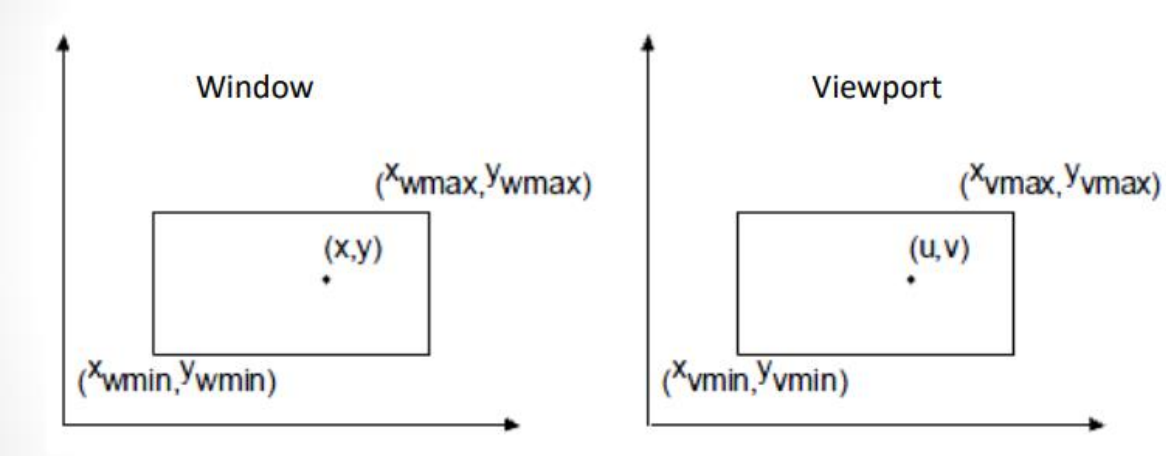
Viewport:

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



Window to view port transformation:

- The window to viewport coordinate transformation defines the mechanism for displaying views of a picture on an output device.
- A window is specified by four world coordinates: X_{wmin}, X_{wmax}, Y_{wmin} and Y_{wmax}.
- Similarly, a viewport is described by four normalized device coordinates: X_{vmin}, X_{vmax}, Y_{vmin} and Y_{vmax}.



1. Translate the window to the origin. That is, apply T ($-X_{wmin}$, $-Y_{wmin}$)
2. Scale it to the size of the viewport. That is, apply S (sx, sy)
3. Translate scaled window to the position of the viewport. That is, apply T(X_{vmin}, Y_{vmin}). Therefore, net transformation,

T_{WV} = T (X_{vmin}, Y_{vmin}). S(sx, sy).T($-X_{wmin}$, $-Y_{wmin}$)

- Let (x, y) be the world coordinate point that is mapped onto the viewport point (u, v), then we must have following relation to maintain the same relative placement of the point in the viewport:

$$\frac{u - x_{v\min}}{x_{v\max} - x_{v\min}} = \frac{x - x_{w\min}}{x_{w\max} - x_{w\min}}$$

$$u = x_{v\min} + \frac{x_{v\max} - x_{v\min}}{x_{w\max} - x_{w\min}} \cdot (x - x_{w\min})$$

$$\frac{v - y_{v\min}}{y_{v\max} - y_{v\min}} = \frac{y - y_{w\min}}{y_{w\max} - y_{w\min}}$$

$$v = y_{v\min} + \frac{y_{v\max} - y_{v\min}}{y_{w\max} - y_{w\min}} \cdot (y - y_{w\min})$$

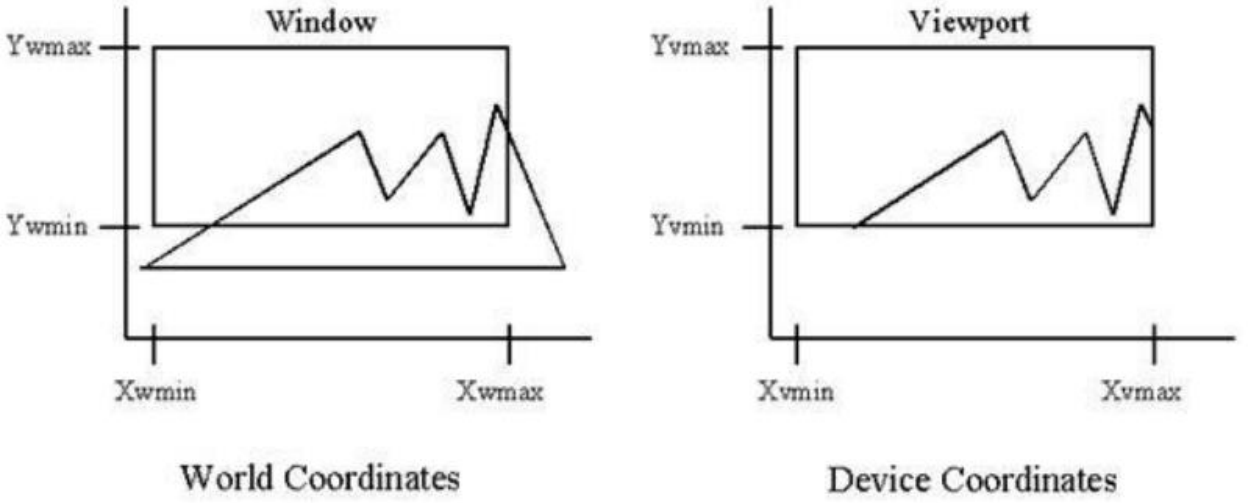
But, we know that

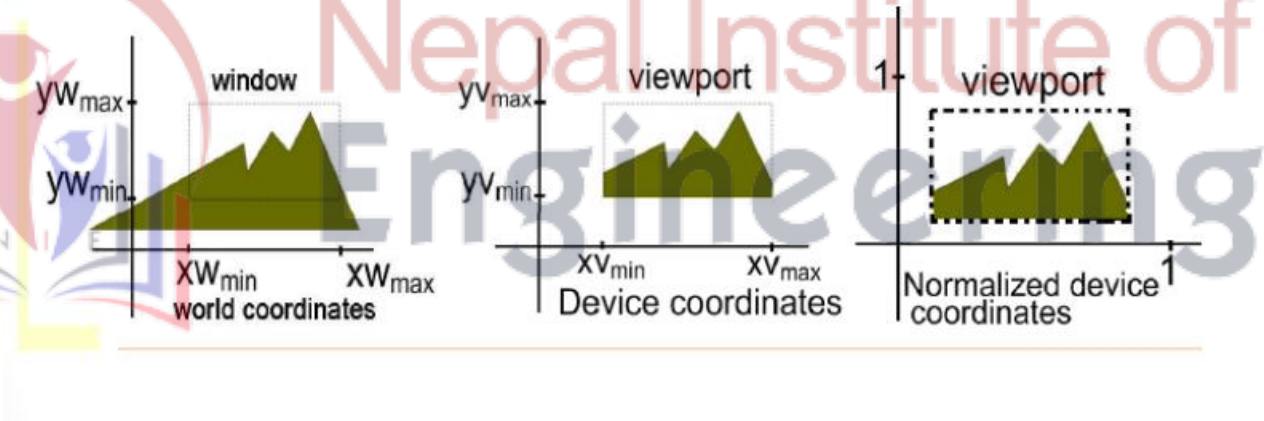
$$u = x_{vmin} + s_x(x - x_{wmin})$$
$$v = y_{vmin} + s_y(y - y_{wmin})$$

Where

$$s_x = \frac{x_{v\max} - x_{w\min}}{x_{w\max} - x_{w\min}}$$

$$s_y = \frac{y_{v\max} - y_{v\min}}{y_{w\max} - y_{w\min}}$$





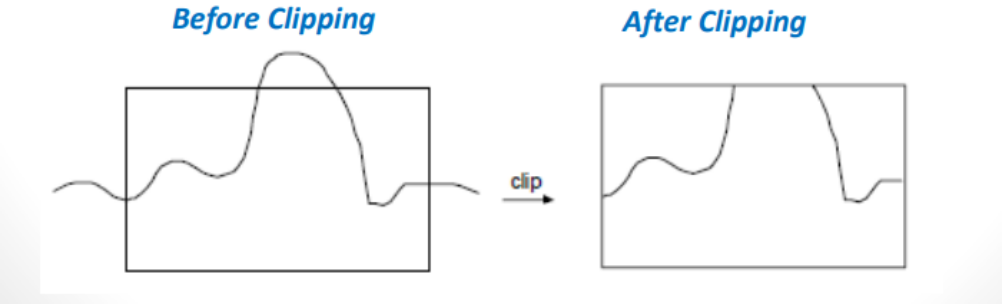
- Window and viewport are often rectangular in standard positions, because it simplifies the transformation process and clipping process.
- Other shapes such as polygons, circles take longer time to process.
- By changing the position of the viewport, we can view objects at different positions on the display area of an output device.
- Also by varying the size of viewports, we can change size of displayed objects.
- Zooming effects can be obtained by successively mapping different-sized windows on a fixed-sized viewport.

Clipping:

- The process of identifying those portions of a picture that are either inside or outside of the specified region of space is called clipping.

Two possible ways to apply clipping in the viewing transformation:

1. Apply clipping in the world coordinate system: ignore objects that lie outside of the window.
2. Apply clipping in the device coordinate system: ignore objects that lie outside of the viewport.



Importance of Clipping:

1. Excludes unwanted graphics from the screen.
2. Improves efficiency, as the computation dedicated to objects that appear off screen can be significantly reduced.

Applications:

1. In drawing and painting, it is used to select picture parts for copying, erasing, moving.
2. Displaying a multi-window environment

Types of Clipping:

- Point Clipping
- Line Clipping (Straight line segment)
- Area Clipping (Polygon)
- Curve Clipping
- Text clipping

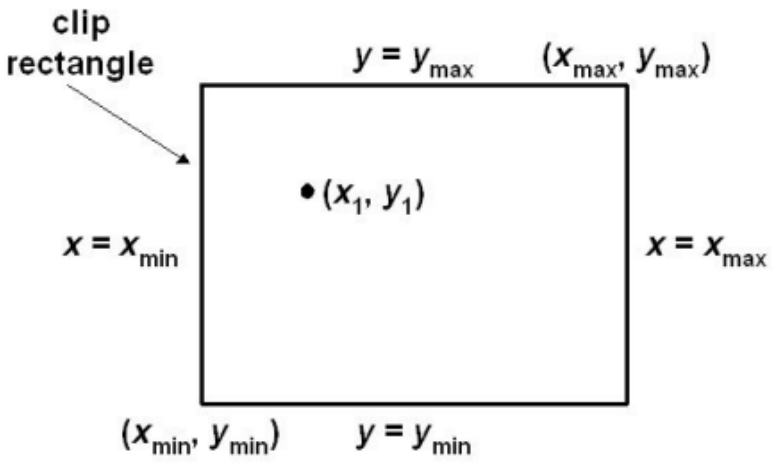
Point Clipping:

- The points are said to be interior to the clipping window if

$X_{wmin} \leq x \leq X_{wmax}$ and

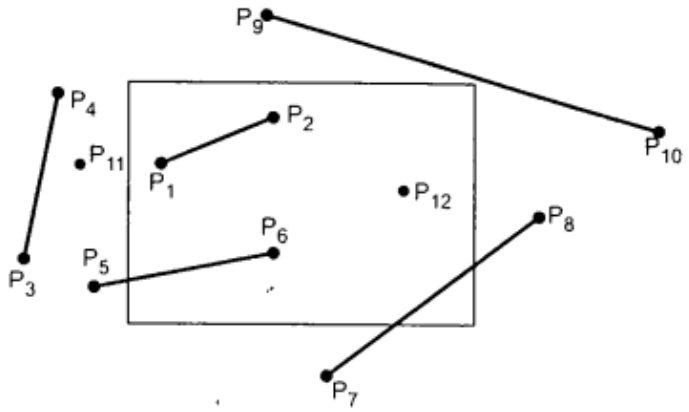
$Y_{wmin} \leq y \leq Y_{wmax}$

- The equal sign indicates that points on the window boundary are included within the window.

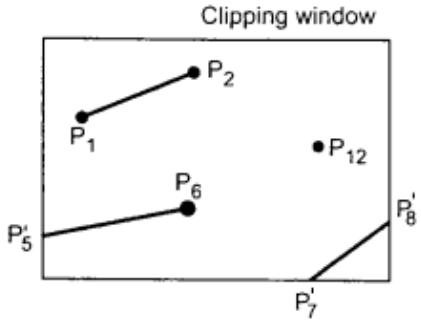


Line Clipping:

- Lines that do not intersect the clipping window are either completely inside the window or completely outside the window.
- On the other hand, a line that intersects the clipping window is divided by the intersection point(s) into segments that are either inside or outside the window.
- For any particular line segment:
  - **a.** Both endpoints are inside the region: No clipping necessary
  - **b.** One endpoint is inside and one is outside of the clipping window: Clip at intersection point.
  - **c.** Both endpoints are outside the region:
    - No intersection
    - Discard the line segment.
    - Line intersects the region
    - Clip line at both intersection points.



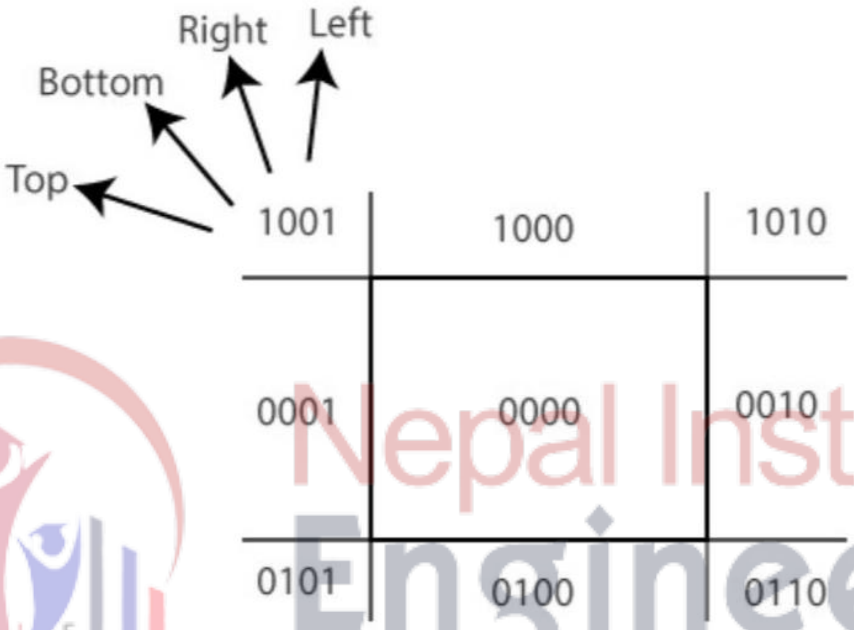
(a) Before clipping



(b) After clipping

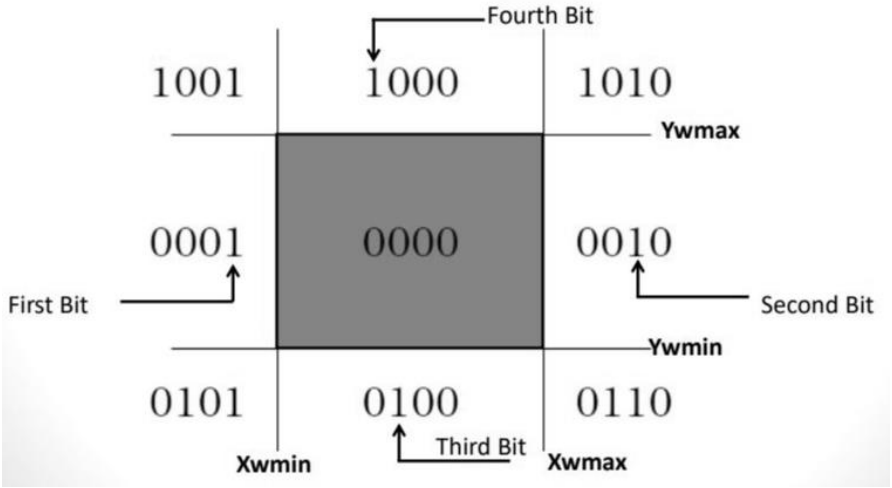
Cohen- Sutherland Line Clipping Algorithm:

- One of the oldest and most popular line-clipping algorithms.
- 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 (region code or out code). Each bit in the code is set to either a 1(true) or a 0(false). Assign a bit pattern to each region as shown:



Step 1: Establish Region code for all line end point

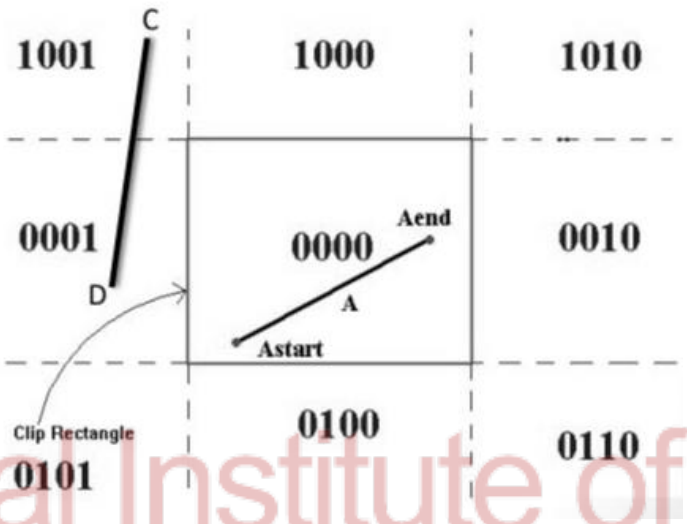
• First bit is 1, if x < Xwmin (Point lies to left of window), else set it to 0
• Second bit is 1, if x > Xwmax (Point lies to right of window), else set it to 0
• Third bit is 1, if y < Ywmin (Point lies to below window), else set it to 0
• Fourth bit is 1, if y > Ywmax (Point lies to above window), else set it to 0



Step 2: Determine which lines are completely inside window and which are not
a) If both end points of line has region codes '0000' line is completely inside window.
b) If logical AND operation of region codes of two end points is NOT '0000'. The line is completely outside (some bit position have 1's).

Astart 0000
Aend 0000
Completely inside

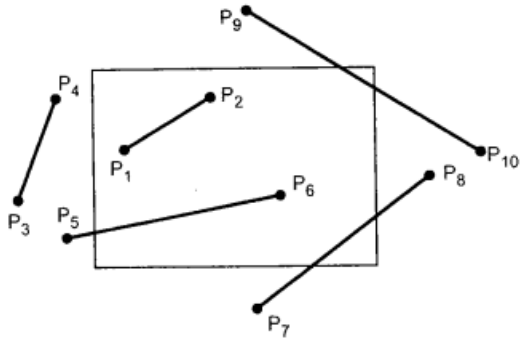
C 1001
D 0001
AND
0001
Completely Outside

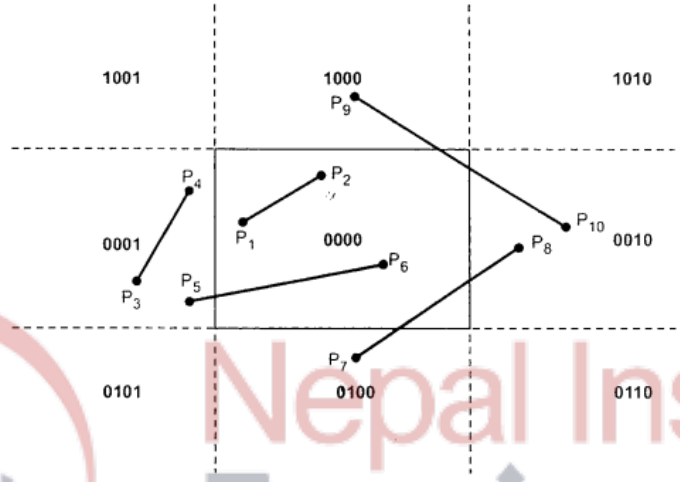


Step 3: If both tests fail then line is partially visible so we need to find the intersection with boundaries of window.
a) If 1st bit is 1 then line intersect with Left boundary and
• $Y_i = Y_1 + m (X – X_1)$ where $X = X_{wmin}$
b) If 2nd bit is 1 then line intersect with Right boundary and
• $Y_i = Y_1 + m (X – X_1)$ where $X = X_{wmax}$
c) If 3rd bit is 1 then line intersect with Bottom boundary and
• $X_i = X_1 + (1/m) (Y – Y_1)$ where $Y = Y_{wmin}$
d) If 4th bit is 1 then line intersect with Top boundary and
• $X_i = X_1 + (1/m) (Y – Y_1)$ where Y= Ywmax

Here, (X_i , Y_i) are (X ,Y) intercepts for that the step 4 repeat step 1 through step 3 until line is completely accepted or rejected.

Example: Consider the clipping window and the line shown in the figure. Find the region codes for each point and identify whether the line is completely visible, partially visible or completely invisible.





Line	End Point Codes		Logical ANDing	Result
$P_1 P_2$	0000	0000	0000	Completely visible
$P_3 P_4$	0001	0001	0001	Completely invisible
$P_5 P_6$	0001	0000	0000	Partially visible
$P_7 P_8$	0100	0010	0000	Partially visible
$P_9 P_{10}$	1000	0010	0000	Partially visible

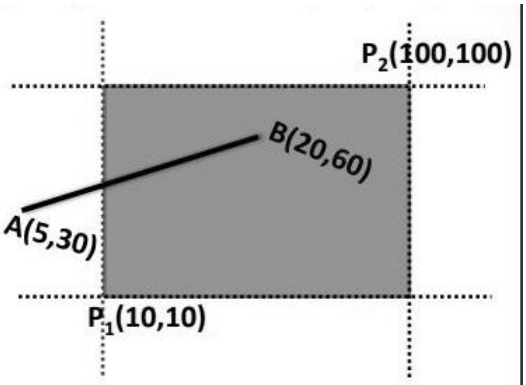
Q.No.1: Clip a line with end points A (5, 30), B (20, 60) against a clip window with lower most left corner at P1 (10, 10) and upper right corner at P2 (100, 100).

Ans:

Step 1: Establish the region

codes for end point A,B

A(5,30)	B(20,60)
5<10 : (true)= 1	20<10: (false) =0
5>100 : (false) =0	20>100: (false)=0
30<10 : (false) =0	60<10 : (false)=0
30>100 : (false) =0	60>100: (false)=0
A(5,30) = 0001	***B(20,60)=0000***



Step 2: Visibility check

A: 0001
AND B: 0000
0000

Partial Visibility

Step 3: Intersection point with boundary

A: 0001 , Cond^{n} of 1st bit is 1

$Y_i = Y_1 + m\,(X - X_1)$ *where ,* X= 10

= 30 + 2 (10-5)

= 40

(X,Y)=(10,40)