C.P.PATEL & F.H.SHAH COMMERCE COLLEGE
(MANAGED BY SARDAR PATEL EDUCATION TRUST)
BCA, BBA (ITM) & PGDCA PROGRAMME
BCA Sem V (COMPUTER GRAPHICS)
UNIT 3 – Transformations and Clipping

UNIT 3 – Transformations and Clipping

- 2-D Geometric Transformations: Translation, Rotation, Scaling, Reflection & Shear (with example)
- Viewing Pipeline, Window-to-Viewport transformation
- Point Clipping
- Line Clipping (Without Program)
- Cohen Sutherland line clipping algorithm
- Polygon Clipping
- Sutherland Hodgeman polygon clipping algorithm
- Text Clipping
- Exterior Clipping

Translation:

T**ranslation** moves every point a constant distance in a specified direction. A translation can be described as a rigid motion: other rigid motions include rotations and reflections. A translation can also be interpreted as the addition of a constant vector to every point, or as shifting the origin of the coordinate system.

Since a translation is an affine transformation but not a linear transformation, homogeneous coordinates are normally used to represent the translation operator by a matrix and thus to make it linear. Thus we write the 3-dimensional vector **w** = (w_x, w_y, w_z) using 4 homogeneous coordinates as **w** = (w_x, w_y, w_z, 1).[1]

To translate an object by a vector **v**, each homogeneous vector **p** (written in homogeneous coordinates) would need to be multiplied by this **translation matrix**:

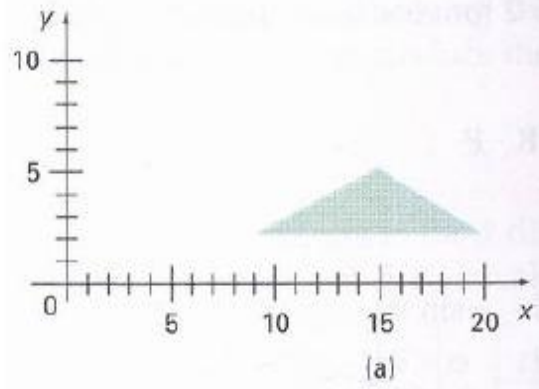


Figure 5-2
Moving a polygon from position (a) to position (b) with the translation vector (-5.50, 3.75).

Rotation

"It is done by repositioning the object by along a circular path in xy plane."

For rotating the object, we specify the two things: **rotation angle Φ**, and **rotation point (pivot point)** about which the object is to be rotated.

We can give **positive** values for rotation in **clockwise** direction and negative values for opposite.
The original polar co-ordinates are given by:

$x = r \cos \Phi$ and $y = r \sin \Phi$ --- (1)

Now, when we increase the angle by Ø (to rotate), the values of new co-ordinates are

$x' = r \cos (\Phi + Ø)$ and $y' = r \sin (\Phi + Ø)$

By expanding the above eqn.,

$X' = r\cos \Phi \cos Ø - r\sin \Phi \sin Ø$
$Y' = r\cos \Phi \sin Ø + r\sin \Phi \cos Ø$

Putting the values of x and y from eqn. (1), we get

$x' = x \cos Ø - Y \sin Ø$
$y' = x \sin Ø + y \cos Ø$ ..... (2)

Similar to the translation, the general eqn. for rotation is given by :
p' = R . P I where the rotation matrix is

$$R = \begin{pmatrix} \cos Ø & -\sin Ø \\ \sin Ø & \cos Ø \end{pmatrix}$$

Representing the co-ordinates in terms of Rows, the matrix will be transposed. And eqn. is given by:

$$P'^{T} = (R.P)^{T} = P^{T}.R^{T}$$

Instead of the center as in eqn.(2), the rotation about any random point (Xr,yr) is given by

$$X' = X_r + (X - X_r) \cos Ø - (Y - Y_r) \sin Ø$$
$$Y' = Y_r + (X - X_r) \sin Ø + (Y - Y_r) \cos Ø$$

Representation of rotation is as below:

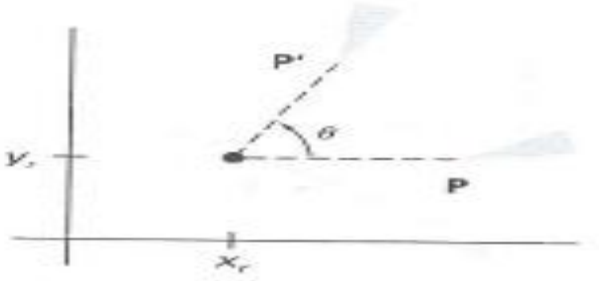


Figure 5-3
Rotation of an object through angle θ about the pivot point (x_r, y_r).

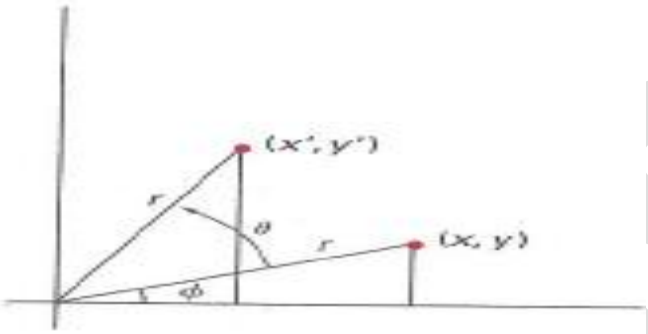


Figure 5-4
Rotation of a point from position (x, y) to position (x', y') through an angle θ relative to the coordinate origin. The original angular displacement of the point from the x axis is ϕ.

Scaling:

A scaling transformation alters the size of an object. This operation can be carried out for polygons by multiplying the coordinates values (x,y) of each vertex by scaling factors s_x,s_y to produce the transformed coordinates(x',y'):

$x' = x.\ s_x$, $y' = y.\ s_y$

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

P' = S.P

Reflection:

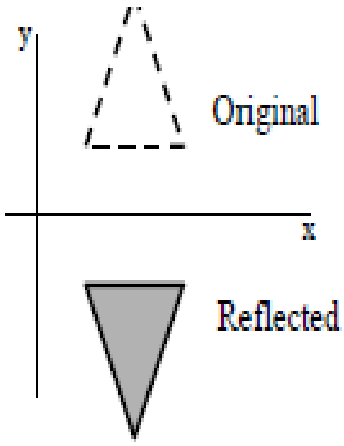
A reflection is a transformation that produces a mirror image of an object. The mirror image for a 2D -reflection is generated relative to an axis of reflection by rotating the object 180' about the reflection axis.

Reflection about the line y =0, the x-axis, is accomplished with the transformation matrix

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

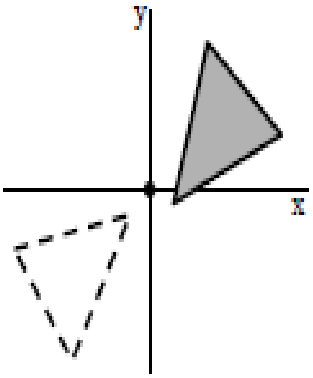
x-axis (y = 0)

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



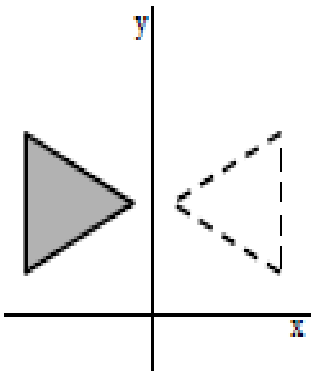
xy-plane

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



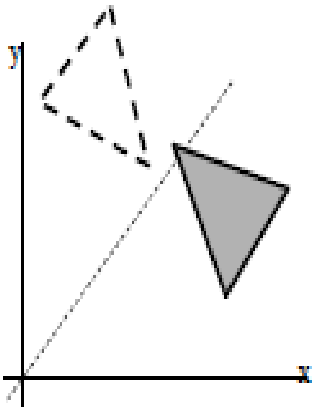
y-axis (x = 0)

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



xy-plane

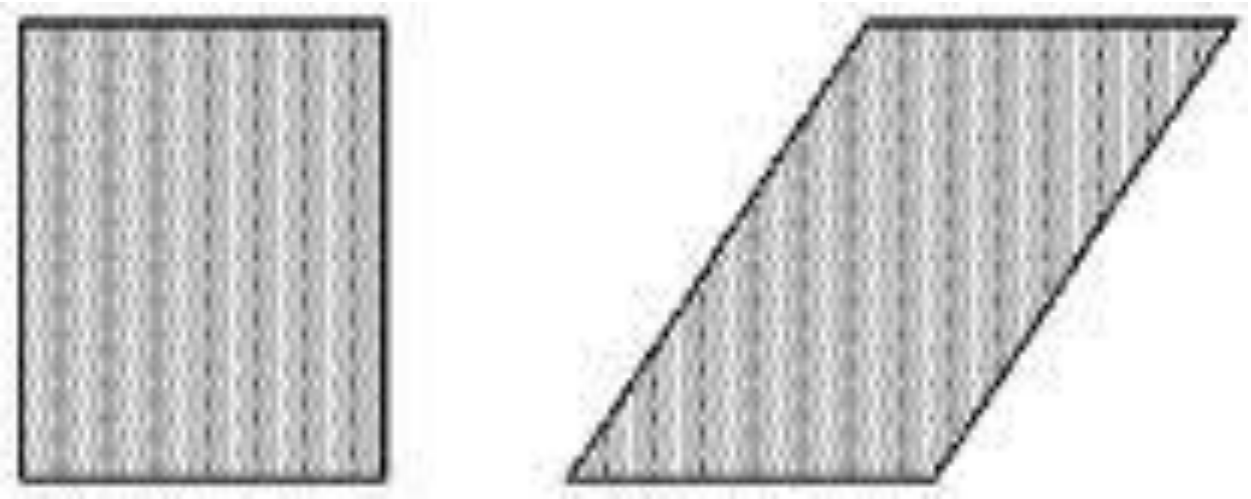
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Shear:

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

$$SH_x = \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad SH_y = \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Viewing Pipeline:

Window: A world co-ordinate area selected for the display.
I
Viewport: An area on display device to which a window is mapped.

Viewing Transformation: The mapping of world co-ordinate scene to device co-ordinate.

The viewing co-ordinate is also called **window to viewport transformation** or the **windowing transformation**.

The viewing transformation is done by different steps as follows:

1. Construct the scene in world co-ordinate by using the output primitive and attributes

2. Obtain the orientation for window and define window in viewing co-ordinate

3. Transform the world co-ordinate to viewing co-ordinate.

4. Define the normalized co-ordinate

5. The viewport is transferred to the device which we require

The viewing pipeline can be represented as below:

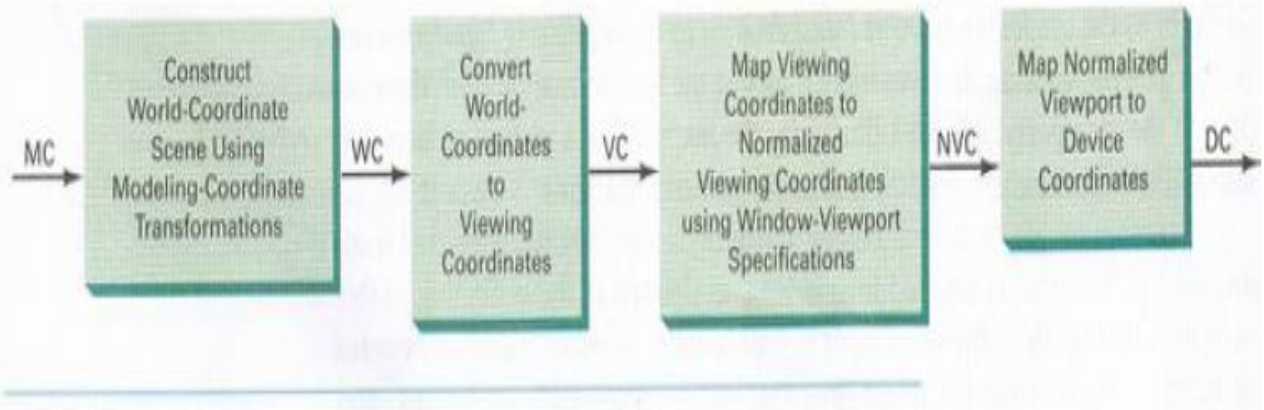


Figure 6-2
The two-dimensional viewing-transformation pipeline.

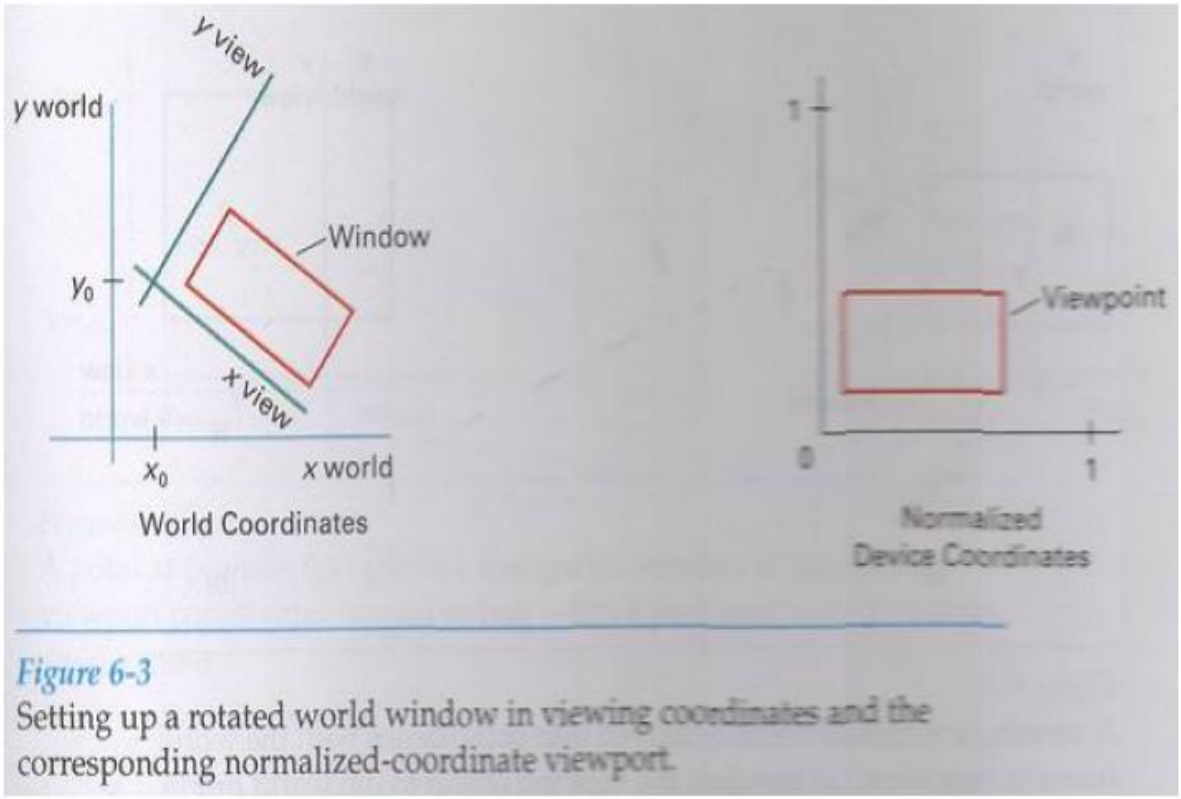


Figure 6-3
Setting up a rotated world window in viewing coordinates and the corresponding normalized-coordinate viewport.

Practically the transformation will be carried out as below:

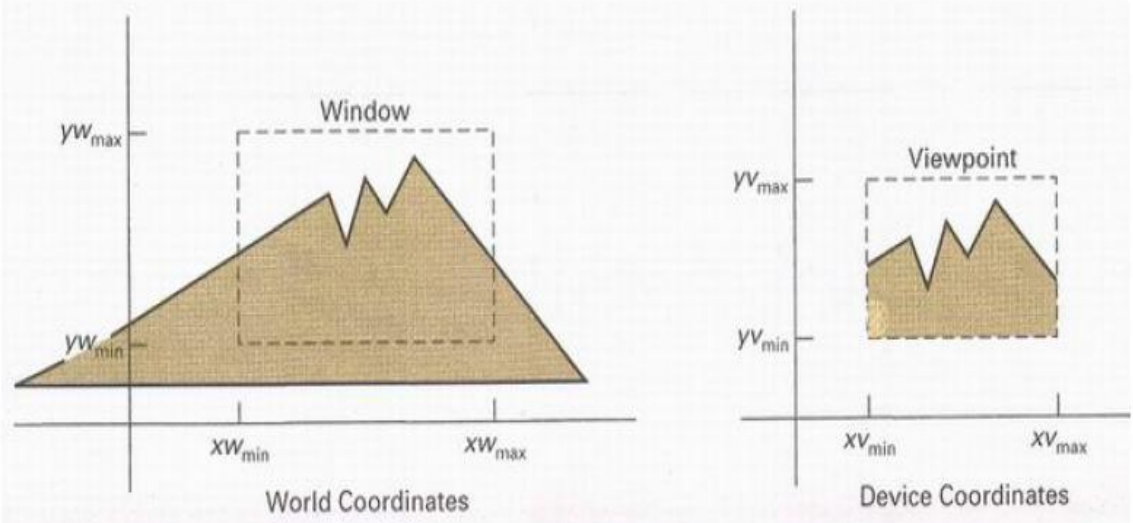
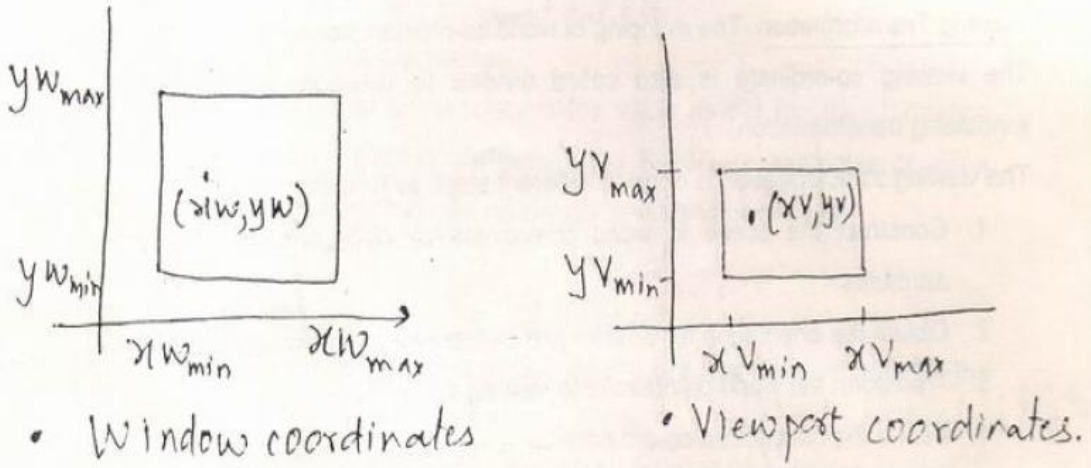


Figure 6-1
A viewing transformation using standard rectangles for the window and viewport.

Window-to-Viewport Coordinate Transformation



• Window coordinates • Viewport coordinates.

The above figure shows the conversion from window coordinate to viewing coordinate. The condition for this conversion is given by:

$$\frac{x_v - xv_{min}}{xv_{max} - xv_{min}} = \frac{x_w - xw_{min}}{xw_{max} - xw_{min}}$$

$$\frac{y_v - yv_{min}}{yv_{max} - yv_{min}} = \frac{y_w - yw_{min}}{yw_{max} - yw_{min}}$$

By solving these equations the coordinates of viewport (xv,yv) has the values

$$xv = xv_{min} + (xw - xw_{min})\, sx$$

$$yv = yv_{min} + (yw - yw_{min})\, sy$$

where, the values of ***sx and sy*** are

$$sx = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}} \qquad sy = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

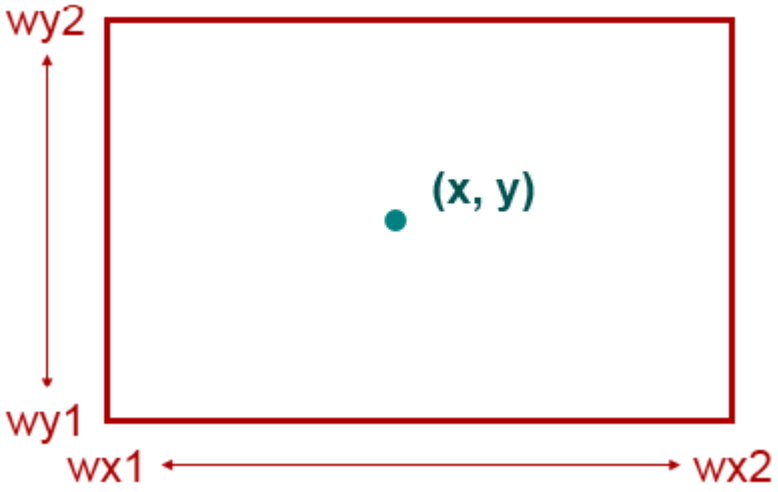
This transformation is performed in following two steps:

1. Perform scaling transformation of (xw_{min} , yw_{min}) to scale from window to viewport
2. Translate the window area into the position of viewport

Point Clipping

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

The X-coordinate of the given point is inside the window, if X lies in between Wx1 ≤ X ≤ Wx2. Same way, Y coordinate of the given point is inside the window, if Y lies in between Wy1 ≤ Y ≤ Wy2.

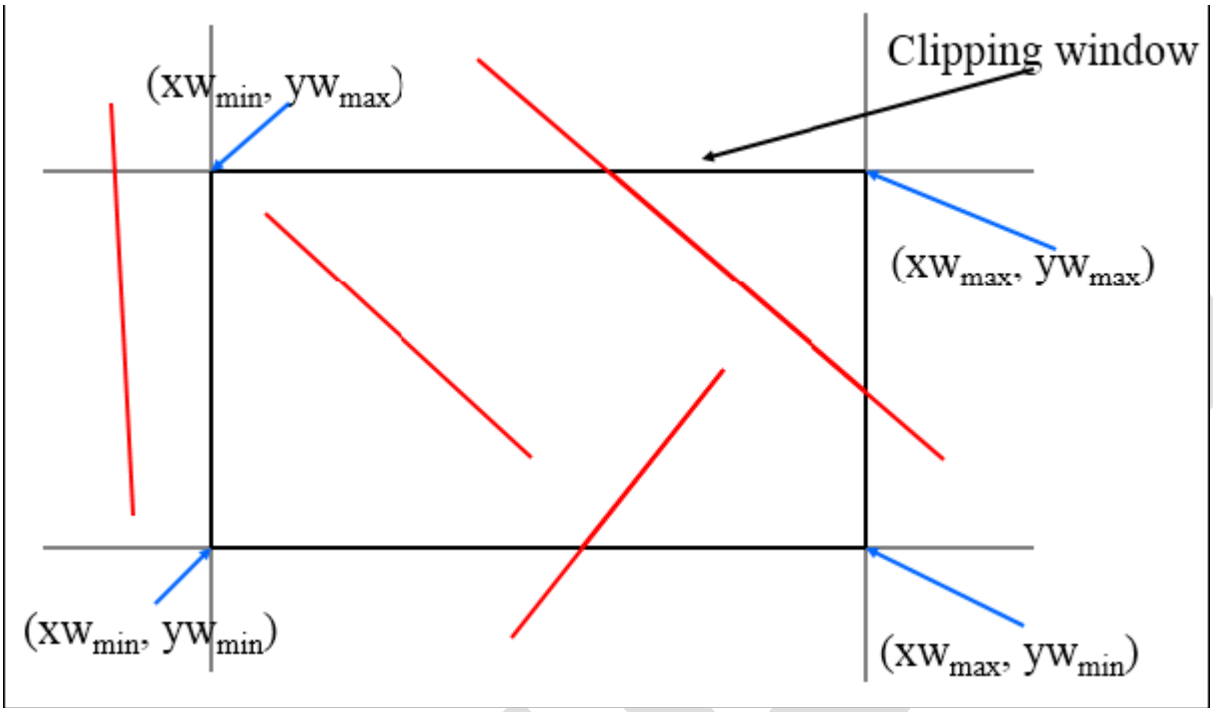


Line Clipping

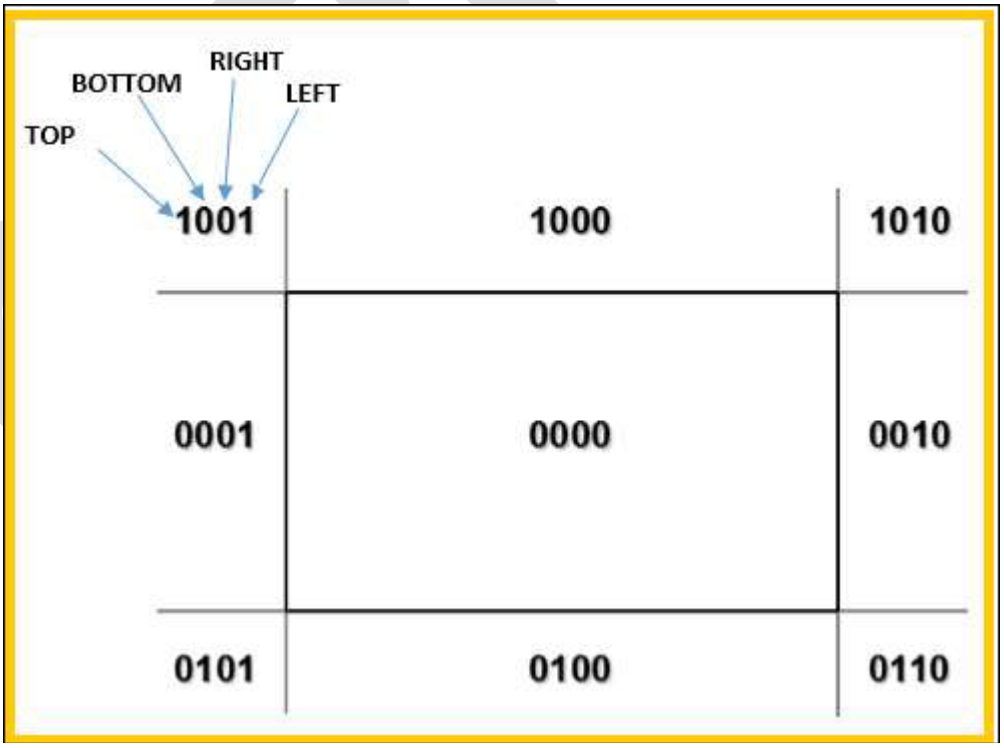
The concept of line clipping is same as point clipping. In line clipping, we will cut the portion of line which is outside of window and keep only the portion that is inside the window.

Cohen Sutherland Line Clipping Algorithm:

This algorithm uses the clipping window as shown in the following figure. The minimum coordinate for the clipping region is (XWmin, YWmin) and the maximum coordinate for the clipping region is (XWmax, YWmax).



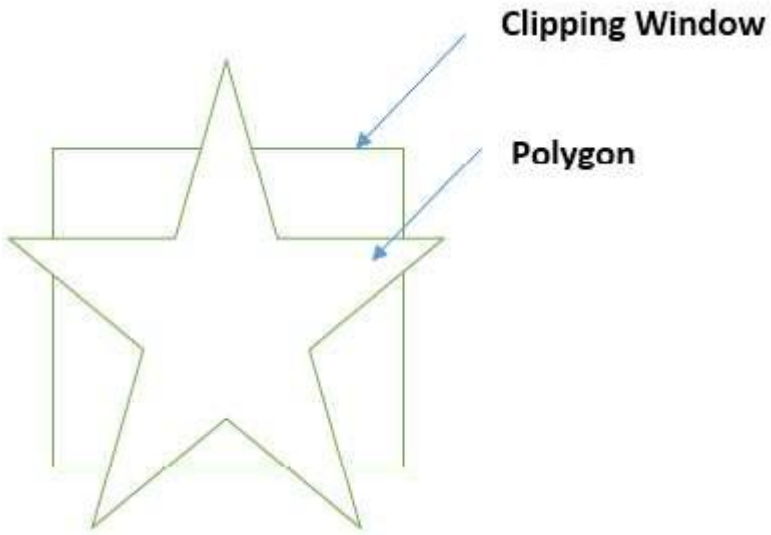
We will use 4-bits to divide the entire region. These 4 bits represent the Top, Bottom, Right, and Left of the region as shown in the following figure. Here, the **TOP** and **LEFT** bit is set to 1 because it is the **TOP-LEFT** corner.



Polygon Clipping (Sutherland Hodgman Algorithm)

A polygon can also be clipped by specifying the clipping window. Sutherland Hodgeman polygon clipping algorithm is used for polygon clipping. In this algorithm, all the vertices of the polygon are clipped against each edge of the clipping window.

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



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

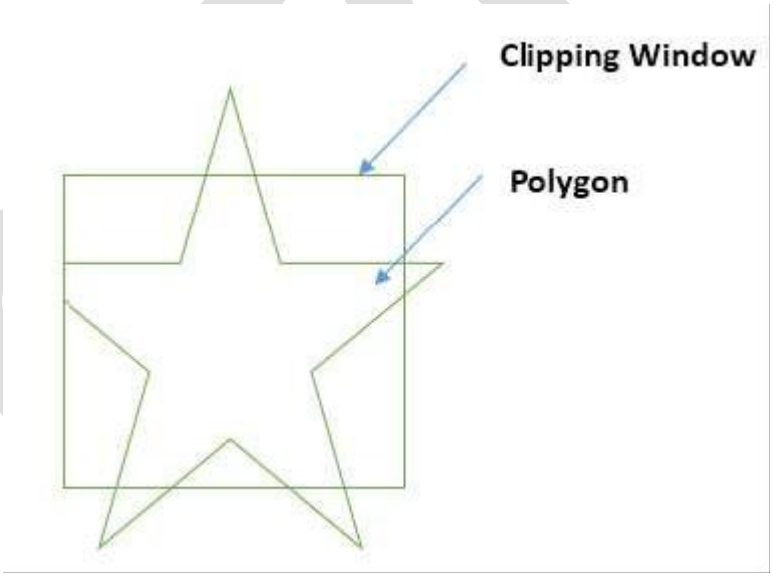


Fig:1

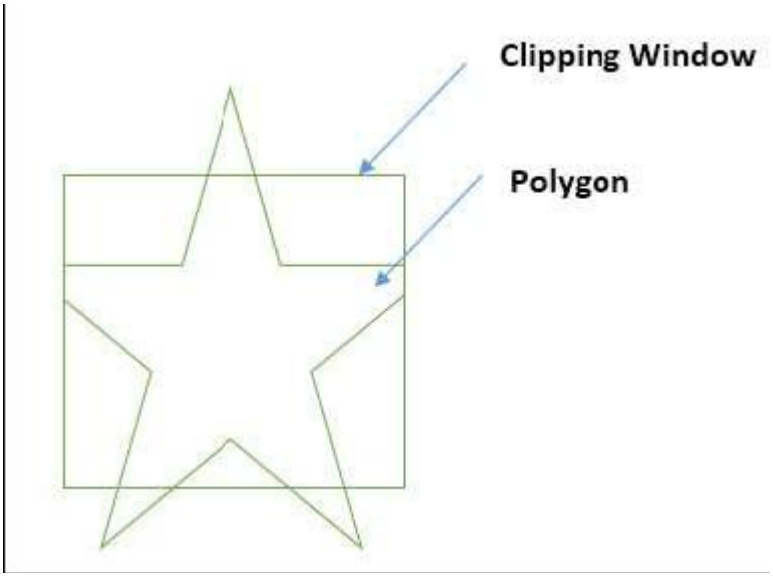

Clipping Window
Polygon

Fig:2

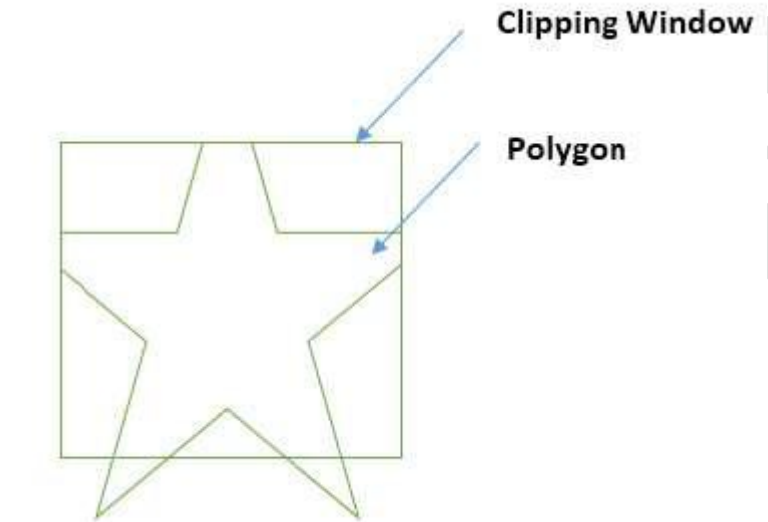

Clipping Window
Polygon

Fig:3

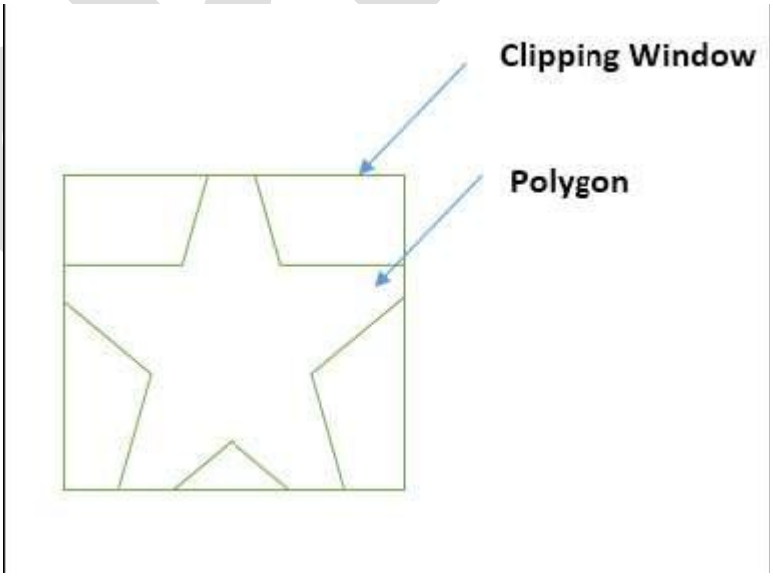
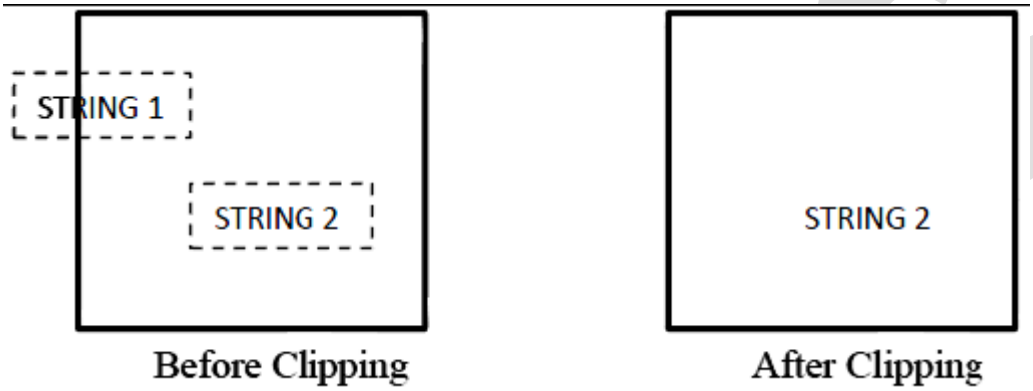

Clipping Window
Polygon

Fig:4

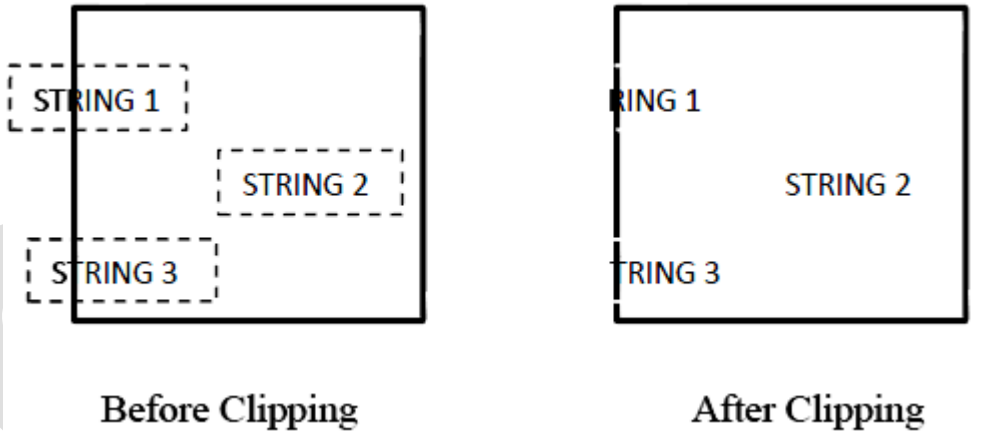
Text Clipping

Various techniques are used to provide text clipping in a computer graphics. It depends on the methods used to generate characters and the requirements of a particular application. There are three methods for text clipping which are listed below:
1. All or none string clipping
2. All or none character clipping
3. Text clipping

The following figure shows all or none string clipping:



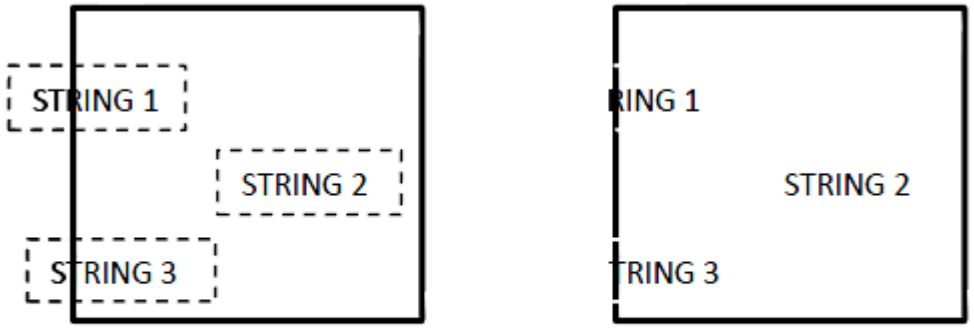
In all or none string clipping method, either we keep the entire string or we reject entire string based on the clipping window. As shown in the above figure, STRING2 is entirely inside the clipping window so we keep it and STRING1 being only partially inside the window, we reject.
The following figure shows all or none character clipping:



This clipping method is based on characters rather than entire string. In this method if the string is entirely inside the clipping window, then we keep it. If it is partially outside the window, then:

- You reject only the portion of the string being outside
- If the character is on the boundary of the clipping window, then we discard that entire character and keep the rest string.

The following figure shows text clipping:



Before Clipping After Clipping

This clipping method is based on characters rather than the entire string. In this method if the string is entirely inside the clipping window, then we keep it. If it is partially outside the window, then

- You reject only the portion of string being outside.
- If the character is on the boundary of the clipping window, then we discard only that portion of character that is outside of the clipping window.