CHAPTER – 4

2D Geometric Transformation

4.1 Basic Transformations

The orientation, size, and shape of the output primitives are accomplished with geometric transformations that alter the coordinate descriptions of objects. The basic geometric transformations are translation, rotation, and scaling. Other transformations that are often applied to objects include reflection and shear. In these all cases we consider the reference point is origin so if we have to do these transformations about any point then we have to shift these point to the origin first and then perform required operation and then again shift to that position.

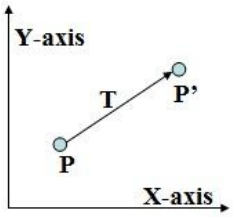
A. Translation

A translation is applied to an object by repositioning it along a straight-line path from one coordinate location to another. We 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').

x' = x + t_x **y' = y + t_y**, where the pair (t_x, t_y) is called the ***translation vector*** or ***shift vector***.

We can write equation as a single matrix equation by using column vectors to represent coordinate points and translation vectors. Thus,

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



*Translating a point from position **P** to position **P'** With translation vector **T**.*

So we can write

$P' = P + T$

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

Translation is a ***rigid-body transformation*** that moves objects without deformation. That is, every point on the object is translated by the same amount

B. Rotation

A two-dimensional rotation is applied to an object by repositioning it along a circular path in the **xy** plane. To generate a rotation, we specify a rotation angle θ and the position (x_r, y_r,) of the rotation point (or pivot point) about which the object is to be rotated.

+ Value for 'θ' define *counter-clockwise* rotation about a point

- Value for 'θ' defines *clockwise* rotation about a point

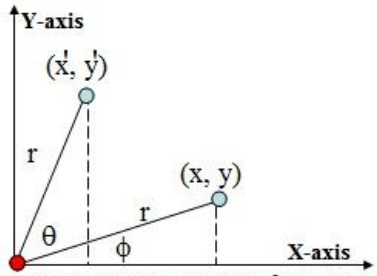


Fig. Rotating a point from position (x, y) to position (x', y ') through an angle θ about rotation point (0, 0). The original angular displacement of the point from the x axis is ϕ.

Let (x, y) is the original point, 'r' the constant distance from origin, & 'Φ' the original angular displacement from x-axis. Now the point (x, y) is rotated through angle 'θ' in a counter clock wise direction

Express the transformed coordinates in terms of 'ϕ' and 'θ' as

x' = r cos(ϕ + θ) = r cosϕ. cosθ - r sinϕ. sinθ ….(i)

y' = r sin(ϕ + θ) = r cosϕ. sinθ + r sinϕ. cosθ ….(ii)

We know that original coordinates of point in polar coordinates are

$x = r\cos\phi$

$y = r\sin\phi$

Substituting these values in (i) and (ii), we get,

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

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

So using column vector representation for coordinate points the matrix form would be

$P' = R \cdot P$

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

Rotation of a point about an arbitrary pivot position can be seen in the figure.

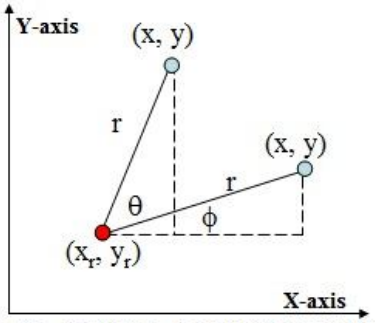


Fig. Rotating a point from position (x, y) to position (x', y ') through an angle θ about rotation point (x_r, y_r).

Here,

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

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

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

Note: - This can also be achieved by translating the arbitrary point into the origin and then apply the rotation and finally perform the reverse translation.

C. Scaling

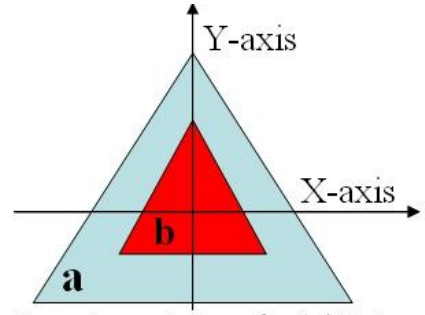
A scaling transformation alters the size of an object. This operation can be carried out for polygons by multiplying the coordinate values (x, y) of each vertex by scaling factors s_x and s_y to produce the transformed coordinates (x', y').

- s_x scales object in 'x' direction
- s_y scales object in 'y' direction

Thus, for equation form, $x' = x.\ s_x$ and $y' = y.\ s_y$

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

- Values greater than 1 for s_x s_y produce ***enlargement***
- Values less than 1 for s_x s_y *reduce* size of object
- $s_x = s_x = 1$ leaves the size of the object ***unchanged***
- When s_x, s_y are assigned the same value $s_x = s_x =$ 3 or 4 etc then a ***Uniform Scaling*** is produced



Turning a triangle (a) Into a triangle (b) with scaling factors $s_x = -2$ and $s_y = -2$

4.2 Other Transformations

Basic transformations such as translation, rotation, and scaling are included in most graphics packages. Some packages provide a few additional transformations that are useful in certain applications. Two such transformations are reflection and shear.

A. Shearing

It distorts the shape of object in either 'x' or 'y' or both direction. In case of single directional shearing (e.g. in 'x' direction can be viewed as an object made up of very thin layer and slid over each other with the *base* remaining where it is). Shearing is a ***non-rigid-body transformation*** that moves objects with deformation.

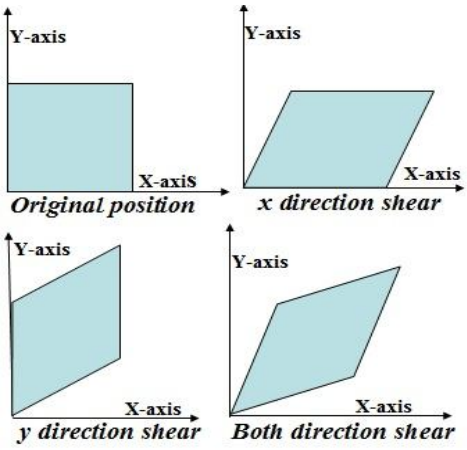


Fig. Two Dimensional shearing

In 'x' direction,

$x' = x + s_{hx}. y$

$y' = y$

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

In 'y' direction,

$x' = x$

$y' = y + s_{hy} . x$

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

In both directions,

$x' = x + s_{hx} . y$

$y' = y + s_{hy} . x$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & s_{hx} \\ s_{hy} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

B. 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. We can choose an axis of reflection in the **xy**-plane or perpendicular to the xy plane. When the reflection axis is a line in the xy plane, the rotation path about this axis is in a plane perpendicular to the **xy**-plane. For reflection axes that are perpendicular to the **xy**-plane, the rotation path is in the xy plane.

(i) Reflection about x axis or about line y = 0

Keeps 'x' value same but flips y value of coordinate points

So $x' = x$

$y' = -y$

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

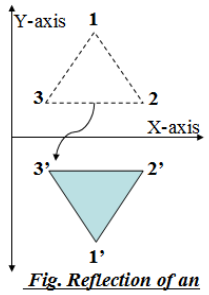


Fig. Reflection of an object about the x axis.

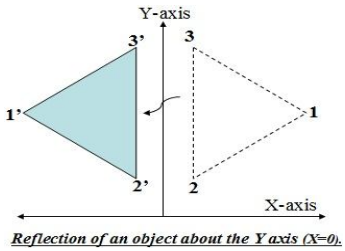
(ii) Reflection about y axis or about line x = 0

Keeps 'y' value same but flips x value of coordinate points

So $x' = -x$

$y' = y$

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



Reflection of an object about the Y axis (X=0).

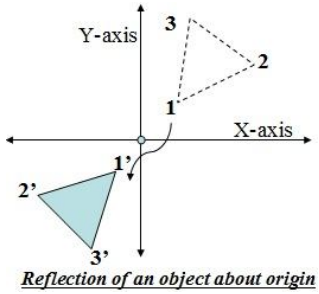
(iii) Reflection about origin

Flip both 'x' and 'y' coordinates of a point

So $x' = -x$

$y' = -y$

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



Reflection of an object about origin

(iv) Reflection about line y = x

Steps required:

- Rotate about origin in clockwise direction by 45 degree which rotates line y = x to x-axis
- Take reflection against x-axis
- Rotate in anti-clockwise direction by same angle

$$R = \begin{pmatrix} \text{Cos}\,\theta & \text{Sin}\,\theta \\ -\text{Sin}\,\theta & \text{Cos}\,\theta \end{pmatrix} \;\&\; R' = \begin{pmatrix} \text{Cos}\,\theta & -\text{Sin}\,\theta \\ \text{Sin}\,\theta & \text{Cos}\,\theta \end{pmatrix}$$

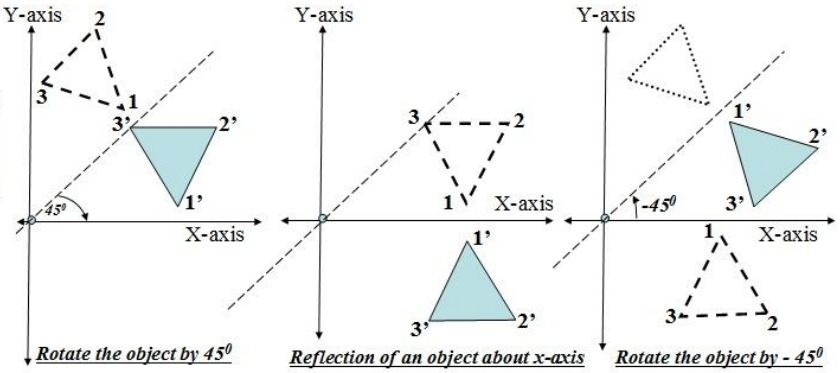
Reflection about in anticlockwise direction (θ =45)

Thus, reflection against x=y-axis (i.e. θ =45)

$$R_{x=y} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Hence

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



Composite matrix = $R_{\theta=-45}\, R_{fx}\, R_{\theta=45}$

Note: If there is more than one transformation should be performed for any task then it is called the composite transformation.

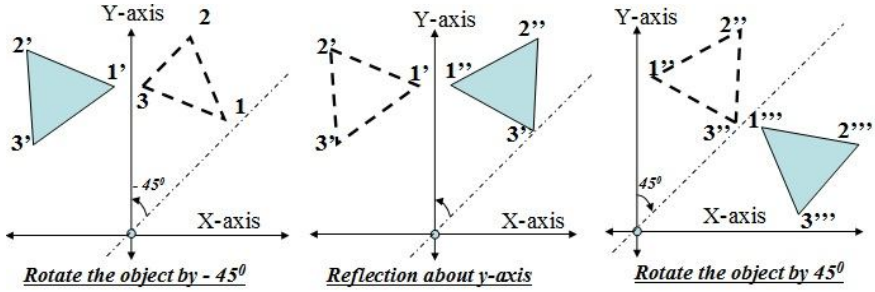
Reflection about in clockwise direction (θ = -45)

Thus, reflection against x=y-axis (i.e. θ = -45)

$$R_{x=y} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$$

Hence

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



Composite matrix = $R_{\theta=45}\, R_{fy}\, R_{\theta=-45}$

(v) Reflection about line y = -x

Steps required:

i. Rotate about origin in clockwise direction by 45
ii. Take reflection against y-axis
iii. Rotate in anti-clockwise direction by same angle

$$R = \begin{pmatrix} \text{Cos}\,\theta & -\text{Sin}\,\theta \\ \text{Sin}\,\theta & \text{Cos}\,\theta \end{pmatrix} \;\&\; R = \begin{pmatrix} \text{Cos}\,\theta & \text{Sin}\,\theta \\ -\text{Sin}\,\theta & \text{Cos}\,\theta \end{pmatrix}$$

Thus, reflection against x=y-axis
in anti-clockwise direction (i.e. θ = 45)

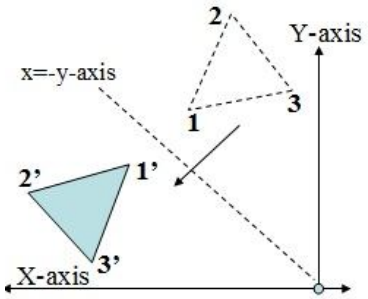
Hence

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad \textit{Where} \quad R_{x=-y} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

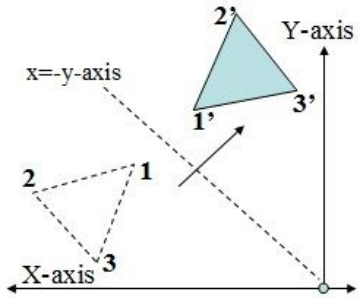
Thus, reflection against x=y-axis
in clockwise direction (i.e. θ = - 45)

Hence

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad \textit{Where} \quad R_{x=-y} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$



Reflection of an object about x= -y in anti-clockwise direction



Reflection of an object about x= -y in clockwise direction

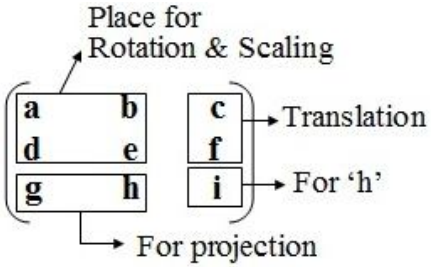
4.3 Homogenous Coordinates

The matrix representations for translation, scaling and rotation are respectively:

Translation: **P' = T + P** (*Addition*)
Scaling: **P' = S . P** (*Multiplication*)
Rotation: **P' = R . P** (*Multiplication*)

Since, the composite transformation such as ***Com* = $R_f T R_\theta T'$** include many sequence of translation, rotation etc and hence the many naturally differ addition & multiplication sequence have to perform by the graphics allocation. Hence, the applications will take more time for rendering. Thus, we need to treat all three transformations in a consistent way so they can be combined easily & compute with one mathematical operation. If points are expressed in homogenous coordinates, all geometrical transformation equations can be represented as matrix multiplications.

Here, in case of homogenous coordinates we add a third coordinate '*h*' to a point (x, y) so that each point is represented by (*h*x, *h*y, *h*). The '*h*' is normally set to 1. If the value of 'h' is more the one value then all the co-ordinate values are scaled by this value.



Coordinates of a point are represented as three element column vectors, transformation operations are written as 3 x 3 matrices.

For translation

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

With T(t_x , t_y) as translation matrix, inverse of this translation matrix is obtained by representing t_x, t_y with $-t_x$, $-t_y$.

For rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \textbf{(a)}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \textbf{(b)}$$

Here, figure-a shows the Counter Clockwise (CCW) rotation & figure-b shows the Clockwise (CW) rotation.

For scaling

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} S_{hx} & 0 & 0 \\ 0 & S_{hy} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Inverse scaling matrix is obtained with $1 / s_{hx}$ and $1 / s_{hx}$.

For Reflection

- Reflection about x-axis

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

- Reflection about y-axis

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

- Reflection about y=x-axis

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

- Reflection about y=-x-axis

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

- Reflection about any line y=mx+c

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{-(m^2-1)}{(m^2+1)} & \frac{2m}{(m^2+1)} & \frac{2mc}{(m^2+1)} \\ \frac{2m}{(m^2+1)} & \frac{(m^2-1)}{(m^2+1)} & \frac{2c}{(m^2+1)} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

4.4 Composite Transformation

With the matrix representation of transformation equations it is possible to setup a matrix for any sequence of transformations as a composite transformation matrix by calculating the matrix product of individual transformation. Forming products of transformation matrices is often referred to as a **concatenation,** or **composition,** of matrices. For column matrix representation of coordinate positions we form composite transformation by multiplying matrices in order from right to left.

- *What do you mean by composite transformation? What is its significance in computer graphics?*

Exercise-I

- *Rotate the triangle (5, 5), (7, 3), (3, 3) about fixed point (5, 4) in counter clockwise (CCW) by 90 degree.*

Solution

Here, the required steps are:

- Translate the fixed point to origin.
- Rotate about the origin by specified angle θ.
- Reverse the translation as performed earlier.

Thus, the composite matrix is given by

$Com = T_{(xf, yf)}\ R_\theta\ T_{(-xf, -yf)}$

$$= \begin{pmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Cos90 & -Sin90 & 0 \\ Sin90 & Cos90 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 & 4 \\ 1 & 0 & -5 \\ 0 & 0 & 1 \end{pmatrix}$$

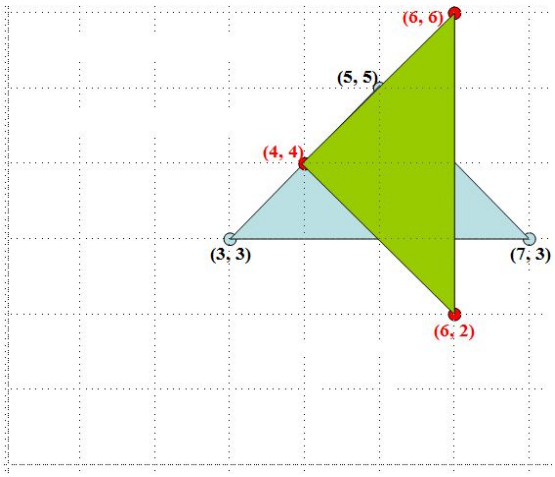
$$= \begin{pmatrix} 0 & -1 & 9 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

Now, the required co-ordinate can be calculated as:

$P' = Com \times P$

$$= \begin{pmatrix} 0 & -1 & 9 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 5 & 7 & 3 \\ 5 & 3 & 3 \\ 1 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 4 & 6 & 6 \\ 4 & 6 & 2 \\ 1 & 1 & 1 \end{pmatrix}$$



Hence, the new required coordinate points are (4, 4), (6, 6) & (6, 2).

Exercise-II

- *Reflect an object (2, 3), (4, 3), (4, 5) about line y = x +1.*

Solution

Here,

The given line is $y = x + 1$.

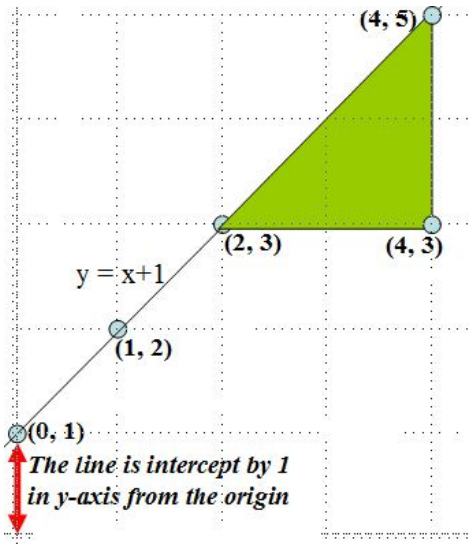
Thus,

When x = 0, y=1
When x = 1, y=2
When x = 2, y=3

Also,

The slope of the line (m) = 1

Thus, the rotation angle (θ) = $Tan^{-1}(m) = Tan^{-1}(1) = 45^0$



Here, the required steps are:

- Translate the line to origin by decreasing the y-intercept with one.
- Rotate the line by angle 45^0 in clockwise direction so that the given line must overlap x-axis.
- Reflect the object about the x-axis.
- Reverse rotate the line by angle -45^0 in counter-clockwise direction.
- Reverse translate the line to original position by adding the y-intercept with one.

Thus, the composite matrix is given by: Com = $\boldsymbol{T' R_\theta' R_{fx} R_\theta T}$

Thus, composite matrix is given by

$$= \underset{\text{Addition y-intercept}}{\begin{pmatrix}1&0&0\\0&1&1\\0&0&1\end{pmatrix}} \underset{\text{CCW Rotation}}{\begin{pmatrix}\text{Cos45}&-\text{Sin45}&0\\\text{Sin45}&\text{Cos45}&0\\0&0&1\end{pmatrix}} \underset{\text{Reflection about x-axis}}{\begin{pmatrix}1&0&0\\0&-1&0\\0&0&1\end{pmatrix}} \underset{\text{CW Rotation}}{\begin{pmatrix}\text{Cos45}&\text{Sin45}&0\\-\text{Sin45}&\text{Cos45}&0\\0&0&1\end{pmatrix}} \underset{\text{Reduce y-intercept}}{\begin{pmatrix}1&0&0\\0&1&-1\\0&0&1\end{pmatrix}}$$

$$= \begin{pmatrix}1&0&0\\0&1&1\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&-1/\sqrt{2}&0\\1/\sqrt{2}&1/\sqrt{2}&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&-1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&1/\sqrt{2}&0\\-1/\sqrt{2}&1/\sqrt{2}&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&1&-1\\0&0&1\end{pmatrix}$$

$$= \begin{pmatrix}1&0&0\\0&1&1\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&-1/\sqrt{2}&0\\1/\sqrt{2}&1/\sqrt{2}&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&-1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&1/\sqrt{2}&-1/\sqrt{2}\\-1/\sqrt{2}&1/\sqrt{2}&-1/\sqrt{2}\\0&0&1\end{pmatrix}$$

$$= \begin{pmatrix}1&0&0\\0&1&1\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&-1/\sqrt{2}&0\\1/\sqrt{2}&1/\sqrt{2}&0\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&1/\sqrt{2}&-1/\sqrt{2}\\1/\sqrt{2}&-1/\sqrt{2}&1/\sqrt{2}\\0&0&1\end{pmatrix}$$

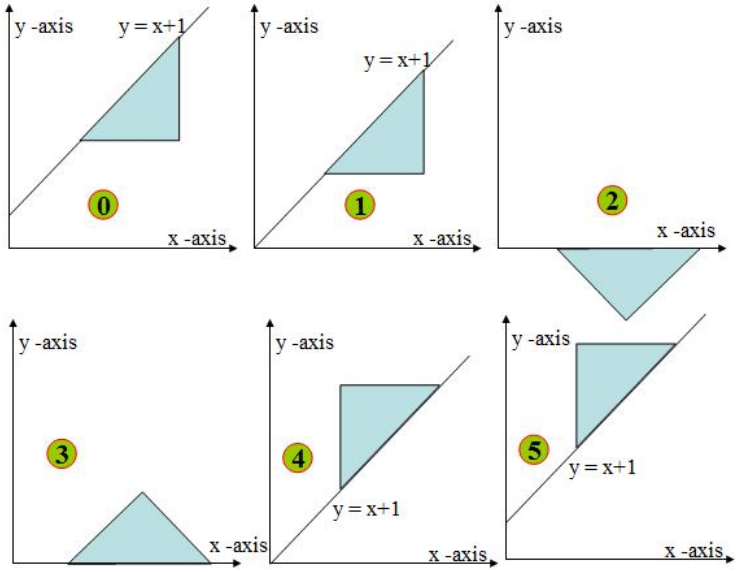
$$= \begin{pmatrix}1&0&0\\0&1&1\\0&0&1\end{pmatrix}\begin{pmatrix}0&1&-1\\1&0&0\\0&0&1\end{pmatrix} = \begin{pmatrix}0&1&-1\\1&0&1\\0&0&1\end{pmatrix}$$

Also, the composite matrix can be calculated as:

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

Where, m =1, c=1

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



Now, the required co-ordinate can be calculated as:

P' = Com x P

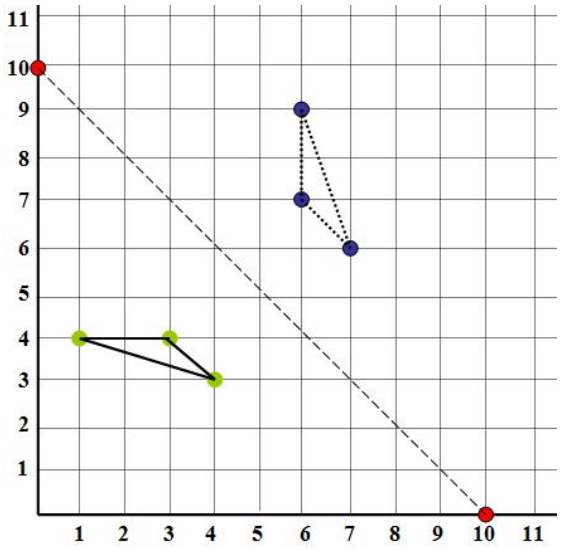
$$= \begin{pmatrix}0&1&-1\\1&0&1\\0&0&1\end{pmatrix}\begin{pmatrix}2&4&4\\3&3&5\\1&1&1\end{pmatrix}$$

$$= \begin{pmatrix}2&2&4\\3&5&5\\1&1&1\end{pmatrix}$$

Hence, the final coordinates are (2, 3), (2, 5) & (4, 5).

Exercise-III (Imp)

- *A mirror is placed such that it passes through (0, 10), (10, 0). Fin the mirror image of an object (6, 7), (7, 6), (6, 9).*



Solution

Here,

The given mirror or line is passing through the points *(0, 10) & (10, 0).*

Now, the slope of the line (m) = $(y_2-y_1) / (x_2-x_1)$ = (0 - 10) / (10 - 0) = -1

Thus, the rotation angle (θ) = $Tan^{-1}(m) = Tan^{-1}(-1) = -45^0$

The composite matrix is given by:

Com = $\boldsymbol{T}_{(0,\,10)\ or\ (10,\,0)}\ \boldsymbol{R}_{\theta\ in\ CW}\ \boldsymbol{R}_{fx}\ \boldsymbol{R}_{\theta\ in\ CCW}\ \boldsymbol{T}_{(0,\,-10)\ or\ (-10,\,0)}$

Here, we can either shift the x-coordinate of mirror to origin or the y-coordinate but not both during the translation. Let us more the x-intercept to origin.

Thus, composite matrix is given by

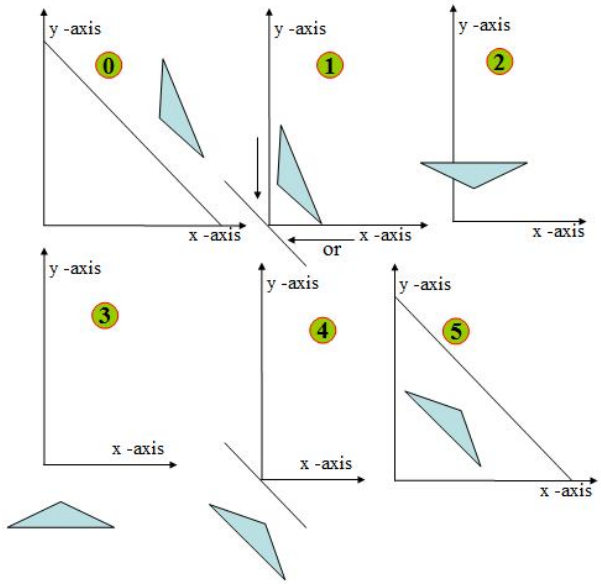
$$= \underset{\text{Addition x-intercept}}{\begin{pmatrix}1&0&10\\0&1&0\\0&0&1\end{pmatrix}} \underset{\text{CW Rotation}}{\begin{pmatrix}Cos45&Sin45&0\\-Sin45&Cos45&0\\0&0&1\end{pmatrix}} \underset{\text{Reflection about x-axis}}{\begin{pmatrix}1&0&0\\0&-1&0\\0&0&1\end{pmatrix}} \underset{\text{CCW Rotation}}{\begin{pmatrix}Cos45&-Sin45&0\\Sin45&Cos45&0\\0&0&1\end{pmatrix}} \underset{\text{Reduce x-intercept}}{\begin{pmatrix}1&0&-10\\0&1&0\\0&0&1\end{pmatrix}}$$

$$= \begin{pmatrix}1&0&10\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&1/\sqrt{2}&0\\-1/\sqrt{2}&1/\sqrt{2}&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&-1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&-1/\sqrt{2}&0\\1/\sqrt{2}&1/\sqrt{2}&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&-10\\0&1&0\\0&0&1\end{pmatrix}$$

$$= \begin{pmatrix}1&0&10\\0&1&1\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&1/\sqrt{2}&0\\-1/\sqrt{2}&1/\sqrt{2}&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&-1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&-1/\sqrt{2}&-10/\sqrt{2}\\1/\sqrt{2}&1/\sqrt{2}&-10/\sqrt{2}\\0&0&1\end{pmatrix}$$

$$= \begin{pmatrix}1&0&10\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&1/\sqrt{2}&0\\-1/\sqrt{2}&1/\sqrt{2}&0\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&-1/\sqrt{2}&-10/\sqrt{2}\\-1/\sqrt{2}&-1/\sqrt{2}&10/\sqrt{2}\\0&0&1\end{pmatrix}$$

$$= \begin{pmatrix}1&0&10\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}0&-1&0\\-1&0&10\\0&0&1\end{pmatrix} = \begin{pmatrix}0&-1&10\\-1&0&10\\0&0&1\end{pmatrix}$$



Now, the required co-ordinate can be calculated as:

P' = Com x P

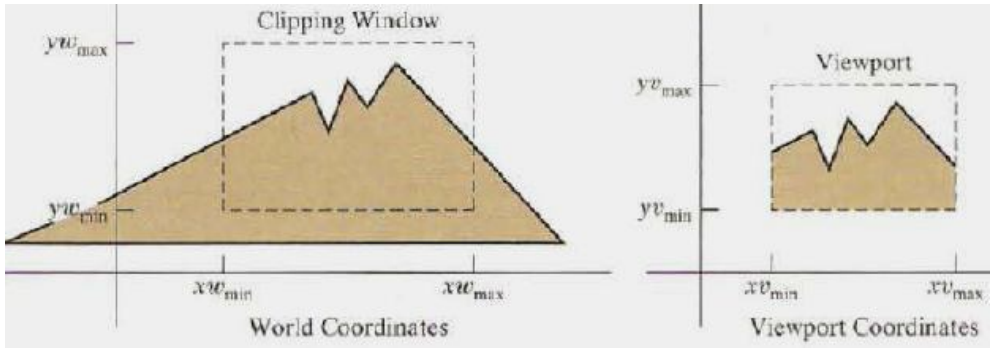
$$= \begin{pmatrix}0&-1&10\\-1&0&10\\0&0&1\end{pmatrix}\begin{pmatrix}6&7&6\\7&6&9\\1&1&1\end{pmatrix}$$

$$= \begin{pmatrix}3&4&1\\4&3&4\\1&1&1\end{pmatrix}$$

Hence, the final coordinates are (3, 4), (4, 3) & (1, 4).

4.5 Window to View port Transformation

A world-coordinate area selected for display is called a ***window***. An area on a display device to which a window is mapped is called a ***viewport***. The window defines *what* is to be viewed; the viewport defines *where* it is to be displayed.



Often, windows and viewports are rectangles in standard position (*sometime polygon shapes and circles but these takes longer process)*), with the rectangle edges parallel to the coordinate axes. In general, the mapping of a part of a world-coordinate scene to device coordinates is referred to as a viewing transformation. Sometimes the 2D viewing transformation is simply referred to as the *window-to-viewport transformation* or the *windowing transformation.*

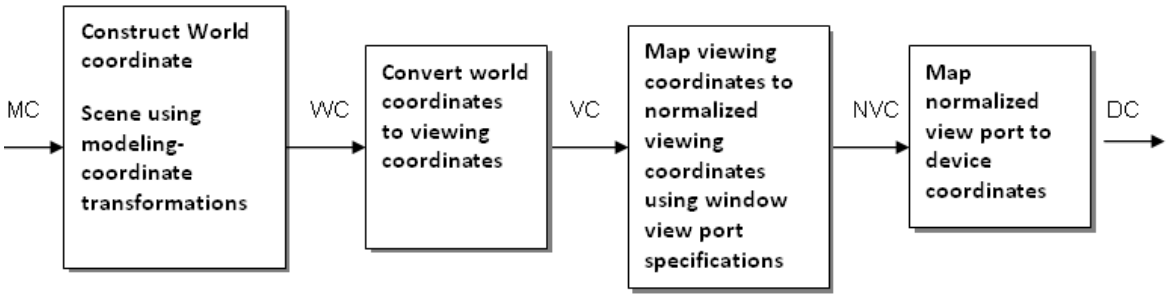
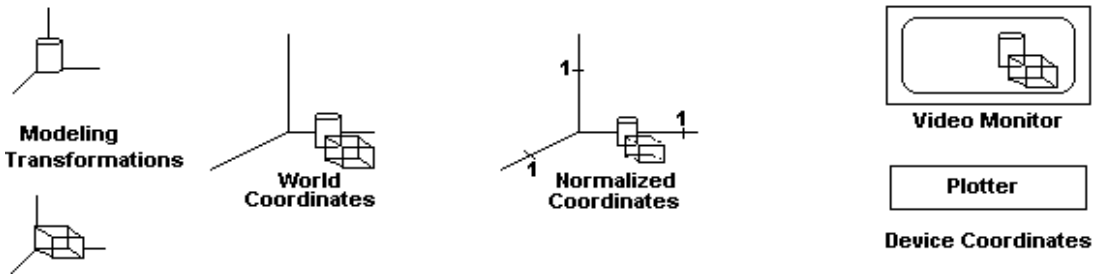


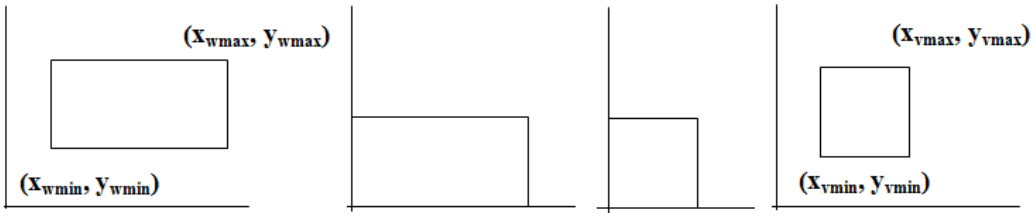
Fig. 2-D Viewing Transformation Pipeline



To transform a window to the view port we have to perform the following steps:

The overall transformation which performs these three steps called the viewing transformation. Let the window coordinates be (x_{wmin}, y_{wmin}) and (x_{wmax}, y_{wmax}) where as the view port coordinates be (x_{vmin}, y_{vmin}) and (x_{vmax}, y_{vmax}).

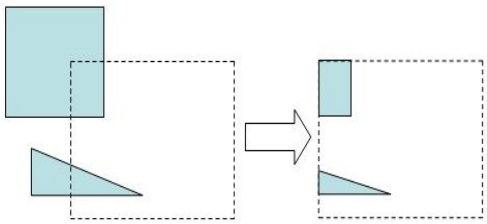
- **Step1:** The object together with its window is translated until the lower left corner of the window is at the origin
- **Step2:** The object and window are scaled until the window has the dimensions of the view port
- **Step3:** Again translate to move the view port to its correct position on the screen



4.6 Clipping in Raster World

Generally, any procedure that identifies those portions of a picture that are either *inside* or *outside* of a specified region of space is referred to as a ***clipping algorithm***, or simply ***clipping***. The region against which an object is to clip is called a clip ***window.*** In other word, the clipping defines how much we show or hide the object on the window. Here, our perception must be like as a machine to detect the visible surface. An application of clipping includes:

- Extracting parts of defined scene for viewing
- Identifying visible surfaces in three dimension views
- Drawing, painting operations that allow parts of picture to be selected for copying, moving, erasing or duplicating etc.



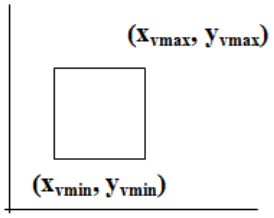
The clipping primitives are:

- Point Clipping
- Line Clipping (straight-line segments)
- Area Clipping (polygons)
- Curve Clipping
- Text Clipping

A) 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:

- $x_{min} \leq x \leq x_{max}$
- $y_{min} \leq y \leq y_{max}$



Here the edges of the clip window (x_{min}, x_{max}, y_{min}, y_{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 (not saved for display).

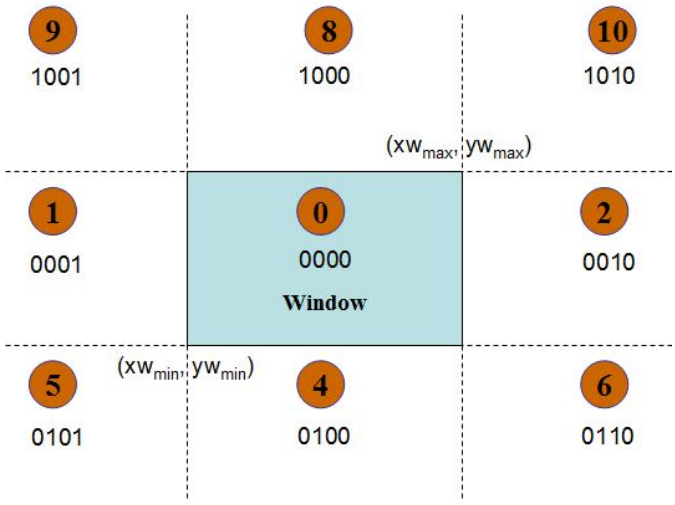
B) Line clipping

Cohen-Sutherland Line Clipping Algorithm

- Divide 2D space into 3x3 = 9-regions.
- Middle region is the **clipping window.**
- Each region is assigned **a** 4-bit code.

Bit 1 is set to 1 if the region is to the **left** of the clipping window, otherwise. Similarly we deal for bits 2, 3 and 4.

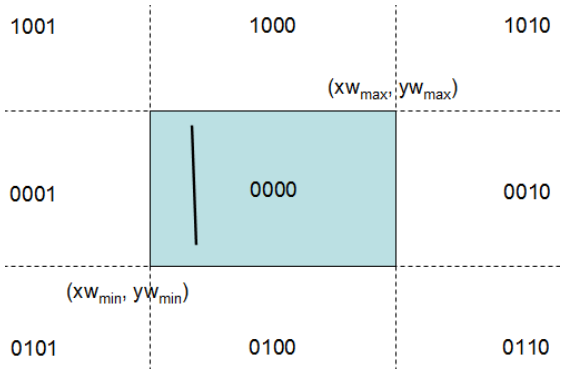
4	3	2	1
Top	Bottom	Right	Left



- If the left = 1 then right positional region code must be zero & similarly, if top = 1 then bottom = 0.
- To clip a line, find out which regions its two endpoints lie in.

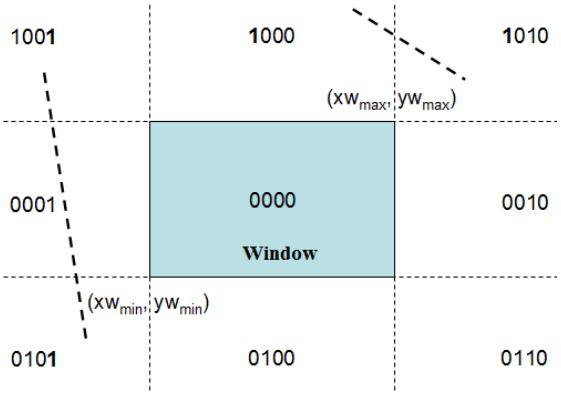
Case I

- If they are **both in** region 0000, then it's completely **in**.



Case II

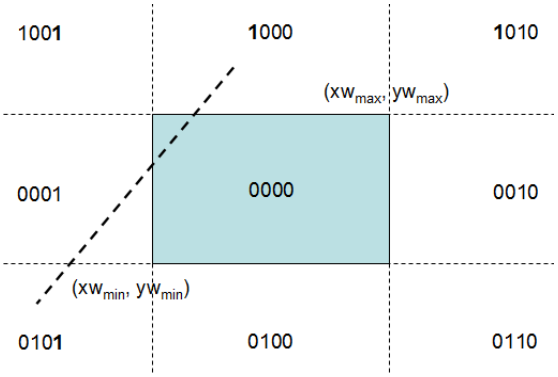
- If the two region numbers both have a 1 in the same bit position, the line is **completely out**.



Case III

- If lines can not be identified as completely inside or outside we have to **do some more calculations.**

- Here we find the intersection points with a clipping boundary using the slope intercept form of the line equation
- Here, to find the visible surface, the intersection points on the boundary of window can be determined as:
  - $y-y_1 = m\,(x-x_1)$
  - where $m = (y_2-y_1)/(x_2-x_1)$

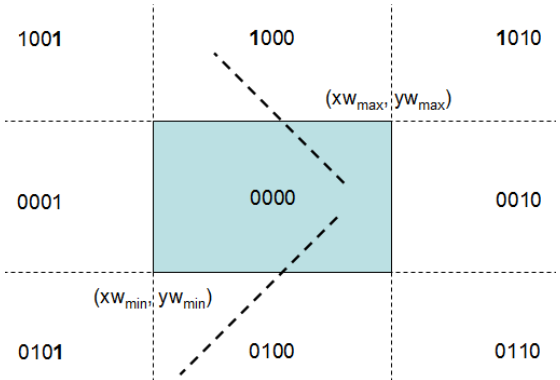


Condition-1

- For a line with end point coordinates (x1,y1) and (x2, y2) the y coordinate of the intersection point with a horizontal boundary is

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

  Where x is set to either xw_{min} or xw_{max}

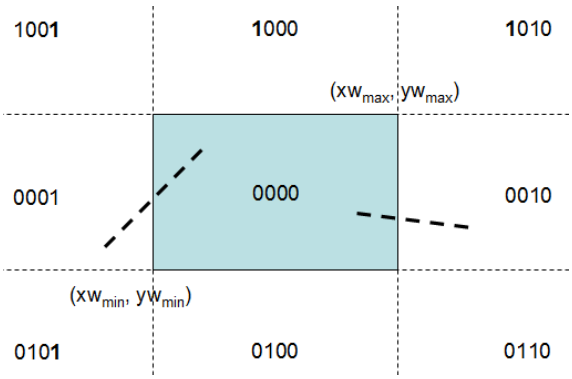


Condition-2

- Similarly for the intersection with a vertical boundary

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

  Where y is set to either yw_{min} or yw_{max}



Example-1

- *Use Cohen Sutherland line clipping algorithm to clip a line with end point coordinates A(70, 90), B(60, 85) against a clip window with its lower left corner at (50,80) and upper right corner at (90,120)*

Solution

Finding the region code

For A (70, 90)	*For B (60, 85)*
▪ $x - xw_{min} = 70 - 50$ = Positive → 0 (Left) ▪ $xw_{max} - x = 90 - 70$ = Positive → 0 (Right) ▪ $y - yw_{min} = 90 - 80$ = Positive → 0 (Bottom) ▪ $yw_{max} - y = 120 - 90$ = Positive → 0 (Top) ***Hence the region code = 0000***	▪ $x - xw_{min} = 60 - 50$ = Positive → 0 (Left) ▪ $xw_{max} - x = 90 - 60$ = Positive → 0 (Right) ▪ $y - yw_{min} = 85 - 80$ = Positive → 0 (Bottom) ▪ $yw_{max} - y = 120 - 85$ = Positive → 0 (Top) ***Hence the region code = 0000***

Here, both the end point of the line has the region code 0000, hence both the end point are in the clip-window. This means the line is totally inside the window & thus totally visible.

Example-2

- *Use Cohen Sutherland line clipping algorithm to clip a line with end point coordinates A(70,90), B(100, 130) against a clip window with its lower left corner at (50,80) and upper right corner at (90,120)*

Solution

1. **Finding the region code**

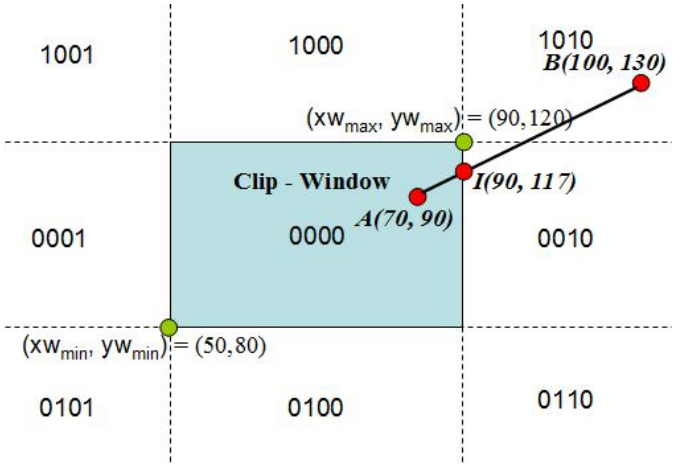
For A (70, 90)	*For B (100, 130)*
▪ $x - xw_{min} = 70 - 50$ = Positive → 0 (Left) ▪ $xw_{max} - x = 90 - 70$ = Positive → 0 (Right) ▪ $y - yw_{min} = 90 - 80$ = Positive → 0 (Bottom) ▪ $yw_{max} - y = 120 - 90$ = Positive → 0 (Top) ***Hence the region code = 0000***	▪ $x - xw_{min} = 100 - 50$ = Positive → 0 (Left) ▪ $xw_{max} - x = 90 - 100$ = Negative → 1 (Right) ▪ $y - yw_{min} = 130 - 80$ = Positive → 0 (Bottom) ▪ $yw_{max} - y = 120 - 130$ = Negative → 1 (Top) ***Hence the region code = 1010***

2. **Finding the position of line & intersection point on the clip window**

Here,
Slope of the line with end point *A(70, 90) & B(100, 130)* is given by
Slope (m) = $(y_2 - y_1)/(x_2 - x_1)$
Or m = (130 – 90) / (100-70) = 4/3
Now,
For x = 90, the value of 'y' can be calculated by using the following equation:

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

Let us take the point B (100, 130) = (x_1, y_1).
Thus,
y = 130 + 4/3*(90-100) = 116.666 = 117
Hence, the intersection point is I (90, 117).



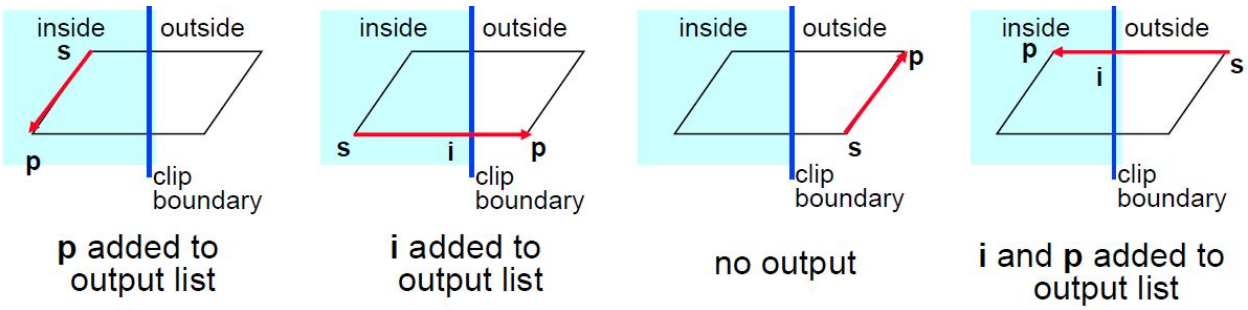
Now, we can conclude that the visible portion of the line has the end points A (70, 90) & I (90, 117).

C) Polygon Clipping: Sutherland - Hodgeman

Any procedure which identifies that portion of a picture which is either inside or outside a region is referred to as a *clipping algorithm* or *clipping.* The region against which an object is to be clipped is called clipping window. The *Sutherland–Hodgeman algorithm* is used for clipping polygons. A single polygon can actually be split into multiple polygons. The algorithm clips a polygon against all edges of the clipping region in turn. This algorithm is actually quite general — the clip region can be any convex polygon in 2D, or any convex polyhedron in 3D.

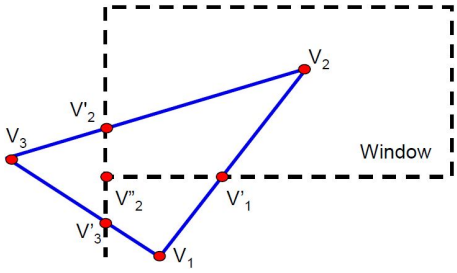
Table 1: Rules for Sutherland-Hodgman clipping.

Case	1st vertex	2nd vertex	output
1	inside	inside	2nd vertex
2	inside	outside	intersection
3	outside	outside	none
4	outside	inside	2nd and intersection



For example:

From	To	1st point	2nd point	Case	Output list
V_1	V_2	Outside	Inside	4	V'_1 and V_2
V_2	V_3	Inside	Outside	2	V'_2
V_3	V_1	Outside	Outside	3	



Assignment: