Two-Dimensional Geometric Transformations

Unit 3

Transformation

- Changing Position, shape, size, or orientation of an object on display is known as transformation.
- Basic transformation includes three transformations *Translation*, *Rotation*, and *Scaling*.
- Other transformations that are often applied to objects include *reflection and shear*.

Homogenous Coordinates

- Expressing position in homogeneous coordinates allows us to represent all geometric transformations equation as matrix multiplications.

$$(x, y) \Rightarrow (x_h, y_h, h)$$

where $x = \frac{x_h}{h}, y = \frac{y_h}{h}$ and h is any non zero value.

- For convenient $h = 1$

$$(x, y) \Rightarrow (x, y, h)$$

Translation

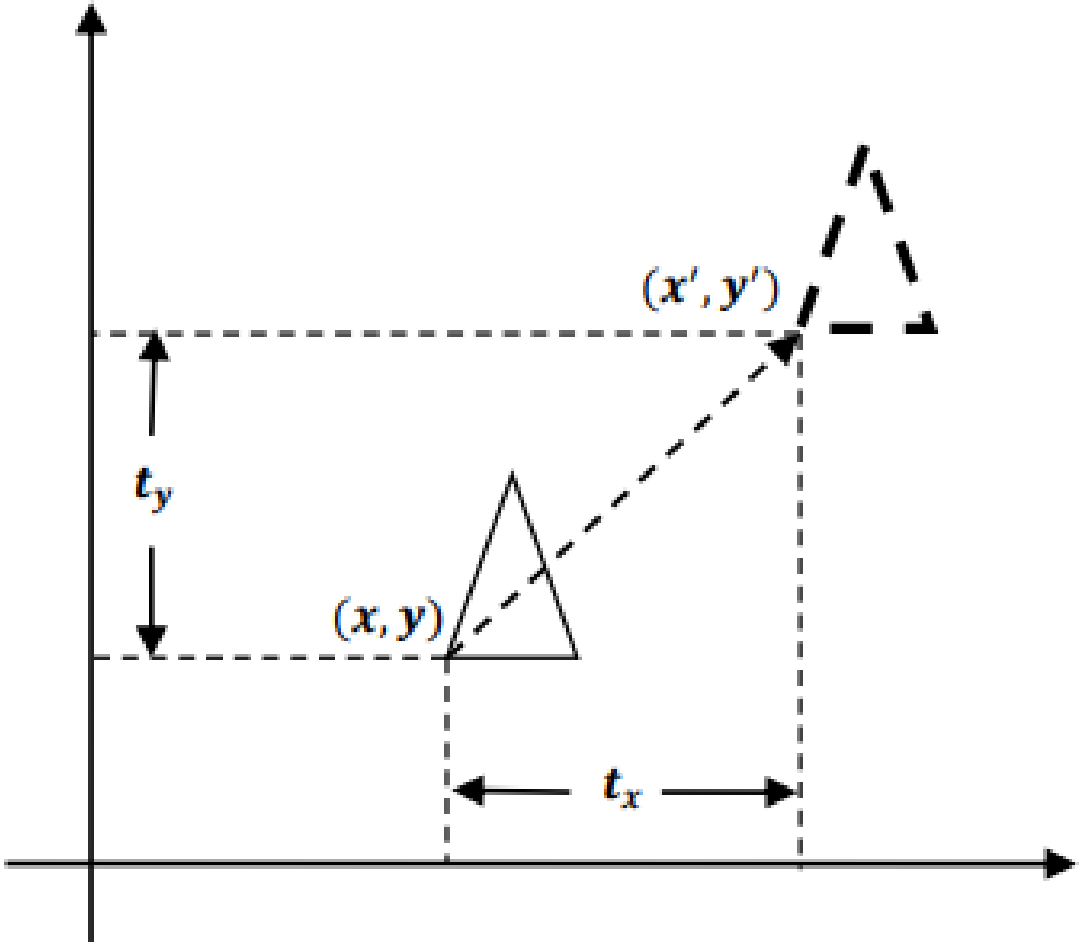


Figure: Translation.

Translation Cont....

- It is a transformation that used to reposition the object along the straight line path from one coordinate location to another.
- We translate two dimensional point by adding translation distance t_x and t_y to the original coordinate position (x, y) to move at new position (x', y') as:

$$x' = x + t_x \qquad \& \qquad y' = y + t_y$$

- Translation distance pair (t_x, t_y)
- is called a *Translation Vector* or *Shift Vector*.
- It is represented into single matrix equation in column vector as:

$$P' = P + T$$

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

- It can also be represented in row vector form as:

$$P' = P + T$$

$$[x' \quad y'] = [x \quad y] + [tx \quad ty]$$

Translation Cont....

- In homogeneous representation if position $P = (x, y)$ is translated to new position $P' = (x', y')$ then:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow P' = T(t_x, t_y).P$$

Translation--Example

Example: - Translate the triangle

$$[A\ (10,10), B\ (15,15), C\ (20,10)]$$

2 unit in x direction and 1 unit in y direction.

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

- It is a transformation that used to reposition the object along the circular path in the XY-plane

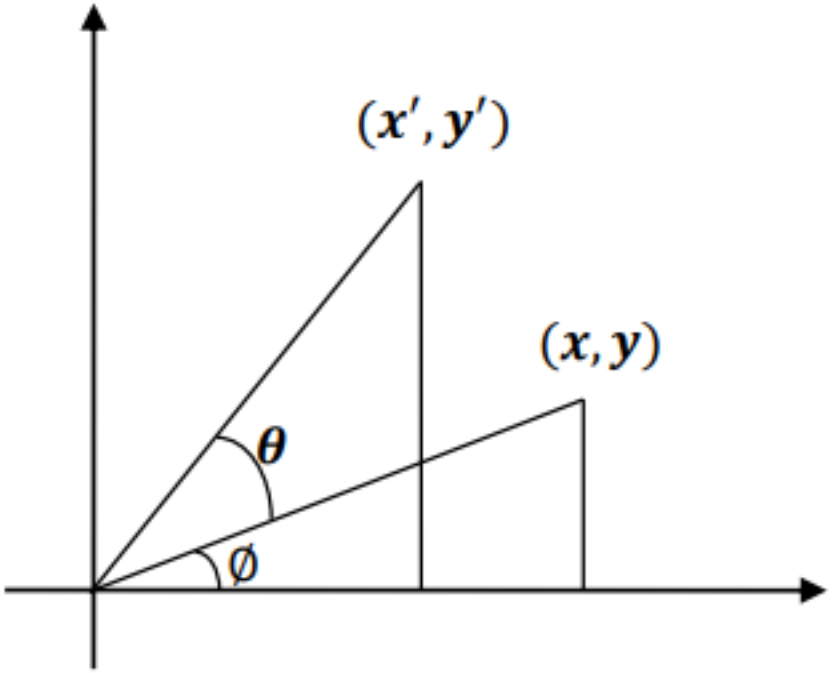


Figure: Rotation

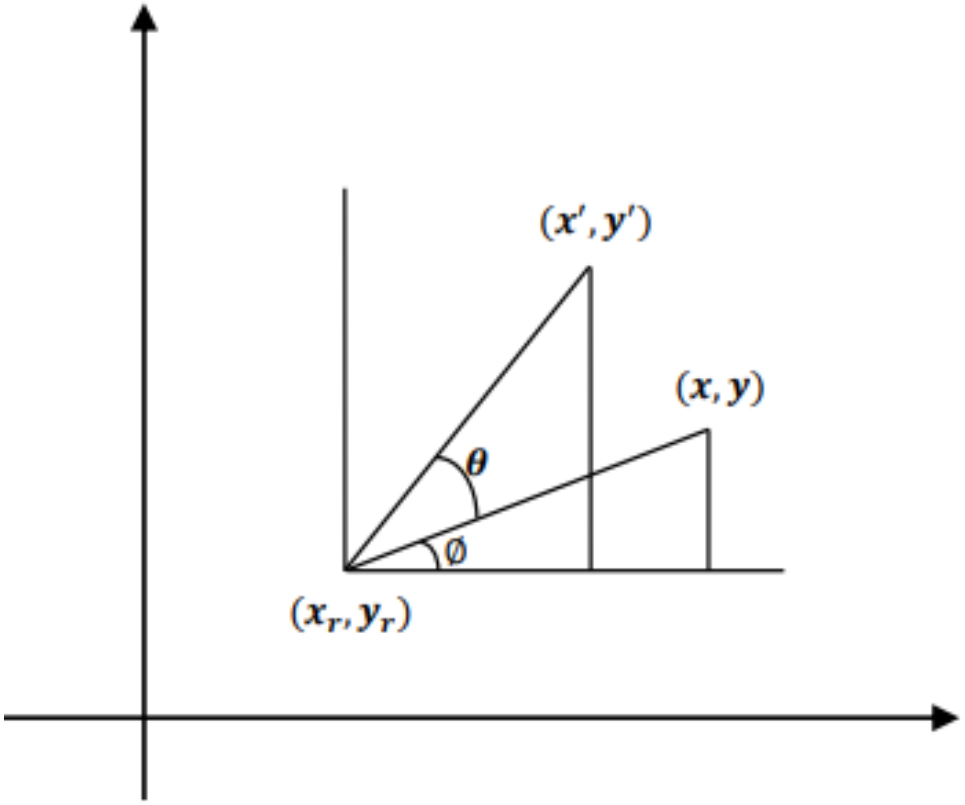


Figure: Rotation about pivot point.

Rotation Cont....

- From figure we can write.

$$x = r \cos \emptyset$$

$$y = r \sin \emptyset$$

and

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

$$y' = r \sin(\emptyset + \theta) = r \cos \emptyset \sin \theta + r \sin \emptyset \cos \theta$$

- Now replace $r \cos \emptyset$ with x and $r \sin \emptyset$ with y in above equation.

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

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

- We can write it in the form of column vector matrix equation as;

$$P' = R \cdot P$$

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

Rotation Cont....

- Transformation equation for rotation of a point about pivot point (x_r, y_r) is:

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

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

- In homogeneous co-ordinate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow P' = R(\theta).P$$

Rotation Cont.…

Example:

- Locate the new position of the triangle [A (5, 4), B (8, 3), C (8, 8)] after its rotation by 90º clockwise about the origin.
- As rotation is clockwise we will take $\theta = -90^{\circ}$.

$$P' = R \cdot P$$

$$P' = \begin{bmatrix} \cos(-90) & -\sin(-90) \\ \sin(-90) & \cos(-90) \end{bmatrix} \begin{bmatrix} 5 & 8 & 8 \\ 4 & 3 & 8 \end{bmatrix}$$

$$P' = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 5 & 8 & 8 \\ 4 & 3 & 8 \end{bmatrix}$$

$$P' = \begin{bmatrix} 4 & 3 & 8 \\ -5 & -8 & -8 \end{bmatrix}$$

- Final coordinates after rotation are [A' (4, -5), B' (3, -8), C' (8, -8)].

Scaling

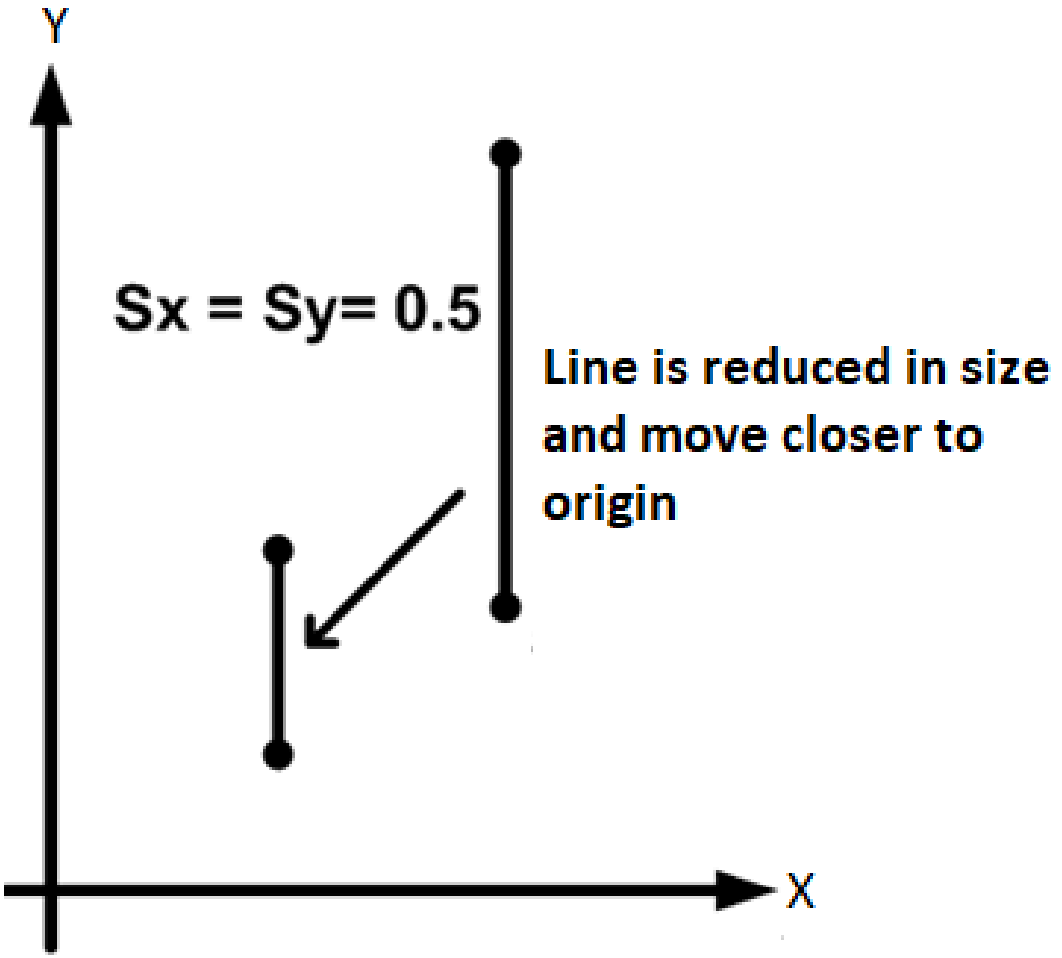


Figure: Scaling

Scaling Cont....

- 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
- So equation for scaling is given by:

$$x' = x \cdot S_x$$

$$y' = y \cdot S_y$$

- These equation can be represented in column vector matrix equation as: $P' = S \cdot P$

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

Scaling Cont....

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

- In homogenous coordinate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow P' = S(s_x, s_y).P$$

Scaling Cont....

Example:

Consider square with left-bottom corner at (2, 2) and right-top corner at (6, 6) apply the transformation which makes its size half.

- ✓ As we want size half so value of scale factor are $s_x = 0.5$, $s_y = 0.5$ and Coordinates of square are $[A(2,2), B(6,2), C(6,6), D(2,6)]$.

$$P' = S \cdot P$$

$$P' = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} 2 & 6 & 6 & 2 \\ 2 & 2 & 6 & 6 \end{bmatrix}$$

$$P' = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 2 & 6 & 6 & 2 \\ 2 & 2 & 6 & 6 \end{bmatrix}$$

$$P' = \begin{bmatrix} 1 & 3 & 3 & 1 \\ 1 & 1 & 3 & 3 \end{bmatrix}$$

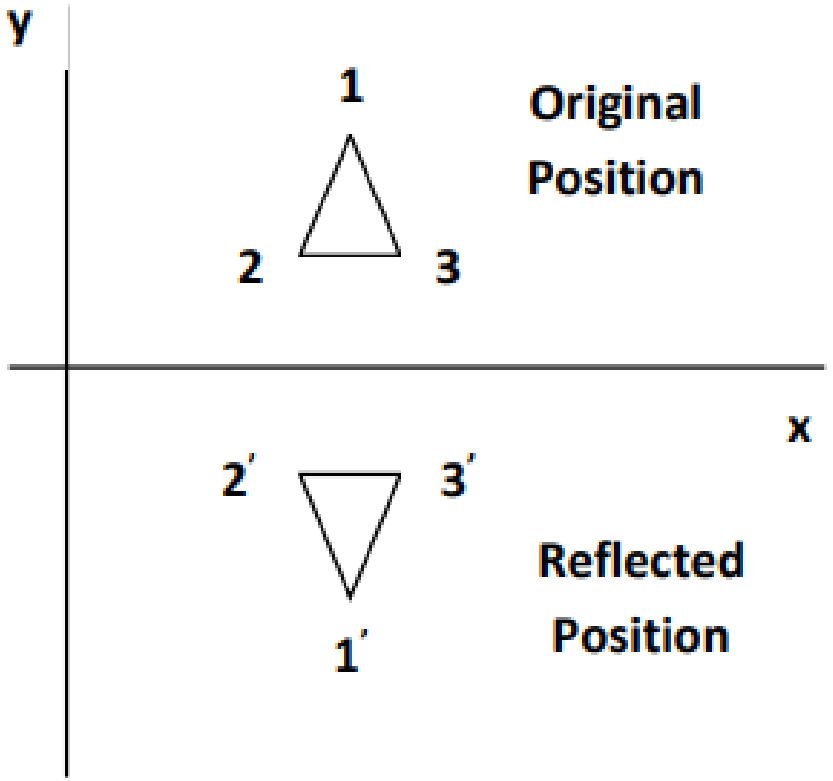
- ✓ Final coordinate after scaling are $[A'(1,1), B'(3,1), C'(3,3), D'(1,3)]$.

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 gives image based on position of axis of reflection.
- Transformation matrix for few positions are :

Reflection Cont....

- ***Transformation matrix for reflection about the line $y = 0$, the x-axis.***



- This transformation keeps x values are same, but flips (Change the sign) y values of coordinate positions.

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

Figure: Reflection about x-axis.

Reflection Cont....

- ***Transformation matrix for reflection about the line $x = 0$, the y-axis.***

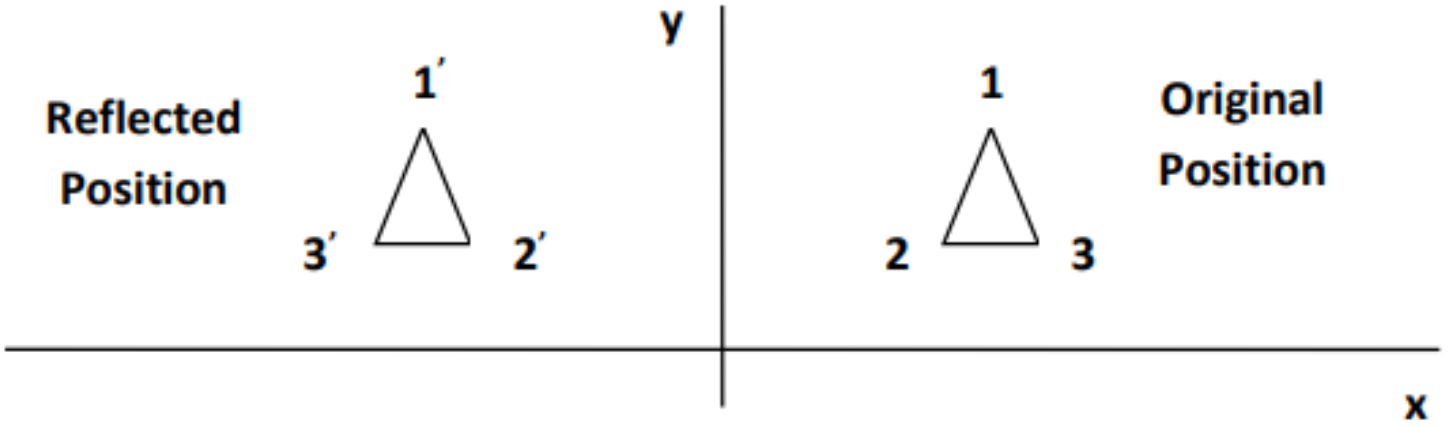


Figure: Reflection about y-axis.

- This transformation keeps y values are same, but flips (Change the sign) x values of coordinate positions.

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

Reflection Cont....

- ***Transformation matrix for reflection about the line x = y***

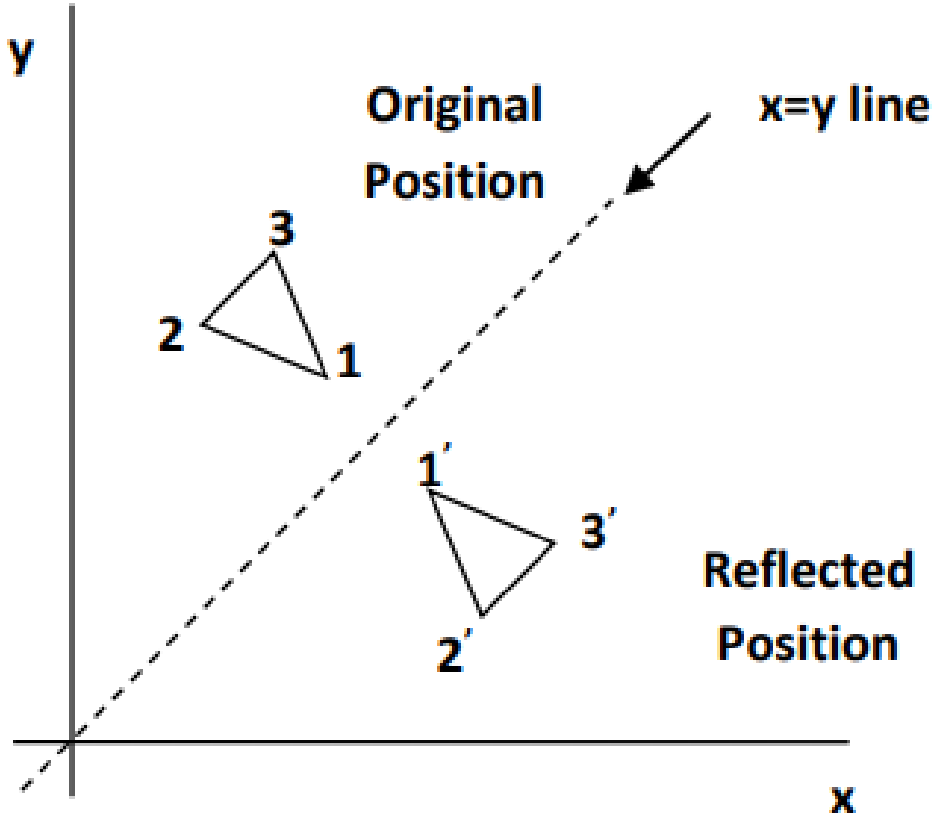


Figure: Reflection about x=y line

- This transformation interchange x and y values of coordinate positions

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

Reflection Cont....

- ***Transformation matrix for reflection about the line x = -y***

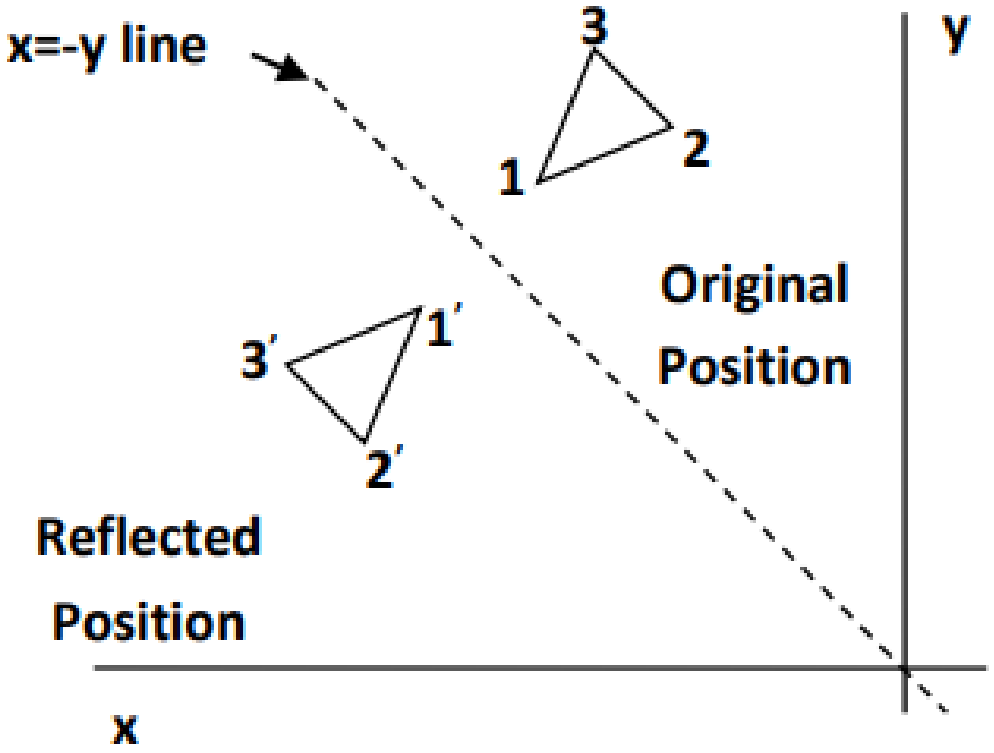


Figure: Reflection about $x = -y$ line

- This transformation interchange x and y values of coordinate positions

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

Reflection Cont....

Example:- Find the coordinates after reflection of the triangle $[A(10,10),\ B(15,15),\ C(20,10)]$ about x axis.

$$P' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 & 15 & 20 \\ 10 & 15 & 10 \\ 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 10 & 15 & 20 \\ -10 & -15 & -10 \\ 1 & 1 & 1 \end{bmatrix}$$

- Final coordinate after reflection are $[A'(10,-10),\ B'(15,-15),\ C'(20,-10)]$

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 coordinate x values and those that shift y values.

Shear Cont....

Shear in x-direction

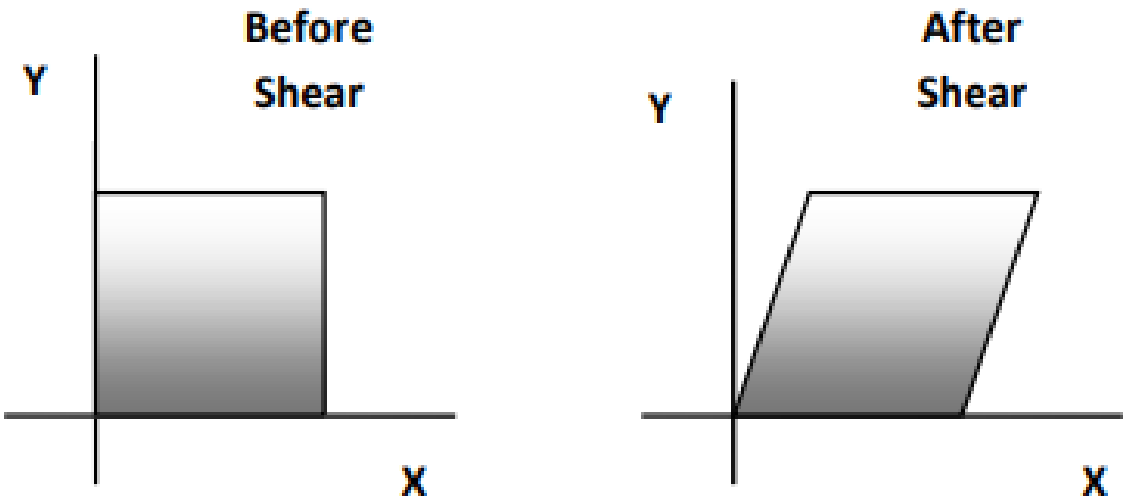


Figure: Shear in x-direction

- Shear relative to $x-axis$ that is $y=0$ line can be produced by following equation:

$$x' = x + sh_x \cdot y\,, \qquad y' = y$$

Shear Cont....

Shear in x-direction

- Transformation matrix for that is: $\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

where sh_x is shear parameter.

- We can generate $x - direction$ shear relative to other reference line $y = y_{ref}$ with following equation:

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

- Transformation matrix for that is:

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y \cdot x_{ref} \\ 0 & 0 & 1 \end{bmatrix}$$

Shear Cont....

Shear in x-direction

Example:- Shear the unit square in x direction with shear parameter $\frac{1}{2}$ relative to line $y = -1$.

- Here $y_{ref} = -1$ and $sh_x = 0.5$
- Coordinates of unit square are [A (0, 0), B (1, 0), C (1, 1), D (0, 1)].

$$P' = \begin{bmatrix} 1 & sh_x & -sh_x \cdot y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 1 & 0.5 & -0.5 \cdot (-1) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

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

$$P' = \begin{bmatrix} 0.5 & 1.5 & 2 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

- Final coordinate after shear are $[A'(0.5, 0),\ B'(1.5, 0),\ C'(2, 1),\ D'(1, 1)]$

Shear Cont....

Shear in y-direction

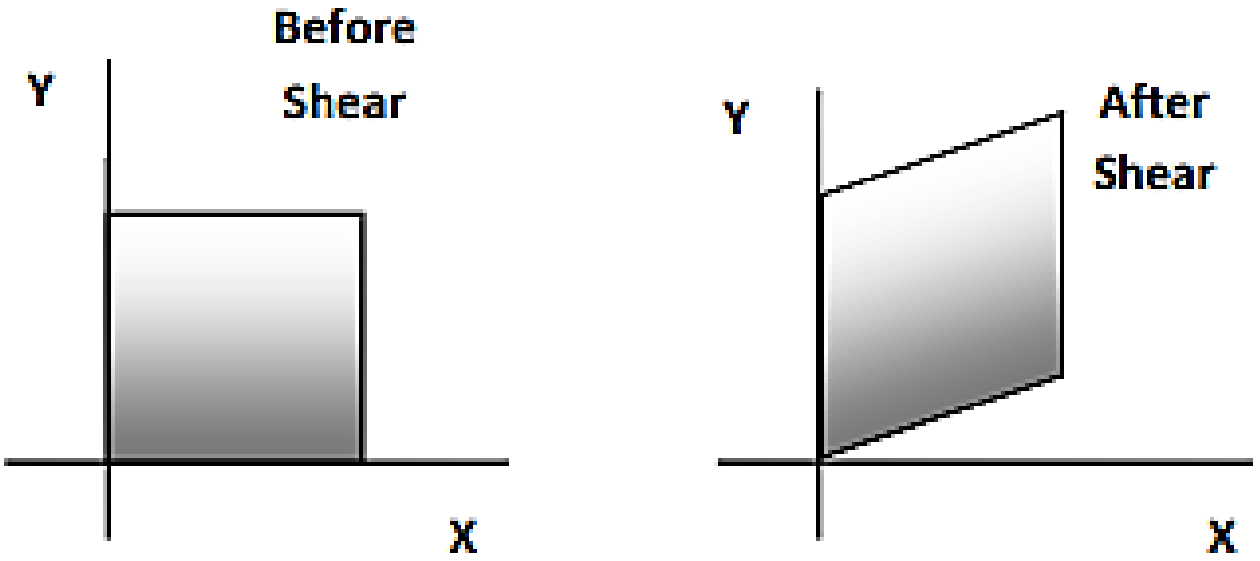


Figure: Shear in y-direction

- Shear relative to $y - axis$ that is $x = 0$ line can be produced by following equation:

$$x' = x, \qquad y' = y + sh_y \cdot x$$

Shear Cont....

Shear in y-direction

- Transformation matrix for that is: $\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

where sh_y is shear parameter.

- We can generate $y - direction$ shear relative to other reference line $x = x_{ref}$ with following equation:

$$x' = x, \qquad y' = y + sh_y \cdot (x - x_{ref})$$

- Transformation matrix for that is:

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y \cdot x_{ref} \\ 0 & 0 & 1 \end{bmatrix}$$

Shear Cont....

Shear in y-direction

Example:- Shear the unit square in y direction with shear parameter $\frac{1}{2}$ relative to line $x = -1$.

- Here $x_{ref} = -1$ and $sh_y = 0.5$
- Coordinates of unit square are [A (0, 0), B (1, 0), C (1, 1), D (0, 1)].

$$P' = \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y \cdot x_{ref} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & -0.5 \cdot (-1) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

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

$$P' = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0.5 & 1 & 2 & 1.5 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

- Final coordinate after shear are $[A'(0, 0.5),\ B'(1, 1),\ C'(1, 2),\ D'(0, 1.5)]$

Composite Transformation

- We can set up a matrix for any sequence of transformations as a *composite transformation matrix* by calculating the matrix product of individual transformation.
- For column matrix representation of coordinate positions, we form composite transformations by multiplying matrices in order from right to left.

Composite Transformation Cont....

Translation

- Two successive translations are performed as:

$$P' = T(t_{x2}, t_{y2}) \cdot \{T(t_{x1}, t_{y1}) \cdot P\}$$

$$P' = \{T(t_{x2}, t_{y2}) \cdot T(t_{x1}, t_{y1})\} \cdot P$$

$$P' = \begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

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

$$P' = T(t_{x1} + t_{x2}, t_{y1} + t_{y2}) \cdot P\}$$

- Here P' and P are column vector of final and initial point coordinate respectively.
- This concept can be extended for any number of successive translations.

Composite Transformation Cont....

Translation

Example: Obtain the final coordinates after two translations on point (2,3) with translation vector (4, 3) and (−1, 2) respectively.

$$P' = T(t_{x1} + t_{x2}, t_{y1} + t_{y2}) \cdot P$$

$$P' = \begin{bmatrix} 1 & 0 & t_{x1} + t_{x2} \\ 0 & 1 & t_{y1} + t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \cdot P = \begin{bmatrix} 1 & 0 & 4 + (-1) \\ 0 & 1 & 3 + 2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \\ 1 \end{bmatrix}$$

✓ Final Coordinates after translations are $p'(5, 8)$.

Composite Transformation Cont....

Rotations

- Two successive Rotations are performed as:

$$P' = R(\theta_2) \cdot \{R(\theta_1) \cdot P\}$$

$$P' = \{R(\theta_2) \cdot R(\theta_1)\} \cdot P$$

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

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

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

$$P' = R(\theta_1 + \theta_2) \cdot P$$

- Here P' and P are column vector of final and initial point coordinate respectively.
- This concept can be extended for any number of successive rotations.

Composite Transformation Cont....

Rotations

- Obtain the final coordinates after two rotations on point (6,9) with rotation angles are 30^o and 60^o respectively.

$$P' = R(\theta_1 + \theta_2) \cdot P$$

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

$$P' = \begin{bmatrix} cos(30 + 60) & -sin(30 + 60) & 0 \\ sin(30 + 60) & cos(30 + 60) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

$$P' = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 6 \\ 9 \\ 1 \end{bmatrix} = \begin{bmatrix} -9 \\ 6 \\ 1 \end{bmatrix}$$

- Final Coordinates after rotations are $p'(-9, 6)$.

Composite Transformation Cont....

Scaling

- Two successive scaling are performed as:

$$P' = S(s_{x2}, s_{y2}) \cdot \{S(s_{x1}, s_{y1}) \cdot P\}$$

$$P' = \{S(s_{x2}, s_{y2}) \cdot S(s_{x1}, s_{y1})\} \cdot P$$

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

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

$$P' = S(s_{x1} \cdot s_{x2}, s_{y1} \cdot s_{y2}) \cdot P$$

- Here P' and P are column vector of final and initial point coordinate respectively.
- This concept can be extended for any number of successive scaling.

Transformation Between Coordinate Systems

- Graphics applications often requires the transformation of object descriptions from one coordinate system to another.
- This is done in two steps.
- Let Cartesian systems with the coordinate origins at (0,0) and (x_0, y_0) and with the orientation angle θ between the x and x' axes.
- Translation from object descriptions from xy coordinates to $x'y'$ coordinates can be done as:
  - Translate so that the origin (x_0, y_0) of the $x'y'$ system is moved to the origin of the xy system.
  - Rotate the x' axis to the x axis.
- Thus,

$$M_{xy,x'y'} = R(-\theta).T(-x_0, -y_0)$$

Transformation Between Coordinate Systems Cont....

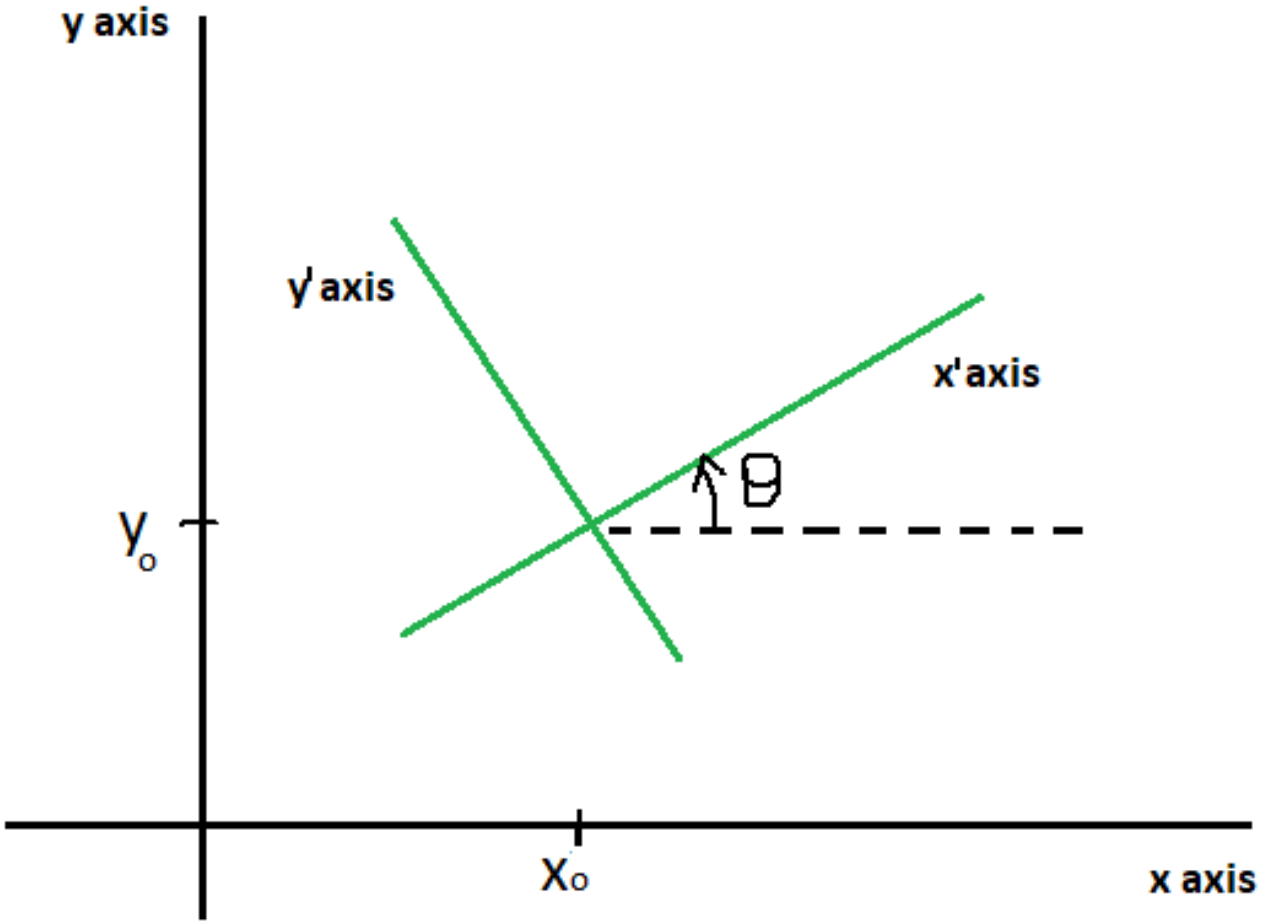


Figure: *A Cartesian $x'y'$ system positioned at (x_0, y_0) with orientation θ in an xy Cartesian system.*

2-D Viewing

- ***Window:*** Area selected in world-coordinate for display is called window. It defines what is to be viewed.
- ***Viewport:*** Area on a display device in which window image is display (mapped) is called viewport. It defines where to display.
- ***Viewing Transformation:*** Finding device coordinates of viewport from world coordinates of window.

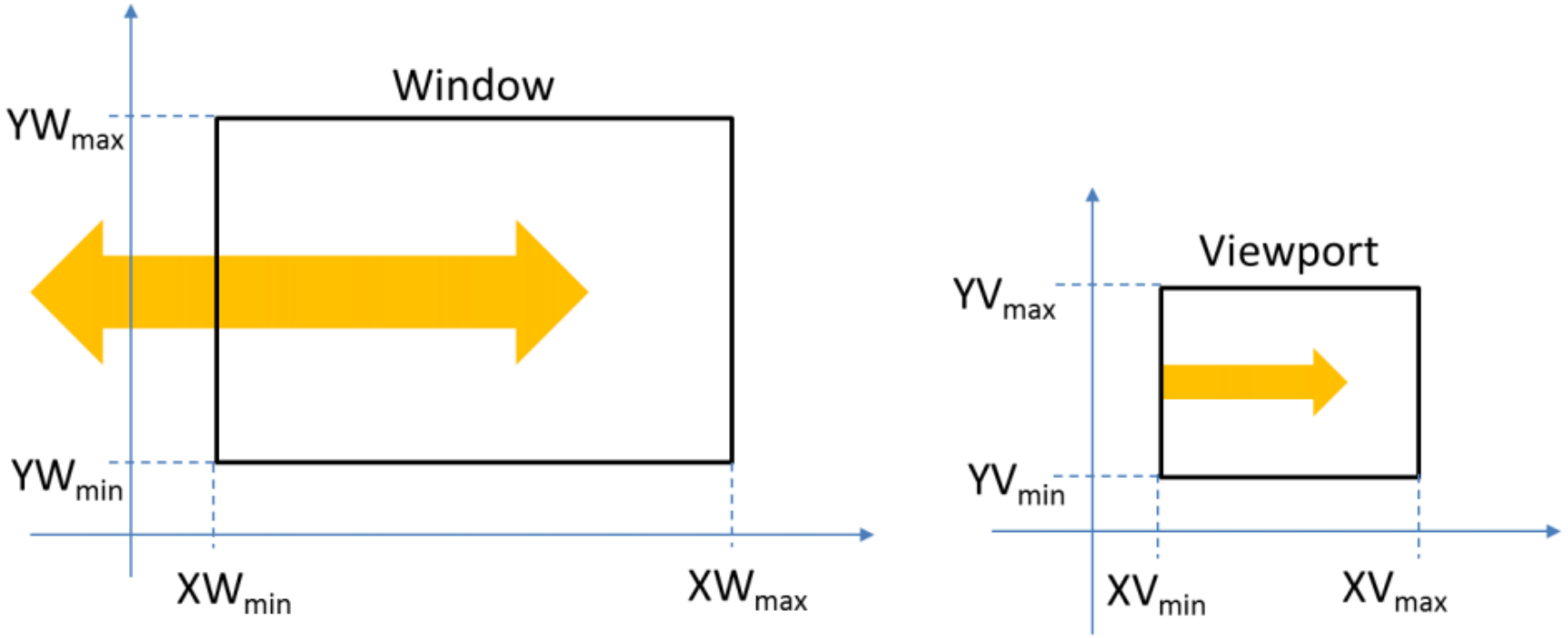


Figure: *A viewing transformation using standard rectangles for the window and viewport*

Viewing Pipeline

- The term Viewing Pipeline describes a series of transformations.
- Steps involves in viewing pipeline are shown in figure below.

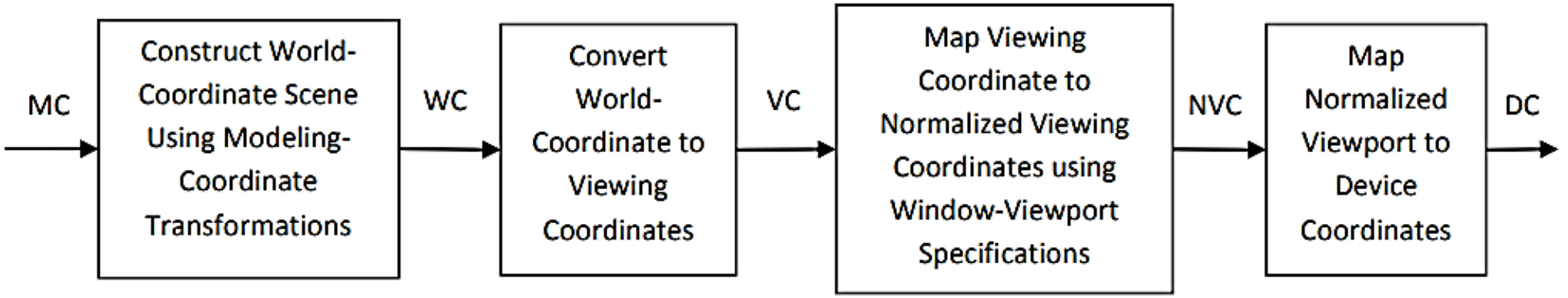


Figure: 2D viewing pipeline.

- As shown in figure above first of all we construct world coordinate scene using modeling coordinate transformation.
- After this we convert viewing coordinates from world coordinates using window to viewport transformation.

Viewing Pipeline Cont....

- Then we map viewing coordinate to normalized viewing coordinate in which we obtain values in between 0 to 1.
- At last we convert normalized viewing coordinate to device coordinate using device driver software which provide device specification.
- Finally device coordinate is used to display image on display screen.
- By changing the viewport position on screen we can see image at different place on the screen.
- By changing the size of the window and viewport we can obtain zoom in and zoom out effect as per requirement.
- Fixed size viewport and small size window gives zoom in effect, and fixed size viewport and larger window gives zoom out effect.
- View ports are generally defines with the unit square so that graphics package are more device independent which we call as normalized viewing coordinate.

Viewing Coordinate Reference Frame

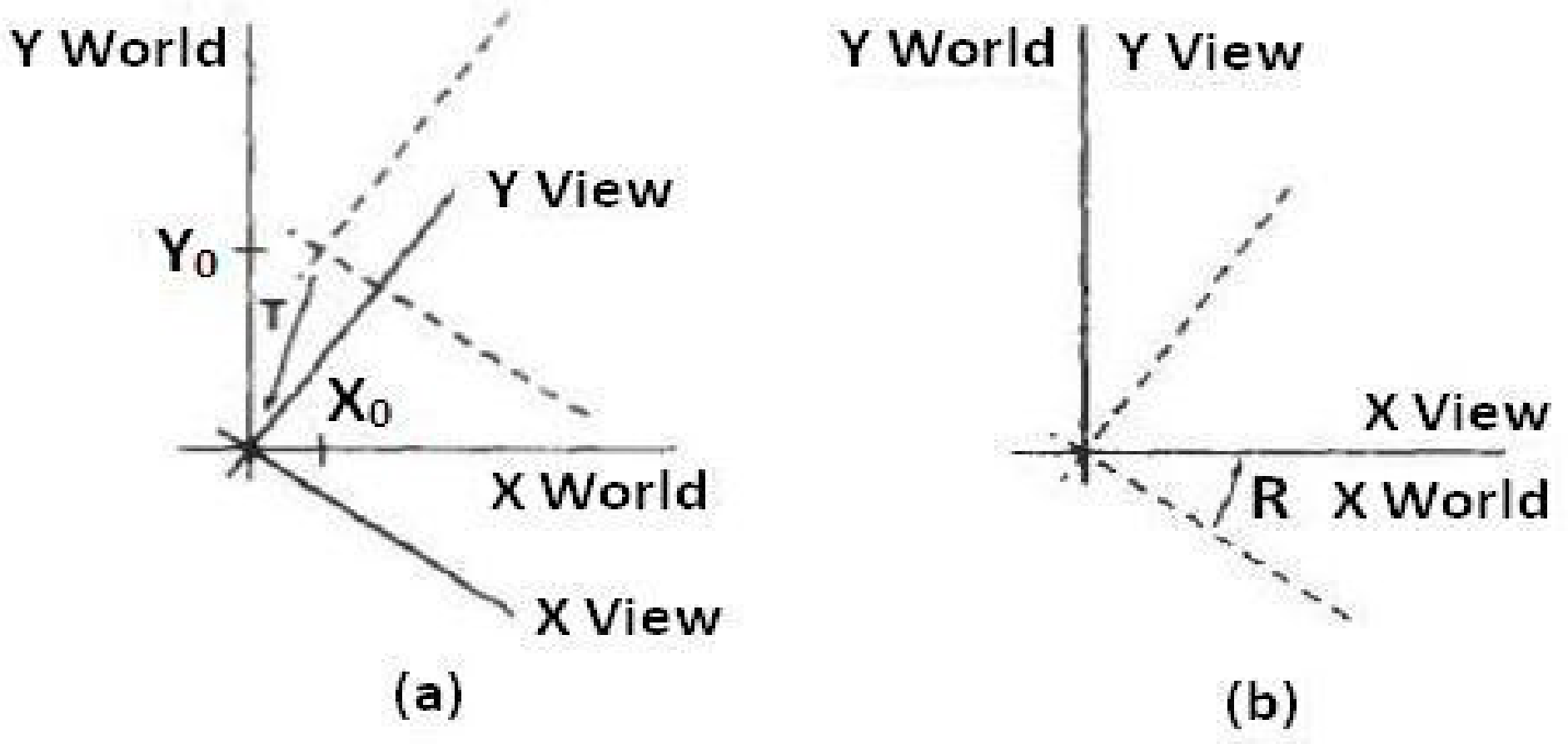


Figure: A viewing-coordinate frame is moved into coincidence with the world frame in two steps: (a) translate the viewing origin to the world origin, and then (b) rotate to align the axes of the two systems.

Viewing Coordinate Reference Frame Cont....

- We can obtain reference frame in any direction and at any position.
- For handling such condition first of all we translate reference frame origin to standard reference frame origin and then we rotate it to align it to standard axis.
- In this way we can adjust window in any reference frame.
- This is illustrate by following transformation matrix:

$$M_{wc,vc} = R.T$$

- Where T is translation matrix and R is rotation matrix.

Window to Viewport Coordinate Transformation

- Mapping of window coordinate to viewport is called window to viewport transformation.
- We do this using transformation that maintains relative position of window coordinate into viewport.
- That means center coordinates in window must be remains at center position in viewport.
- We find relative position by equation as follow:

$$\frac{x_v - x_{vmin}}{x_{vmax} - x_{vmin}} = \frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}}$$

$$\frac{y_v - y_{vmin}}{y_{vmax} - y_{vmin}} = \frac{y_w - y_{wmin}}{y_{wmax} - y_{wmin}}$$

Window to Viewport Coordinate Transformation Cont....

- Solving by making viewport position as subject we obtain:

$$x_v = x_{vmin} + (x_w - x_{wmin})s_x$$
$$y_v = y_{vmin} + (y_w - y_{wmin})s_y$$

- Where

$$s_x = \frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}$$
$$s_y = \frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}}$$

- We ca viewport with the set of transfo e following sequence of transformations:
  1. Perform a scaling transformation using a fixed-point position of (xWmin,ywmin) that scales the window area to the size of the viewport.
  2. Translate the scaled window area to the position of the viewport.

Clipping Operations

- 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*.
- Clip window can be general polygon or it can be curved boundary.

Application of Clipping

- It can be used for displaying particular part of the picture on display screen.
- Identifying visible surface in 3D views.
- Antialiasing.
- Creating objects using solid-modeling procedures.
- Displaying multiple windows on same screen.
- Drawing and painting.

Point Clipping

- In point clipping we eliminate those points which are outside the clipping window and draw points which are inside the clipping window.
- Here we consider clipping window is rectangular boundary with edge (Xwmin, Xwmax, Ywmin, Ywmax).
- So for finding whether given point is inside or outside the clipping window we use following inequality:

  $Xwmin \leq x \leq Xwamx$

  $Ywmin \leq y \leq Ywamx$

- If above both inequality is satisfied then the point is inside otherwise the point is outside the clipping window.

Line Clipping

- Line clipping involves several possible cases.
  1. Completely inside the clipping window.
  2. Completely outside the clipping window.
  3. Partially inside and partially outside the clipping window.

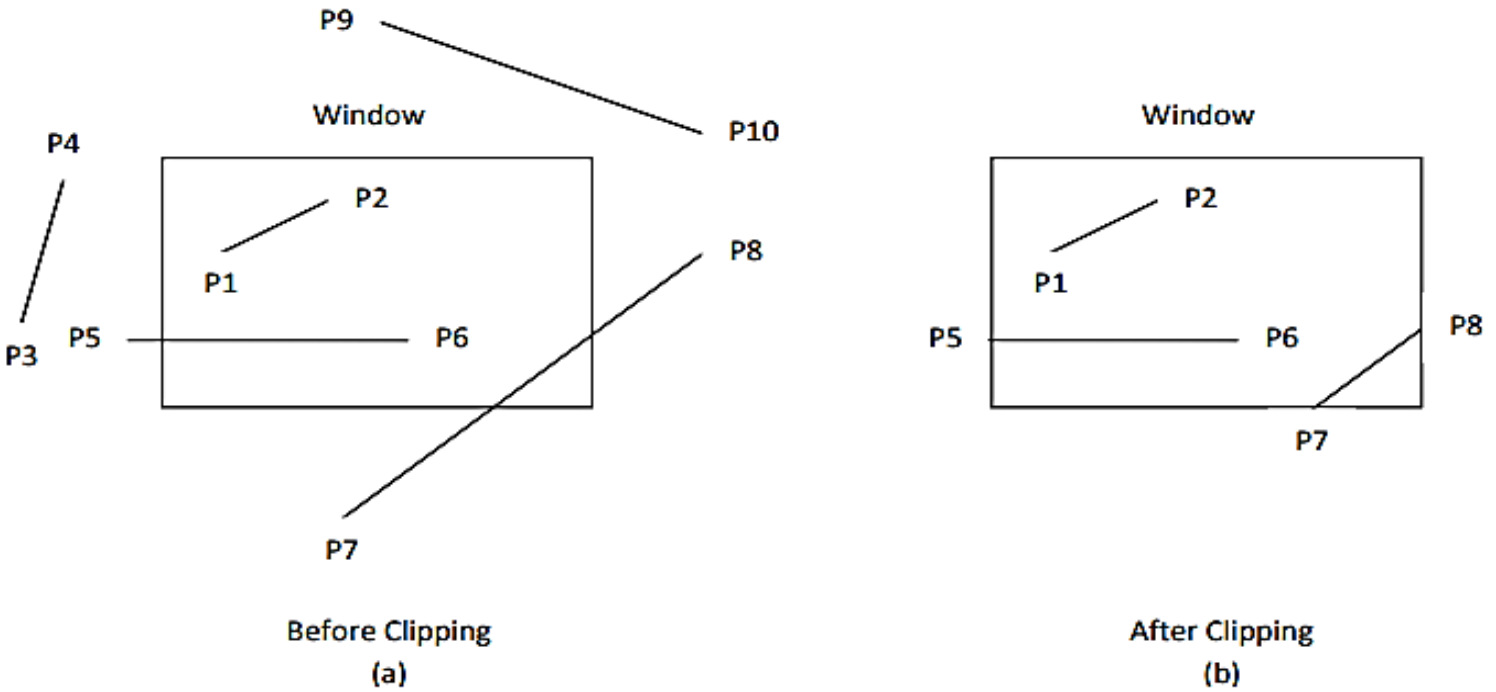


Fig. - Line clipping against a rectangular window.

Line Clipping Cont....

- Line which is completely inside is display completely. Line which is completely outside is eliminated from display. And for partially inside line we need to calculate intersection with window boundary and find which part is inside the clipping boundary and which part is eliminated.
- For line clipping several scientists tried different methods to solve this clipping procedure. Some of them are discuss below.

Cohen-Sutherland Line Clipping

- This is one of the oldest and most popular line-clipping procedures.

Region and Region Code

- In this we divide whole space into nine region and assign 4 bit code to each endpoint of line depending on the position where the line endpoint is located.

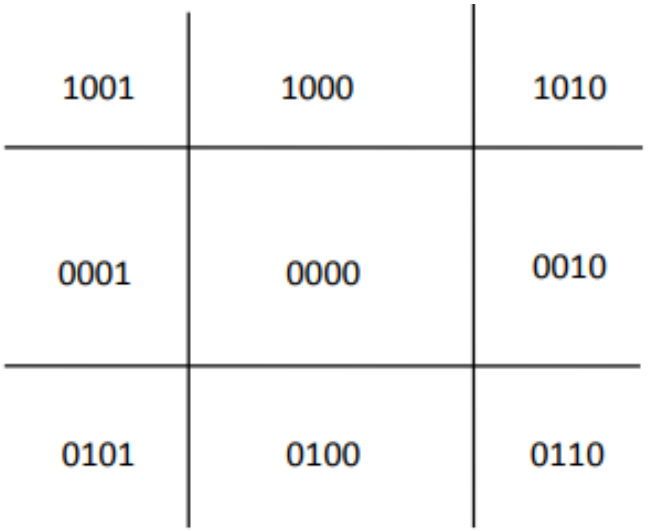


Fig: - Workstation transformation.

- Figure above shows code for line end point which is fall within particular area.
- Code is deriving by setting particular bit according to position of area.
  - Set bit 1: For left side of clipping window.
  - Set bit 2: For right side of clipping window.
  - Set bit 3: For below clipping window.
  - Set bit 4: For above clipping window.
- All bits as mention above are set means 1 and other are 0.

Algorithm

Step-1:

- Assign region code to both endpoint of a line depending on the position where the line endpoint is located.

Step-2:

- If both endpoint have code ‘0000’
  - Then line is completely inside.
- Otherwise
  - Perform logical ending between this two codes.
  - If result of logical ending is non-zero
    - Line is completely outside the clipping window.
  - Otherwise
    - Calculate the intersection point with the boundary one by one.
    - Divide the line into two parts from intersection point.
    - Recursively call algorithm for both line segments.