Chapter 3
Two-Dimensional Geometric Transformations and Viewing

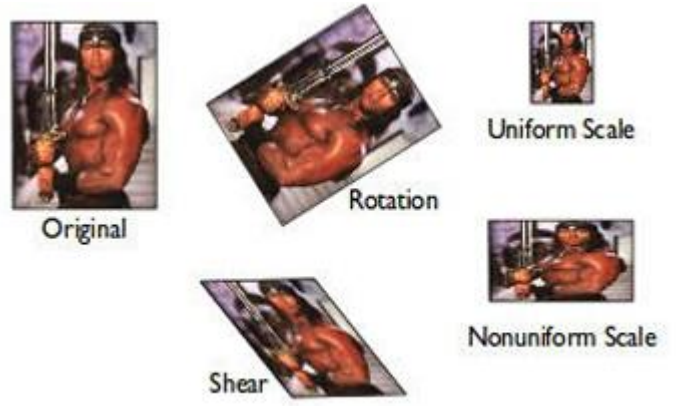
Credit hours: 8 hrs

Contents

3.1 Two Dimensional Transformations

3.1 Two Dimensional Transformations

- Geometric transformation is **the change in orientation, shape and size of graphic objects which alter the coordinate description of objects.**
- Transformation means a change in the object's orientation, size, and shape.
- They position the object, change its shape, and even change how something is viewed.
- Transformation plays a major role in computer graphics, repositioning the graphics on the screen and changing their size or orientation.
- The basic geometrical 2D Transformation in Computer Graphics are:
  - **Translation**
  - **Rotation**
  - **Scaling**
- The derived geometrical transformation is:
  - **Reflection**
  - **Shearing**



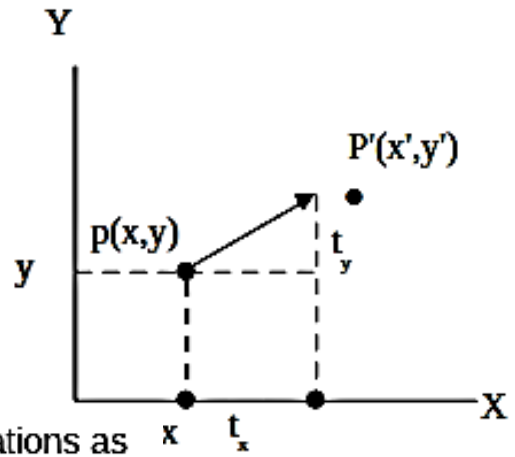
a) Translation

Repositioning of object along a straight-line path from one coordinate location to another is called translation. 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') as:

$$x' = x + t_x$$

$$y' = x + t_y$$

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

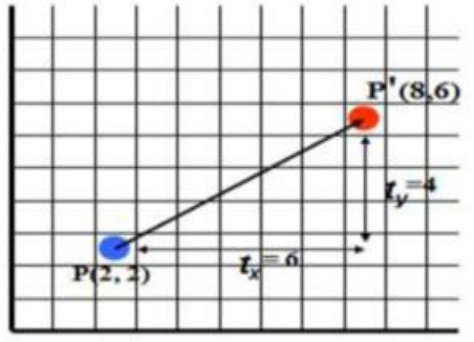


The translatio $y + t_y$ pair (t_x,t_y) is known as translation vector or shift vector. We can express translation equations as matrix representations as

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

For Example:

Given a point P with original position (2,2). Then after performing translation operation with $t_x = 6$ and $t_y = 4$, we get new transformed coordinate P' with coordinate (8,6).



1.

Q. Translate the given points (2,5) by the translating value (3,3).

Solution:

Given Point P (2, 5) and translation distance Tx= 3, and Ty= 3
We have From Translation Matrix

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

i.e.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$$

Therefore, P(2,5) is translated to new point P'(5,8).

2. **Q. Translate the given square having coordinate A (0,0), B (3,0), and C (3,3) and D (0, 3) by the translating value 2 in both directions.**

Solution:

Given points A (0,0), B (3,0), and C (3,3) and D (0, 3) and translating value 2 ($t_x=t_y=2$)

We have From Translation Matrix

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

i.e.

$$A' = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$C' = \begin{bmatrix} 3 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

$$D' = \begin{bmatrix} 0 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

Hence the final co-ordinate are A'(2,2)

Hence the final co-ordinate are A'(2,2), B'(5,2), C'(5,5) and D'(2,5)

3. Given a circle C with radius 10 and center coordinates (1, 4). Apply the translation with distance 5 towards X axis and 1 towards Y axis. Obtain the new coordinates of C without changing its radius.

Given-

- Old center coordinates of C = $(X_{old}, Y_{old}) = (1, 4)$
- Translation vector = $(T_x, T_y) = (5, 1)$

Let the new center coordinates of C = (X_{new}, Y_{new}).

Applying the translation equations, we have-

- $X_{new} = X_{old} + T_x = 1 + 5 = 6$
- $Y_{new} = Y_{old} + T_y = 4 + 1 = 5$

Thus, New center coordinates of C = (6, 5).

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

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

we can express the transformed coordinates in terms of 'Φ' and 'θ' as

$x' = r\cos(\Phi + \theta) = r\cos\Phi \cdot \cos\theta - r\sin\Phi \cdot \sin\theta$ ....(i)

$y' = r\sin(\Phi + \theta) = r\cos\Phi \cdot \sin\theta + r\sin\Phi \cdot \cos\theta$ ....(ii)

We know that original coordinates of point in polar coordinates are

$x = r\cos\Phi$

$y = r\sin\Phi$

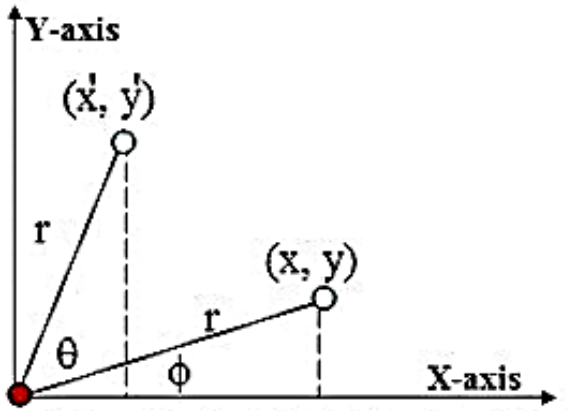


Fig. Rotating a point from position (x, y) to position (x', y') through an angle θ about rotation point (0, 0). The original angular displacement of the point from the x axis is ϕ.

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 . 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

- Translate the point (x_r, y_r) and P(x,y) by translation vector $(-x_r, -y_r)$ which translates the pivot to origin and P(x,y) to $(x-x_r, y-y_r)$.
- Now apply the rotation equations when pivot is at origin to rotate the translated point $(x-x_r, y-y_r)$ as:

$$x_1 = (x - x_r)\cos\theta - (y - y_r)\sin\theta$$

$$y_1 = (x - x_r)\sin\theta + (y - y_r)\cos\theta$$

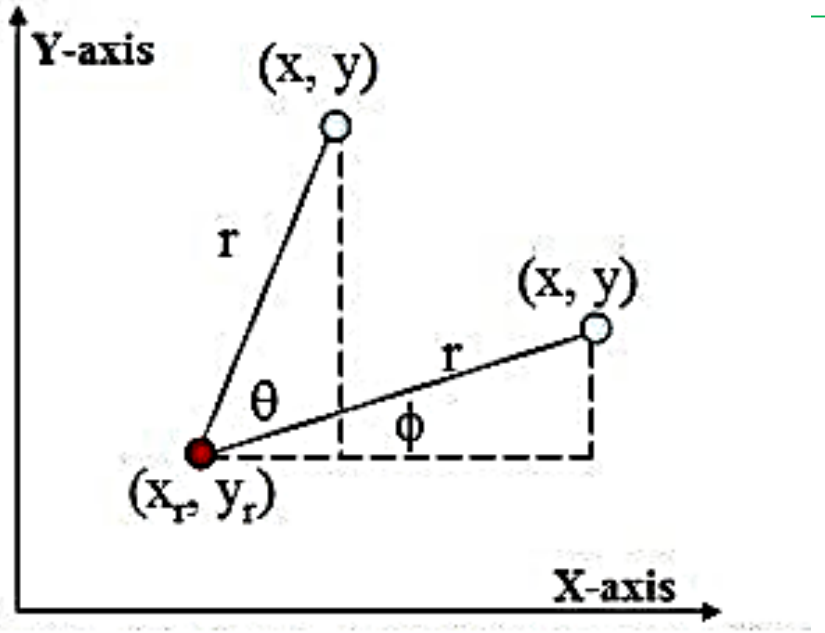
- Re- translate the rotated point (x_1, y_1) with translation vector (x_r, y_r) which is reverse translation to original translation.

Finally we get the equation after successive transformation as

$$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta \quad \ldots\ldots\ldots\ldots 1$$

$$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta \quad \ldots\ldots\ldots\ldots 2$$

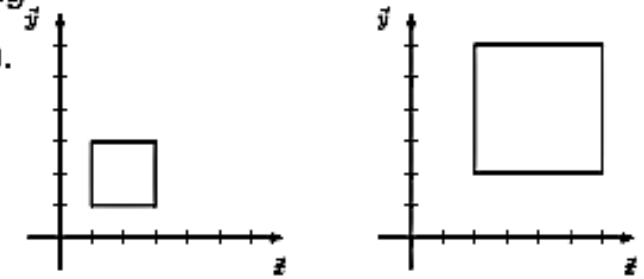
Which are actually the equations for rotation of (x,y) from the pivot point (x_r, y_r)



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{pmatrix} \text{Cos}\theta & -\text{Sin}\theta \\ \text{Sin}\theta & \text{Cos}\theta \end{pmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{pmatrix} 1-\text{Cos}\theta & -\text{Sin}\theta \\ -\text{Sin}\theta & 1-\text{Cos}\theta \end{pmatrix} \begin{bmatrix} x_r \\ y_r \end{bmatrix}$$

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
- $s_x = s_y = 1$ leaves the size of the object unchanged
- When s_x , s_y are assigned the same value $s_x = s_y = 4$ then a Uniform Scaling is produced.

- Scaling **About arbitrary point**

If p(x,y) be scaled to a point p'(x', y') by s_x time in X-units and s_y times in y-units about arbitrary point (x_f, y_f) then the equation for scaling is given as

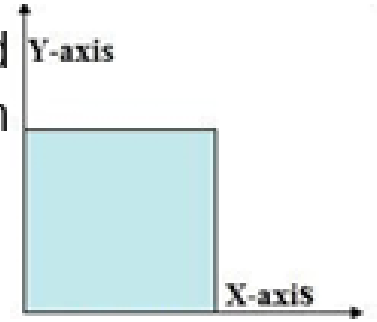
$$x' = x_f + sx.(x-x_f)$$

$$y' = y_f + sy.(y-y_f)$$

d) Shearing

- 2D Shearing transformation slants or distorts an object or coordinate system along either the x-axis or y-axis in a 2D plane. It involves shifting the position of points in a specific direction based on their original coordinates.

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.



X-direction Shear:

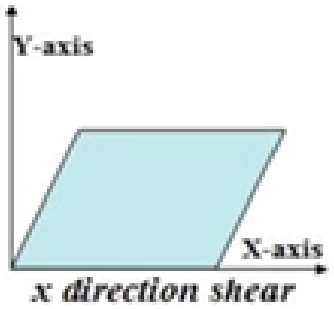
An X-direction shear relative to x-axis is produced with transformation matrix equation.

In 'x' direction,

$x' = x + s_{hx} \cdot y$

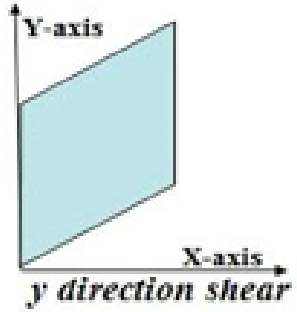
$y' = y$

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



Y-direction Shear:

An y-direction shear relative to y-axis is produced by following transformation equations.



In 'y' direction,

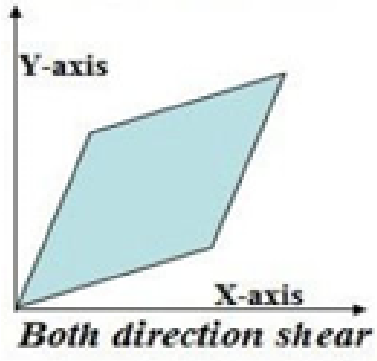
$x' = x$

$y' = y + s_{hy} \cdot x$

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

Both direction Shear:

Both direction share relative to x-axis and y-axis is produced by following transformation equations



In both directions,

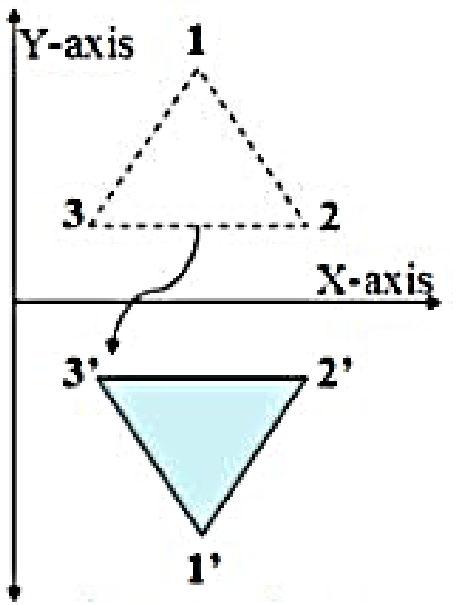
$x' = x + s_{hx} \cdot y$

$y' = y + s_{hy} \cdot x$

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

e) Reflection

A reflection is a transformation that produce a mirror image of an object. In 2D-transformation, reflection is generated relative to an axis of reflection. The reflection of an object to an relative axis of reflection , is same as 180° rotation about the reflection axis.



i. **Reflection about x axis or about line y = 0**

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

$$x' = x$$
$$y' = -y$$

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

Q. What is Arbitrary axis?

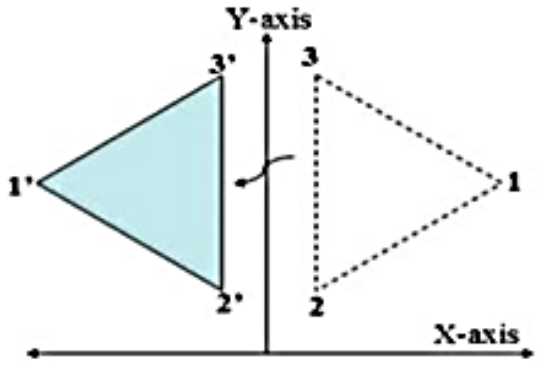
Arbitrary axis simply means the axis chosen in any way we like within the co-ordinate system. We choose any line in space, and then put a pin into our object along that line, and transform the object around the pin

ii. Reflection about y axis or about line x = 0

- Keeps ‘y’ value same but flips ‘x’ value of coordinate points

$$x' = -x$$
$$y' = y$$

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

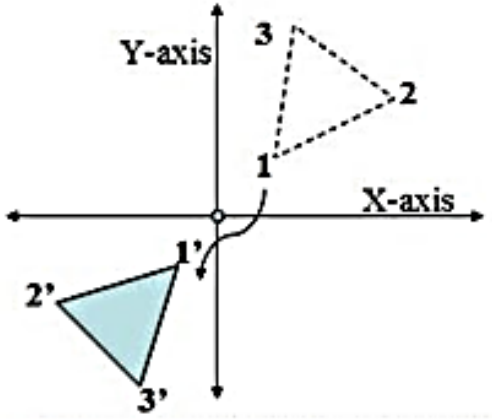


iii. Reflection about origin

Flip both ‘x’ and ‘y’ coordinates of a point

$$x' = -x$$
$$y' = -y$$

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

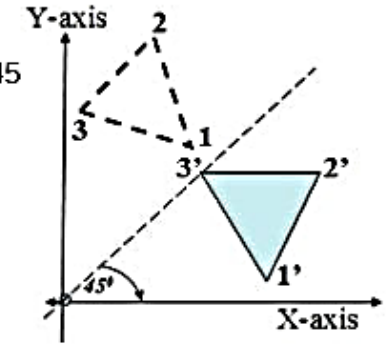


iv. **Reflection on an arbitrary axis**

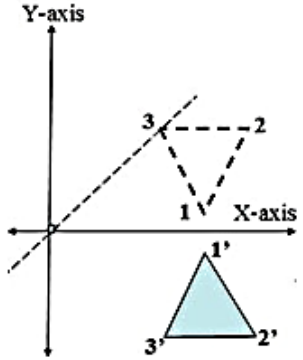
The reflection on any arbitrary axis of reflection can be achieved by sequence of rotation and co-ordinate axes reflection matrices.

Reflection about line y = x ($\theta=45^0$)

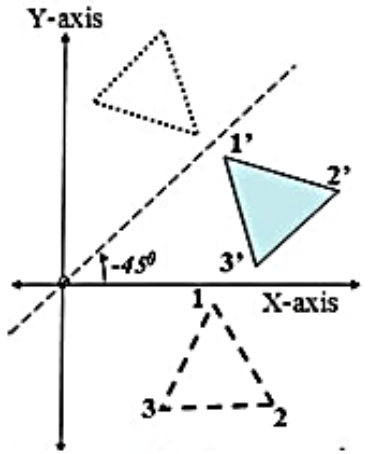
Step 1: - Rotate about origin in clockwise direction by 45 degree which rotates line y = x to x-axis



Step 2: - Take reflection against x-axis



Step 3: - Rotate in anti-clockwise direction by same angle



Here, we have multiple transformation at once, So when more than one transformations are applied for performing a task then such transformation is called **composite transformation**.

So the composite transformation required for reflecting the given object about y=x axis is

$$T = R_{\theta=45} R_x R_{\theta=-45}$$

Forming products of transformation matrices is often referred to as a concatenation, or composition, of matrices.

Q. What is the basic purpose of composite transformation?

The basic purpose of composing transformations is to gain efficiency by applying a single composed transformation to a point, rather than applying a series of transformation, one after another.

3.2 Matrix Representations of Transformations

3.2 Matrix Representations of Transformations

3.2 Matrix Representations of Transformations

- Already included in section 3.1

3.3 Homogeneous Coordinates

3.3 Homogeneous Coordinates

Homogeneous Coordinates in 2D transformation

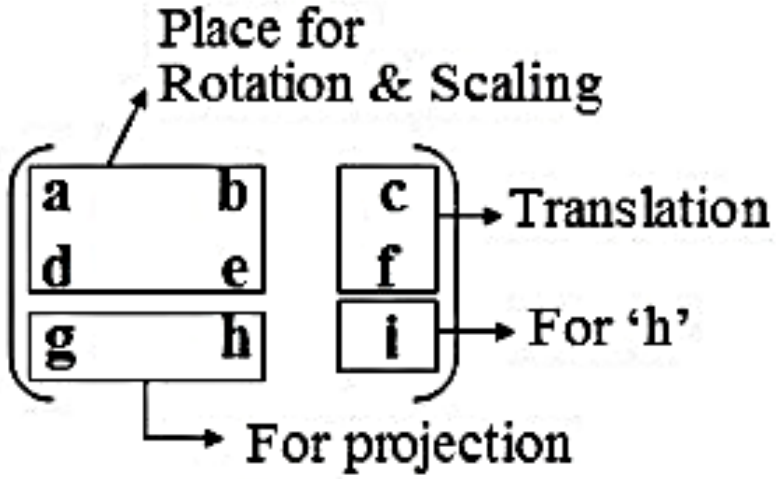
- In 2D computer graphics, homogeneous coordinates are a mathematical tool used to represent points in a projective space.
- The usual Cartesian coordinates (x, y) are extended to homogeneous coordinates by adding an additional coordinate, typically denoted as w.
- A point in homogeneous coordinates is represented as (x, y, w).
- The primary advantage of using homogeneous coordinates in 2D transformations lies in their ability to represent translation, rotation, scaling, and other transformations as matrix operations.
- These transformations can be expressed using 3x3 matrices, where the third row of the matrix is typically (0, 0, 1) to ensure the correct representation of points.
- Homogeneous coordinates simplify the representation and concatenation of these transformations, making it easier to combine multiple transformations into a single matrix and apply them efficiently.

To perform a sequence of transformation such as translation followed by rotation and scaling, we need to follow a sequential process –

- Translate the coordinates,
- Rotate the translated coordinates, and then
- Scale the rotated coordinates to complete the composite transformation.

To shorten this process, we have to use 3×3 transformation matrix instead of 2×2 transformation matrix. To convert a 2×2 matrix to 3×3 matrix, we have to add an extra **dummy coordinate**. So, we can represent the point by 3 numbers instead of 2 numbers, which is called Homogenous Coordinate system.

In this system, we can represent all the transformation equations in matrix multiplication.
Any Cartesian point P(X, Y) can be converted to homogenous coordinates by P' (X_h, Y_h, 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.

1 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 $-t_x, -t_y$.

2. For rotation

a. Counter Clock Wise (CCW):

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

b. Clock Wise(CCW):

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

3. For scaling

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

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

4. For shearing

For X-direction shear

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

For Y-direction shear

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

For both-direction shear

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

5. For Reflection

a) Reflection about x-axis

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

b) Reflection about y-axis

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

c) Reflection about y=x-axis

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

d) Reflection about Origin

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

e) Reflection about any line y=mx+c

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

3.4 Composite Transformations

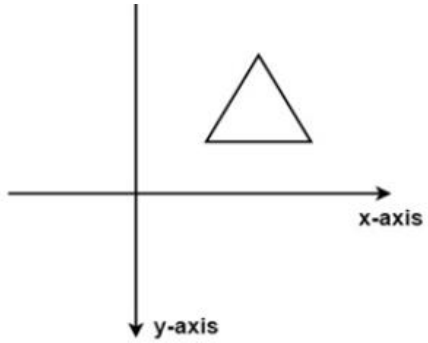
3.4 Composite Transformations

- Composite transformations in 2D graphics refer to the **process of combining multiple individual transformations** (such as translation, rotation, scaling) into a single transformation matrix.
- This matrix can then be applied to a point or an object to achieve the cumulative effect of all the individual transformations.
- Using homogeneous coordinates and matrix multiplication, these transformations can be efficiently concatenated.

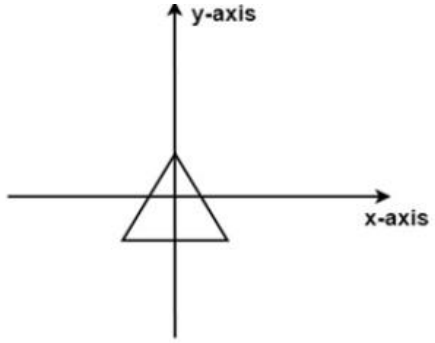
- A number of transformations or sequence of transformations can be combined into single one called as composition. The resulting matrix is called as composite matrix. The process of combining is called as concatenation.
- Suppose we want to perform rotation about an arbitrary point, then we can perform it by the sequence of three transformations
  - Translation
  - Rotation
  - Reverse Translation
- The ordering sequence of these numbers of transformations must not be changed. If a matrix is represented in column form, then the composite transformation is performed by multiplying matrix in order from right to left side. The output obtained from the previous matrix is multiplied with the new coming matrix.

Example (Enlargement wrt. Origin):

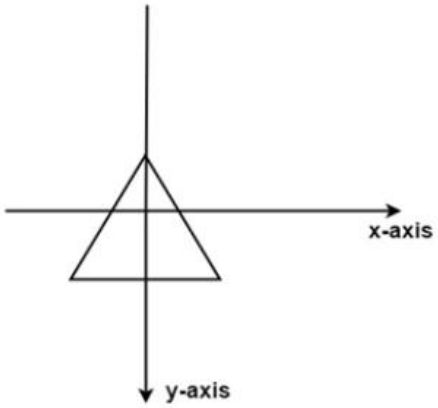
- For this following sequence of transformations will be performed and all will be combined to a single one.
  - **Step1:** The object is kept at its position as in fig (a)
  - **Step2:** The object is translated so that its center coincides with the origin as in fig (b)
  - **Step3:** Scaling of an object by keeping the object at origin is done in fig (c)
  - **Step4:** Again translation is done. This second translation is called a reverse translation. It will position the object at the origin location.
- Above transformation can be represented as $T_V . S.T_V^{-1}$



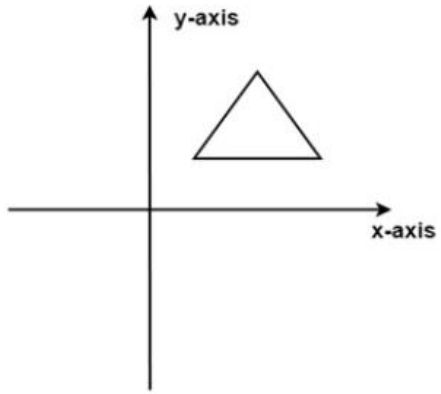
(original position of object)
Fig:(a)



(object is translated to origin)
Fig:(b)



(object is enlarged)
Fig:(c)



(object is retranslated to original position)
Fig:(d)

Two Successive translations are additive.

- **Let T1 (tx1, ty1) and T2 (tx2,ty2) be two translations applied on a point P(x,y) in succession.**
  - First T1 is applied on P as T1 (tx1, ty1). {P}
  - Then T2 is applied on the resultant position
  - **i.e. P'=T2 (tx2, ty2).{T1 (tx1, ty1).{P}}**
- It makes no difference if T2 and T1 are first multiplied and the resultant 3x3 matrix (called composite) is multiplied with P on the right.
- So **we can rearrange** the braces as below (which does not alter the end result) **P'={T2 (tx2, ty2).T1 (tx1, ty1)}.{P}**
- Converting the above notation **into matrix form we have:**

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

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

- **Thus** we can conclude that **two successive translations are additive**.

  P'=T (tx2+tx1, ty2+ty1).{P}

Two Successive rotations are additive.

- **Let R1 (θ1) and R2(θ2) be two rotations applied on a point P(x,y) in succession.**
  - First R1 is applied on P as {R1 (θ1) .P}
  - Then R2 is applied on the resultant position **i.e. P'= R2 (θ2). { R1 (θ1) . P}**
- **This can be written as: P'= { R2 (θ2). R1 (θ1)} . P**
- Converting the above notation **into matrix form we have:**

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta 2 * cos\theta 1 - \sin\theta 2 * sin\theta 1 & -\cos\theta 2 * sin\theta 1 - sin\theta 2 * cos\theta 1 & 0 \\ \sin\theta 2 * cos\theta 1 + cos\theta 2 * sin\theta 1 & -\sin\theta 2 * sin\theta 1 + cos\theta 2 * cos\theta 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

- **Thus** we can conclude that **two successive rotations are additive**(as evident from the composite matrix) which can be written in notation as:

  P'=R (θ2+θ1).{P}

- where R is the composite matrix representing the complex transformation, two successive rotations.

Two Successive scaling are multiplicative.

- **Let S1 (sx1,sy1) and S2 (sx2, sy2) be two scaling applied on a point P(x,y) in succession, with respect to origin.**
  - First S1 is applied on P as S2 (sx1,sy1). {P}
  - Then S2 is applied on the resultant position. **i.e. S2 (sx2, sy2) . {S1 (sx1,sy1)).P}**
- **This can be rewritten as: P'= {S2 (sx2, sy2) . S1 (sx1,sy1))}.{P}**
- Converting the above notation **into matrix form we have:**

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

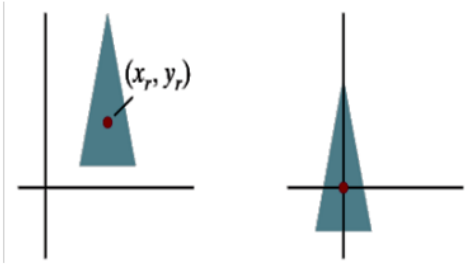
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} Sx_2.Sx_1 & 0 & 0 \\ 0 & Sy_2.Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- **Thus** we can conclude that **two successive scaling operations are multiplicative** (as evident from the composite matrix) which can be written in notation as:

  P'=S (sx2.sx1, sy2.sy1).{P}

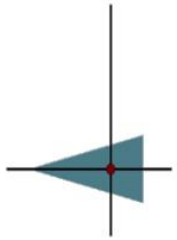
- where S is the composite matrix representing the complex transformation, two successive scaling.

Example-1:General Pivot Point Rotation:

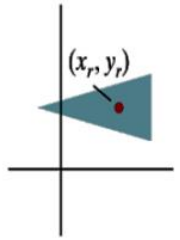


Original position with pivot point (x ,y)

Pivot point translated such that it is at origin.



Rotation with respect to origin performed.



Pivot point translated back to its original position.

- Imagine a randomly chosen **pivot point other than origin, say at (x_r ,y_r)** as shown in the figure above.
- The rotation involves a series a steps as below:
  - **Step-1: Pivot point translated such that it is at origin. T(-x_r ,-y_r)**
  - **Step-2: Rotation with respect to origin performed. R(θ)**
  - **Step-3: Pivot point translated back to its original position. T(x_r , y_r)**
- The above three steps are to performed in that order. When we can write that into notation form (we have to put the steps in the reverse order)

$$P' = \{T(x_r, y_r).R_{(\theta)}.T(-x_r, -y_r)\}.\{P\}$$

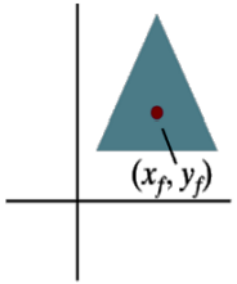
- Converting the above notation to matrix form we have:

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

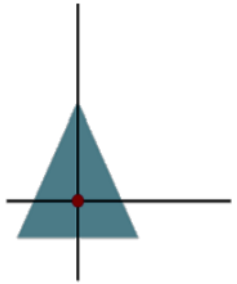
- After finding the composite we have

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r \sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) - x_r \sin\theta \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

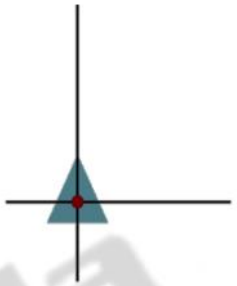
Example-2: General fixed Point Scaling:



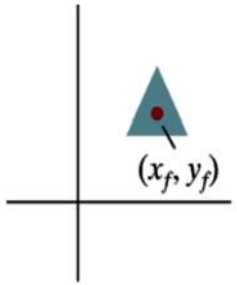
Original position with fixed point (x_f, y_f)



Fixed point translated such that it is at origin.



Scaling with respect to origin performed.



Fixed point translated back to its original position.

- Let's assume the case of scaling with respect to a **fixed point (xf ,yf)** that is any point in space other than origin.
- We can treat this as a complex transformation and break-up into a series of basic transformations and simply multiply them as below:
  - **Step-1: Fixed point translated such that it is at origin. T(-xf ,-yf)**
  - **Step-2: Scaling with respect to origin performed. S(sx,sy)**
  - **Step-3: Fixed point translated back to its original position. T(xf , yf)**
- We can convert that into notation form as:

  P'= {T(xf ,yf). S(sx,sy). T(-xf ,-yf)}. {P}
- Converting the above notation to matrix form we have

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- After finding the composite we have

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

Numerical:

1. Q. Rotate a triangle A(0,0) , B(2,2) , C(4,2) about the origin by the angle of 45 degree.

Solution:
The given triangle ABC can be represented by a matrix, formed from the homogeneous coordinates of the vertices.

$$\begin{bmatrix} 0 & 2 & 4 \\ 0 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

Also, we have

$$R_{45^0} = \begin{bmatrix} cos45^0 & -sin45^0 & 0 \\ sin45^0 & cos45^0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

So the coordinates of the rotated triangle ABC are

$$R_{45^0}[ABC] = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 2 & 4 \\ 0 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \sqrt{2} \\ 0 & 2\sqrt{2} & 3\sqrt{2} \\ 1 & 1 & 1 \end{bmatrix}$$

Hence the final co-ordinate points are A'(0,0), B'(0, 2√2) and C'(√2, 3√2).

2.

Q. N. > Rotate the triangle (5, 5), (7, 3), (3, 3) in counter clockwise (CCW) by 90 degree.

A :

$$P' = R.\ P$$

$$= \begin{pmatrix} \text{Cos90} & -\text{Sin90} & 0 \\ \text{Sin90} & \text{Cos90} & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 5 & 7 & 3 \\ 5 & 3 & 3 \\ 1 & 1 & 1 \end{pmatrix}$$

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

$$= \begin{bmatrix} -5 & -3 & -3 \\ 5 & 7 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$

Hence, the new points are (-5,5) (-3,7) and (-3, 3).

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

Solution

The required steps are:

1. Translate the fixed point to origin.
2. Rotate about the origin by 90 degree.
3. Reverse the translation as performed earlier.

Thus, the composite matrix is given by

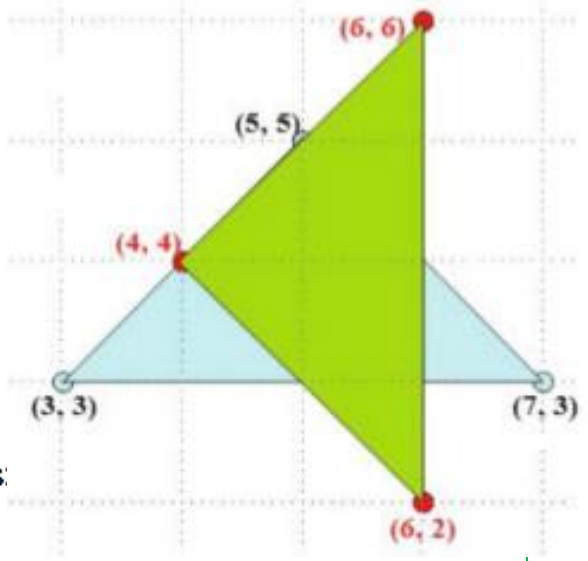
$$Com = T_{(xf, yf)} \, R_{\theta} \, T_{(-xf, -yf)}$$

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

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

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

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



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

$$P' = Com * P$$

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

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

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

4. Q. Rotate triangle ABC by 450 clockwise about origin and scale it by (2,3) about origin.

Here,

The steps required are:

1. Rotate by 45 clockwise
2. scale by $t_x = 2$ and $t_y = 3$

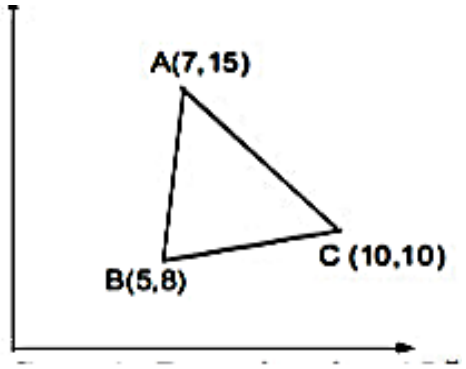
Thus the composite matrix is given by

$$com = S(2,3).R_{-45}$$

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45 & Sin45 & 0 \\ -Sin45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2/\sqrt{2} & 2/\sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



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

$$A' = com * A$$

$$= \begin{bmatrix} 2/\sqrt{2} & 2/\sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 15 \\ 1 \end{bmatrix} =$$

$$B' = com * B$$

$$= \begin{bmatrix} 2/\sqrt{2} & 2/\sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 8 \\ 1 \end{bmatrix} =$$

$$C' = com * C$$

$$= \begin{bmatrix} 2/\sqrt{2} & 2/\sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} =$$

Hence, the final co-ordinates are A'( ), B' ( ) and C'( ).

Reflection about an arbitrary line y=mx+c

Q. Derive the composite matrix for reflecting an object about any arbitrary line y=mx+c.

In order to reflect an object about any line y=mx+c, we need to perform composite transformation as below

$$T = T_{(0,c)} \cdot R_{\theta} \cdot R_{x} \cdot R_{-\theta} \cdot T_{(0,-c)}$$

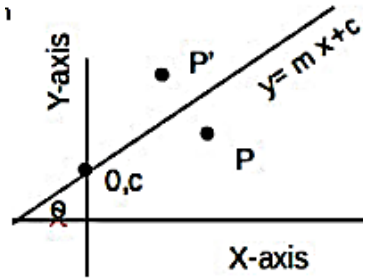
And

Slope m= tan Θ,

Also we have,

$$\cos^2\Theta = \frac{1}{\tan^2\theta+1} = \frac{1}{m^2+1}$$

$$\cos\Theta = \frac{1}{\sqrt{m^2+1}}$$



Also we have,

$$\sin^2\Theta + \cos^2\Theta = 1$$

$$\sin^2\Theta = 1 - \cos^2\Theta$$

$$\sin^2\Theta = 1 - \frac{1}{m^2+1} = \frac{m^2+1-1}{m^2+1}$$

$$\sin\Theta = \frac{m}{\sqrt{m^2+1}}$$

So,

$$T = T_{(0,c)} \cdot R_{\theta} \cdot R_{x} \cdot R_{-\theta} \cdot T_{(0,-c)}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -c \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & -c\sin\theta \\ -\sin\theta & \cos\theta & -c\cos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & -c\sin\theta \\ \sin\theta & -\cos\theta & c\cos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

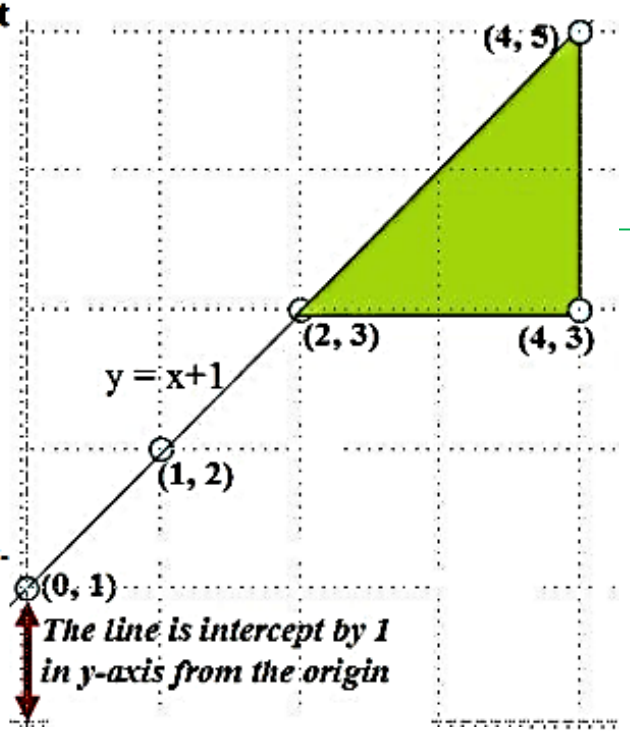
$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{(m^2+1)}} & \frac{-m}{\sqrt{(m^2+1)}} & 0 \\ \frac{-m}{\sqrt{(m^2+1)}} & \frac{1}{\sqrt{(m^2+1)}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{(m^2+1)}} & \frac{m}{\sqrt{(m^2+1)}} & \frac{-cm}{\sqrt{(m^2+1)}} \\ \frac{m}{\sqrt{(m^2+1)}} & \frac{-1}{\sqrt{(m^2+1)}} & \frac{c}{\sqrt{(m^2+1)}} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1-m^2}{m^2+1} & \frac{2m}{m^2+1} & \frac{-2cm}{m^2+1} \\ \frac{2m}{m^2+1} & \frac{m^2-1}{m^2+1} & \frac{c-cm^2}{m^2+1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1-m^2}{m^2+1} & \frac{2m}{m^2+1} & \frac{-2mc}{m^2+1} \\ \frac{2m}{m^2+1} & \frac{m^2-1}{m^2+1} & \frac{2c}{m^2+1} \\ 0 & 0 & 1 \end{bmatrix}$$

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

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:

1. Translate the line to origin by decreasing the y-intercept with one.



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

Thus, the composite matrix is given by:

$$Com = T' R_\theta' R_x R_\theta T$$

Thus, composite matrix is given by

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

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

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

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

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

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

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

$$P' = Com * P$$

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

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

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

Note: Composite Matrix can also 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.

Assignment:

1. Translate a square ABCD, A(0,0), B(3,0), C(3,3), D(0,3) three units in X-direction and 3 units in y-direction.
2. Scale a square ABCD, A(0,0), B(3,0), C(3,3) and D(0,3), three units in X-direction and 3 units in Y-direction wrt. origin.
3. Translate a square ABCD, A(0,0), B(3,0), C(3,3) and D(0,3) 2 units in both direction and scale it by 1.5 units in X direction and 0.5 units in Y-direction.
4. Magnify a triangle with vertices A(0,0) , B(0,1) and C(5,2) to twice its size while keeping C(5,0) fixed, i.e. fixed point scaling.
5. Find the coordinate of a triangle A(1,3), B(2,5) and C(3,3) after twice its original size about fixed point (2,3).
6. Write 2x2 transformation matrix for each of the following rotation about the origin: i) counter clockwise by pie ii) Clockwise by Pi / 2
7. Preform a 45 rotation of a triangle a(0,0), B(1, 1) and C(5,2):
   a) About the origin
   b) About the point (-1,-1)
8. Reflect the triangle ABC about the line 3x-4y+8 = 0. the position vector of the coordinates ABC is given as: A(4,1), B(5,2) and C94,3)
9. Rotate a triangle ABC by 90 degree anti-clockwise about (5,8) and scale it by (2,2) about (10,10).
10. A mirror is placed such that it passes through (0,10) and (10,0). Find the mirror image of an object (6,7), (7,6) , (6,9).

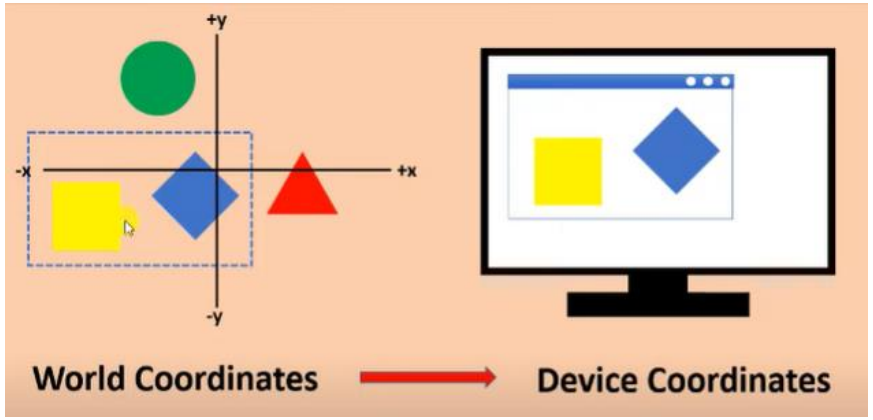
3.5 Windowing Concepts, Two Dimensional Viewing Pipeline

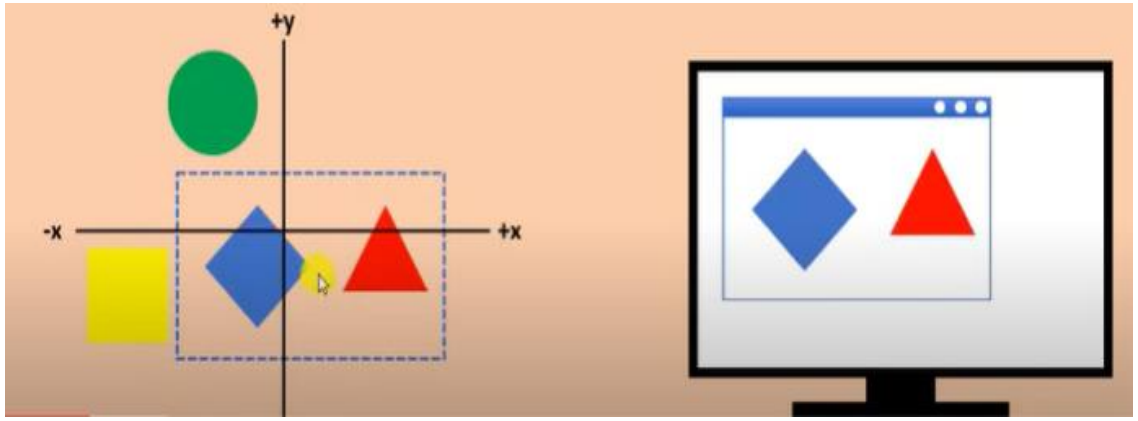
Two Dimensional Viewing Pipeline

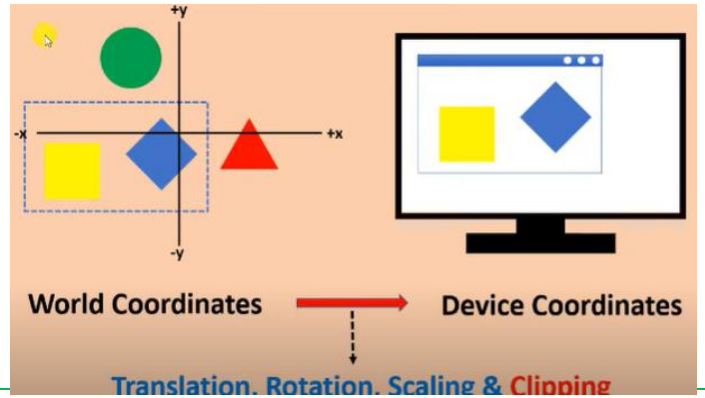
- Two-dimensional viewing is **the mechanism for displaying views of a picture on an output device.**
- Much like what we see in real life through a small window on the wall or the viewfinder of a camera, a computer generated image often depicts a partial view of a large scene.
- For a 2D picture, a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

Coordinate Systems

- Which part of a defined picture is to be displayed?
- Where that part is to be placed on the display device?



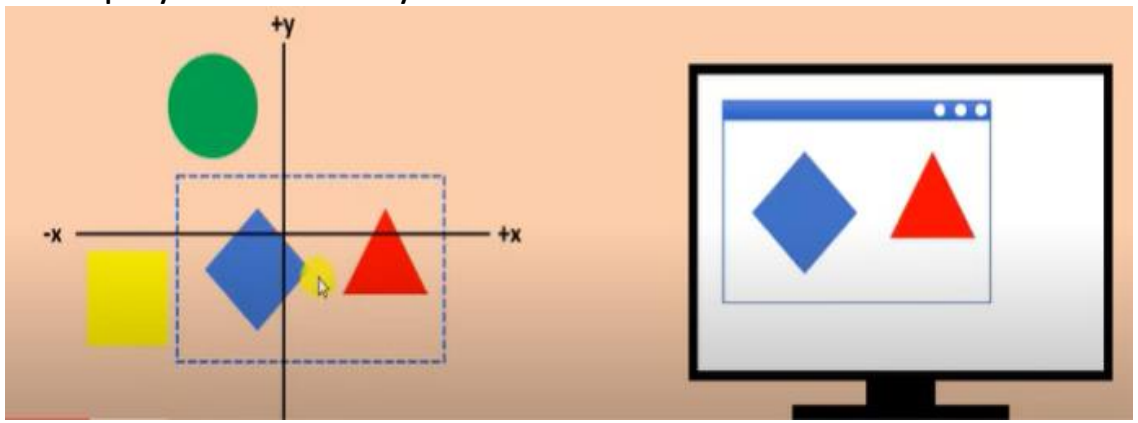


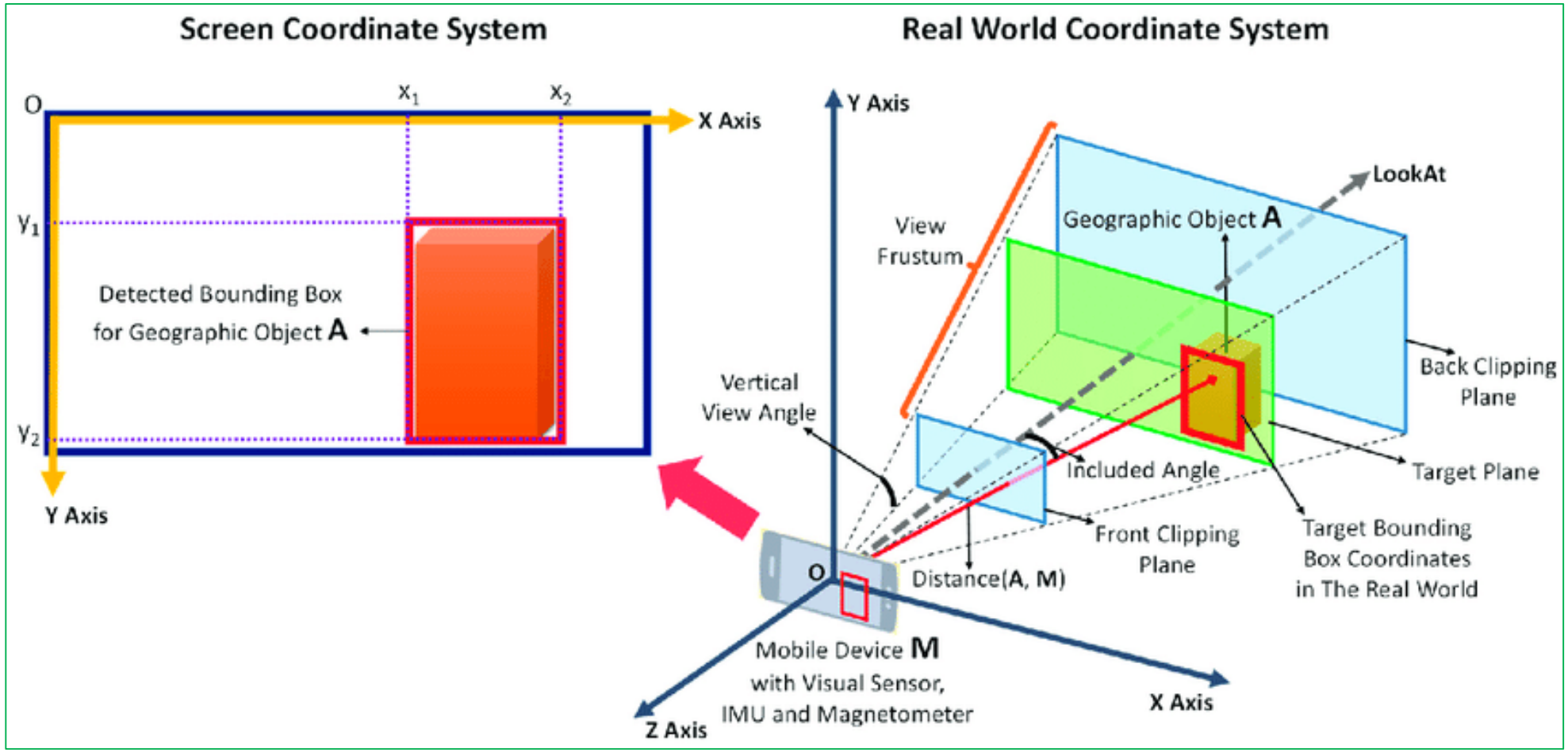


Viewing coordinate

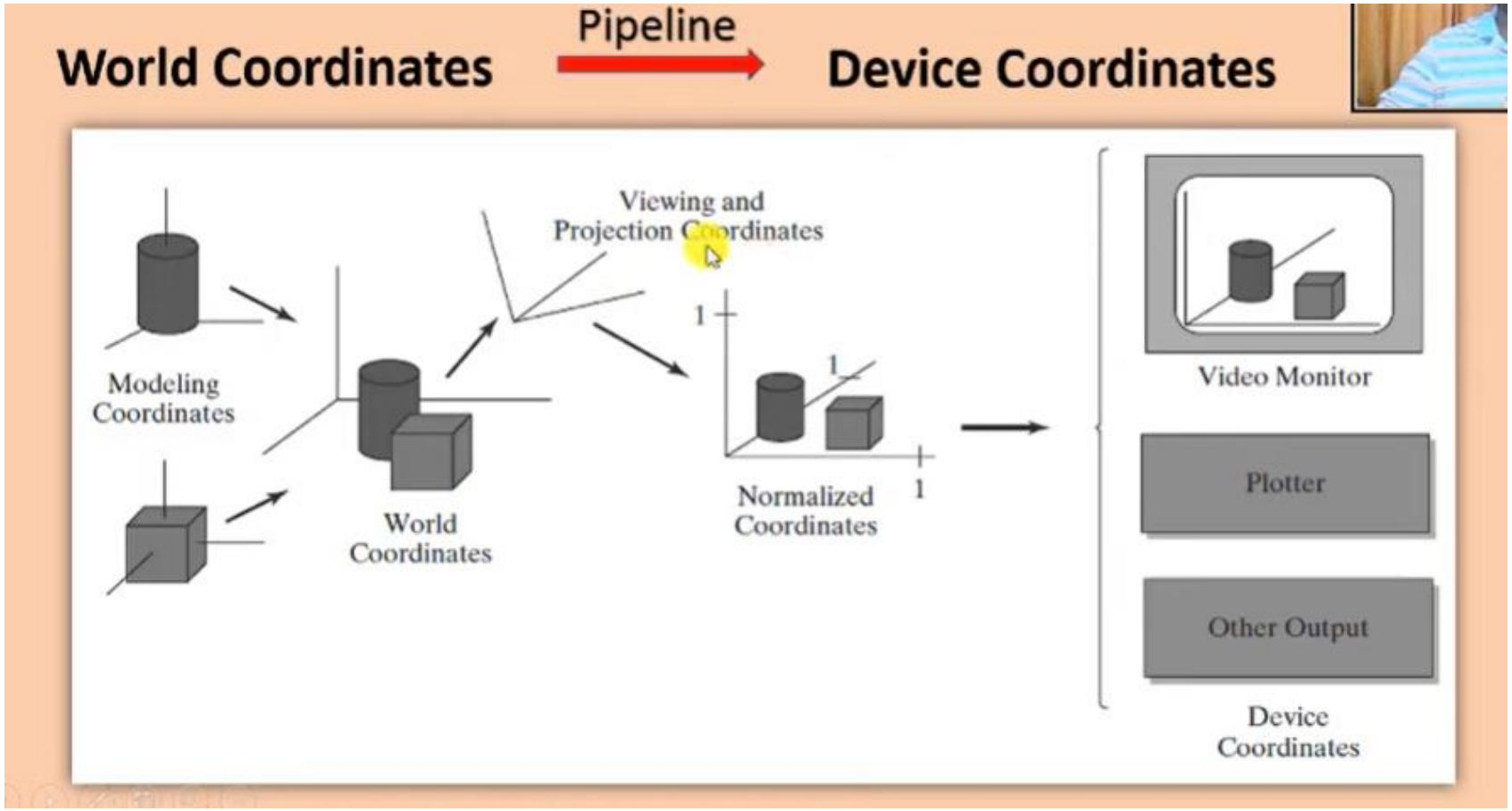
Coordinate System

1. **Model Coordinates (MC):** Every individual object or model has its own coordinate system. This coordinate system is called model (or local/master) coordinate.
2. **World Coordinates (WC):** once individual objects have been identified, these objects are placed into appropriate position within a scene using reference frame to interact each other. This reference frame is called world-coordinate.
3. **Viewing coordinates (VC):** Viewing coordinate is used to define window in the world coordinate plane with any possible orientation.
4. **Normalized Viewing Coordinates (NVC):** In normalized coordinate, the coordinates are in range 0 to 1. WC are converted to NC before final conversion to specified Device coordinate (DC)
5. **Device Coordinates (DC):** When the WC description of the scene is transformed to one or more o/p device reference frame for display, then the display coordinate system is referred to DC.




Screen Coordinate System
Real World Coordinate System
O
x_1
x_2
X Axis
y_1
y_2
Detected Bounding Box
for Geographic Object A
Y Axis
Y Axis
LookAt
View
Frustum
Geographic Object A
Back Clipping
Plane
Vertical
View Angle
Included Angle
Target Plane
Front Clipping
Plane
Target Bounding
Box Coordinates
in The Real World
Distance(A, M)
O
Mobile Device M
with Visual Sensor,
IMU and Magnetometer
Z Axis
X Axis

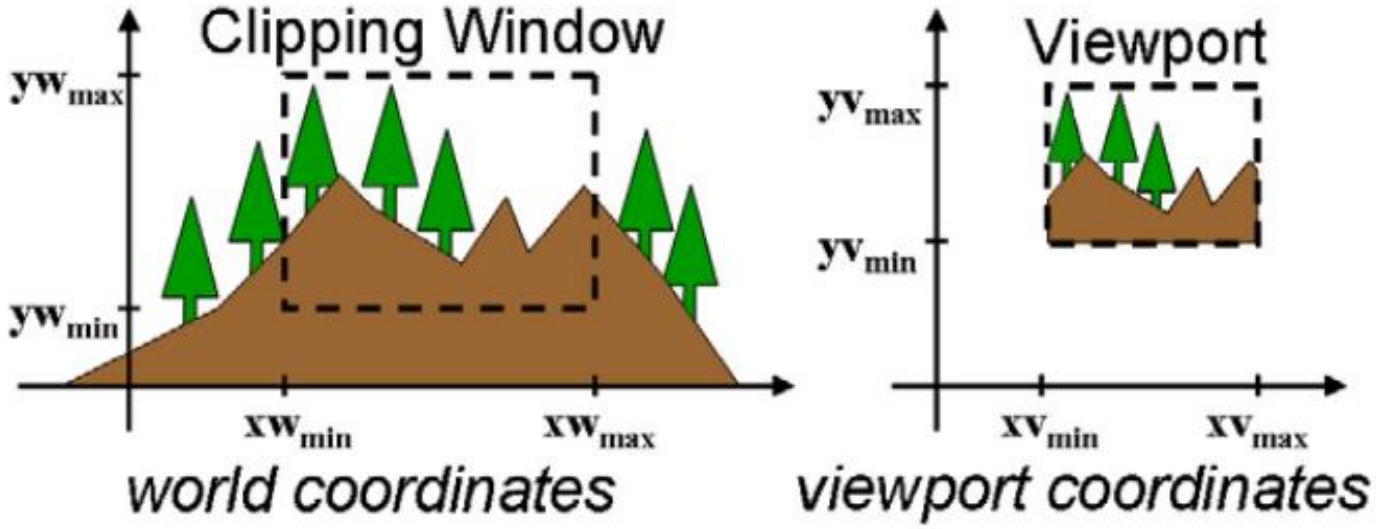
- For more study:



Reference : https://www.youtube.com/watch?app=desktop&v=PuAwwI0QlT8

Windowing Concepts

- 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.**
- Window defines what is to be viewed and the view-port defines where it is to be displayed.
- The mapping of a part of a world-coordinate scene to device-coordinate is referred to as **viewing transformation/ window transformation** (ie. Window to viewport transformation).



What is window?

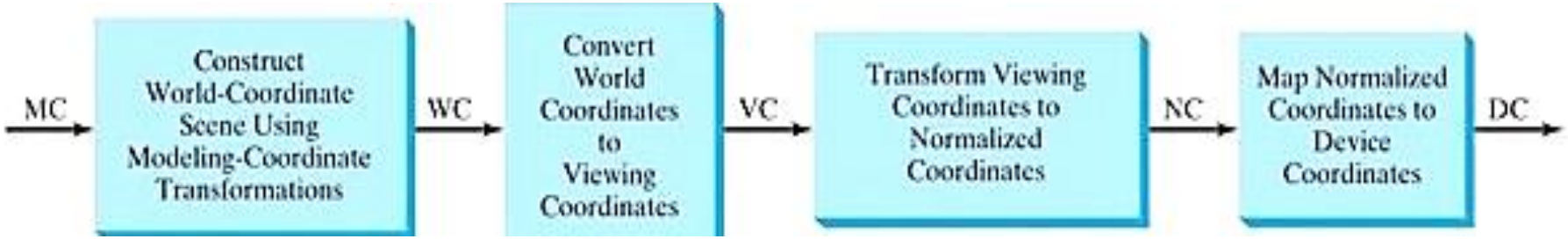
- A world-coordinate area selected for display is called a window.
- You can define the window to be larger than, the same size as, or smaller than the actual range of data values, depending on whether you want to show all of the data or only part of the data.
- **Window defines what is to be viewed.**

What is viewport?

- An area on a display device to which a window Is mapped is called a viewport.
- The rectangular portion of the interface window that defines where the image will actually appear.
- **Viewport defines where the window to be displayed.**

2D Viewing Pipeline

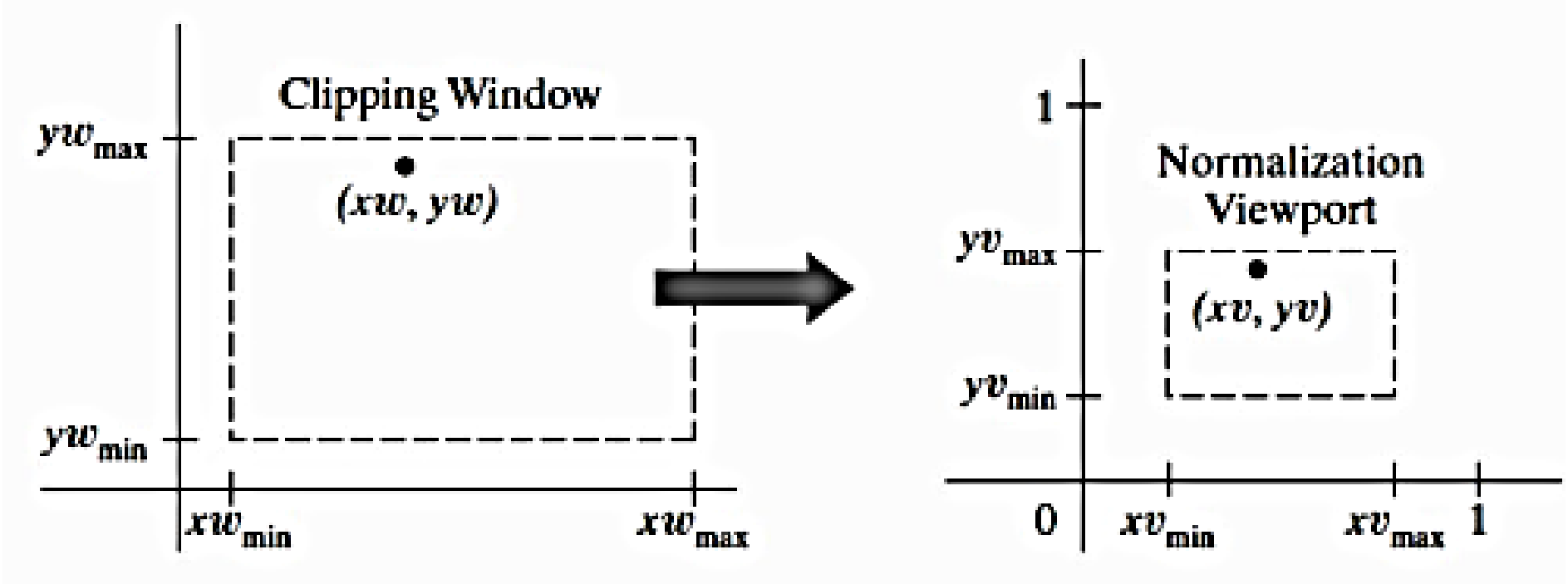
- The term Viewing Pipeline describes a series of transformations, which are passed by geometry data to end up as image data being displayed on a device.



- The steps involved in viewing transformation:-
  1. **Construct the scene in world co-ordinate** using the output primitives and attributes.
  2. **Set a 2-D viewing co-ordinate system** in the world coordinate plane and **define a window** in the viewing co-ordinate system.
  3. Once the viewing reference frame is established, **transform descriptions in world co-ordinates to viewing co-ordinates.**
  4. **Define a view port in normalized co-ordinates** and map the viewing co-ordinates description of the scene to normalized co-ordinates.
  5. **Clip all the parts of the picture which lie outside the viewport**, and the contents of the viewport are **transferred to Device coordinate.**

3.6 Window to Viewport Transformation

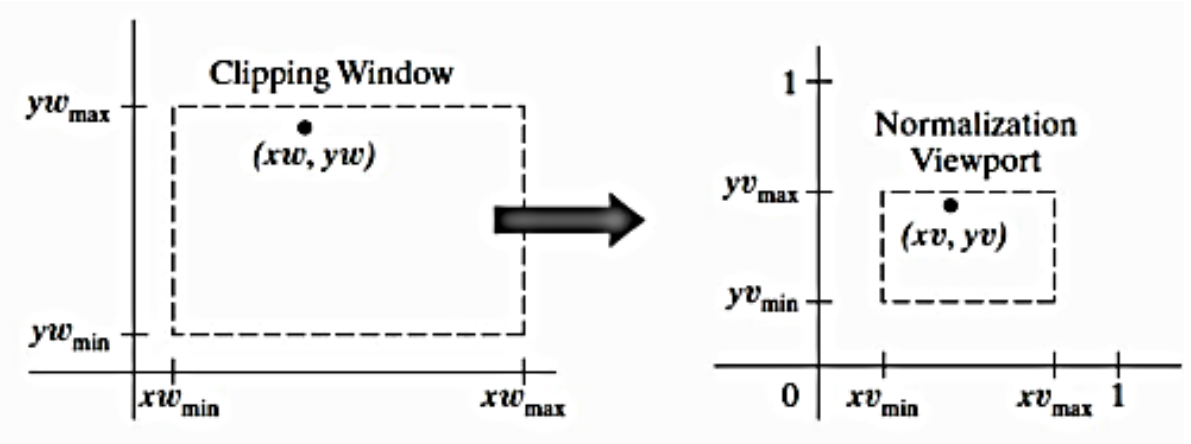
A window can be specified by four world coordinates: xw_{min}, xw_{max}, yw_{min} and yw_{max}. Similarly, a view port can be described by four device coordinates: xv_{min}, xv_{max}, yv_{min} and yv_{max}



- The following proportional ratios must be equal:

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}}$$



Solving these expressions, we get

$$xv = xv_{min} + (xw - xw_{min})S_x$$

$$yv = yv_{min} + (yw - yw_{min})S_y$$

where,

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

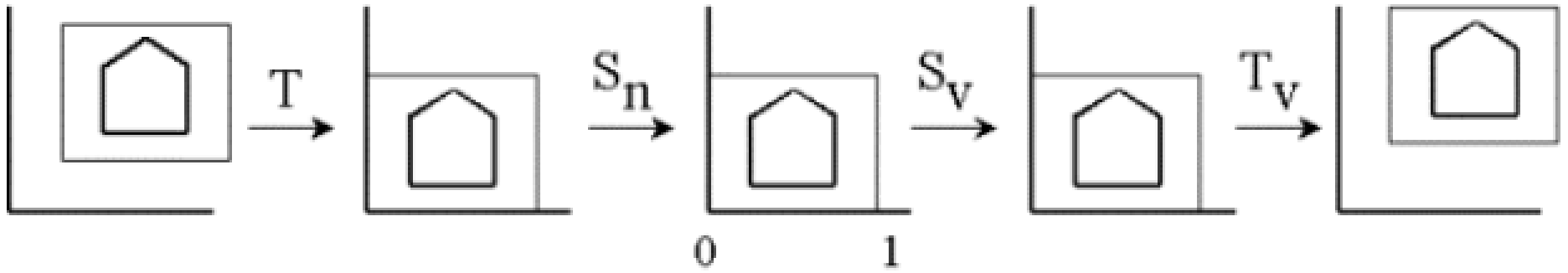
$$s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

- Steps for transformation of window to view-port:

1. Translate the window to the origin. That is, apply T (-Xwmin, -Ywmin)
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(Xvmin, Yvmin).

Therefore, net transformation,

T_{wv} = T (Xvmin, Yvmin). S(sx, sy).T(-Xwmin, -Ywmin)



1. **Q. Window port is given by (100,100,300,300) and view port is given by (50,50,150,150). Convert the window port coordinate (200,200) to the view port coordinate.**

Solution:

$(xw_{min}, yw_{min}) = (100, 100)$
$(xw_{max}, xw_{max}) = (300, 300)$
$(xv_{min}, yv_{min}) = (50, 50)$
$(xv_{max}, xv_{max}) = (150, 150)$
$(xw,yw)=(200,200)$
Then, we have

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}} = (150- 50)/ (300 - 100) = 0.5$$

$$s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}} = (150- 50)/ (300 - 100) = 0.5$$

The equations for mapping window co-ordinate to view port coordinate is given by

$xv = xv_{min} + (xw - xw_{min})S_x$
$yv = yv_{min} + (yw - yw_{min})S_y$

Hence,

$xv = 50 + (200 - 100) * 0.5 = 100$
$yv = 50 + (200 - 100) * 0.5 = 100$

The transformed view port coordinate is (100,100).

2. Q. Find the normalization transformation matrix for window to view port which uses the rectangle whose lower left corner is at (2,2) and upper right corner is at (6,10) as a window and the view port that has lower left corner at (0,0) and upper right corner at (1,1)

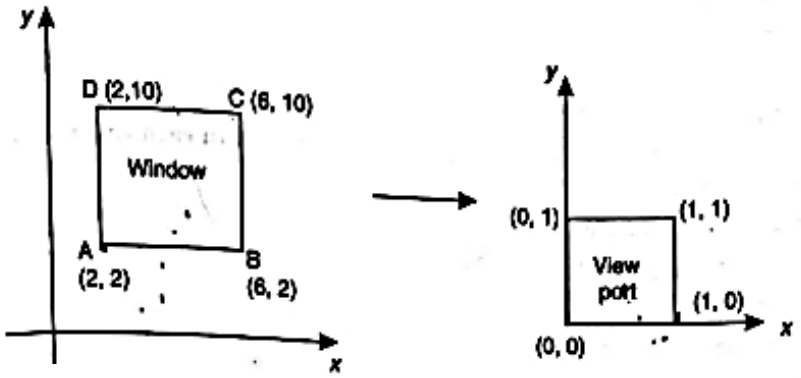
The composite transformation matrix for transforming the window co-ordinate to viewport coordinate is given as

$$T = S(s_x, s_y)T(-2, -2)$$

Now we know,

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

$$s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$



$S_x = (1-0)/(6-2) = 0.25$

$S_y = (1-0)/(10-2) = 0.125$

Then, transformation matrix,

$$T = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 0.125 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.25 & 0 & -0.5 \\ 0 & 0.125 & -0.25 \\ 0 & 0 & 1 \end{bmatrix}$$

Assignment:

- Consider the window is located from X_{wmin}=20, X_{wmax}=80, Y_{wmin}=40, Y_{wmax}=80 and a point is located in (30,80). Identify the new location of the point in the view port considering viewport size X_{vmin}=30, X_{vmax}=60, Y_{vmin}=40, Y_{vmax}=60.

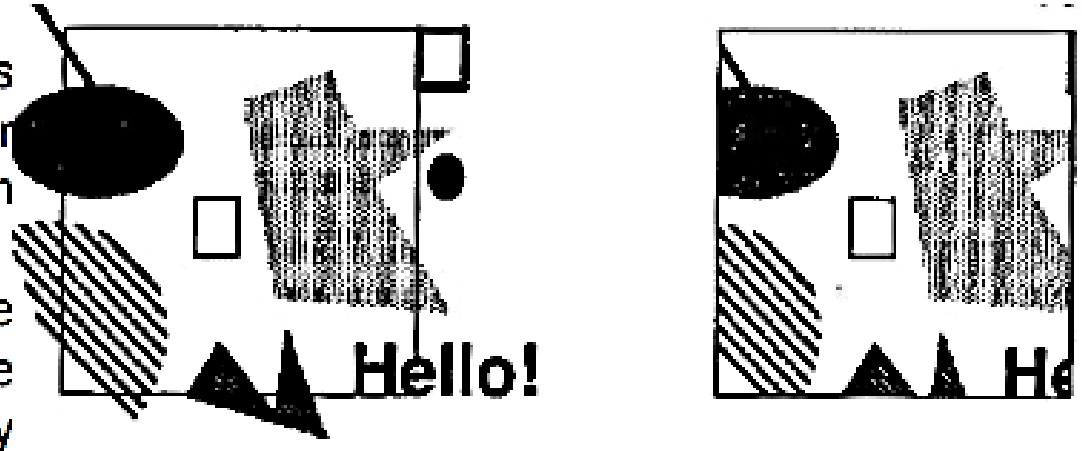
3.7 Clipping

Clipping

The process of discarding those parts of a picture which are outside of a specified region or window is called **clipping**. The procedure using which we can identify whether the portions of the graphics object is within or outside a specified region or space is called **clipping algorithm**.

The region or space which is used to see the object is called window and the region on which the object is shown is called **view port**.

Clipping is necessary to remove those portions of the objects which are not necessary for further operation's. excludes unwanted graphics from the screen. So there are three cases

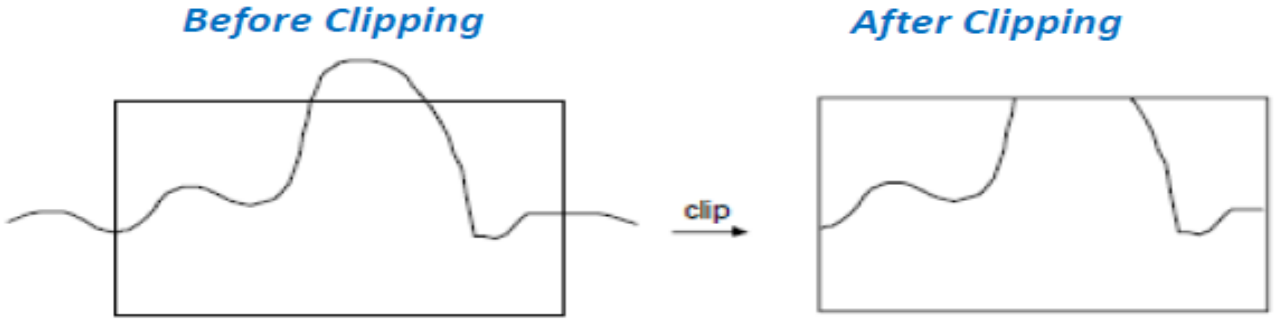


i. The object may be completely outside the viewing area defined by the window port.
ii. The object may be seen partially in the window port.
iii. The object may be seen completely in the window port

For case i and ii, clipping operation is necessary but not for case iii.

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.



Applications of Clipping

i. Extracting part of a defined scene for viewing.
ii. Identifying visible surfaces in three-dimensional views.
iii. Anti aliasing line segments or object boundaries.
iv. Drawing and painting operations that allow parts of a picture to be selected for copying, moving erasing, or duplicating.

Types of Clipping

i. Point Clipping
ii. Line Clipping (straight-line segments)
iii. Area Clipping (polygons)
iv. Curve Clipping
v. Text Clipping

i) Point Clipping

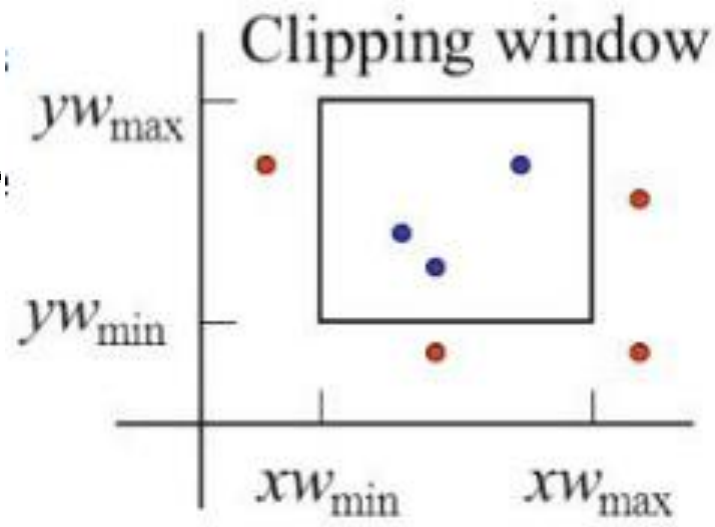
Point clipping is the process of removing all those points that lies

that lies outside a given region or window. Let

Point clipping is the process of removing all those points that lies out
xw_{min} and xw_{max} be the edge of the boundaries of the clipping window parallel to Y axis and yw_{min} and yw_{max} be the edge of the boundaries of the clipping window parallel to X axis as shown in figure below.
So any point P(x,y) of world coordinate can be saved for display if the following conditions are satisfied

$$xw_{min} \leq x \leq xw_{max}$$

$$yw_{min} \leq y \leq yw_{max}$$

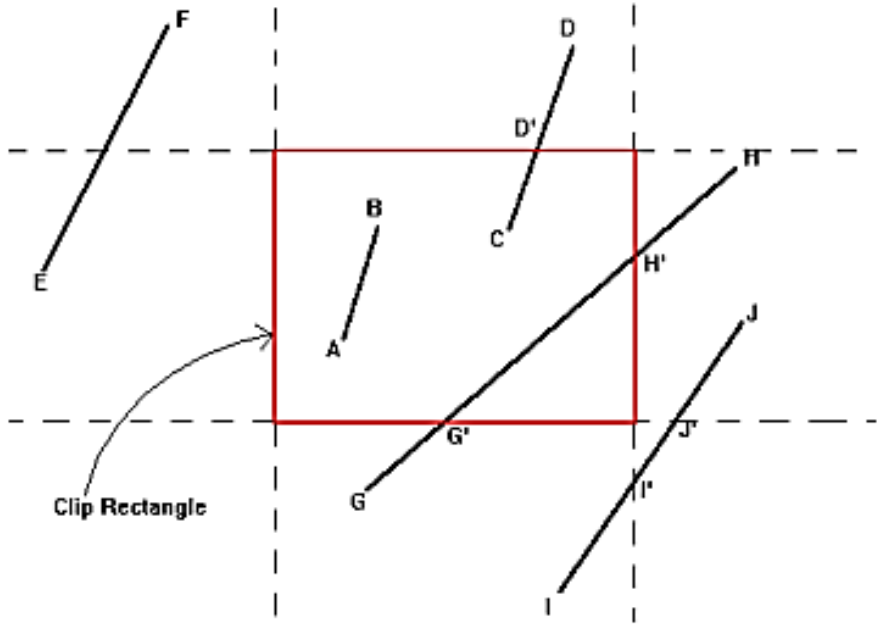


If one of these four inequalities is not satisfied,
The point is clipped (ie. Not saved for display)

ii) Line Clipping

In line clipping, a line or part of line is clipped if it is outside the window port. All the line segment falls into one of the following clipping cases.

a. The line is totally outside the window port so is directly clipped.
b. The line is partially clipped if a part of the line lies outside the window port.
c. The line is not clipped if all whole line lied within the window port.



So in the above figure, line FE require total clipping, CD, GH, IJ require partial clipping and AB requires no clipping.

3.7 Line Clipping Algorithm: Cohen-Sutherland

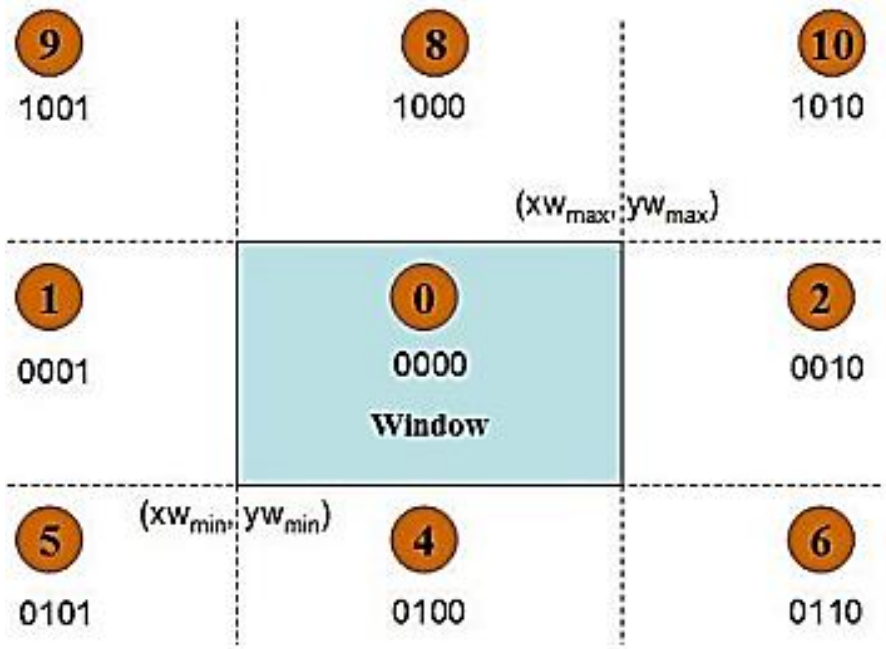
3.7 Line Clipping Algorithm: Cohen-Sutherland

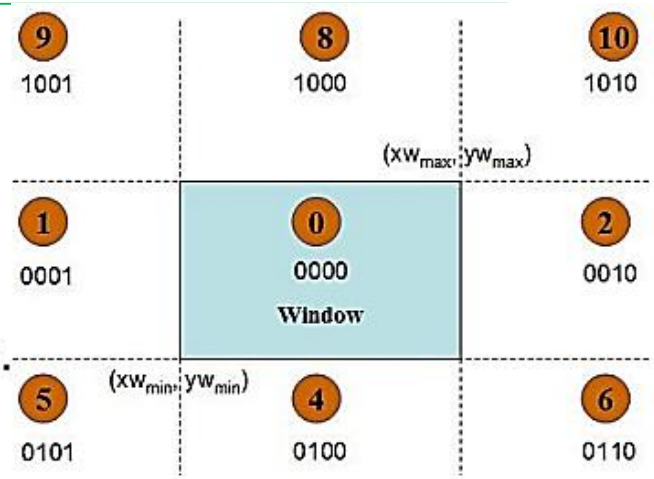
- In this we divide whole space into nine region and assign 4 bit code to each endpoint of line depending on the position where the line endpoint is located.

  - 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.

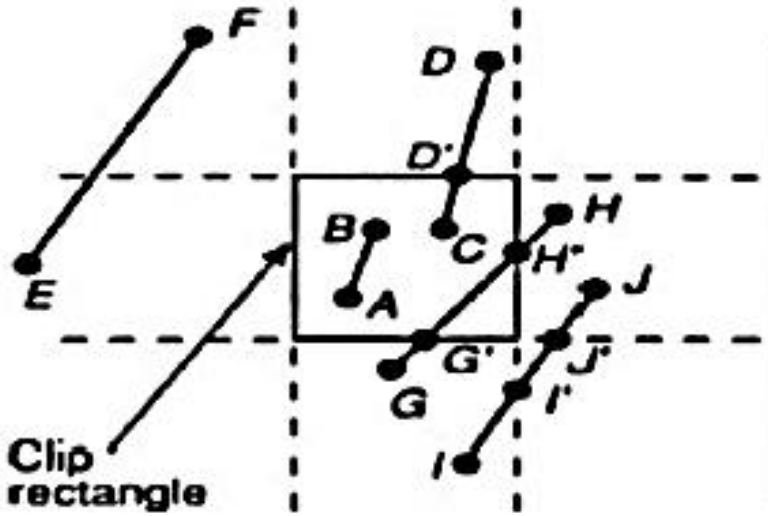
4	3	2	1
Top	Bottom	Right	Left





- Any point inside the clipping window has a region code 0000.
- Any endpoint (x, y) of a line segment, the code can be determined as follows:
  - If $x < xw_{min}$, first bit is 1, (Point lies to left of window(Left)) (0th bit) Otherwise 0.
  - If $x > xw_{max}$, second bit is 1, (Point lies to right of window(Right)) (1st bit), otherwise 0.
  - If $y < yw_{min}$, third bit is 1, (Point lies to below window(Bottom)) (2nd bit), otherwise 0.
  - If $y > yw_{max}$, fourth bit is 1, (Point lies to above window(Top)) (3rd bit), otherwise 0.

Example:

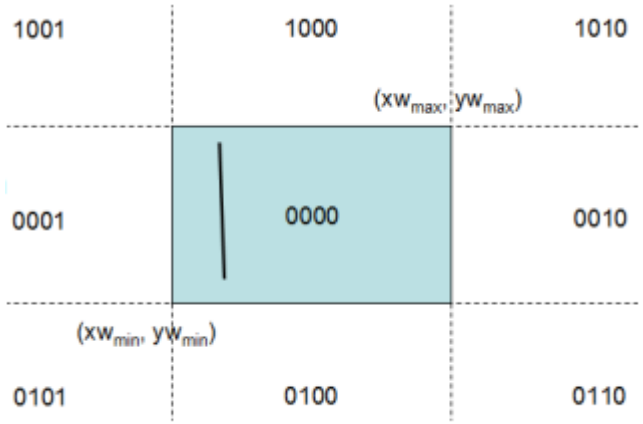


Line	Code 1	Code 2
A⇒B	0000	0000
C⇒D	0000	1000
E⇒F	0001	1001
G⇒H	0100	0010
I⇒J	0100	0010

- Cases of acceptance and rejections:

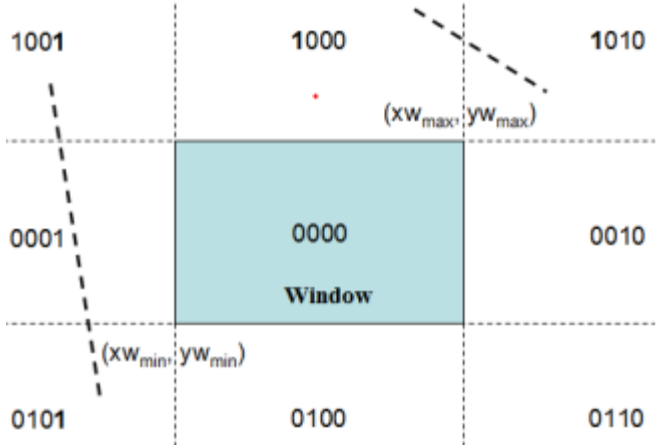
Case I: **Line completely inside the window**

If both end point code is 0000, then the line segment is completely inside the window. i.e. if bitwise OR of the codes yields 0000, then the line segment is accepted for display.



Case II: **Line completely outside the window**

If the two end point region code both have a 1 in the same bit position, then the line segment lies completely outside the window, so discarded. i.e. if bitwise AND of the codes not equal to 0000, then the line segment is rejected.



Case III: **Line partially inside the window**

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:

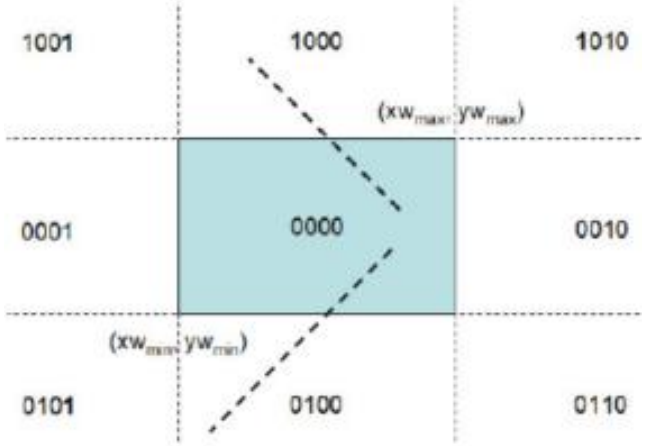
$$y - y_1 = m\,(x - x_1)$$

where $m = (y_2 - y_1)/(x_2 - x_1)$

i. If the intersection is with vertical boundary

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

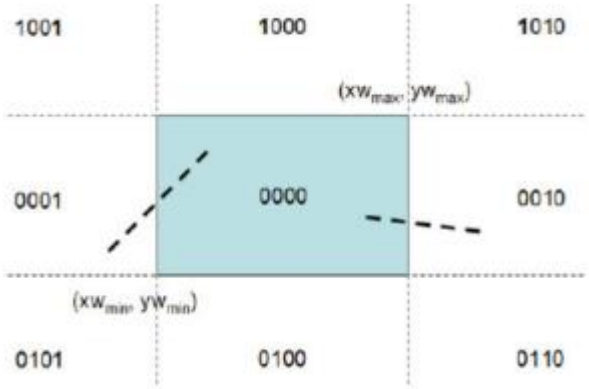
Where y is set to either yw_{min} or yw_{max}



ii. If the intersection is with Horizontal boundary

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

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



- Completely inside the window:

E	0000
F	0000
OR	0000
AND	0000
accept	

- Completely outside the window:

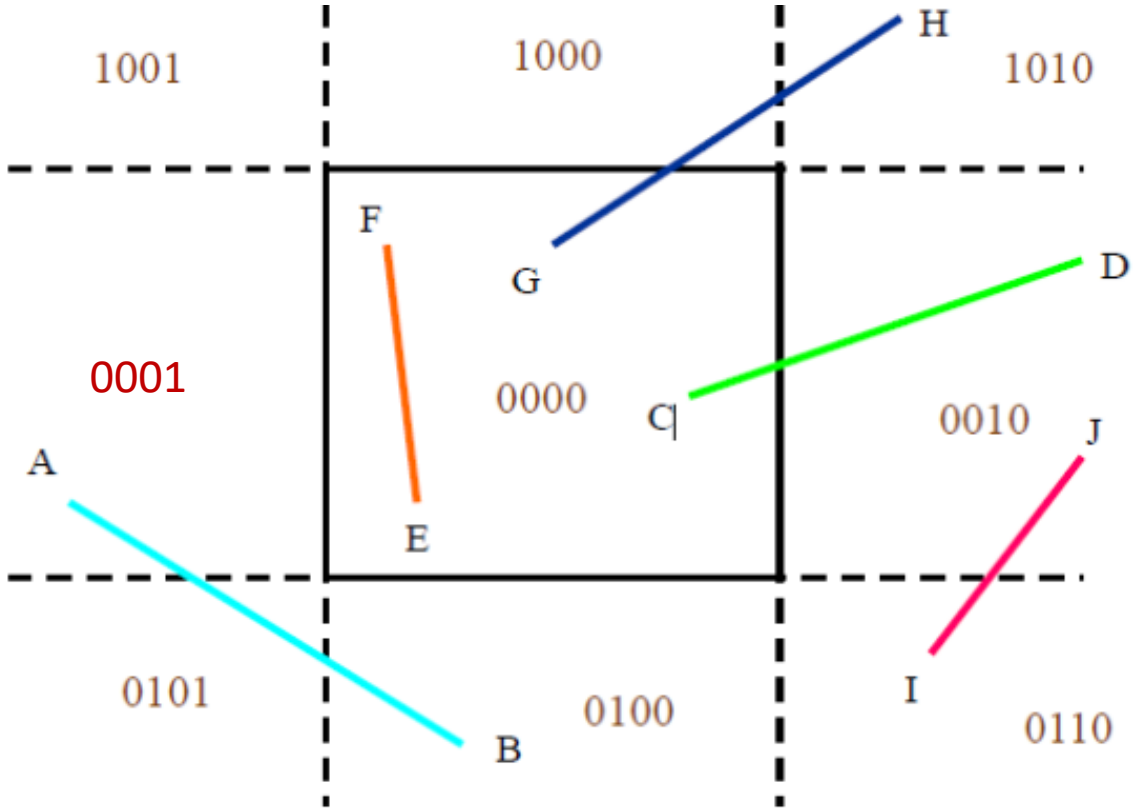
I	0110
J	0010
OR	0110
AND	0010
reject	

- Partially inside the window:

C	0000
D	0010
OR	0010
AND	0000
clip	

G	0000
H	1010
OR	1010
AND	0000
clip	

A	0001
B	0100
OR	0101
AND	0000
clip	



Algorithm:

Step 1: Establish region code for all line end points.

Bit 1 is set to '1' if $x < x_{min}$ else set to '0'.

Bit 2 is set to '1' if $x > x_{max}$ else set to '0'.

Bit3 is set to '1' $y < y_{min}$ else set '0'

Bit 4 is set to '1' $y > y_{max}$ else set to '0'

Step 2: Determine whether line is completely inside or outside window using test.

a) If both end pint have region code ' 0000' line is completely inside.

b) If logical AND of end points of a line not '0000' line is completely outside.

Step 3: If both condition of step 2 fails. i.e. Logical AND give '0000' we need to find the intersection with window boundary.

Here,

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

a) If bit 1 is 1, line intersection with left boundary, so,

$y_i = y_1 + m(x - x_1)$ where $x = x_{min}$

b) If bit 2 is 1, line intersect with right boundary, so,

$y_i = y_1 + m(x - x_1)$ where $x = x_{max}$

c) If bit 3 is 1; line intersect with lower boundary, so,

$x_i = x_1 + \frac{1}{m}(y - y_1)$ where $y = y_{min}$

d) If bit 4 is 1, line intersects with upper boundary, so,

$x_i = x_1 + \frac{1}{m}(y - y_1)$ where $y = y_{max}$

Here, x_i and y_i are x, y intercepts for that line, update

Step 4: Repeat step 2 and 4 till completely accepted

Q. Clip a line with end point 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)

Algorithm:

Step 1: Establish region code for all line end points.

Bit 1 is set to '1' if $x < x_{min}$ else set to '0'.

Bit 2 is set to '1' if $x > x_{max}$ else set to '0'.

Bit3 is set to '1' $y < y_{min}$ else set '0'

Bit 4 is set to '1' $y > y_{max}$ else set to '0'

Step 2: Determine whether line is completely inside or outside window using test.

a) If both end pint have region code ' 0000' line is completely inside.

b) If logical AND of end points of a line not '0000' line is completely outside.

Step 3: If both condition of step 2 fails. i.e. Logical AND give '0000' we need to find the intersection with window boundary.

Here,

$m = \frac{y_2 - y_1}{x_2 - x_1}$

a) If bit 1 is 1, line intersection with left boundary, so,

$y_i = y_1 + m(x - x_1)$ where $x = x_{min}$

b) If bit 2 is 1, line intersect with right boundary, so,

$y_i = y_1 + m(x - x_1)$ where $x = x_{max}$

c) If bit 3 is 1; line intersect with lower boundary, so,

$x_i = x_1 + \frac{1}{m}(y - y_1)$ where $y = y_{min}$

d) If bit 4 is 1, line intersects with upper boundary, so,

$x_i = x_1 + \frac{1}{m}(y - y_1)$ where $y = y_{max}$

Here, x_i and y_i are x, y intercepts for that line, update

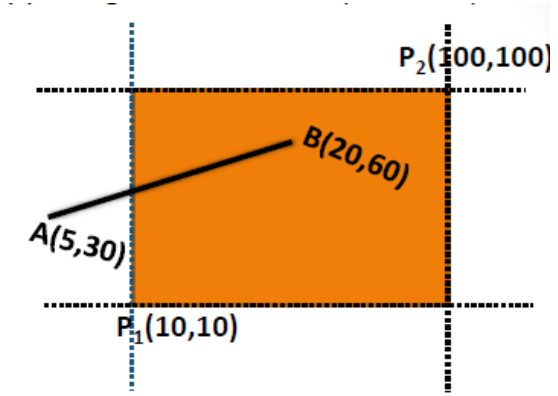
Step 4: Repeat step 2 and 4 till completely accepted

Solution: Here, X_{min}= 10, X_{max}= 100, Y_{min}= 10 and Y_{max}=100

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

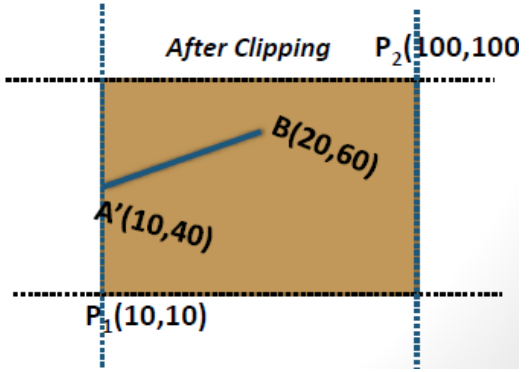
A: 0001 , Cond^n of 1st bit is 1

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

= 30 + 2 (10-5)

= 40

(X,Y)=(10,40)



Hence, Intersection point is (10,40).

Assignment:

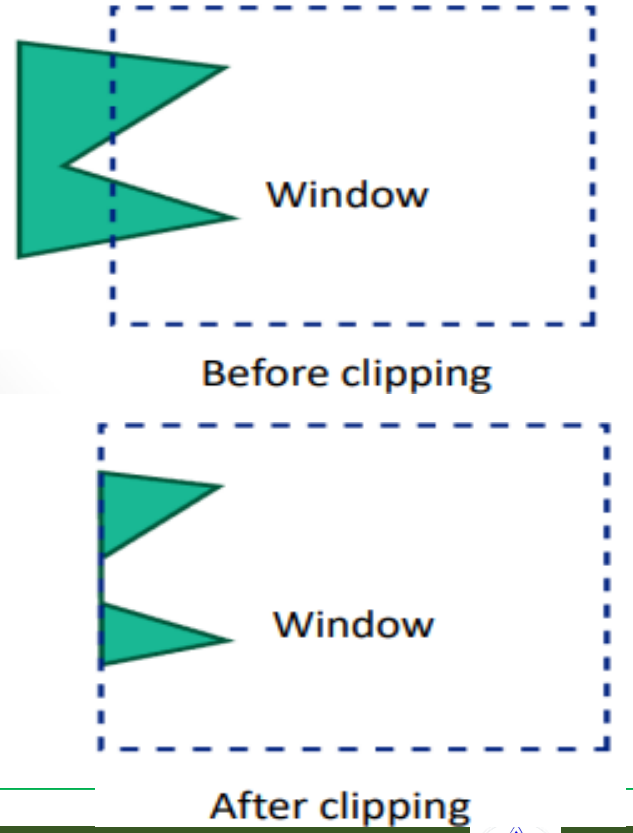
1. Use the Cohen Sutherland algorithm to clip a line P1(70,20) and P2(100, 10) against a window with lower-left corner (50,10) and upper-right corner (80,40).
2. Let R be the rectangular window whose lower left hand corner is at L(-3, 1) and upper right hand corner is at R(2,6). Use Cohen-Sutherland algorithm to clip the line segments A(-4,2) and B(-1,7).
3. Given a clipping window a(10,10), B(40,10), C(40,40) and D(10,40). Using Cohen-Sutherland line clipping algorithm, find region code of each end points of lines P1P2, P3P4 and P5P6 where coordinates are P1(5,15), P2(25,30), P3(15,15) P4(35,30) , P5(5,8) and P(40,15). Also find clipped lines using above parameters.

3.8 Polygon Clipping: Sutherland-Hodgeman

3.8 Polygon Clipping: Sutherland-Hodgeman

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.

- A polygon can be defined as a geometric object "consisting of a number of points (called vertices) and an equal number of line segments (called sides or edges). Polygon clipping is defined as the process of removing those parts of a polygon that lie outside a clipping window. Consider a general polygon that is clipped with respect to a rectangular viewing region.



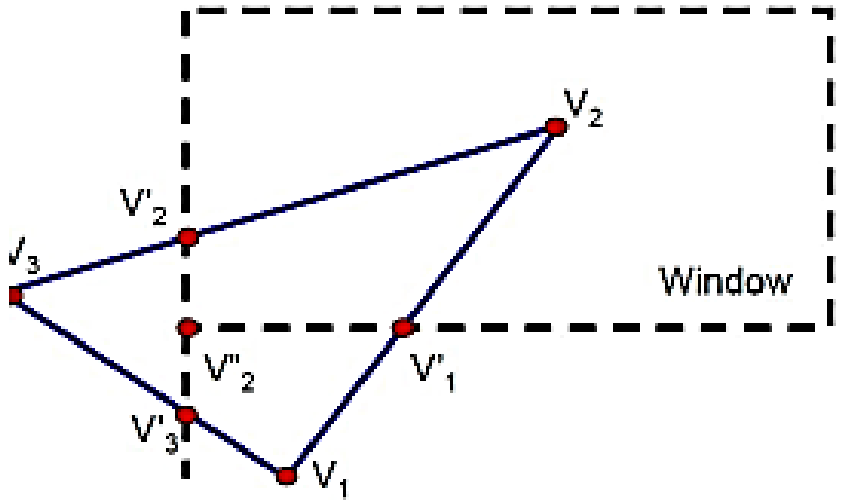
There are four possible cases for any given edge of given polygon against clipping edge.

1. **Both vertices are inside :**
   Only the second vertex is added to the output list
2. **First vertex is outside while second one is inside :**
   Both the point of intersection of the edge with the clip boundary and the second vertex are added to the output list
3. **First vertex is inside while second one is outside :**
   Only the point of intersection of the edge with the clip boundary is added to the output list
4. **Both vertices are outside :**

   No vertices are added to the output list

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

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	

Exam questions:

Exam questions:

1. Explain the 2D viewing pipeline along with the derivation for the window to viewport transformation. [2011 fall]
2. Reflect the triangle with vertices A(2,2), B(4,1) and C(5,3) along the line x=3. [2011 fall/spring]
3. Define window and viewport. What are the different steps of 2D world to screen viewing transformation? Describe with matrix representation at each steps. [2011 fall]
4. Clip the line from (-2,3) to (18,13) against the window dimension, lower left corner (0,0) and upper right corner (20,100) using Chen Sutherland algorithm. [2012 fall]
5. Given a diamond shaped polygon with vertices V1(5,5), V2(3,3), V3(5,1) and V4(7,3), reflect the object about a line y=x+2. [2012fall]
6. Clip the line P1P2 with PI(0,120) and P2(130,5) suing Cohen-Sutherland Line Algorithm. Given that rectangular window ABCD has end-points A(10,100), B(150,100), C(150,10) and D(10,10). [2012 spring]
7. Explain transformation of 2D object to screen viewing with matrix derivation. [2012 spring]
8. Calculate viewing transformation matrix with given information: given triangle with sides A(5,5), B(15,5), C(10,10), given window coordinate (7,4) (13,4)(13,8), (7,8) and view port location is (17,7),(18,7),(18,8),(17,8)? [2012 spring, 214 fall]
9. Scale the triangle with vertices A(0,00, B(1,1), C(5,2) to half its size while keeping B(1,1) fixed. [2012 spring]
10. Rotate the triangle A(2,3), B(5,3) and C(3,1) about a fixed point by 30 degree, [2013 fall]

Exam questions:

11. Explain the 2D viewing pipeline along with the derivation for the window to viewport transformation. [2013 fall]
12. Differentiate between window port and view port. Derive the transformation matrix for window to view port transformation. [2013 spring]
13. What is the significance of clipping operation? Explain the clipping operation used foe clipping lines in 2D. [2013 spring]
14. Scale a triangle A(0,0), B(1,1), C(3,2) by twice its original size, about origin and about point P(-1,-2). [2013 spring]
15. Perform a 45 degree rotation of a line A(5,3) and B(10,15) about the origin. [2014 fall]
16. What is clipping? Explain in detail about Sutherland-Hodgeman polygon clipping algorithm with example. [2014 fall, 2015 fall]
17. Perform a 45 degree rotation of a line A(8,3) and B(14,10): i. About the origin. ii. About a fixed point (4,2). [2014 spring]
18. Reflect a rectangle A(2,2) B(5,2) C92,4) D(5,4) about a line x=y. [2014 spring]
19. Define window and view-port. Derive the matrix that is responsible for placing an object from a window to viewport. [2015 fall]
20. What is line clipping? Explain in detail about Sutherland line clipping algorithm with example. [2015 spring]

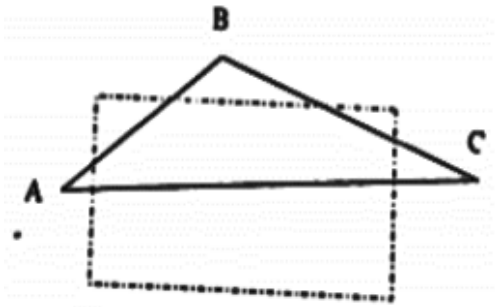
Exam questions:

21. Use Liang Barskly line clipping algorithm to clip a line starting from (-11,5) and ending at (15,11) against the window having its lower left corner ar (-6,-4) and upper right corner at (10,8). [2016 fall]
22. Discuss why homogeneous coordinates are sued in computer graphics for transformation computations? Also explain homogeneous transformation matrix for various 2D basic transformations. [2016 fall]
23. A point (-5,3) is required to be rotated by 45 degrees in clockwise direction and then scaled by a factor of 3, what will be the final transformed position after applying these transformations. [2016 fall]
24. What will be the final coordinates of a polygon with vertices A(3,4), B(5,4), C(5,2), D(3,4) after it is reflected about a line y=2x+1? [2016 spring]
25. Explain the steps of 2D viewing pipeline? How is the complexity added in 3D viewing process in comparison to 2D viewing process? [2016 spring]
26. Why do we need clipping? Explain in detail about Cohen-Sutherland line clipping algorithm. [2016 spring]
27. Show that the composition of two successive rotation are additive. [2017 Fall]
28. Derive the composite transformation matrix for reflection of an object about a line y=mx+c. Apply the derived matrix for the object A(4,2), B(7,3), C(9,2), D(10,1) on the line y=3x.
29. A mirror is placed vertically such that it passes through the points (10,0) and (0,10). Find the related view of triangle ABC with coordinates A(5,50), B(20,40) and C(10,70). [2017 spring]
30. Describe the rotation of an object about an axis, which is parallel to any of three coordinate axes of coordinate system.

Exam questions:

31. What will be the final coordinates of a triangle with vertices A(2,3), B(3,3) and C(3,2) after reflecting it about the line y=x? [2018 Fall]
32. What is windowing and clipping? Derive window too viewport transformation matrix. [2018 spring]
33. Determine window to viewport transformation matrix for window (5,10), (15,20) and for viewport (8,12) (12,18). Note the coordinates values are for lower left and upper right corner. [2019 Fall]
34. Why do you need clipping? Explain the Cohen Sutherland line clipping algorithm. [2019 Fall/Spring]
35. Explain 2D transformation using Homogenous Coordinate system. [2019 Spring]
36. Find the transformation matrix for window to viewport transformation. [2020 Fall]
37. What is windowing and clipping; how a polygon can be clipped? explain.
38. Rotate a triangle A(5,5) B(10,10) and C(6,4) by 45 in CW and scale to half of its original size about origin.[2020 spring]
39. What is the importance of window and viewport transformation, explain with its practical application? Why we used to homogenous coordinate system in transformation, is it true that without homogenous coordinate system we could not have the final transformation matrix while performing animation? [2020 spring]
40. Why clipping is necessary in computer graphics? Clip the given polygon

    using Sutherland-Hodgeman Algorithm and explain every step considered.



Exam questions:

41. Prove that the successive translation and rotation is additive. [2021 Fall]
42. Explain the role of composite transformation in 2D/3D geometric transformation. Explain viewing pipelining in 2D. [2021 fall]
43. What is 3D transformation? Rotate the triangle A(0,0), B(2,2) C(4,2) about the origin by an angle of 45 degree.
44. Explain the Sutherland Hodgeman polygon clipping algorithm considering the four different cases. [2021 spring]
45. Let R be the rectangular window whose lower left hand corner is at L(-3, 1) and upper right hand corner is at R(2,6). Use Cohen-Sutherland algorithm to clip the line segments A(-4,2) and B(-1,7).
46. Derive a composite transformation matrix for reflecting an object about a line y=x+4 in 2D.
47. Differentiate between window and viewport. Derive a matrix for window to viewport transformation.
48. Prove that: i) Two successive translation are additive. ii) two successive scaling are multiplicative [2022 fall]
49. Window port is given by (100,100,300,300) and viewport is given by (50,50,150,150). Convert the window port coordinate (200,200) to the viewport coordinate.
50. A mirror is placed vertically such that it passes through the points (5,0) and (0,5). Find the related view of triangle ABC with coordinates A(5,30), B(30,50) and C(20,60). [2022 Fall]

End of Chapter