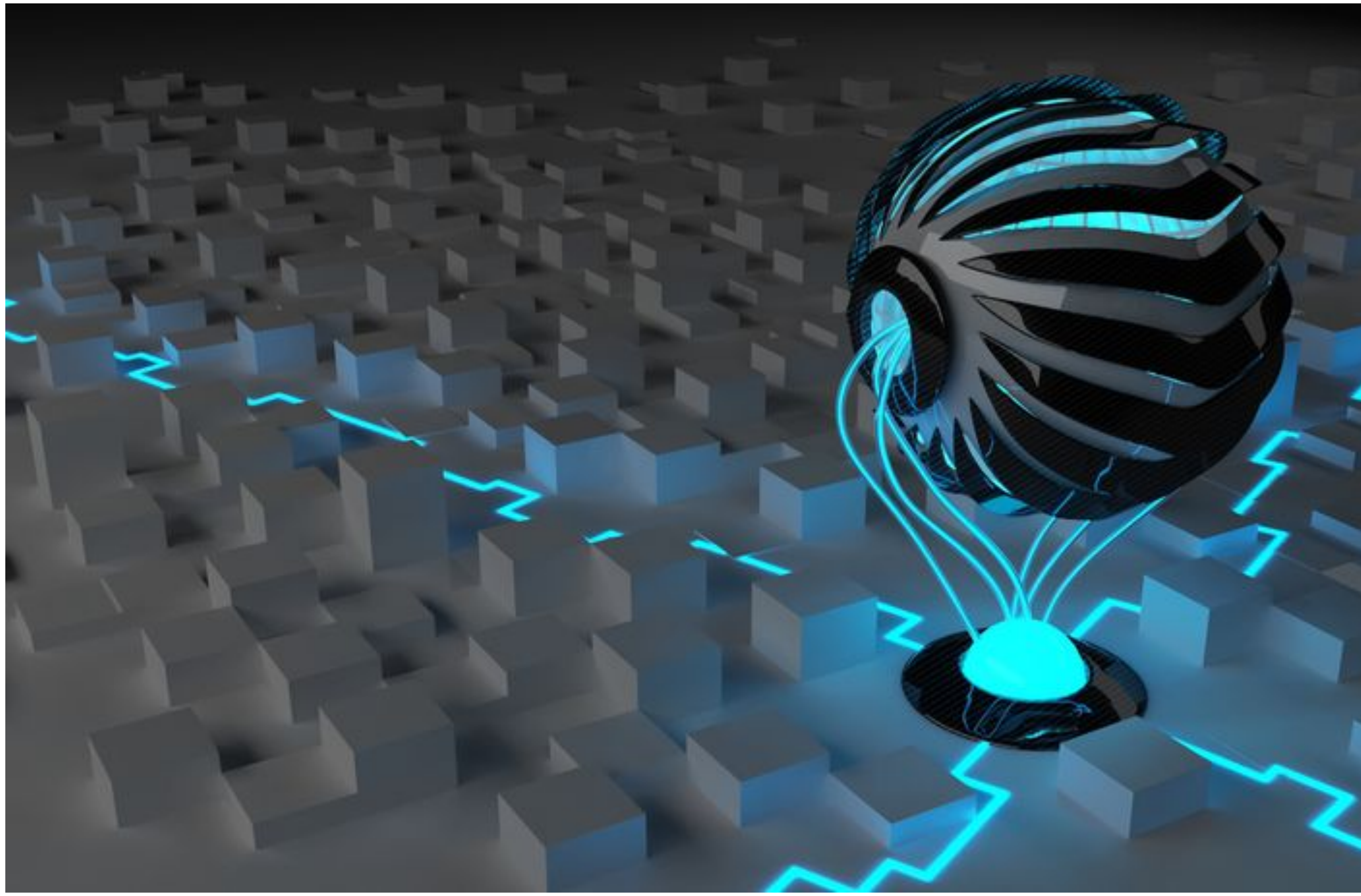
Computer Graphics and Visualization

Chapter 3

2D and 3D Coordinate Systems and Viewing Transformations



Prepared by:

Asst. Prof. Sanjivan Satyal

3. 2D and 3D Coordinate Systems and Viewing Transformations [9 hours]

- ❑3.1. Two –dimensional Transformation: Translation, Rotation, Scaling, Reflection, She transforms
- ❑ 3.2. Two-dimensional composite transformation
- ❑3.3. Window-to-Viewport coordinate transformation 3.4. Three–dimensional Display Methods
- ❑ 3.4.1. Parallel Projection
- ❑ 3.4.2. Perspective Projection
- ❑3.5. Three –dimensional Transformation: Translation, Rotation, Scaling, Reflection, She transforms
- ❑3.6. Three-dimensional Composite Transformation
- ❑ 3.7. Projection and Viewing transformation

Transformation

- Transformations are fundamental part of computer graphics
- In order to manipulate object in two dimensional space, we must apply various transformation functions to object
- Transformation: changing co-ordinate description of an object
- In computer graphics, transformations of 2D objects are essential to many graphics applications
- Many applications use the geometric transformations to change the position , orientation, and size or shape of the objects in drawing

Transformation

- The three basic transformations are:
  1. **Translation**
  2. **Rotation**
  3. **Scaling**

Other transformation:

1. **Reflection**
2. **Shearing**

Types:

Rigid Body Transformations:

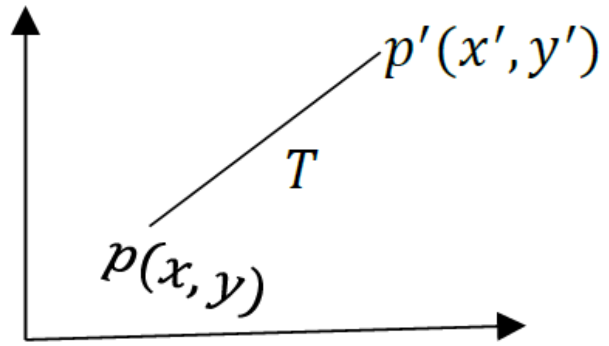
- Preserve distances and angles.
- Do not change the shape or size of the object.
- Include translation, rotation, and reflection or any sequence of these.

Non-Rigid Transformations:

- Do not necessarily preserve distances and angles.
- Can change the shape or size of the object.
- Include scaling, non-uniform scaling, and general affine transformations.

Translation

- Repositioning of object along a straight-line path from one coordinate location to another is called translation
- Translation is performed on a point by adding offset to its coordinate so as to generate a new coordinate position
- Let **p(x, y)** be translated to $p'(x', y')$ by using offset t_x and t_y in x & y direction. Then,
- $x' = x + t_x$
- $y' = y + t_y$
- the pair **(t_x , t_y)** is called the **translation vector**

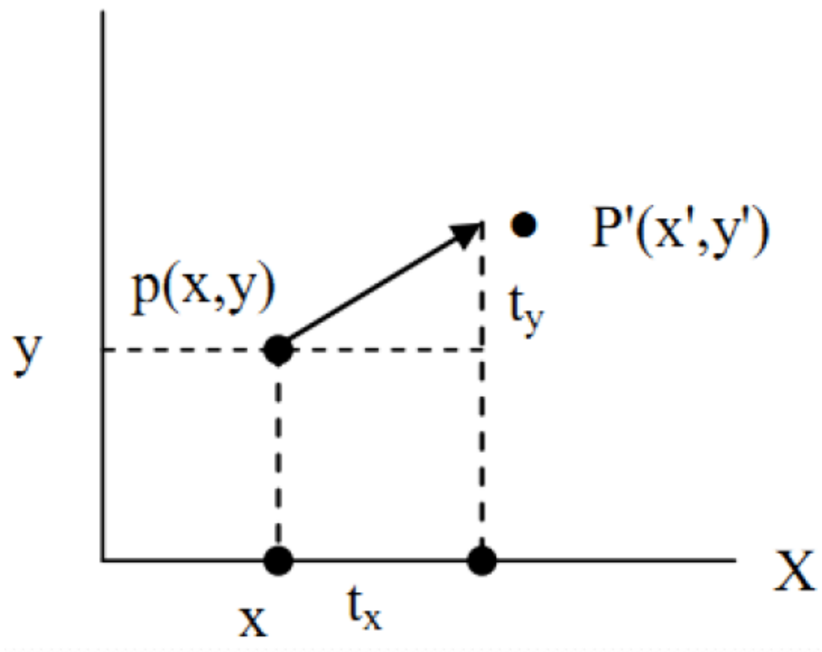


- In matrix form,
- i.e. $P' = P + T$ where T is transformation matrix

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

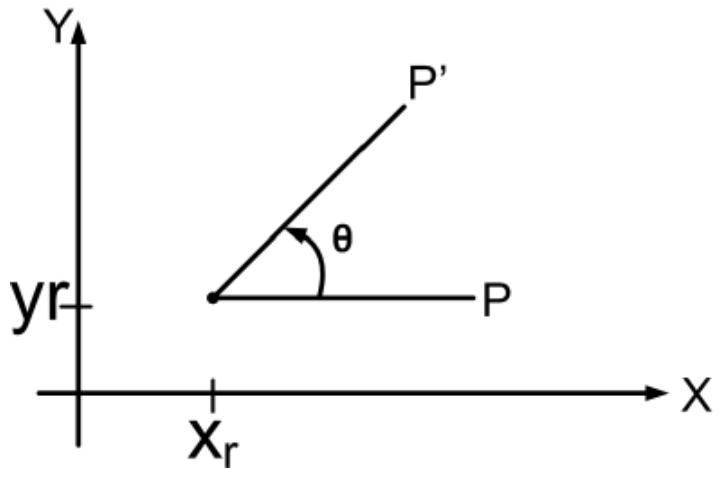
$$\therefore P' = P + T$$

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



Rotation

- Changing the co-ordinate position along a circular path is called rotation
- 2D rotation is applied to re-position the object along a circular path in XY-plane
- Rotation is generated by specifying rotation angle (θ) and pivot point (rotation point)
  - **+** value for 'θ' define *counter-clockwise* rotation about a point
  - **-** value for 'θ' define *clockwise* rotation about a point



Rotation (Rotation from Origin)

- Let $p(x, y)$ be a point rotated by θ about origin to new point $p'(x', y')$

$x' = rcos(\varnothing + \theta)$

$= rcos\varnothing cos\theta - rsin\varnothing sin\theta$

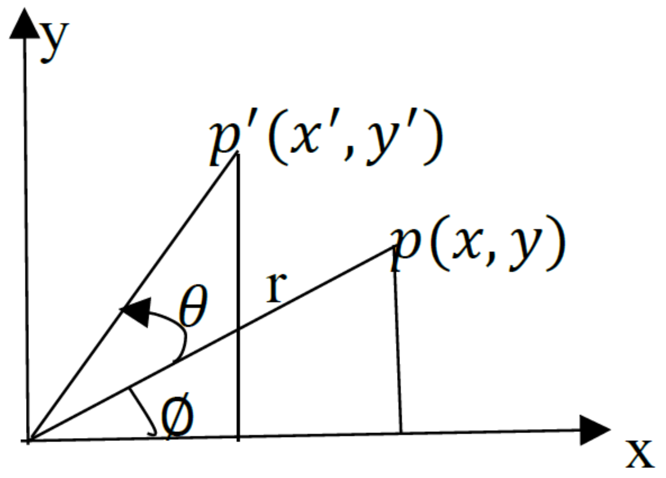
- But $x = rcos\varnothing$ & $y = rsin\varnothing$

$\therefore \boldsymbol{x' = xcos\theta - ysin\theta}$ .... (i)

Similarly,

$\therefore \boldsymbol{y' = xsin\theta + ycos\theta}$ .... (ii)

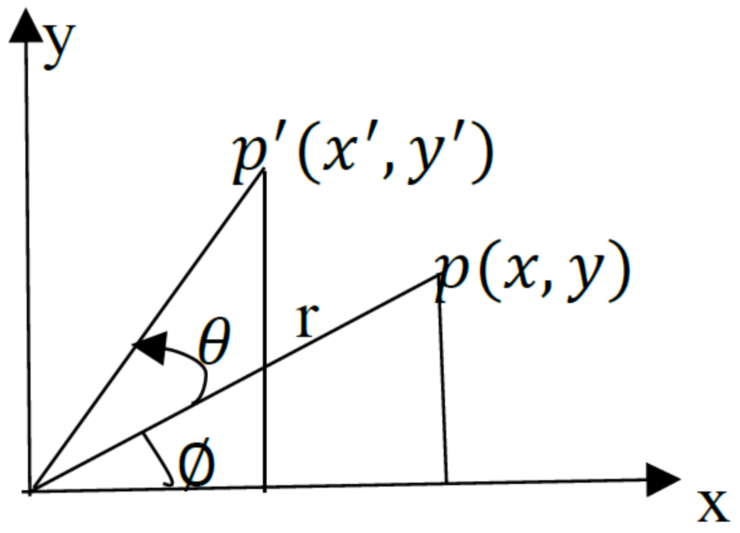
which are equation for rotation of (x, y) with angle θ and taking pivot as origin



- **Rotation (Rotation from Origin)**

$\boldsymbol{x}' = \boldsymbol{x}\boldsymbol{cos}\theta - \boldsymbol{y}\boldsymbol{sin}\theta$ .... (i)

$\boldsymbol{y}' = \boldsymbol{x}\boldsymbol{sin}\theta + \boldsymbol{y}\boldsymbol{cos}\theta$ .... (ii)



- In matrix form
- $P' = R.P$

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

2D Rotation (Rotation from any Point)

- If the arbitrary fixed pivot point is at$(\boldsymbol{x_r}, \boldsymbol{y_r})$

$\cos(\varnothing + \theta) = (x' - x_r)/r$

or, $r\cos(\varnothing + \theta) = (x' - x_r)$

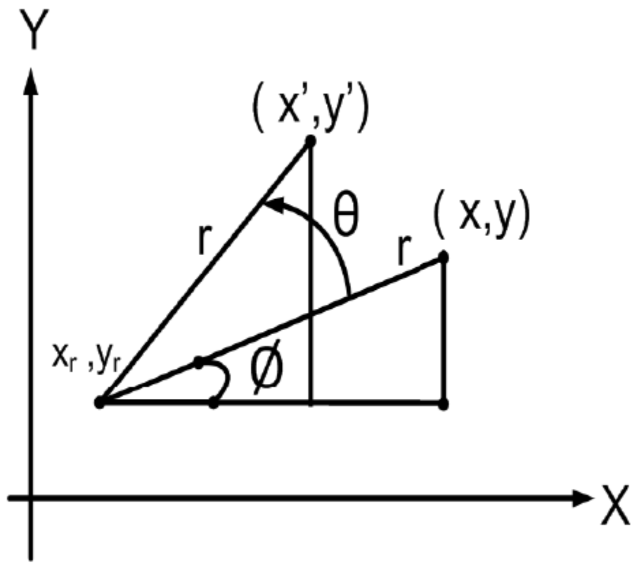
or, $(x' - x_r) = r\cos\varnothing\cos\theta - r\sin\varnothing\sin\theta$

Since,

$r\cos\varnothing = (x - x_r)$,

$r\sin\varnothing = (y - y_r)$

$\therefore \boldsymbol{x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta}$ .... (i)



Rotation

2D Rotation (Rotation from any Point)

Similarly,

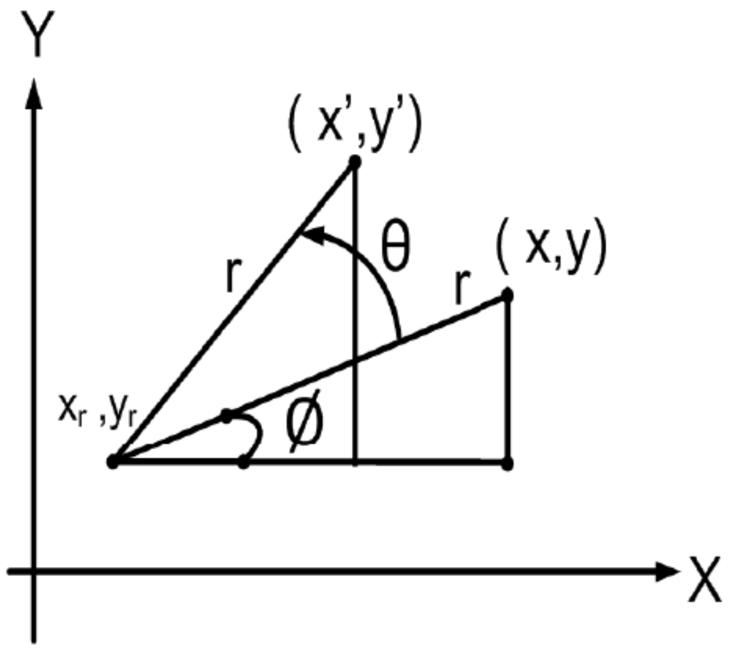
$\sin(\varnothing + \theta) = (y' - y_r)/r$

or, $r \sin(\varnothing + \theta) = (y' - y_r)$

or, $(y' - y_r) = r\sin\varnothing\cos\theta + r\cos\varnothing\sin\theta$

Since, $r\cos\varnothing = (x - x_r)$, $r\sin\varnothing = (y - y_r)$

$\therefore \boldsymbol{y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta}$ ……… (ii)



- These equations (i) and (ii) are the equations for rotation of a point (x, y) with angle θ taking pivot point $\mathbf{(x_r , y_r)}$.

- The transformation can be represented in matrix form as:
- $\boldsymbol{x'} = \boldsymbol{x_r} + (\boldsymbol{x} - \boldsymbol{x_r})\boldsymbol{cos}\theta - (\boldsymbol{y} - \boldsymbol{y_r})\boldsymbol{sin}\theta$ ……… (i)
- $\boldsymbol{y'} = \boldsymbol{y_r} + (\boldsymbol{x} - \boldsymbol{x_r})\boldsymbol{sin}\theta + (\boldsymbol{y} - \boldsymbol{y_r})\boldsymbol{cos}\theta$ ……… (ii)

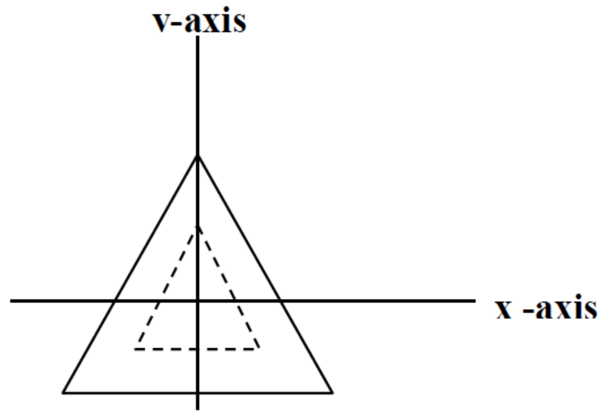
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x_r \\ y_r \end{pmatrix} + \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x - x_r \\ y - y_r \end{pmatrix}$$

Scaling

- Scaling transformation alters the size of object
- Scaling is the process of expanding or compressing the dimension of an object
- A simple two dimensional scaling operation is performed by multiplying object position (x, y) with scaling factors s_x & s_y along x & y direction to produce (x', y')

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

- Scaling factor ***S_x*** scales object in x-direction and ***S_y*** scales in y-directions



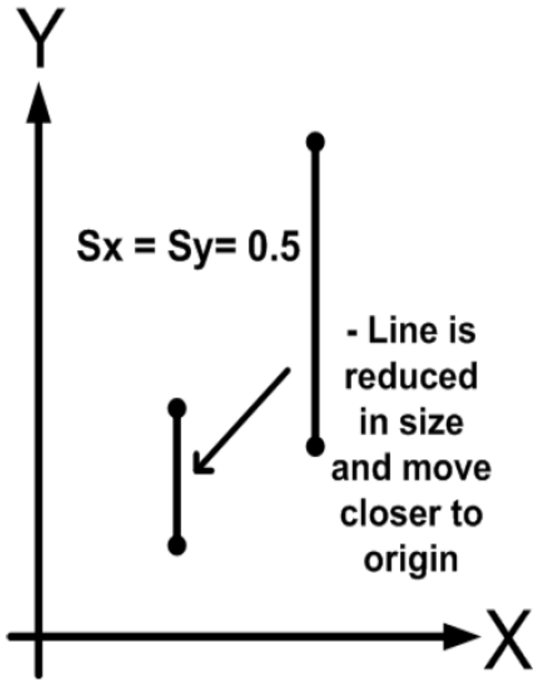
In matrix form,

$P' = S.P$

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

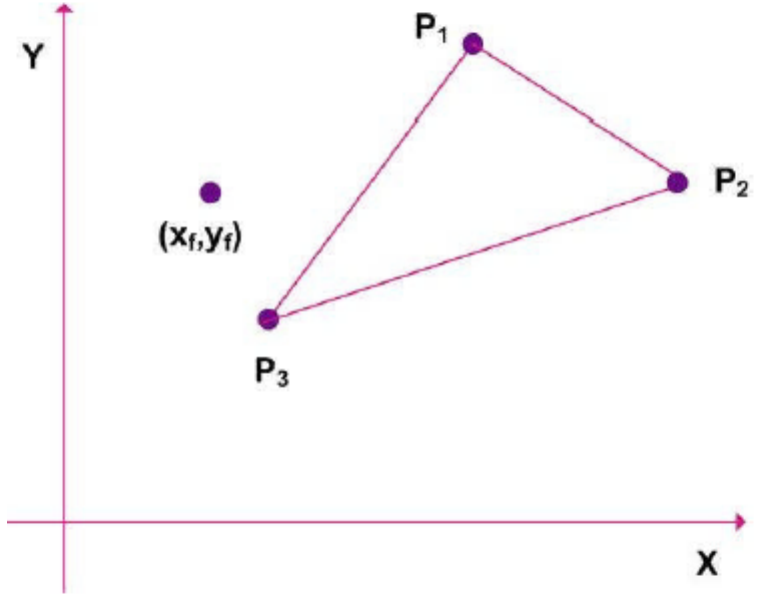
- If both scaling factors have same value: **uniform scaling**
- If the value of s_x and s_y are different: **differential scaling**
- The differential scaling is mostly used in the graphical package to change the shape of the object

- If **scaling factor < 1,**
  - size of object is **decreased**
  - and move the object closer to the coordinate origin
- If **scaling factor > 1**
  - size of object is **increased**
  - move farther from the origin
- The **scaling factor = 1** for both direction does not change the size of the object



- **For Scaling about an arbitrary fixed pivot point** (x_f, y_f)
- An object is scaled relative to the fixed point by scaling the
- distance from each vertex to the fixed point
- If fixed point is (x_f , y_f) about which scaling is made, then

$$x' = x_f + (x - x_f).s_x,$$
$$y' = y_f + (y - y_f).s_y$$



- We can write this equation to separate the multiplicative and additive terms

$$x' = x.s_x + x_f(1 - s_x)$$
$$y' = y.s_y + y_f(1 - s_y)$$

- Additive terms $\mathbf{x_f(1\text{-}s_x)}$ and $\mathbf{y_f(1\text{-}s_y)}$ are constant for all points in the object

Scaling, fixed pivot point (x_f , y_f)

- **For Scaling about an arbitrary fixed pivot point** (x_f , y_f)

$$x' = x.s_x + x_f(1 - s_x)$$
$$y' = y.s_y + y_f(1 - s_y)$$

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

P' **=** S**.** P **+ C**

- C has additive terms **$x_f(1\text{-}s_x)$** and **$y_f(1\text{-}s_y)$** which are constant for all points in the object

Review Matrix Representation

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

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

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

Classwork

1. **Translate the triangle with vertices A(10, 10), B(15, 15), and C(20, 10) three units to the right and four units upwards.**
2. **Rotate the endpoints (3, 4) and (8, 9) about the origin in the anti-clockwise direction**
3. **Perform the scaling transformation on the triangle with vertices P(6, 9), Q(10, 5), and R(4, 3) with scaling factors Sx = 3 and Sy = 2**

Classwork

1. **Translate the triangle with vertices A(10, 10), B(15, 15), and C(20, 10) three units to the right and four units upwards.**

   A'(13, 14), B'(18, 19), and C'(23, 14)

2. **Rotate the endpoints (3, 4) and (8, 9) about the origin in the anti-clockwise direction**

   (0, 3), (-9, 8)

3. **Perform the scaling transformation on the triangle with vertices P(6, 9), Q(10, 5), and R(4, 3) with scaling factors Sx = 3 and Sy = 2**

   P'(18, 18), Q'(30, 10), and R'(12, 6)

Composite Transformation

- ❑Many graphics applications involve sequences of geometric transformations
  - ❑Animations design and picture construction applications
- ❑Composite transformations in computer graphics involve applying multiple geometric transformations
- ❑It include sequences of translations, rotations, or scaling
- ❑Using matrix representation, create a composite transformation matrix by multiplying individual transformation matrices

- ❑ Allow multiple geometric transformations to be combined into a single matrix operation, which can be used to transform objects in a single step

- ❑ **$P' = M_2 . M_1 . P = M . P$**

- ❑ This is especially efficient and powerful when working with homogeneous coordinates

- ❑ For column matrix representation of coordinates, multiply matrices from right to left

- ❑ The order of multiplication is important because matrix multiplication is not commutative

Composite Transformations

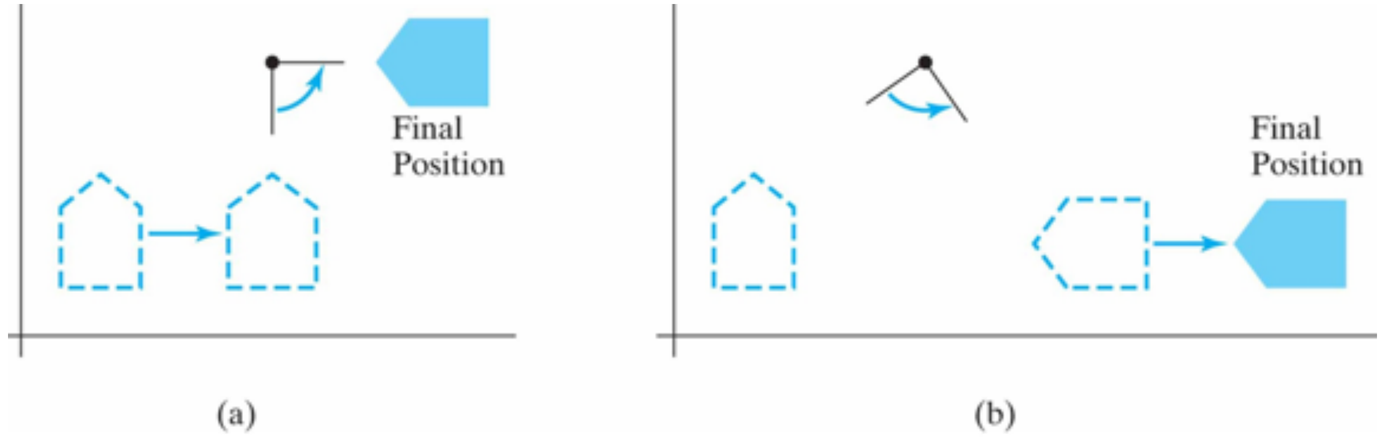
- Matrix Concatenation Properties:

- Matrix multiplication is associative
  - **$M3 \cdot M2 \cdot M1 = (M3 \cdot M2) \cdot M1 = M3 \cdot (M2 \cdot M1)$**

- Matrix multiplication is not commutative
  - **$M2 \cdot M1 \neq M1 \cdot M2$**

Reversing the order

In which a sequence of transformations is performed may affect the transformed position of an object.



In (a), an object is first translated in the x direction, then rotated counterclockwise through an angle of 45°.

In (b), the object is first rotated 45° counterclockwise, then translated in the *x* direction

Matrix representation & Homogeneous coordinate

- ❑The homogeneous co-ordinate system provides a uniform framework for handling different geometric transformations including translations, rotations, scaling, and perspective projections

- ❑All geometrical transformation equations can be represented as matrix multiplications

- ❑They are used to perform more than one transformation at a time

- ❑They reduce unwanted calculations, intermediate steps, saves time and memory and produce a sequence of transformations

Matrix representation & Homogeneous coordinate

- We represent each Cartesian coordinate position *(x, y)* with the homogeneous coordinate triple (x_h , y_h , h)

  - $x = x_h / h$
  - $y = y_h / h$

- h is 1 usually for 2D case

- Therefore, **(x, y)** in Cartesian system is represented as **(x, y, 1)** in homogeneous co-ordinate system

Matrix representation & Homogeneous coordinate

- ❑If **h** is more than 1, it scales all the coordinate values by h
- ❑Consider a point (2,3) in Cartesian coordinates.
- ❑ homogeneous coordinates with h=1 are (2,3,1)
- ❑ homogeneous coordinates with h=2, would be (4,6,2)

Homogeneous Matrix

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

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

Rotation

a. Counter Clock Wise (CCW): $\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \text{Cos}\theta & -\text{Sin}\,\theta & 0 \\ \text{Sin}\theta & \text{Cos}\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

b. Clock Wise(CCW): $\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \text{Cos}\theta & \text{Sin}\,\theta & 0 \\ -\text{Sin}\theta & \text{Cos}\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

For Translation: $\boldsymbol{T}(\boldsymbol{t}_x, \boldsymbol{t}_y)$

- The matrix representation in homogeneous coordinate is

$$P' = T.P$$

Where, $P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$, $T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$ & $P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

By which we can get,

$x' = x + t_x$

$y' = y + t_y$

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

For Rotation: R(θ)

- The matrix representation in homogeneous coordinate is

$$P' = R.P$$

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

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

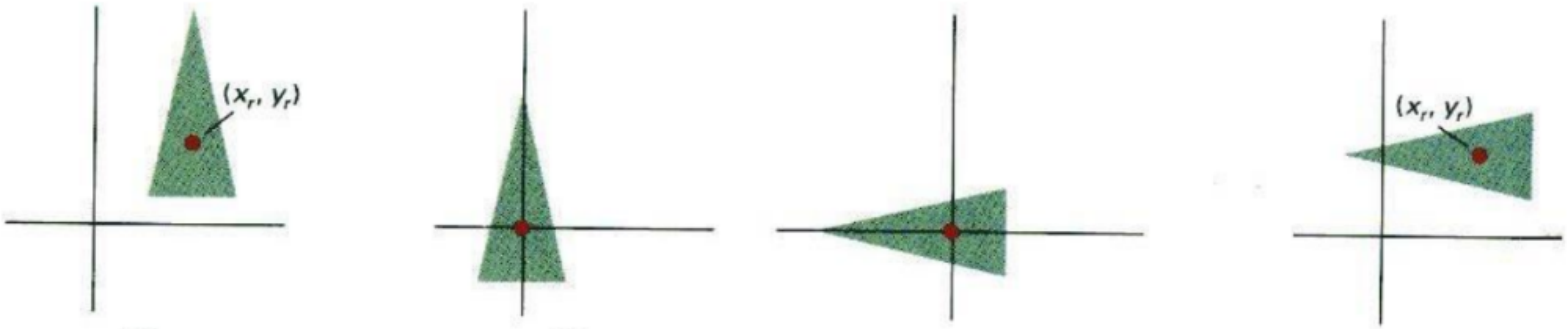
$$\boldsymbol{x' = xcos\theta - ysin\theta}$$
$$\boldsymbol{y' = xsin\theta + ycos\theta}$$

For Rotation about an arbitrary fixed pivot point (x_r, y_r)

- Using a graphics package that only provides a rotation function about the coordinate origin, we can achieve rotations about any arbitrary point by :
    1. Translate fixed point(x_r, y_r) to the co-ordinate origin by $T(-x_r, -y_r)$
    2. Rotate with angle $\rightarrow R(\theta)$
    3. Translate back to original position by $T(x_r, y_r)$
- This composite transformation is represented as

$$P' = T(x_r, y_r).\ R(\theta).\ T(-x_r, -y_r).P$$

Matrix representation & Homogeneous coordinate



(a)
Original Position of Object and Pivot Point

(b)
Translation of Object so that Pivot Point (x_r, y_r) Is at Origin

(c)
Rotation about Origin

(d)
Translation of Object so that the Pivot Point Is Returned to Position (x_r, y_r)

For Rotation about an arbitrary fixed pivot point (x_r,y_r)

- The matrix representation in homogeneous coordinate is

$$P' = T(x_r, y_r).\ R(\theta).\ T(-x_r, -y_r).P$$

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

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

For Scaling with scaling factors $(\boldsymbol{s_x}, \boldsymbol{s_y})$

- The matrix representation in homogeneous coordinate is

$P' = S.P$

Where, $P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$, $S = \begin{bmatrix} \boldsymbol{s_x} & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ & $P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

This gives the equations,
$x' = x.s_x$ & $y' = y.s_y$

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

For Scaling about an arbitrary fixed pivot point (x_f , y_f)

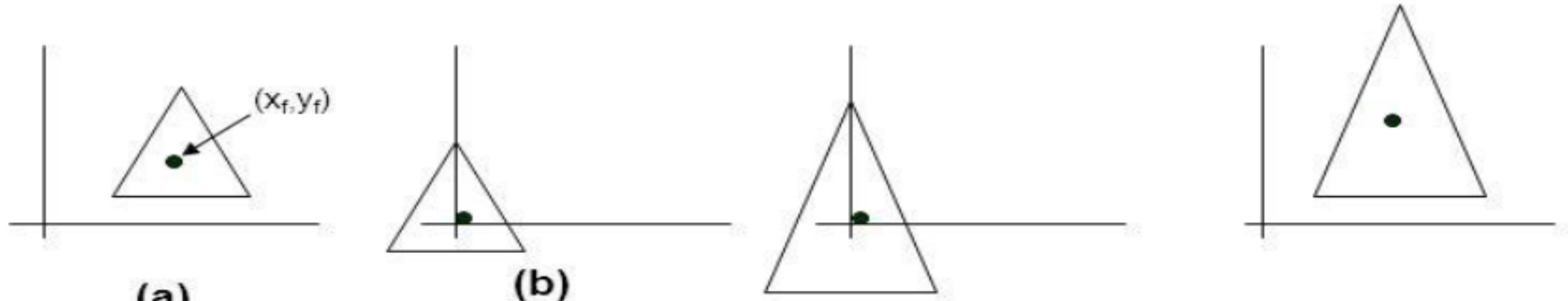
For a given fixed point (x_f , y_f)

1. Shift the scaling point to origin

2. Scale with respect to coordinate origin

3. Restore

   - ❑ Inverse the translation of step 1 to return to the original position



(a) Original Position of Object and Fixed point

(b) Translation of object so that fixed point (x_f, y_f) is at origin

(c) scaling was about origin

(d) Translation of the object so that the Fixed point is returned to position (x_f, y_f)

Matrix representation & Homogeneous coordinate

- This composite transformation is represented as:

$$P' = T(x_f, y_f).\ S(S_x, S_y).\ T(-x_f, -y_f).P$$

- The matrix representation in homogeneous coordinate:

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

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

Matrix representation & Homogeneous coordinate

- The combined transformation matrix for scaling about an arbitrary point (x_f , y_f) is:

$$\mathbf{M} = \begin{pmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{pmatrix}$$

- This matrix can be used to transform any point (x,y) by first converting it to homogeneous coordinates (x,y,1), then multiplying by the matrix M to get the scaled point

Composite translation

- If two successive translation vector (tx_1, ty_1) & (tx_2, ty_2) is applied on position $p(x, y)$, then translated point is given by,

$$P' = \{T(tx_2, ty_2)\ T(tx_1, ty_1)\}P$$

$$= \begin{bmatrix} 1 & 0 & tx_2 \\ 0 & 1 & ty_2 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & tx_1 \\ 0 & 1 & ty_1 \\ 0 & 0 & 1 \end{bmatrix} {}^*P$$

$$= \begin{bmatrix} 1 & 0 & tx_1 + tx_2 \\ 0 & 1 & ty_1 + ty_2 \\ 0 & 0 & 1 \end{bmatrix} {}^*P$$

$$P' = T(tx_1 + tx_2, ty_1 + ty_2)\ {}^*P$$

Composite translation

$$P' = T(tx_1 + tx_2, ty_1 + ty_2) * P$$

- Here P and P’ are column vector of final and initial point coordinate respectively

- This shows that two ***successive translation is additive***

- This concept can be extended for any number of successive translations

Composite rotation

- If two successive rotation vector R(θ_1) and R(θ_2) are applied to a position $p(x, y)$, then transformed point P' is given by,

$$P' = \left(R(\theta_2)\right).\left(R(\theta_1)\right).P$$

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

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

Composite rotation

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

$$= \mathrm{R}\,(\theta_1 + \theta_2).\,P$$

- Here P and P' are column vector of final and initial point coordinate respectively
- This shows that two ***successive rotation is additive***
- This concept can be extended for any number of successive rotation

Composite Scaling

- If two successive scaling vector S(s_{x1}, s_{x2}) and S(s_{x1}, s_{x2}) are applied to a position P(x, y), then transformed point P' is given by,

$$P' = \mathbf{S}\left(s_{x_2}, s_{y_2}\right) \cdot \mathbf{S}\left(s_{x_1}, s_{y_1}\right) . P$$

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

Composite Scaling

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

❑Here P and P’ are column vector of final and initial point coordinate respectively

❑This shows that two ***successive scaling is multiplicative***

❑This concept can be extended for any number of successive scaling

Review Composite Matrix

Composite Translation $= \begin{bmatrix} 1 & 0 & tx_1 + tx_2 \\ 0 & 1 & ty_1 + ty_2 \\ 0 & 0 & 1 \end{bmatrix}$

Composite Rotation $= \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Composite Scaling $= \begin{bmatrix} s_{x1*}s_{x2} & 0 & 0 \\ 0 & s_{y1*}s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Assignment

1.Find out the final coordinates of a figure bonded by the coordinates (1,1) (3,4) (5,7) and (10,3) when rotated about a point (8,8) by 30 degree in counter clockwise direction and scaled by two unit x-direction, three unit y-direction.

2. A triangle having vertices A(3, 3), B(8, 5) & C(5, 8) is first translated by 2 units then scaled twice its size about fixed point (5, 6) & finally rotated 90 degree anticlockwise about pivot point (2, 5). Find the final position of triangle.

3. A triangle having vertices A(3, 3), B(8, 5) & C(5, 8) is first translated by 2 units about fixed point (5, 6) & finally rotated 90 degree anticlockwise about pivot point (2, 5). Find the final position of triangle

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

5. Reflect a triangle with vertices A(2,3), B(6,3) and C(4,8) about the line y=3x+4y

Classwork

- ***Q1. Find the scaled triangle with vertices A(0, 0), B(1, 1) & C(5, 2) after it has been magnified twice its size***
- ***Q2. Rotate a triangle A(0, 0), B(2, 2), C(4, 2) about the origin by the angle of 45 degree.***
- ***Q3. Rotate a triangle (5, 5), (7, 3), (3, 3) about fixed point (5, 4) in counter clockwise by 90 Degree***

Classwork

- ***Q. Find the scaled triangle with vertices A(0, 0), B(1, 1) & C(5, 2) after it has been magnified twice its size***
  - A'(0,0), B'(2,2), C'(10,4).
- ***Rotate a triangle A(0, 0), B(2, 2), C(4, 2) about the origin by the angle of 45 degree.***
- A'(0,0), B'(0,2√2), C'(√2, 3√2).
- ***Q. Rotate a triangle (5, 5), (7, 3), (3, 3) about fixed point (5, 4) in counter clockwise by 90 Degree***
- (4, 4), (6, 6), (6, 2).

Classwork

- ***Q4. A square with vertices A(0, 0), B(2, 0), C(2, 2) & D(0, 2) is scaled 3 units in x & y direction about the fixed point (1, 1). Find the coordinates of the vertices of new square.***

Classwork

- ***Q. A triangle having vertices A(3, 3), B(8, 5) & C(5, 8) is first translated by 2 units then scaled twice its size about fixed point (5, 6) & finally rotated 90 degree anticlockwise about pivot point (2, 5). Find the final position of triangle.***

- **Step 1 Translation**
  - T(2,2)
- **Step 2 Scaling**
  - T(5,6)
  - S(2,2)
  - T(-5,-6)
- **Step 3 Rotation**
  - T(2,5)
  - R(90)
  - T(-2,-5)

The composite matrix is given by

$M = R.\ S.\ T$

Now, the transformation points are;

$A' = M.\ A$ = (3, 8)

$B' = M.\ B$ = (-1, 18)

$C' = M.\ C$ = (-7, 12)

Classwork

Q. A triangle with A(5,8), B(2,1) and C(8,1) is to be enlarged twice its initial size about the fixed point (5,6). Find its final co-ordinates as per the following transformations:

i. ***After translation by T1(4,3) and T2(-1,2)***

ii. ***After rotation by an angle of 60 degree in clockwise and 120 degree in anti-clockwise direction.***

iii. ***After scaling with the scaling factor S1(2,3) and S2(2,2).***

Composite Transformation

Q. Rotate a triangle A(7, 15), B(5, 8) & C(10, 10) by 45 degree clockwise about origin and scale it by (2, 3) about origin.

Q. A square with vertices A(0, 0), B(2, 0), C(2, 2) & D(0, 2) is scaled 2 units in x & y direction about the fixed point (1, 1). Find the coordinates of the vertices of new square.

Q. A triangle having vertices A(3, 3), B(8, 5) & C(5, 8) is first translated by 2 units about fixed point (5, 6) & finally rotated 90 degree anticlockwise about pivot point (2, 5). Find the final position of triangle

Q. Rotate the △ABC by 90° anti-clock wise about (5, 8) and scale it by (2, 2) about (10, 10).

Other transformation

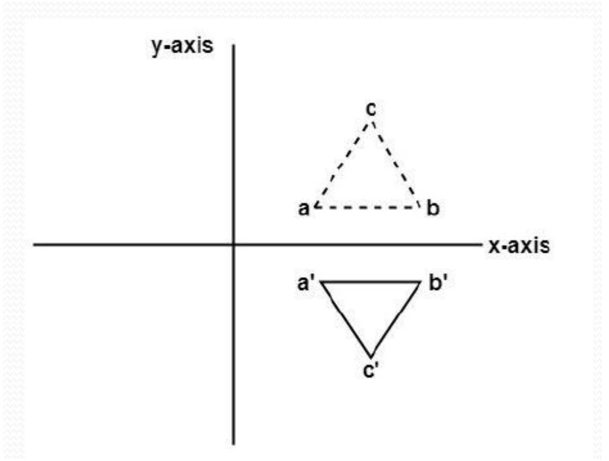
1. **Reflection**
2. **Shearing**

Reflection

- Providing a mirror image about an axis of an object is called reflection
- Equivalent to rotation of an object about the reflection axis
- Each point and its image are the same distance from the reflection axis

Reflection

- The mirror image for a two-dimensional reflection is generated relative to an axis of reflection by rotating the object 180 degrees about the reflection axis
- In reflection, the size of the object does not change
- There are 3 basic types of reflections:
  - **About the x-axis**
  - **About the y-axis**
  - **About both (x, y) axes**



Reflection

- **Reflection about x-axis (y=0)**

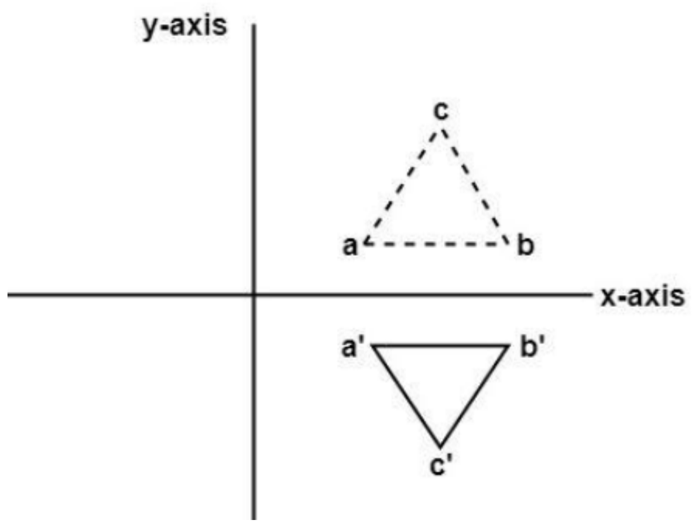
- The reflection of a point (P(x, y) on the x-axis changes the y-coordinate sign, i.e., **P(x, y)** changes to **P'(x, -y)**

- In matrix form,

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

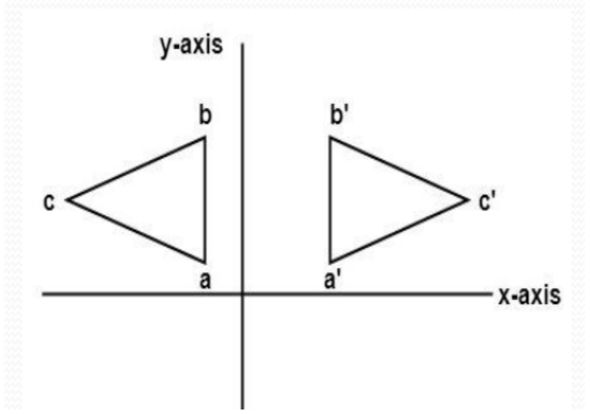
- $P' = Rf_x \cdot P$

- R_{fx}= Reflection matrix about x-axis



- **Reflection about y-axis (x=0)**
- The reflection of a point (P(x, y) on the y-axis changes the y-coordinate sign, i.e., **P(x, y)** changes to **P'(-x, y)**
- In matrix form,

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



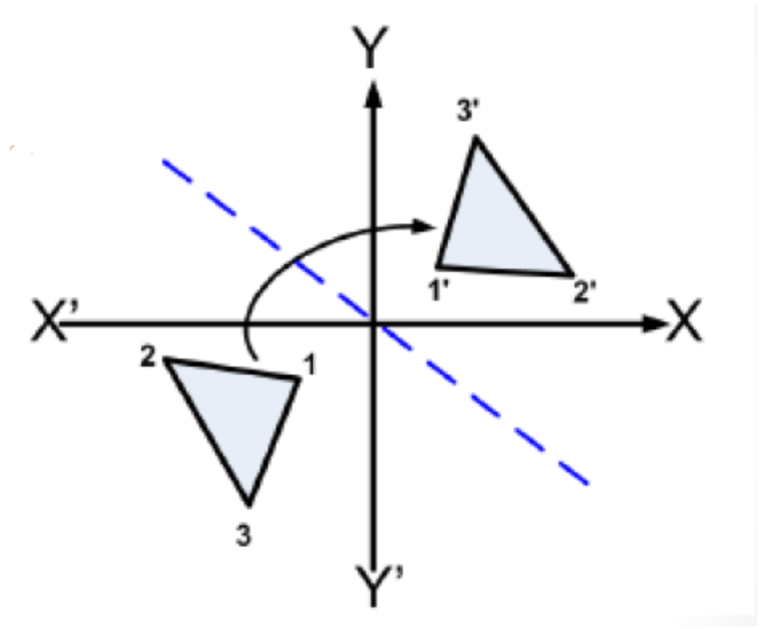
- $P' = Rf_{\mathbf{y}} \,.\, P$
- R_{fy}= Reflection matrix about y-axis

Reflection

- **Reflection about origin**
- The reflection of a point (P(x, y) about origin changes the x and y-coordinate sign, i.e., **P(x, y)** changes to **P’(-x, -y)**
- In matrix form

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

- $P' = Rf_o \,.\, P$
- R_{fo}= Reflection matrix about origin

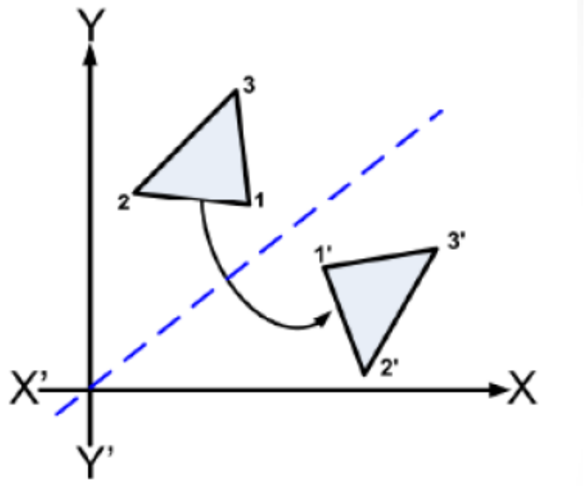


Reflection

- **Reflection about y=x line**
- First, the object is rotated 45° clockwise.
- After that, reflection is done about **the x-axis**
- The last step is rotating the object back to its original position, which is a 45° counter clockwise rotation

OR

- Reflection about x-axis
- Rotate anticlockwise **90˚**



Reflection

- **Reflection about y=x line**
- In matrix form,

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

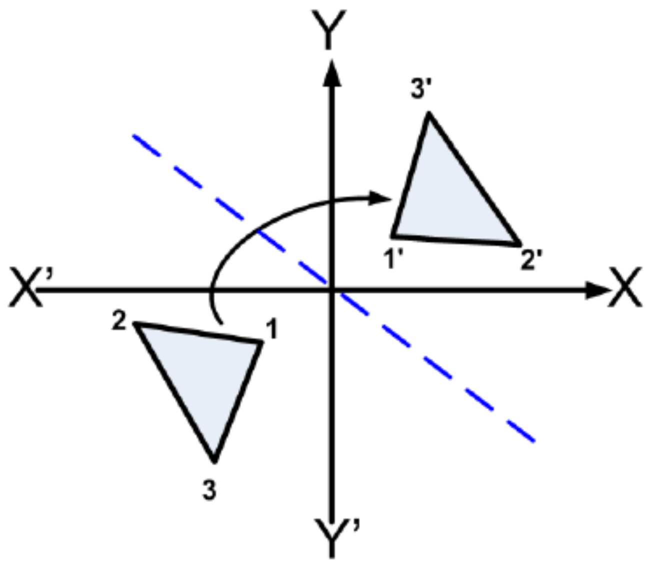
- $P' = Rf_{y=x} * P$
- $Rf_{y=x}$=Reflection matrix about $_{y=x}$ line

Reflection

- **Reflection about y= - x line**
- First, the object is rotated 45° clockwise
- After that, reflection is done about the **y-axis**
- The last step is rotating the object back to its original position, which is a 45° anti clockwise Rotation

OR

- Reflect the object about the y-axis.
- Rotate the object 90° clockwise.



Reflection

- **Reflection about y=-x line**
- In matrix form,

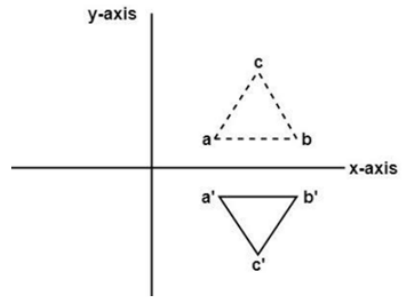
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- P'**=**$Rf_{\mathbf{y=-x}}$*****P
- $Rf_{y=-x}$=Reflection matrix about $_{y=-x}$ line

Reflection Matrix Review

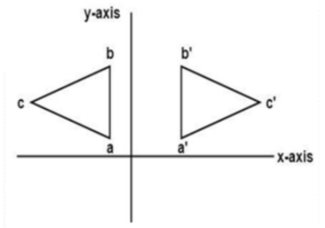
Reflection about x-axis (y=0)

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



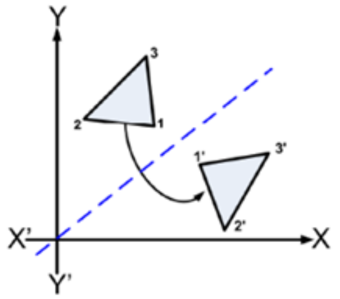
Reflection about y-axis (x=0)

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



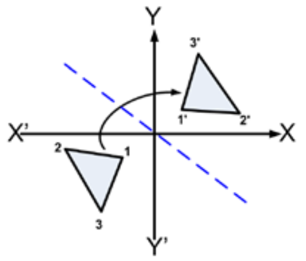
Reflection about y=x line

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



Reflection about y=-x line

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



Classwork (Reflection)

- Reflect a line segment having endpoints (9,3) and (12,10) about a line Y=7. Draw initial and final result graph as well.
- Reflect a line segment having end points (9,3) and (12,10) about a line X=7. Draw initial and final result graph as well.
- Translate a triangle ABC with co-ordinates A(0, 0), B(5, 0) and C(5, 5) by 2 units in x-direction and 3 units in y-directions.

Reflection

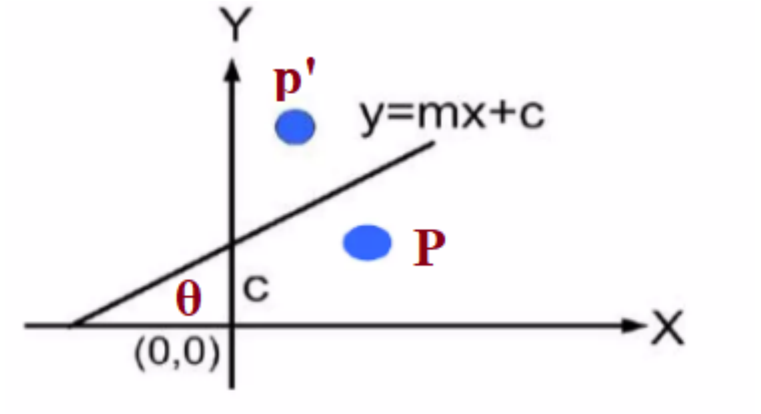
- **Reflection about y=mx+c line**

Perform the following transformation:

1. **Translate** the (0,c) to coincide with origin i.e. T(0,-c).
2. **Rotate** the given line about origin through an **clockwise direction** or angle (-θ) i.e R(-θ)

   so that it coincides with any coordinate axis
3. Apply **Reflection** in the x-axis i.e. R_f (y=0)

4. **Rotate** about the origin by **counter clockwise** angle θ i.e. R(θ)
5. **Translate** to the original position i.e. T(0,c)



Reflection

$$T = T(\mathbf{0},c).\ R(\theta).\ Rfx\ .\ R(-\theta).\ T(\mathbf{0},-c)$$

- Substituting value of tanθ, sinθ & cosθ we will get the reflection matrix

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

Reflection

$$T = T(0,c).\ R(\theta).\ Rfx\ .\ R(-\theta).\ T(0,-c)$$

Slope m=$tan\theta$

Also we have,

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

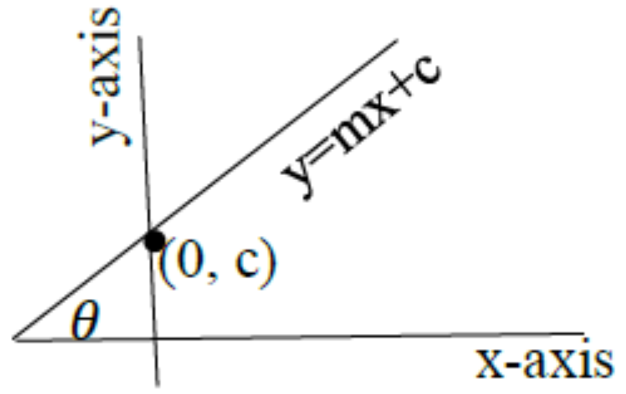
$$\therefore cos\theta = \frac{1}{\sqrt{m^2+1}}$$

Also we have,

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

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

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



Reflection

$$T = T_{(0,c)} . R_{(\theta)} . R_{fx} . R_{(-\theta)} . T_{(0,-c)}$$

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

$=$

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

$=$

Reflect a triangle with vertices A(2,3), B(6,3) and C(4,8) about the line y=3x+4y

- **Composite Transformation(M) =** $T(0,c) . R(\theta) . Rfx . R(-\theta) . T(0,-c)$

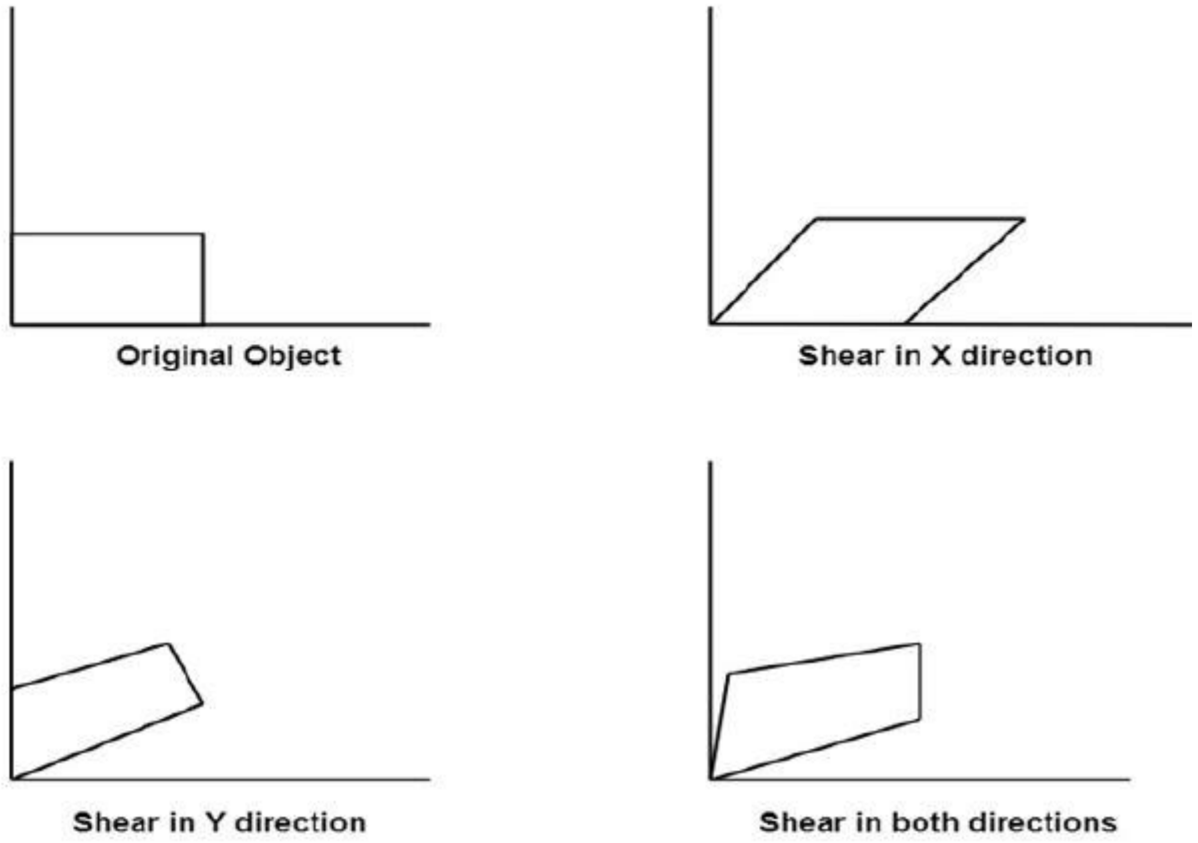
$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix} \qquad R_{\text{reflect}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{\text{clockwise}} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad T^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix}$$

2. Shearing

- ❑A transformation that distorts the shape of an object
- ❑Such that the transformed shape appears as if the object is composed of internal layers and these layers are caused to slide over is shearing
- ❑The shear can be in one direction or in two directions
- ❑Shearing factor (Sh_x, Sh_y)
- ❑Shear can occur in:
  - ❑X-direction
  - ❑Y-direction

Shearing


Original Object
Shear in X direction
Shear in Y direction
Shear in both directions

Shearing

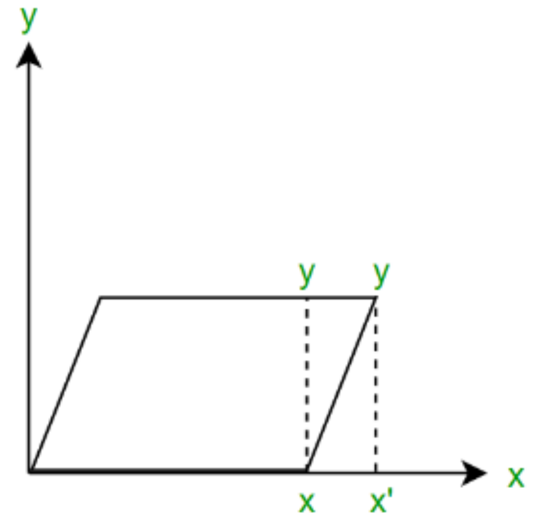
Shearing in X-direction

- The y co-ordinates remain the same but the x co-ordinates changes
- The transformation is given by the following matrix:

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

$$x' = x + sh_x * y$$

$$y' = y$$



Shearing

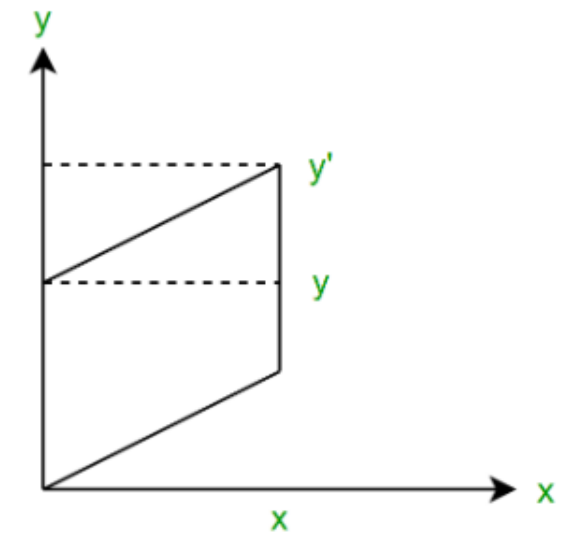
Shearing in Y-direction

- The x co-ordinates remain the same but the y co-ordinates changes.
- The transformation is given by the following matrix:

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

$$x' = x$$

$$y' = sh_y * x + y$$



Shearing

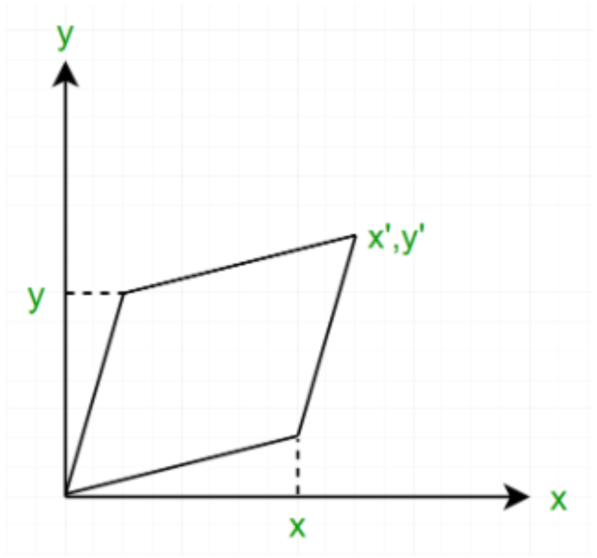
Shearing in both-direction

- In x-y shear, both the x and y co-ordinates changes.
- The transformation is given by the following matrix:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_x & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$x' = x + sh_x * y$$

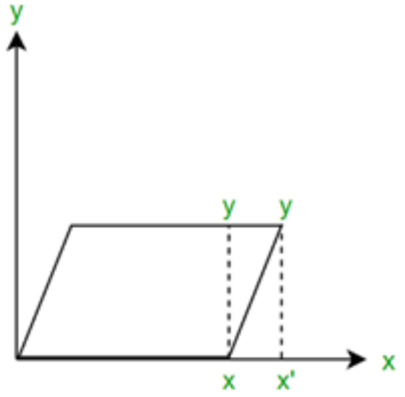
$$y' = sh_y * \mathrm{x} + \mathrm{y}$$



Review Shearing Transformation Matrix

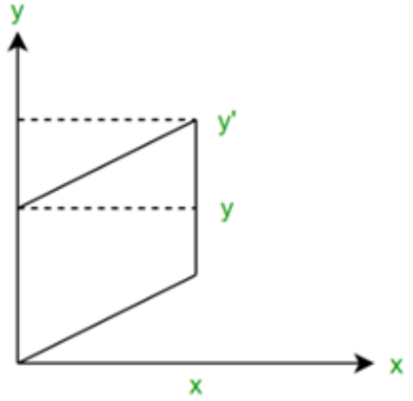
Shearing in X-Direction

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



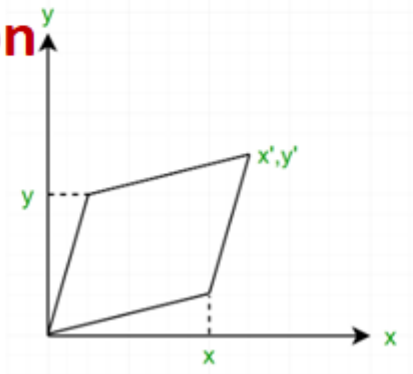
Shearing in Y-Direction

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



Shearing in Both-Direction

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_x & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Shearing

Shearing in x- direction relative to other reference line y=y_{ref}

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

$$y' = y$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Shx & -Shx \cdot y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

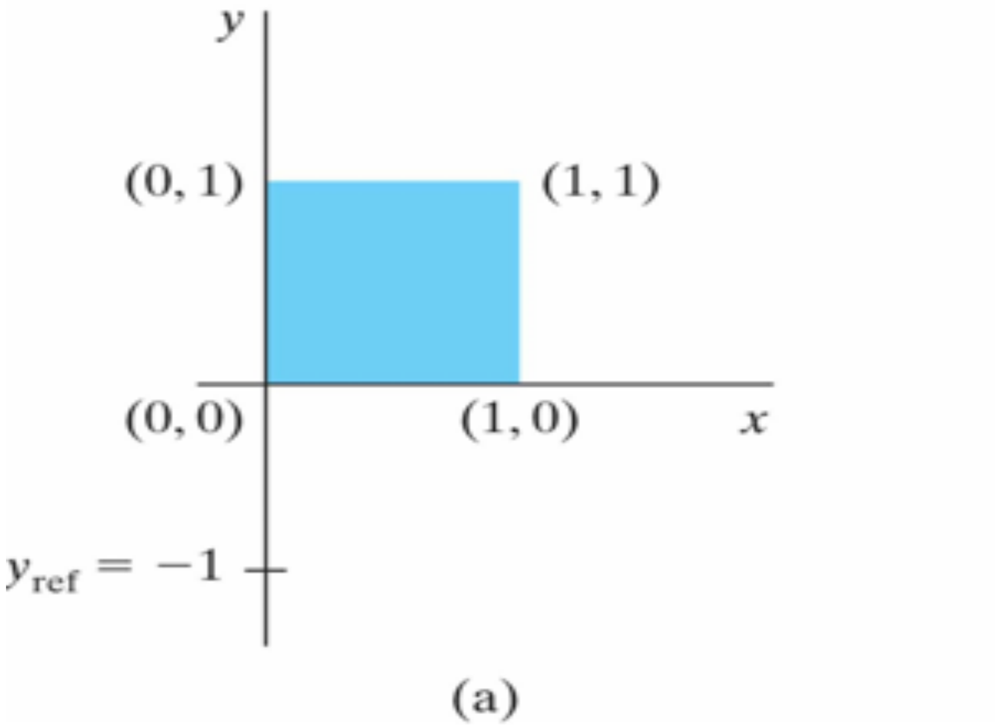
Shearing

Shearing in x- direction relative to other reference line y=y$_{ref}$

$$\begin{bmatrix} 1 & sh_x & -sh_x * y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$y' = y$$



***sh$_x$* = 0.5 and *y*$_{ref}$ = −1**

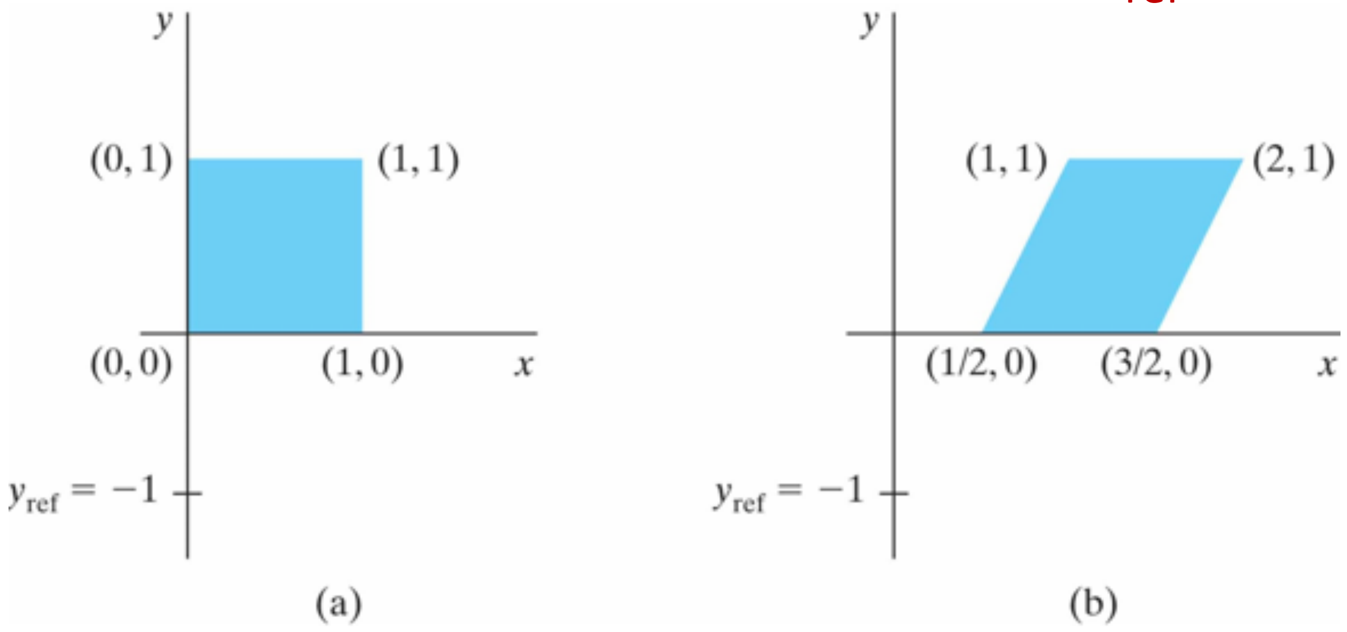
Shearing

Shearing in x- direction relative to other reference line y=y$_{ref}$

$$\begin{bmatrix} 1 & sh_x & -sh_x * y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$y' = y$$



A unit square (a) is transformed to a shifted parallelogram (b) with $sh_x = 0.5$ and $y_{ref} = -1$

Shearing

Shearing in y- direction relative to other reference line x=x_{ref}

$$x' = x$$
$$y' = x + sh_y * \left(x - x_{ref}\right)$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ Shy & 1 & -Shy \cdot X_{ref} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

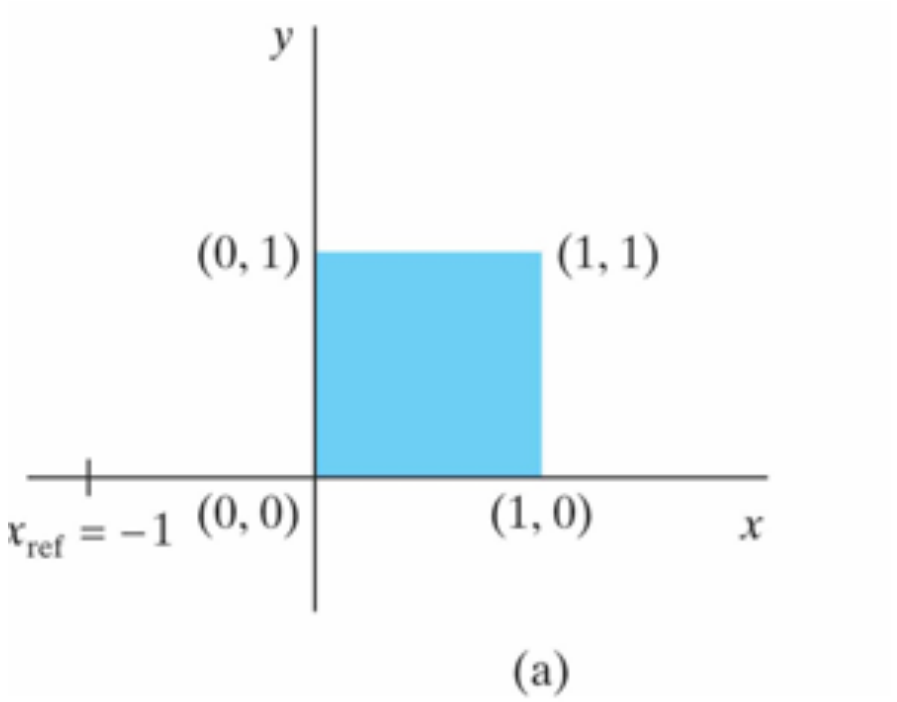
Shearing

Shearing in y- direction relative to other reference line x=x$_{ref}$

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

$$x' = x$$

$$y' = x + sh_y * (x - x_{ref})$$



$sh_y = 0.5$ and $x_{ref} = -1$

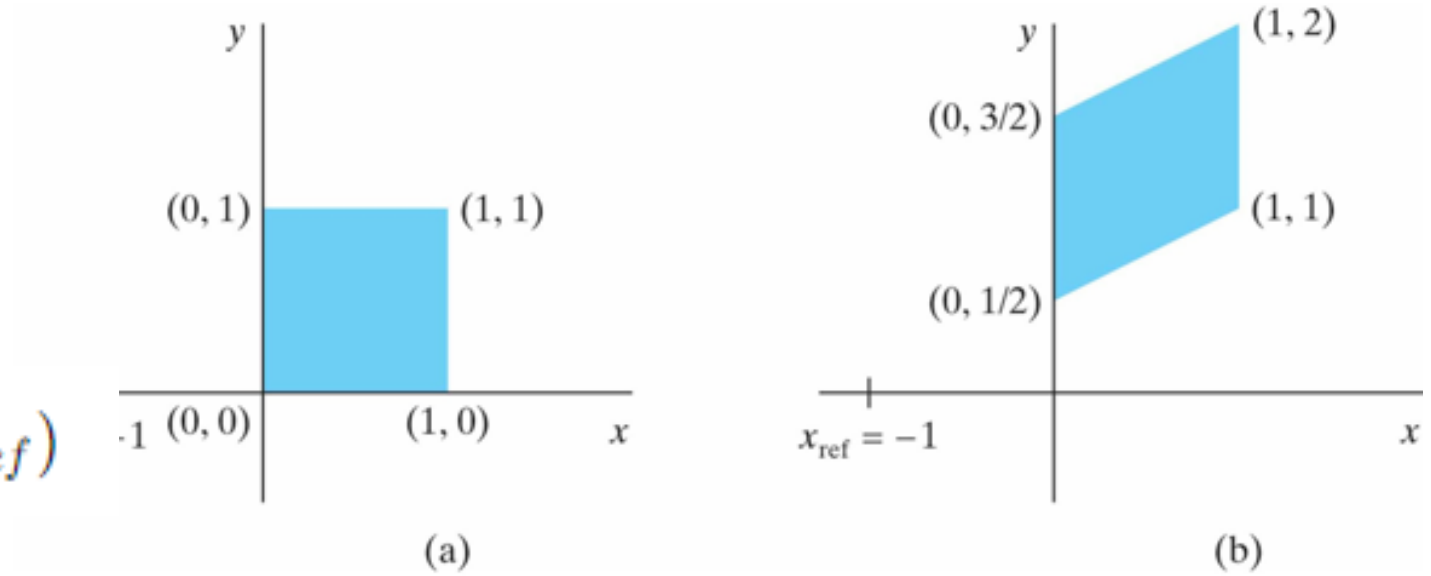
Shearing

Shearing in y- direction relative to other reference line x=x$_{ref}$

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

$$x' = x$$

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



A unit square (a) is turned into a shifted parallelogram (b) with parameter values $sh_y = 0.5$ and $x_{ref} = -1$ in the *y* -direction shearing transformation

2D Viewing

- 2D viewing in computer graphics refers to the process of displaying a two-dimensional image on an output device

- This involves mapping objects defined in a 2D space onto a display device, typically a computer monitor, phone screen or printer

- Key aspects of 2D viewing include defining the viewing area (or window), transforming objects within that window, and projecting the transformed objects onto the viewport (the display area on the screen)

2D Coordinate System

1.Modeling coordinates

- This system is used to define the shapes and positions of objects
- It typically originates from an object's local reference point or origin and extends according to the object's dimensions

2. World Coordinates

- Used to organize the individual objects(Points, lines, circles etc.) into a scene
- A scene is made up of a collection of objects
- These objects make up the "Scene" or "World" that we want to view, and the coordinates that we use to define the scene are called world coordinates
- It represents relative positions of objects

3. Viewing Coordinates

- Viewing coordinates specify the portion of the output device that is to be used to present the view
- It is also known as the camera coordinate system, is specific to a particular viewpoint or camera within a scene
- It defines the positions and orientations of objects relative to the viewpoint or camera
- Coordinates in this system are often used for tasks such as determining visibility, perspective projection, and rendering
- Transformations (Translation, Rotation, Scaling) applied within this coordinate system affect how the scene is viewed from a particular viewpoint

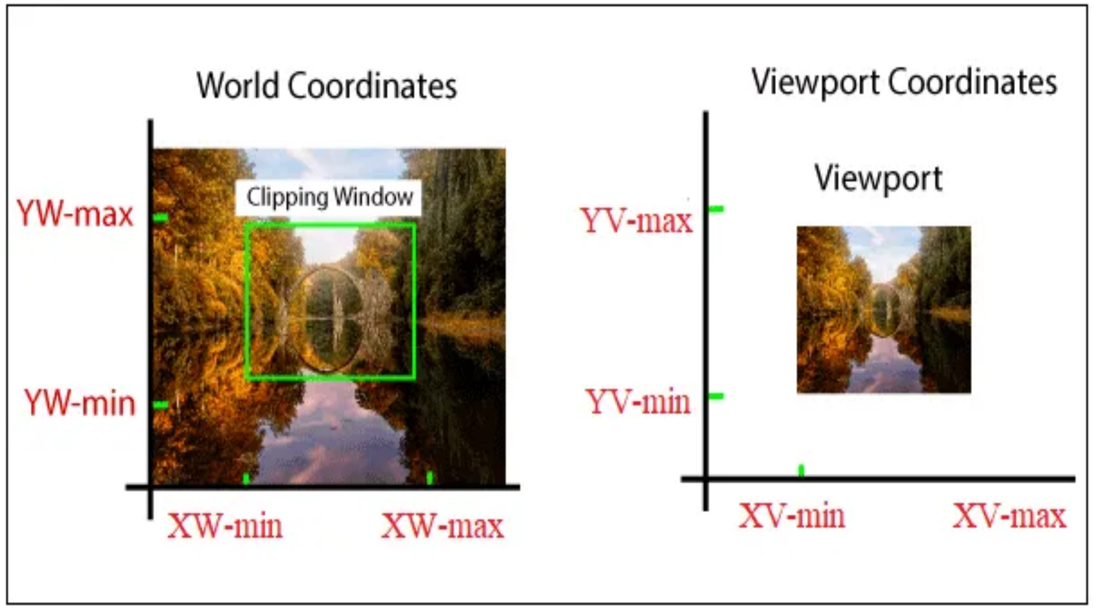
4. Normalized viewing coordinates

- Usually viewing coordinates between 0 and 1 in each direction
- They are used to make the viewing process independent of the output device (monitor, paper, mobile)

5. Device Coordinates or Screen Coordinates

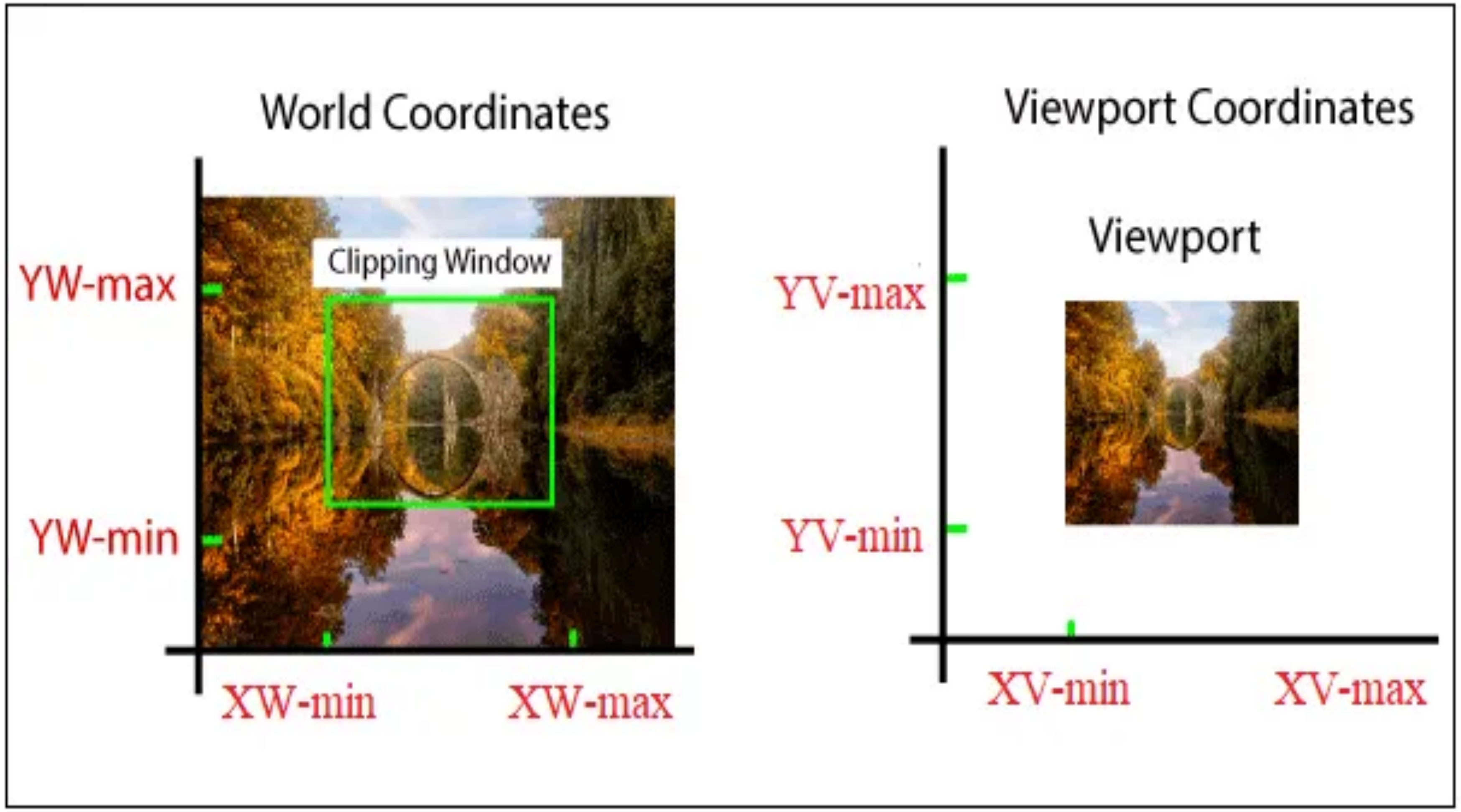
- Device coordinates are display coordinates that are specific to output device
- The device coordinate system represents the physical display or output device onto which the scene is rendered
- It maps the virtual world coordinates onto the actual pixels or display units of the output device
- Transformations within this coordinate system may include scaling to match the resolution of the output device
- **Device coordinates** are integers within the range **(0, 0) to (x_{max}, y_{max})** for a particular output device.

2D Viewing



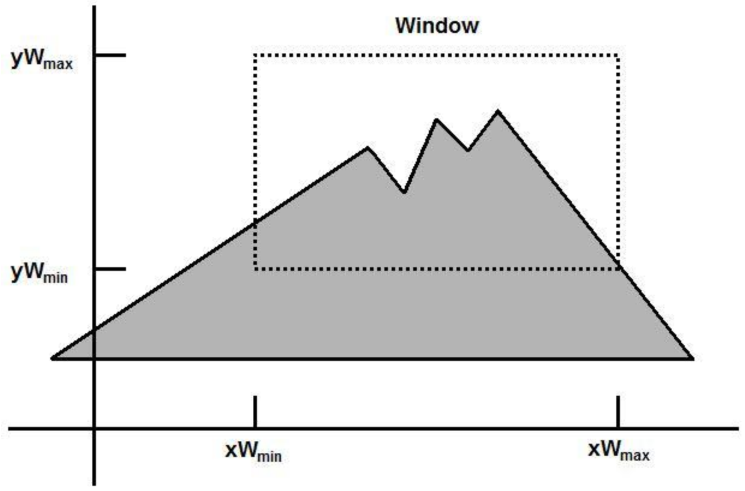
- **Window**
  - a world-coordinate area selected for display
  - define **what is to be viewed**
- **View port**
  - an area on a display device to which a window is mapped
  - define **where it is to be displayed**
  - define within the unit square
  - the unit square is mapped to the display area for the particular output device in use at that time
- **Windows & Viewport**
  - be rectangles in standard position, with the rectangle edges parallel to the coordinate axes

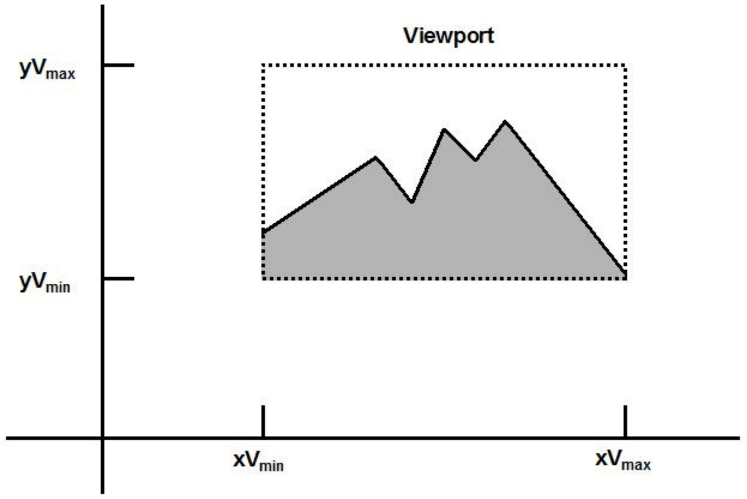
Window and Viewport



Viewing transformation

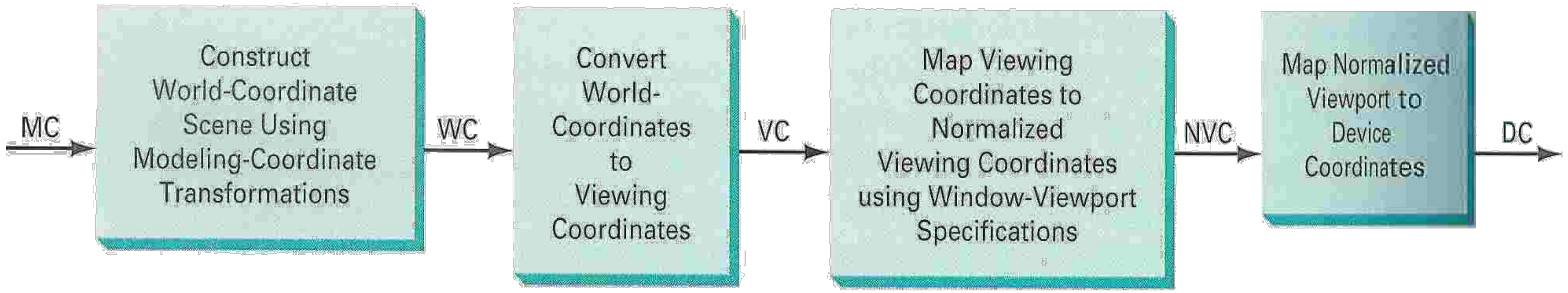
- The **mapping** of a part of a **world-coordinate** scene to **device coordinates** is referred to as a **viewing transformation**
- **2D viewing transformation** is simply referred to as the **window-to-viewport transformation** or the **windowing transformation**





2D Viewing Pipeline

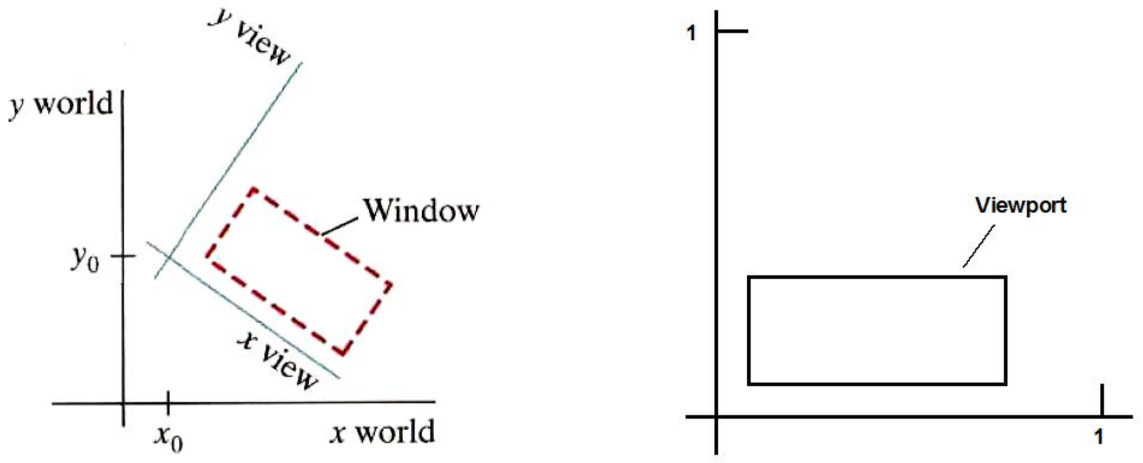
- The **viewing transformation** is carried out in following steps:
  - Construct **world-coordinate scene** using modeling-coordinate transformations
  - Convert world-coordinates to **viewing coordinates**
  - Transform viewing-coordinates to **normalized-coordinates** (ex: between 0 and 1, or between -1 and 1)
  - Map normalized-coordinates to **device-coordinates**


MC
Construct World-Coordinate Scene Using Modeling-Coordinate Transformations
WC
Convert World-Coordinates to Viewing Coordinates
VC
Map Viewing Coordinates to Normalized Viewing Coordinates using Window-Viewport Specifications
NVC
Map Normalized Viewport to Device Coordinates
DC

Construct the scene in **world coordinate (WC)** using the output primitives such as line and circle and their attributes

- **Set Up Viewing Coordinate System (VC)**
  - Establish a **viewing reference frame** in the world coordinate plane
  - Define the origin and orientation (rotation) of the viewing coordinate system
  - Specify the **window** (rectangular area of interest) within the viewing coordinate system
- **Transform World Coordinates (WC) to Viewing Coordinates (VC)**
  - Translate the origin of the WC system to the origin of the VC system
  - Rotate the WC system to align with the orientation of the VC system
- **Define a Viewport in Normalized Coordinates (NVC)**
  - Define the viewport in normalized device coordinates (NVC), ranging from 0 to 1
  - This defines the area on the screen where the window will be mapped
- **Map Viewing Coordinates (VC) to Normalized Coordinates (NVC)**
  - Scale and translate the VC window to fit into the NVC viewport

- **Clipping and Transfer to Device Coordinates (DC)**
- At the **final step**, all parts of the picture that lie outside the viewport are **clipped**, and the contents of the viewport are transferred to **device coordinates (DC)**



A **rotated** viewing-coordinate reference frame and the mapping to normalized coordinates.

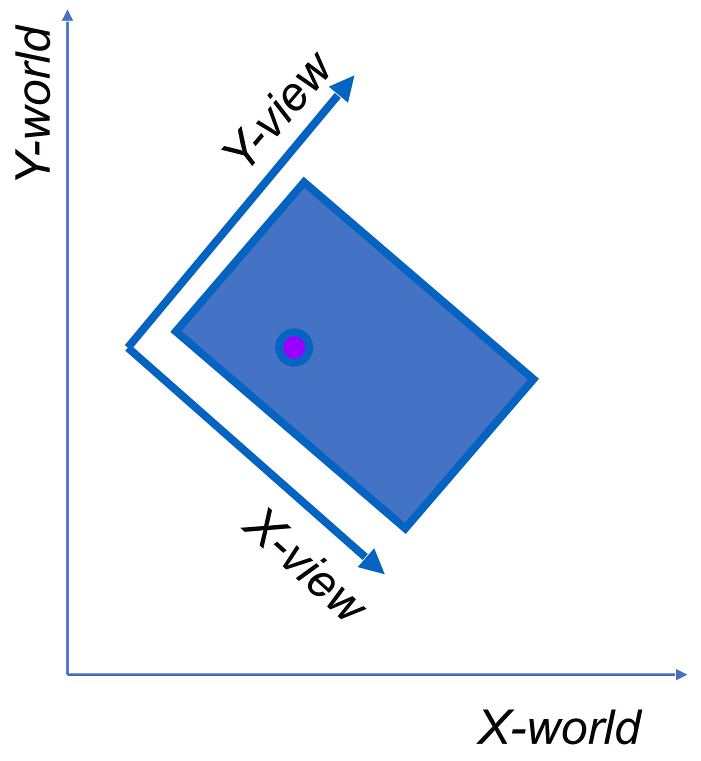
Viewing Coordinate Reference Frame

- To obtain the **matrix** for converting **world-coordinate** positions to **viewing coordinates:**
  - translating and rotating the world coordinates to align with the viewing coordinates
- **First**, we translate the viewing origin to the world origin
- **Second**, we rotate to align the two coordinate reference frame

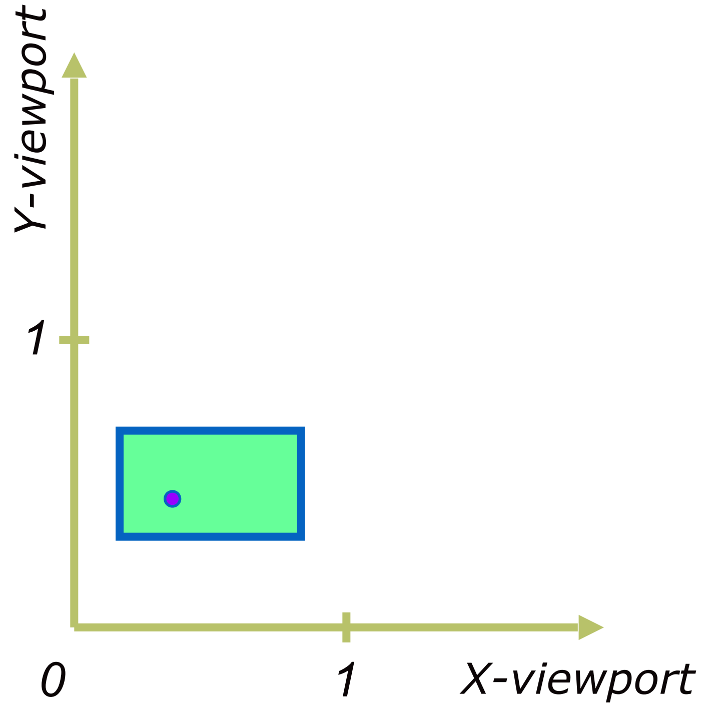
$$\mathbf{M_{wc,\,vc} = R.T}$$

- Where **T** is the **translation** matrix and **R** is the **rotation** matrix

Window-to-viewport coordinate transformation



Window

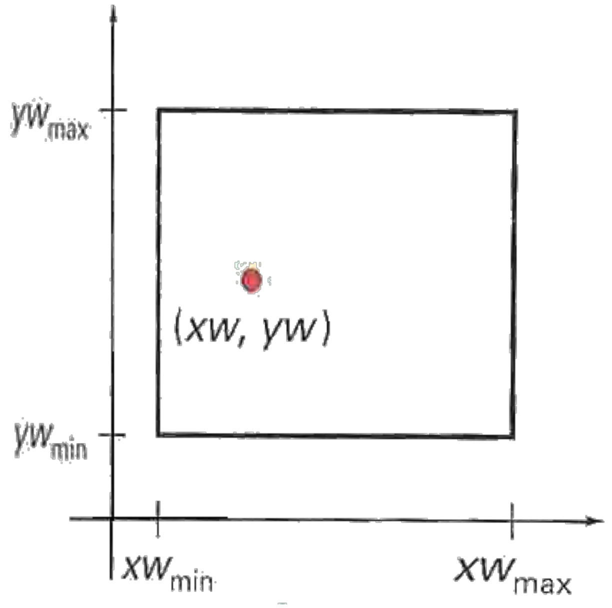


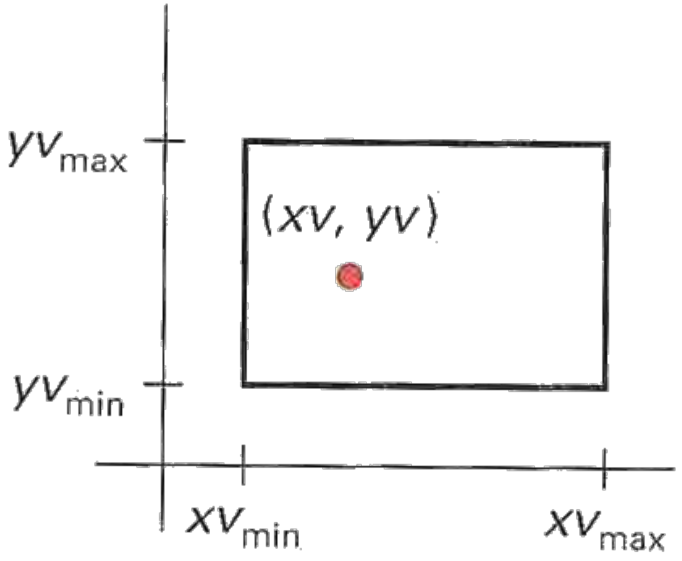
Viewport

Window-to-viewport coordinate transformation

- The window-to-viewport coordinate transformation is a crucial step in computer graphics to map a portion of a scene (window) to a display area (viewport).
- This process involves transforming coordinates **from the window defined in world coordinates** to the **viewport defined in device coordinates** or normalized device coordinates.
- window to viewport coordinate transformation maintains the same relative placement of objects in normalized space as in viewing coordinates

- A point at position (**x_w,y_w**) in the window is mapped to the position **($x_{v,}x_v$)** in the associated viewport
- **transfer to the viewing reference frame**
  - choose the window extents in viewing coordinate
  - select the viewport limits in normalized coordinate





- **to maintain the same relative placement in the viewport as in the window**

$$\frac{xv - xv_{\min}}{xv_{\max} - xv_{\min}} = \frac{xw - xw_{\min}}{xw_{\max} - xw_{\min}} \qquad \frac{yv - yv_{\min}}{yv_{\max} - yv_{\min}} = \frac{yw - yw_{\min}}{yw_{\max} - yw_{\min}}$$

- **Thus**

$$xv = xv_{\min} + (xw - xw_{\min})sx$$
$$yv = yv_{\min} + (yw - yw_{\min})sy$$

Where scaling factors,

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

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

- **If sx = sy, the proportion is maintained otherwise the scene is stretched**

$$xv = xv_{\min} + (xw - xw_{\min})sx$$
$$yv = yv_{\min} + (yw - yw_{\min})sy$$

- This can also be obtained by following transformations:
  - ❑Translate the window to the origin That is, apply **T(-xw$_{\min}$, -yw$_{\min}$)**
  - ❑Scale it to the size of the viewport That is, apply **S(sx, sy)**
  - ❑Translate scaled window to the position of the viewport. That is, apply **T(xv$_{\min}$, yv$_{\min}$)**
  - ❑Therefore, net transformation, **T(xv$_{\min}$, yv$_{\min}$). S(sx, sy). T(-xw$_{\min}$, -yw$_{\min}$)**

Advantage of Viewing Transformation:

- We can display picture at device or display system according to our need and choice.

Note that

- World coordinate system is selected suits according to the application program.
- Screen coordinate system is chosen according to the need of design.
- Viewing transformation is selected as a bridge between the world and screen coordinate.

- Window port is given by (100, 100, 300, 300) and Viewport: (50, 50, 150, 150) .Convert the window-port coordinate (200, 200) to the viewport coordinate.

- Window port is given by (100, 80, 40, 80) and Viewport: (30, 60, 40, 60) .Convert the window port coordinate (30, 80) to the viewport coordinate.