Computer Graphics (CSE2066)

2D Geometric Transformations

Translation, Scaling, Rotation in computer Graphics

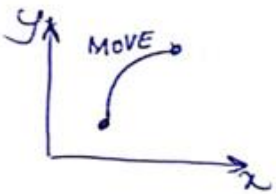
PRESIDENCY UNIVERSITY
GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS
Private University Estd. in Karnataka State by Act No. 41 of 2013
PRESIDENCY GROUP OVER 40 YEARS OF ACADEMIC WISDOM

2D Geometric Transformations:

- In Computer graphics, Transformation is a process of modifying and re-positioning the existing graphics.
- 2D Transformations take place in a two dimensional plane

Representation of points :

$\overset{R}{2} \times \overset{c}{1}$ matrix $\begin{bmatrix} x \\ y \end{bmatrix}$



- Transformations are helpful in changing the position, size, orientation, shape etc of the object.

Two-Dimensional Translation:

- In Computer graphics, 2D Translation is a process of **moving** an object from one position to another in a two dimensional plane.
- Consider a point object O has to be moved from one position to another in a 2D plane.

Let-

Initial coordinates of the object O = (X_{old}, Y_{old})

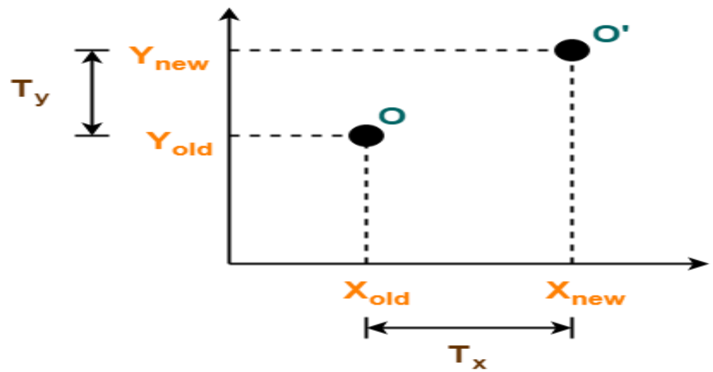
New coordinates of the object O' after translation = (X_{new}, Y_{new})

Translation vector or Shift vector = (T_x, T_y)

Given a Translation vector (T_x, T_y)

T_x defines the distance the X_{old} coordinate has to be moved.

T_y defines the distance the Y_{old} coordinate has to be moved.



2D Translation in Computer Graphics

This translation is achieved by adding the translation coordinates to the old coordinates of the object as-

$X_{new} = X_{old} + T_x$ (This denotes translation towards X axis)

$Y_{new} = Y_{old} + T_y$ (This denotes translation towards Y axis)

In Matrix form, the above translation equations may be represented as

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

Translation Matrix

The translation values of X_{new} and Y_{new} is calculated as

$$\mathbf{X_{new} = X_{old} + T_x} \qquad \mathbf{Y_{new} = Y_{old} + T_y}$$

The translation distance pair (T_x, T_y) is called a **translation vector** or **shift vector Column vector representation is given as**

$$P' = \begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} \quad P = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} \quad T = \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

This allows us to write the two-dimensional translation equations in the matrix Form

$$\mathbf{P' = P + T}$$

Problem-01:

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.

Solution-

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).

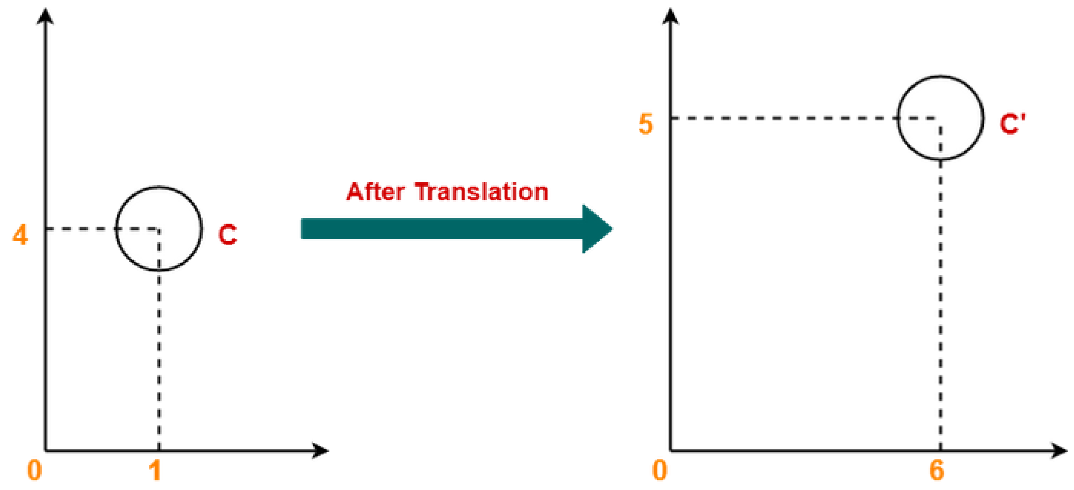
In matrix form, the new center coordinates of C after translation may be obtained as

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix} + \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \end{bmatrix}$$

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

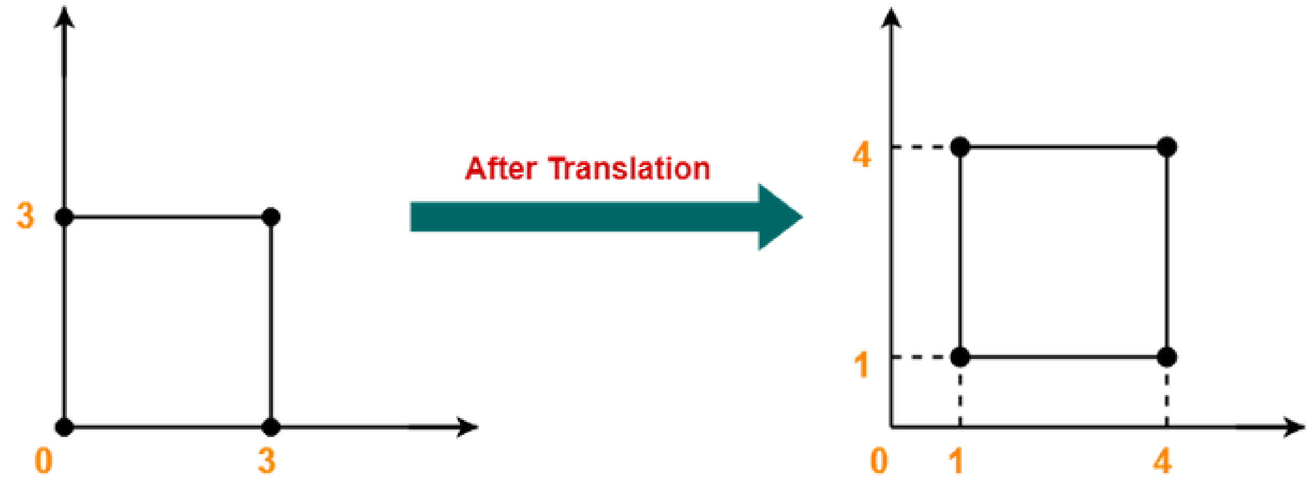


PRESIDENCY UNIVERSITY
Private University Estd. in Karnataka State by Act No. 41 of 2013

Problem-02:

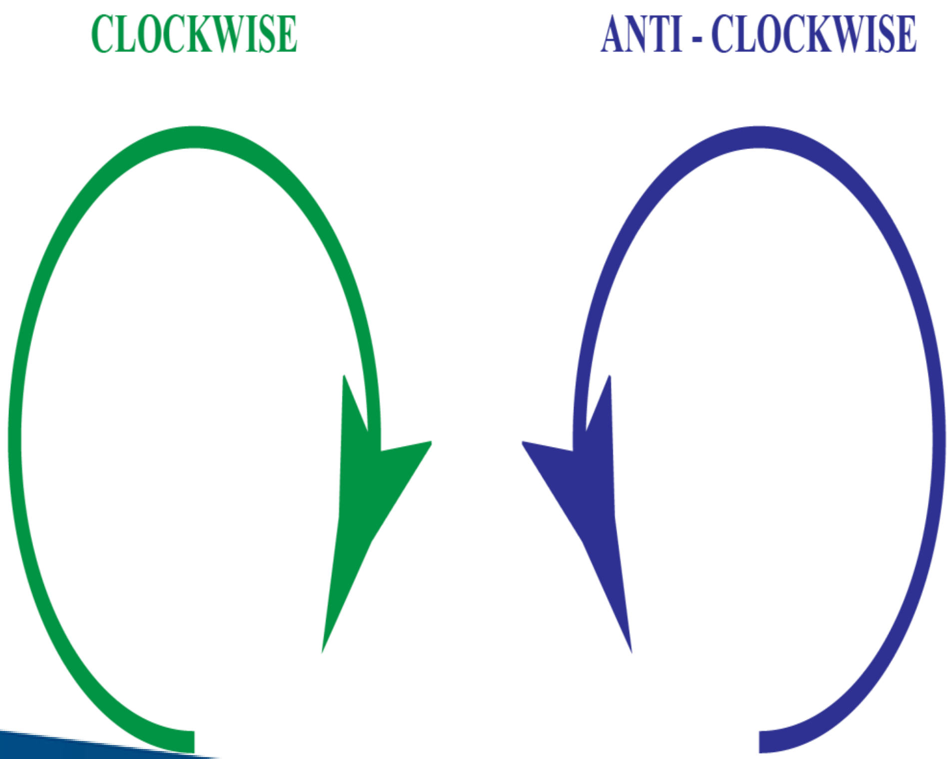
Given a square with coordinate points A(0, 3), B(3, 3), C(3, 0), D(0, 0). Apply the translation with distance 1 towards X axis and 1 towards Y axis. Obtain the new coordinates of the square.

Solution:



Two-Dimensional Rotation

- In Computer graphics, 2D Rotation is a process of rotating an object with respect to an angle in a two dimensional plane.



Anti-Clockwise / Counter Clockwise Rotation Matrix:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Clockwise Rotation Matrix:

$$R(-\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

PRESIDENCY UNIVERSITY
Private University Estd. in Karnataka State by Act No. 41 of 2013

Consider a point object O has to be rotated from one angle to another in a 2D plane.

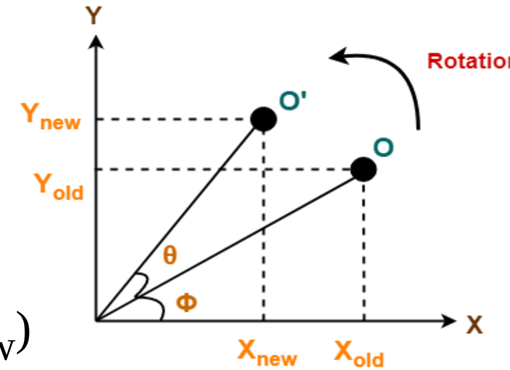
Let-

Initial coordinates of the object O = (X_{old}, Y_{old})

Initial angle of the object O with respect to origin = Φ

Rotation angle = θ

New coordinates of the object O' after rotation = (X_{new}, Y_{new})

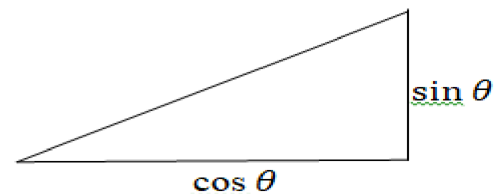


2D Rotation in Computer Graphics

This rotation is achieved by using the following rotation equations

$$X_{new} = X_{old} \times \cos\theta - Y_{old} \times \sin\theta$$

$$Y_{new} = X_{old} \times \sin\theta + Y_{old} \times \cos\theta$$



In Matrix form, the above rotation equations may be represented as

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Rotation Matrix

Anti-Clockwise Direction

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Rotation Matrix

Clockwise Direction

We can write the rotation equations in the matrix form

$$\mathbf{P'} = \mathbf{R} \cdot \mathbf{P}$$

Where the rotation matrix is

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

For Brief Calculation read below procedure (for reference purpose only)

* Parameters of 2D rotation are the rotation angle θ, position (x_r, y_r) called rotation point or pivot point (intersection point/position of rotation axis ~~at~~ with xy plane).

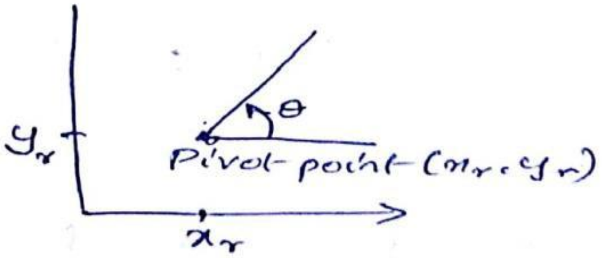
* +ve value for angle θ → counterclockwise rotation

* -ve value – clockwise.

Formula in Trignometry

$$\cos\theta = \frac{\text{adjacent side}}{\text{hypoteneus}}$$

$$\sin\theta = \frac{\text{opposite side}}{\text{hypo}}$$



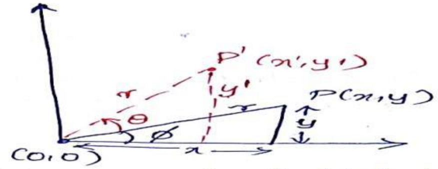
$$\cos(A+B) = \cos A \cos B - \sin A \sin B$$

$$\sin(A+B) = \sin A \cos B + \cos A \sin B$$

$\cos(A-B) = \cos A \cos B + \sin A \sin B$
$\sin(A-B) = \sin A \cos B - \cos A \sin B$

$\cos\phi = \frac{x}{r} \Rightarrow x = r\cos\phi$

$\sin\phi = \frac{y}{r} \Rightarrow y = r\sin\phi$



Rotation in Anticlock wise

The new angle after rotation from $P \to P' = (\phi + \theta)$

so $\cos(\phi+\theta) = \frac{x'}{r}$

$\boxed{x' = r.\cos(\phi+\theta)} = r[\cos\phi \cos\theta - \sin\phi . \sin\theta]$

$= r\cos\phi\cos\theta - r.\sin\phi.\sin\theta$

$\boxed{x' = x\cos\theta - y\sin\theta} \rightarrow (1)$ $\quad \boxed{\because x = r\cos\phi \quad y = r\sin\phi}$

$\text{a}l^{y}$ $\sin(\phi+\theta) = \frac{y'}{r}$

$\boxed{y' = r.\sin(\phi+\theta)} = r.\sin\phi\cos\theta + \cos\phi.\sin\theta$

$= r.\sin\phi\cos\theta + r.\cos\phi.\sin\theta$

$\boxed{y' = y\cos\theta + x\sin\theta} \rightarrow (2)$

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

$\therefore P' = R.P$

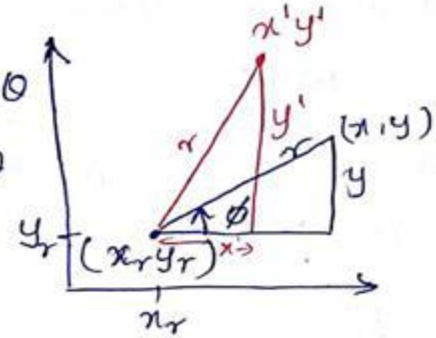
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

If we solve this matrix we will get eqn. (1) & (2)

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

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

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



Problem-01:

Given a line segment with starting point as (0, 0) and ending point as (4, 4). Apply 30 degree rotation anticlockwise direction on the line segment and find out the new coordinates of the line.

Given-

Old ending coordinates of the line = $(X_{old}, Y_{old}) = (4, 4)$

Rotation angle = $\theta = 30^o$

Let new ending coordinates of the line after rotation = (X_{new}, Y_{new}).

Applying the rotation equations, we have

X_{new}

$= X_{old} \times \cos\theta - Y_{old} \times \sin\theta$

$= 4 \times \cos 30^o - 4 \times \sin 30^o$

$= 4 \times (\sqrt{3} / 2) - 4 \times (1 / 2)$

$= 2\sqrt{3} - 2$

$= 2(\sqrt{3} - 1)$

$= 2(1.73 - 1)$

$= 1.46$

Y_{new}

$= X_{old} \times \sin\theta + Y_{old} \times \cos\theta$

$= 4 \times \sin 30^o + 4 \times \cos 30^o$

$= 4 \times (1 / 2) + 4 \times (\sqrt{3} / 2)$

$= 2 + 2\sqrt{3}$

$= 2(1 + \sqrt{3})$

$= 2(1 + 1.73)$

$= 5.46$

Thus, New ending coordinates of the line after rotation = (1.46, 5.46).

Alternatively,

In matrix form, the new ending coordinates of the line after rotation may be obtained as-

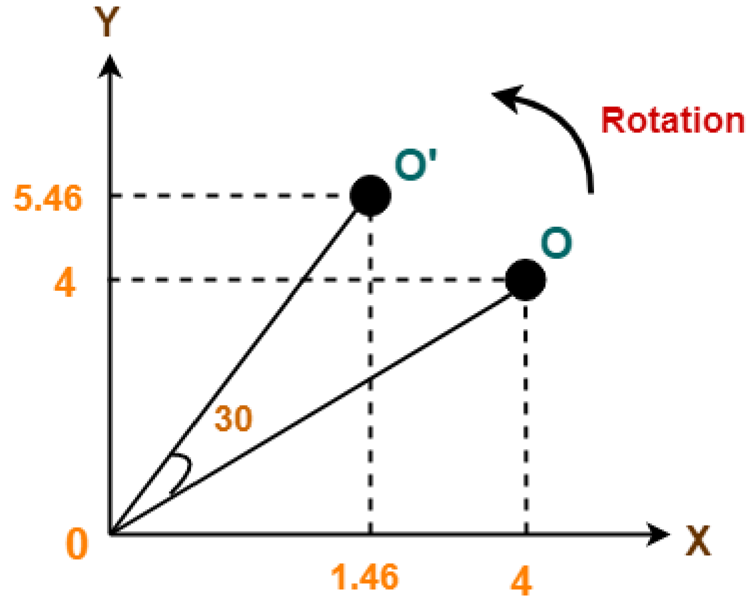
$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos30 & -\sin30 \\ \sin30 & \cos30 \end{bmatrix} \times \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 4 \times \cos30 - 4 \times \sin30 \\ 4 \times \sin30 + 4 \times \cos30 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 4 \times \cos30 - 4 \times \sin30 \\ 4 \times \sin30 + 4 \times \cos30 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1.46 \\ 5.46 \end{bmatrix}$$

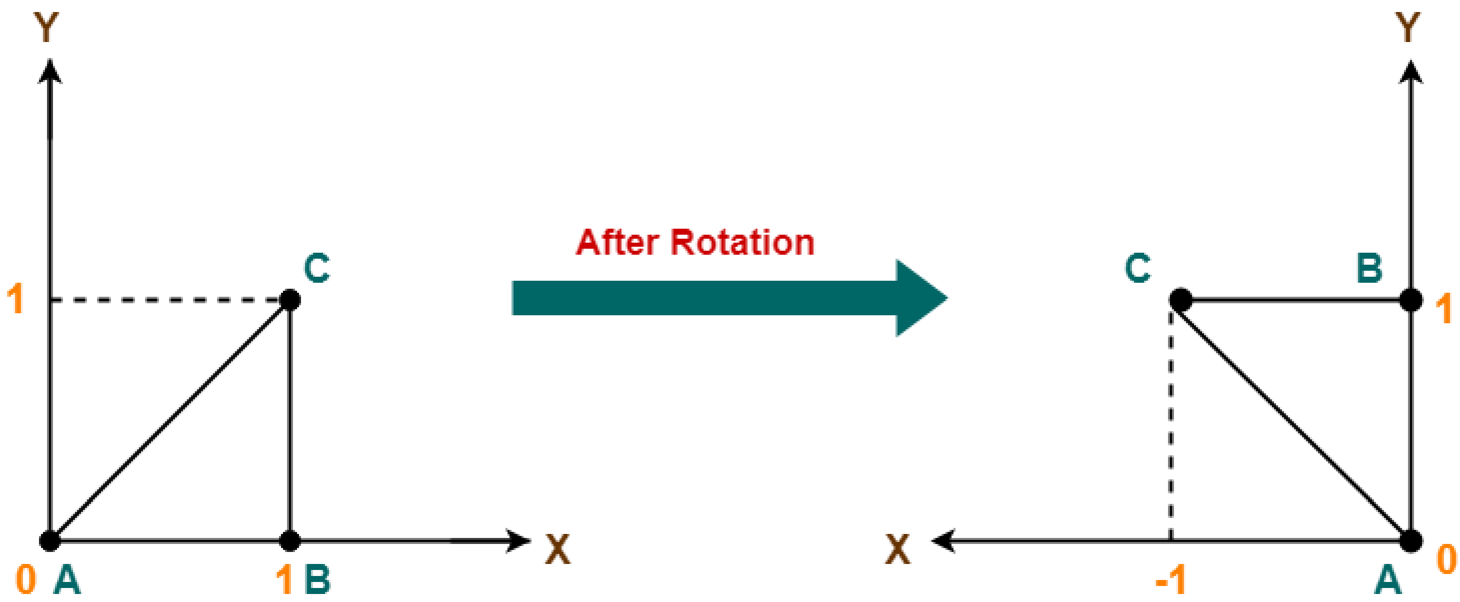


Thus, New ending coordinates of the line after rotation = (1.46, 5.46).

Problem-02:

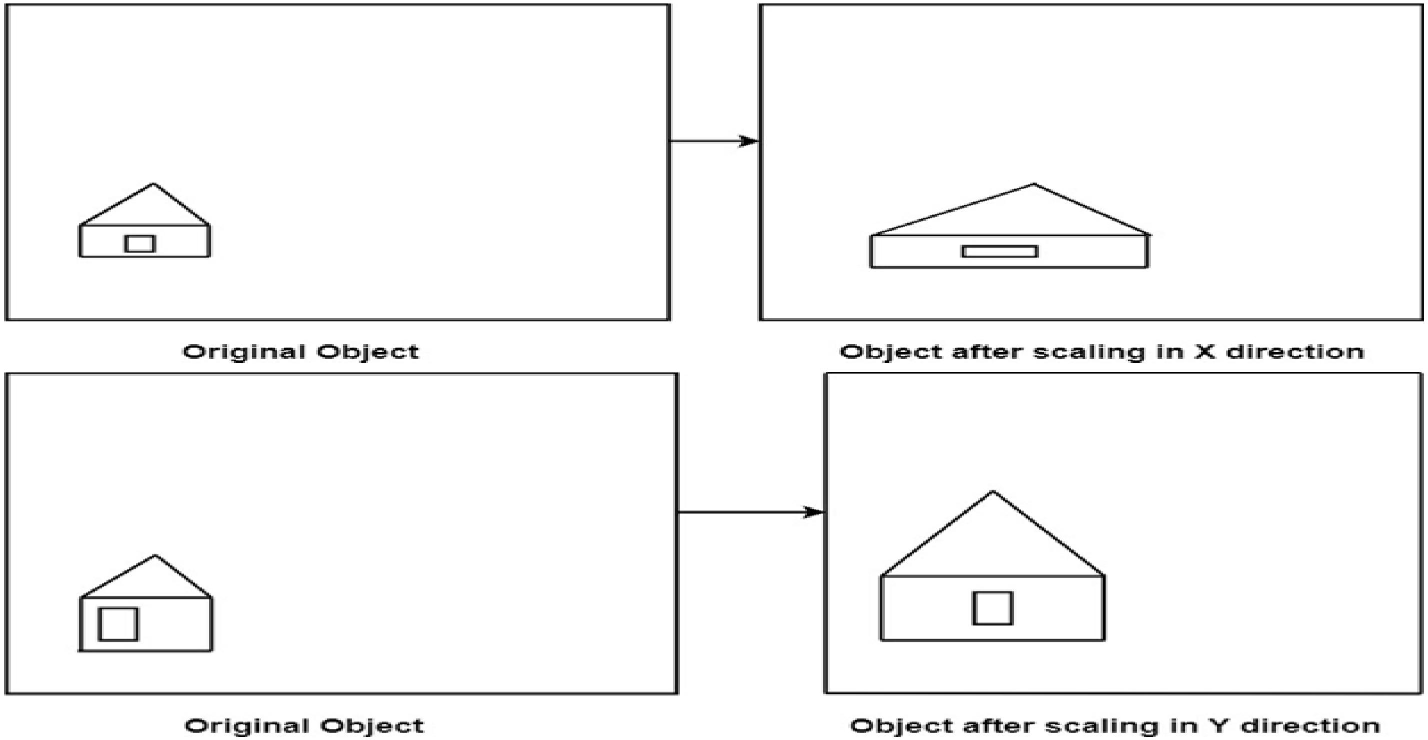
Given a triangle with corner coordinates (0, 0), (1, 0) and (1, 1). Rotate the triangle by 90 degree anticlockwise direction and find out the new coordinates.

Solution:



Two-Dimensional Scaling

- In computer graphics, scaling is a process of modifying or altering the size of objects.
- Scaling may be used to increase or reduce the size of object.
- Scaling subjects the coordinate points of the original object to change.
- Scaling factor determines whether the object size is to be increased or reduced.
- If scaling factor > 1, then the object size is increased.
- If scaling factor < 1, then the object size is reduced.



Consider a point object O has to be scaled in a 2D plane.

Let-

Initial coordinates of the object O = (X_{old}, Y_{old})

Scaling factor for X-axis = S_x

Scaling factor for Y-axis = S_y

New coordinates of the object O′ after scaling = (X_{new}, Y_{new})

This scaling is achieved by using the following scaling equations-

$$X_{new} = X_{old} \times S_x$$

$$Y_{new} = Y_{old} \times S_y$$

In Matrix form, the above scaling equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Scaling Matrix

Note:

- Specifying a value of 1 for both *Sx* and *Sy* leaves the size of objects unchanged.
- **Uniform Scaling**: Both S_x & S_y are assigned to same value.
- **Differential Scaling**: Both S_x & S_y are assigned to different value. (i.e., values are not same)

Problem:

Given a square object with coordinate points A(0, 3), B(3, 3), C(3, 0), D(0, 0). Apply the scaling parameter 2 towards X axis and 3 towards Y axis and obtain the new coordinates of the object.

Solution:

Given-

Old corner coordinates of the square = A (0, 3), B(3, 3), C(3, 0), D(0, 0)

Scaling factor along X axis = 2

Scaling factor along Y axis = 3

For Coordinates A(0, 3)

Let the new coordinates of corner A after scaling = (X_{new}, Y_{new}).

Applying the scaling equations, we have-

$X_{new} = X_{old} \times S_x = 0 \times 2 = 0$

$Y_{new} = Y_{old} \times S_y = 3 \times 3 = 9$

Thus, New coordinates of corner A after scaling = (0, 9).

For Coordinates B(3, 3)

Let the new coordinates of corner B after scaling = (X_{new}, Y_{new}).

Applying the scaling equations, we have-

$X_{new} = X_{old} \times S_x = 3 \times 2 = 6$

$Y_{new} = Y_{old} \times S_y = 3 \times 3 = 9$

Thus, New coordinates of corner B after scaling = (6, 9).

For Coordinates C(3, 0)

Let the new coordinates of corner C after scaling = (X_{new}, Y_{new}).

Applying the scaling equations, we have-

$X_{new} = X_{old} \times S_x = 3 \times 2 = 6$

$Y_{new} = Y_{old} \times S_y = 0 \times 3 = 0$

Thus, New coordinates of corner C after scaling = (6, 0).

For Coordinates D(0, 0)

Let the new coordinates of corner D after scaling = (X_{new}, Y_{new}).

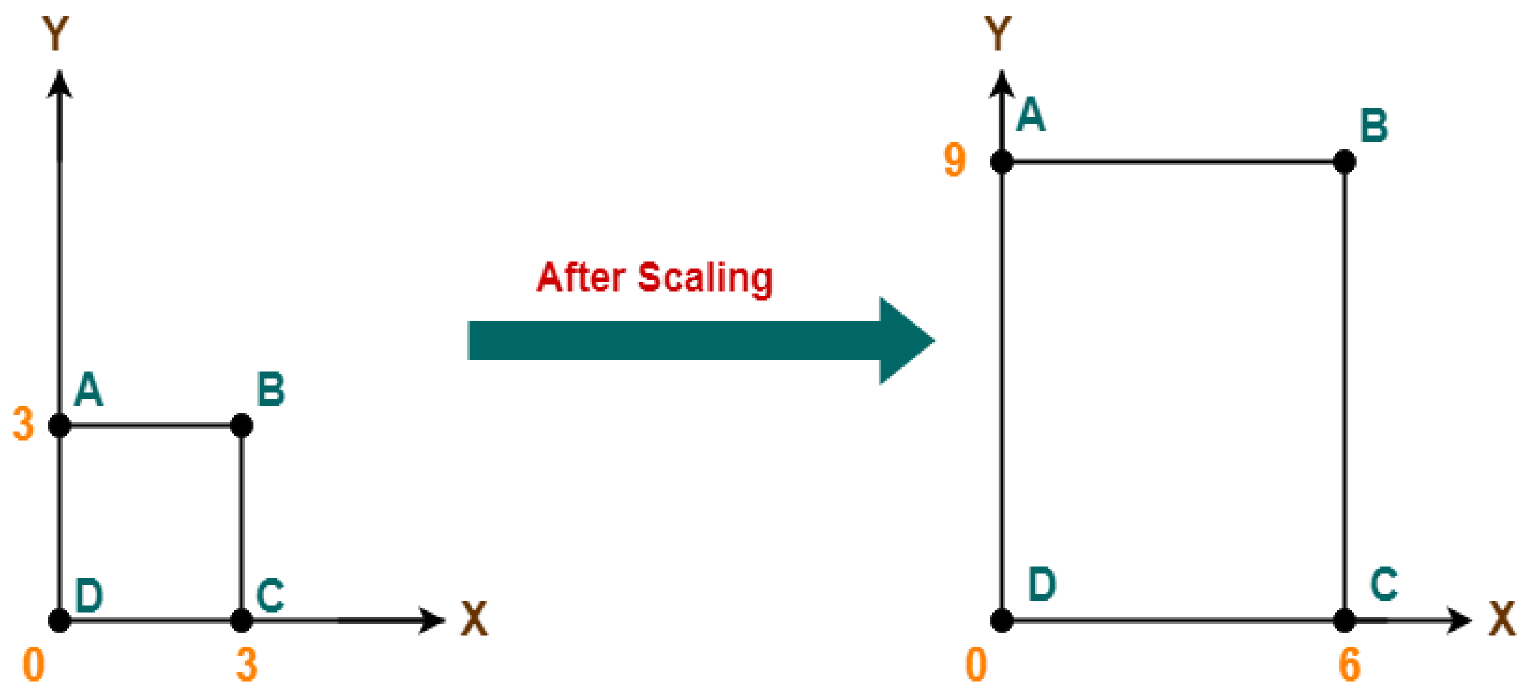
Applying the scaling equations, we have-

$X_{new} = X_{old} \times S_x = 0 \times 2 = 0$

$Y_{new} = Y_{old} \times S_y = 0 \times 3 = 0$

Thus, New coordinates of corner D after scaling = (0, 0).

Thus, New coordinates of the square after scaling = A (0, 9), B(6, 9), C(6, 0), D(0, 0)


Y
A
B
3
D
C
X
0
3
After Scaling
Y
A
B
9
D
C
X
0
6

Matrix representation for translation, scaling and rotation.

MATRIX REPRESENTATIONS

- Each of the three basic two-dimensional transformations (translation, rotation, and scaling) can be expressed in the general matrix form

$$P' = M_1 \cdot P + M_2$$

- With coordinate positions **P** and P' represented as column vectors.
- Matrix **M**1 is a 2 × 2 array containing multiplicative factors, and **M**2 is a two-element column matrix containing translational terms.

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

Translation Matrix

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} X \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Rotation Matrix
Anti-Clockwise Direction

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} X \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Rotation Matrix
Clockwise Direction

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} X \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Scaling Matrix

Homogeneous coordinates for translation, scaling and rotation.

HOMOGENEOUS COORDINATES

A standard technique to expand the matrix representation for a 2D co-ordinate (x,y) position to a 3 element (column matrix) representation (X_h, Y_h,h) called Homogeneous Coordinates.

h- homogeneous parameter (non-zero value)

i.e. (x, y) is converted into new coordinate values as (X_h, Y_h, h)

$$\mathbf{x = \frac{x_h}{h}} \qquad \mathbf{y = \frac{y_h}{h}}$$

$$\mathbf{x_h = x*h} \qquad \mathbf{y_h = y*h}$$

Example:

<u>Convert 2D coordinate into homogeneous coordinate:</u>

Suppose x=2, y=3 & if h=1 then (x_h, y_h, h) = (2*1, 3*1, 2) = (2,3,1)

If h=2, then (x*h, y*h, h) = (2*2, 3*2, 2) = (4, 6, 2)

<u>Convert homogeneous coordinate into 2D coordinate:</u>

Suppose homogeneous coordinates is (x_h,y_h,h) =(4, 6, 2) then 2D coordinate is

$$x = \frac{x_h}{h} = \frac{4}{2} = 2 \qquad y = \frac{y_h}{h} = \frac{6}{2} = 3$$ so (x,y)=(2,3)

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

Translation Matrix

(Homogeneous Coordinates Representation)

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

Rotation Matrix

(Homogeneous Coordinates Representation)

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

Scaling Matrix

(Homogeneous Coordinates Representation)

INVERSE TRANSFORMATIONS

For translation, the inverse matrix by negating the translation distances

$$T^{-1} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$

An inverse rotation is accomplished by replacing the rotation angle by its negative (anti-clockwise).

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

An inverse scaling transformation by replacing the scaling parameters with their reciprocals. The inverse transformation matrix

$$S^{-1} = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

An inverse matrix generates an opposite scaling transformations, so any scaling matrix multiply its inverse produces Identity Matrix.

2D Composite transformations

TWO-DIMENSIONAL COMPOSITE TRANSFORMATIONS

Forming products of transformation matrices is often referred to as a **concatenation,** or **composition,** of matrices if we want to apply two transformations to point position **P**, the transformed location would be calculated as

$$\mathrm{P}' = \mathrm{M}_2 \cdot \mathrm{M}_1 \cdot \mathrm{P}$$
$$= \mathrm{M} \cdot \mathrm{P}$$

Where M= M2.M1

The coordinate position is transformed using the composite matrix **M**, rather than applying the individual transformations **M**1 and then **M**2.

PRESIDENCY UNIVERSITY

GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS

Private University Estd. in Karnataka State by Act No. 41 of 2013

PRESIDENCY GROUP OVER 40 YEARS OF ACADEMIC WISDOM

Composite Two-Dimensional Translations

- If two successive translation vectors (t_{1x}, t_{1y}) and (t_{2x}, t_{2y}) are applied to a two dimensional coordinate position **P**, the final transformed location P′ is calculated as

$$\mathbf{P}' = \mathbf{T}(t_{2x}, t_{2y}) \cdot \{\mathbf{T}(t_{1x}, t_{1y}) \cdot \mathbf{P}\}$$

$$= \{\mathbf{T}(t_{2x}, t_{2y}) \cdot \mathbf{T}(t_{1x}, t_{1y})\} \cdot \mathbf{P}$$

where **P** and ***P′*** are represented as three-element, homogeneous-coordinate column vectors.

- Also, the composite transformation matrix for this sequence of translations is

$$\begin{bmatrix} 1 & 0 & t_{2x} \\ 0 & 1 & t_{2y} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{1x} \\ 0 & 1 & t_{1y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{1x} + t_{2x} \\ 0 & 1 & t_{1y} + t_{2y} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}(t_{2x}, t_{2y}) \cdot \mathbf{T}(t_{1x}, t_{1y}) = \mathbf{T}(t_{1x} + t_{2x}, t_{1y} + t_{2y})$$

- **This demonstrates 2 successive translation matrix are additive.**

Composite Two-Dimensional Rotation

- Two successive rotations applied to a point **P** produce the transformed position

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

- By multiplying the two rotation matrices, we can verify that two successive rotations are additive:

$$\mathbf{R}(\vartheta 2) \cdot \mathbf{R}(\vartheta 1) = \mathbf{R}(\vartheta 1 + \vartheta 2)$$

- So that the final rotated coordinates of a point can be calculated with the composite rotation matrix

$$\mathbf{P}' = \mathbf{R}(\vartheta 1 + \vartheta 2) \cdot \mathbf{P}$$

- **This demonstrates 2 successive rotation matrix are additive.**

PRESIDENCY UNIVERSITY

GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS

Private University Estd. in Karnataka State by Act No. 41 of 2013

PRESIDENCY GROUP OVER 40 YEARS OF ACADEMIC WISDOM

Composite Two-Dimensional Scaling

- Concatenating transformation matrices for two successive scaling operations in two dimensions produces the following composite scaling

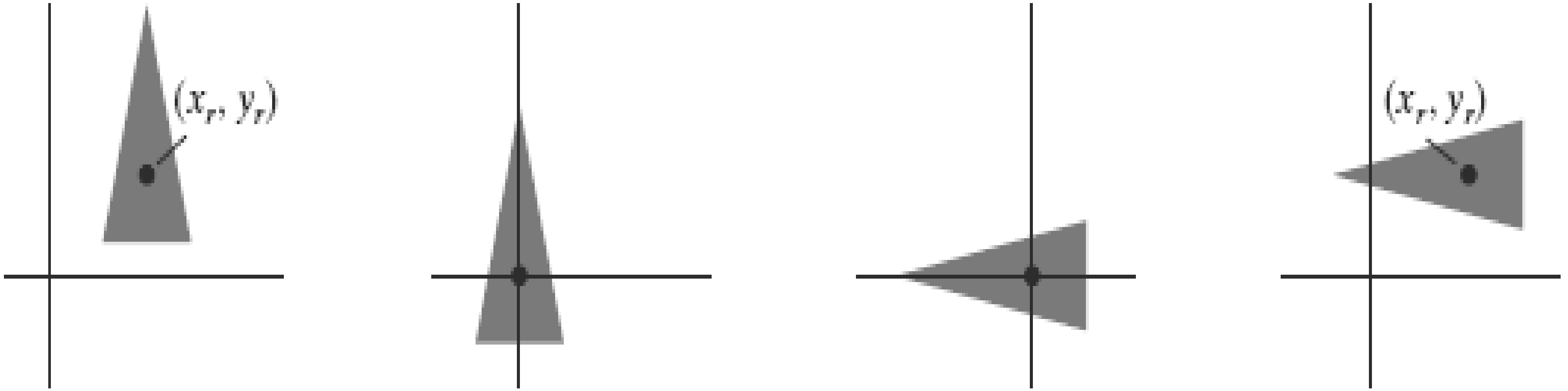
$$\begin{bmatrix} s_{2x} & 0 & 0 \\ 0 & s_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{1x} & 0 & 0 \\ 0 & s_{1y} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{1x} \cdot s_{2x} & 0 & 0 \\ 0 & s_{1y} \cdot s_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{S}(s_{2x}, s_{2y}) \cdot \mathbf{S}(s_{1x}, s_{1y}) = \mathbf{S}(s_{1x} \cdot s_{2x}, \ \ s_{1y} \cdot s_{2y})$$

- **This demonstrates two successive scaling matrix are multiplicative.**

General pivot point rotation and scaling

General Two-Dimensional Pivot-Point Rotation



(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)

General Two-Dimensional Pivot-Point Rotation

- We can generate a two-dimensional rotation about any other pivot point $(x_r\ ,\ y_r\ )$ by performing the following sequence of translate-rotate-translate operations:
- **Translate** the object so that the pivot-point position is moved to the coordinate origin.
- **Rotate** the object about the coordinate origin.
- **Translate** the object so that the pivot point is returned to its original position.
- The composite transformation matrix for this sequence is obtained with the concatenation

$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

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

Refer this for Matrix Multiplication

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

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

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

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

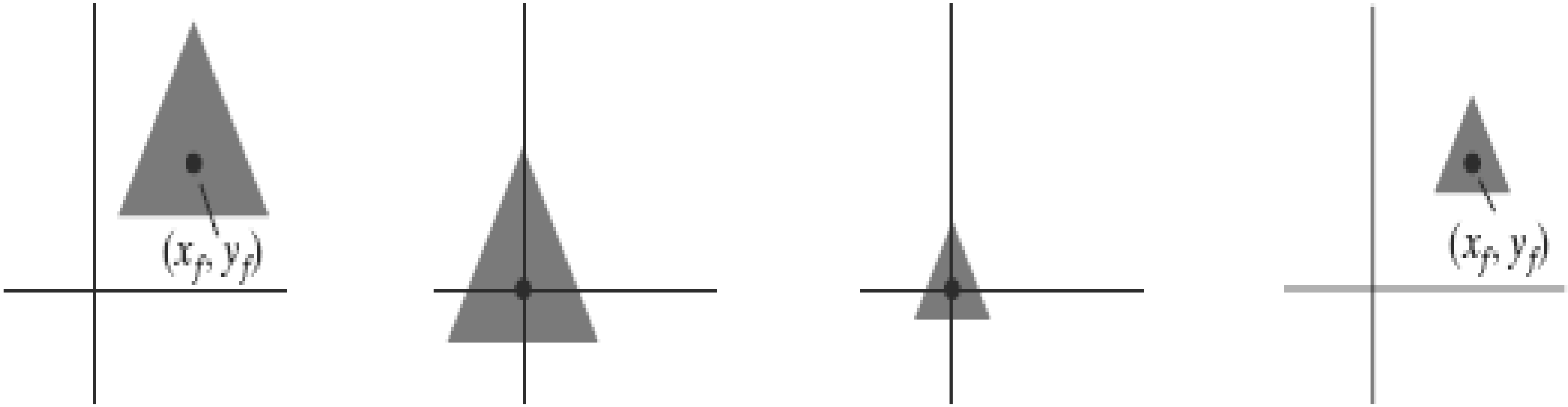
$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta)+y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta)-x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

which can be expressed in the form

$$\mathbf{T}(x_r, y_r) \cdot \mathbf{R}(\theta) \cdot \mathbf{T}(-x_r, -y_r) = \mathbf{R}(x_r, y_r, \theta)$$

where $\mathbf{T}(-x_r\ , -y_r\ ) = \mathbf{T}^{-1}(x_r\ , y_r\ )$

General Two-Dimensional Pivot-Point Scaling



(a) Original Position of Object and Fixed Point

(b) Translate Object so that Fixed Point (x_f, y_f) is at Origin

(c) Scale Object with Respect to Origin

(d) Translate Object so that the Fixed Point is Returned to Position (x_f, y_f)

General Two-Dimensional Pivot-Point Scaling

- To produce a two-dimensional scaling with respect to a selected fixed position (x_f , y_f), when we have a function that can scale relative to the coordinate origin only. This sequence is
- **Translate** the object so that the fixed point coincides with the coordinate origin.
- **Scale** the object with respect to the coordinate origin.
- Use the **inverse of the translation in** step (1) to return the object to its original position.
- Concatenating the matrices for these three operations produces the required scaling matrix:

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

:

$$\mathbf{T}(x_f, y_f) \cdot \mathbf{S}(s_x, s_y) \cdot \mathbf{T}(-x_f, -y_f) = \mathbf{S}(x_f, y_f, s_x, s_y)$$

Refer this for Matrix Multiplication

$$= \begin{bmatrix} S_x+0+0 & 0+0+0 & x_f \\ 0+0+0 & 0+S_y+0 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x & 0 & -S_x \cdot x_f + x_f \\ 0 & S_y & -S_y \cdot y_f + y_f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & x_f(1-S_x) \\ 0 & S_y & y_f(1-S_y) \\ 0 & 0 & 1 \end{bmatrix}$$

OpenGL 2D Geometric Transformation Functions

PRESIDENCY UNIVERSITY
GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS
Private University Estd. in Karnataka State by Act No. 41 of 2013
PRESIDENCY GROUP OVER 40 YEARS OF ACADEMIC WISDOM

Translation matrix is constructed with the following routine:

glTranslate*(tx, ty, tz);

- Translation parameters **tx**, **ty**, and **tz** can be assigned any real-number values, and the single suffix code to be affixed to this function is either **f** (float) or **d** (double).
- For two-dimensional applications, we set **tz** = 0.0;
- example: **glTranslatef (25.0, -10.0, 0.0);**

Rotation matrix is generated with

glRotate*(theta, vx, vy, vz);

- where the vector **v** = (**vx**, **vy**, **vz**) can have any floating-point values for its components defines the orientation for a rotation axis that passes through the coordinate origin.
- The suffix code can be either **f** or **d**, and parameter **theta** is to be assigned a rotation angle in degree
- For example, the statement: **glRotatef (90.0, 0.0, 0.0, 1.0);**

Scaling matrix with respect to the coordinate origin with the following routine:

glScale* (sx, sy, sz);

- The suffix code is again either **f** or **d**, and the scaling parameters can be assigned any real-number values.
- Scaling in a two-dimensional system involves changes in the *x* and *y* dimensions, so a typical two-dimensional scaling operation has a *z* scaling factor of 1.0
- **Example: glScalef (2.0, -3.0, 1.0);**

Basics of 2D viewing and Clipping:

PRESIDENCY UNIVERSITY
GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS
Private University Estd. in Karnataka State by Act No. 41 of 2013
PRESIDENCY GROUP OVER 40 YEARS OF ACADEMIC WISDOM

Basics of 2D viewing :

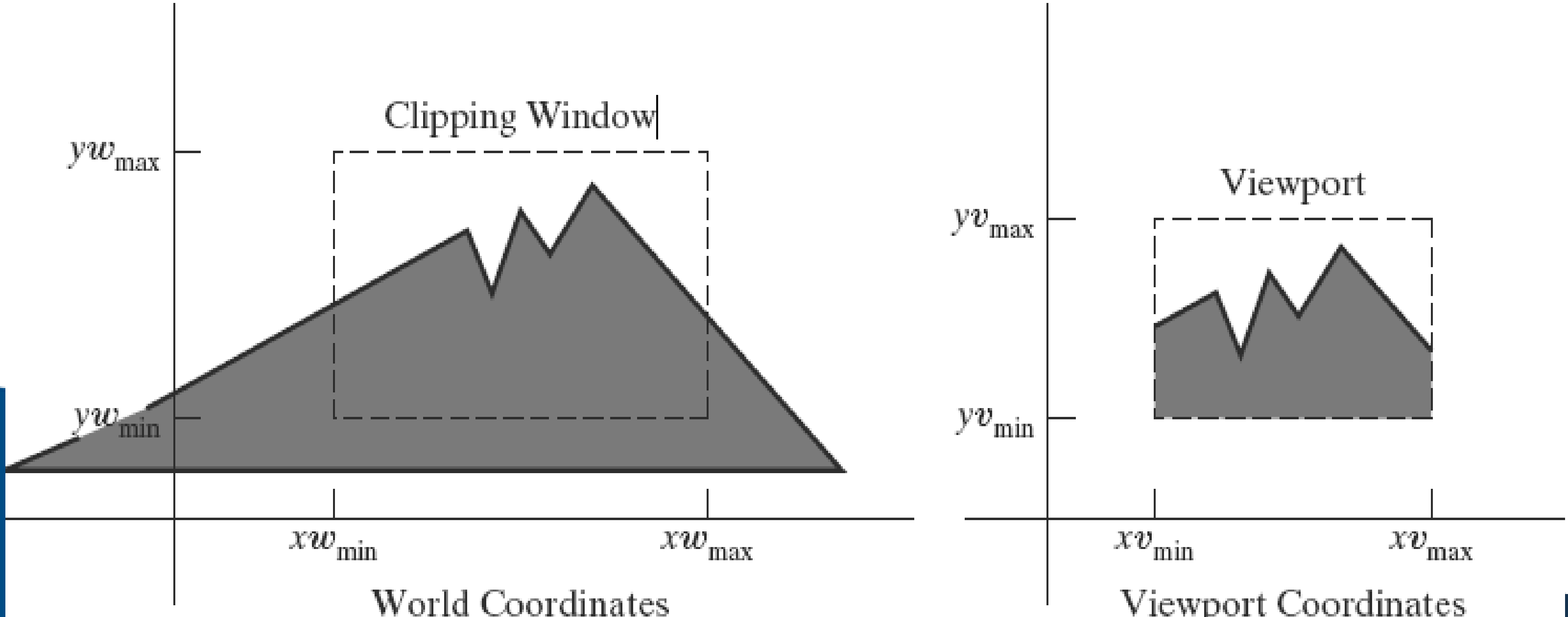
PRESIDENCY UNIVERSITY
GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS
Private University Estd. in Karnataka State by Act No. 41 of 2013
PRESIDENCY GROUP OVER 40 YEARS OF ACADEMIC WISDOM

2D Viewing Pipeline

A section of a two-dimensional scene that is selected for display is called a clipping Window.

- Clipping window is referred as the *world window* or the *viewing window*
- Graphics packages allow us also to control the placement within the display window using another "window" called the viewport.
- **Clipping window-** selects ***what*** we want to see
- **Viewport** – indicates ***where*** it is to be viewed on the output device
- Clipping windows and viewports are rectangles in standard position, with the rectangle edges parallel to the coordinate axes.

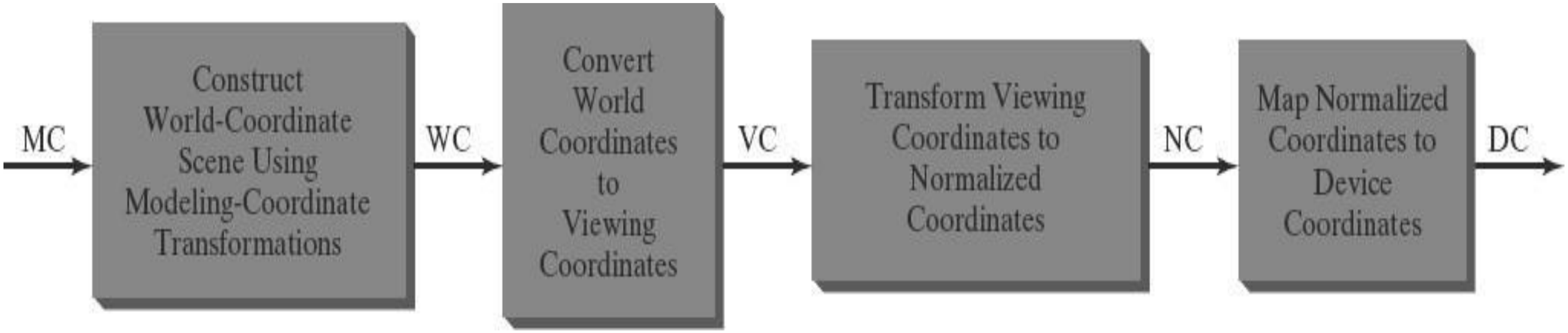
Consider only rectangular viewports and clipping windows, as illustrated in Figure



2D Viewing Pipeline Architecture

- The mapping of a two-dimensional world-coordinate scene description to device coordinates is called a **2D viewing transformation**.
- This transformation is referred as window-to-viewport transformation or the windowing transformation as shown in fig next slide.

2D Viewing Pipeline Architecture

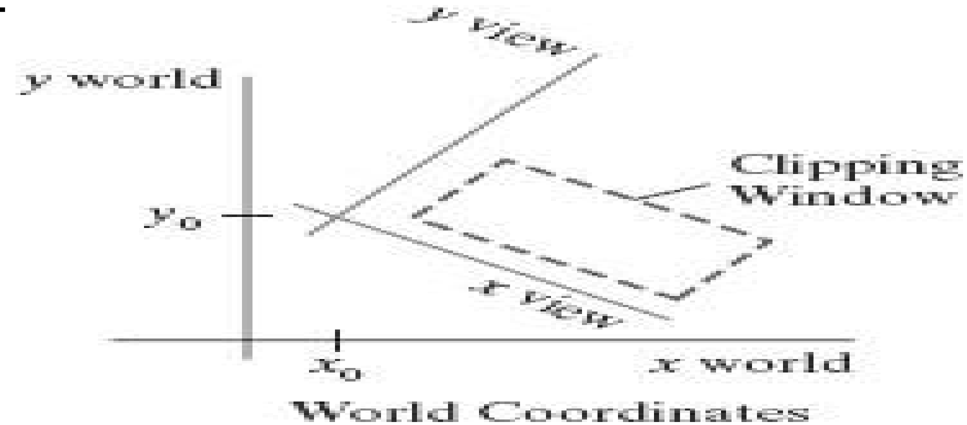


- Once a world-coordinate scene has been constructed, set up a separate 2D **viewing coordinate reference frame** for specifying the clipping window.
- To make the viewing process independent of the requirements of any output device, graphics systems convert object descriptions to normalized coordinates and apply the clipping routines.
- Systems use normalized coordinates in the range from 0 to 1, and others use a normalized ranges from −1 to 1.
- At the final step of the viewing transformation, the contents of the viewport are transferred to positions within the display window.
- Clipping is usually performed in normalized coordinates, allows us to reduce computations by first concatenating the various transformation matrices ranges from −1 to 1.

Viewing Transformation systems :

1. **Viewing-Coordinate Clipping Window**

A general approach to the two-dimensional viewing transformation is to set up a viewing coordinate system within the world-coordinate frame



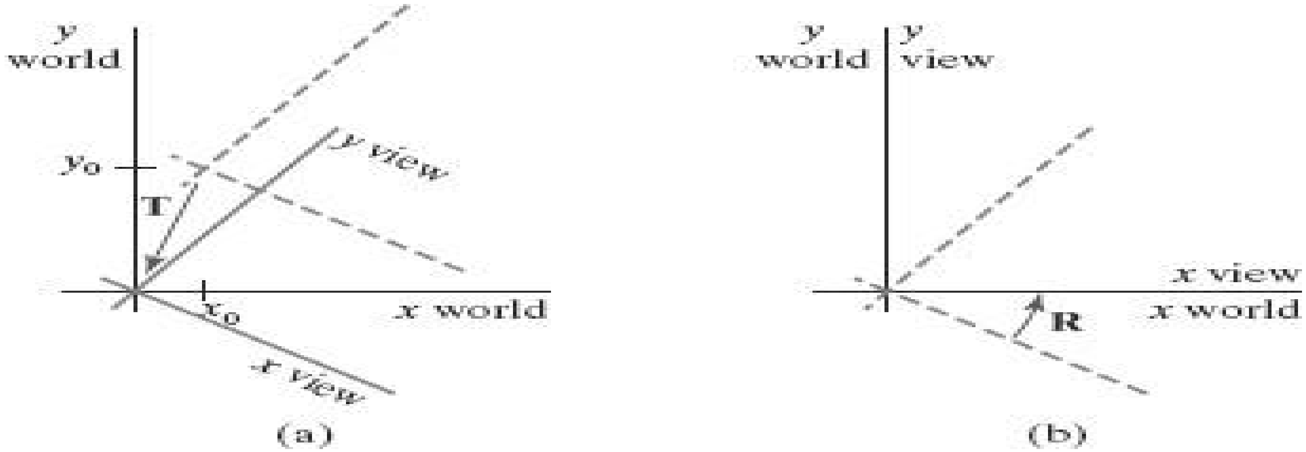
- Choose an origin for a two-dimensional viewing-coordinate frame at some world position P0 = (x0, y0), and establish the orientation using a world vector V that defines the yview direction.
- Vector V is called the two-dimensional view up vector.
- An alternative method is to give a rotation angle relative to either the x or y axis in the world frame.
- The first step in the transformation sequence is to translate the viewing origin to the world origin.

Viewing-Coordinate Clipping Window (cont..)

- Next, we rotate the viewing system to align it with the world frame.
- Given the orientation vector V, we can calculate the components of unit vectors v = (vx, vy) and u = (ux, uy) for the yview and xview axes, respectively.
- Where,

  T is the translation matrix,

  R is the rotation matrix
- A viewing-coordinate frame is moved into coincidence with the world frame is shown in below figure



(a) applying a translation matrix **T** to move the viewing origin to the world origin, then
(b) applying a rotation matrix **R** to align the axes of the two systems.

2. World-Coordinate Clipping Window

- A routine for defining a standard, rectangular clipping window in world coordinates is provided in a graphics-programming library.
- Simply specify two world-coordinate positions, which are then assigned to the two opposite corners of a standard rectangle.
- Once the clipping window has been established, the scene description is processed through the viewing routines to the output device.
- Thus, we simply rotate (and possibly translate) objects to a desired position and set up the clipping window all in world coordinates.

triangle
(a) with a selected reference point and orientation vector, is translated and rotated to position
(b) within a clipping window.

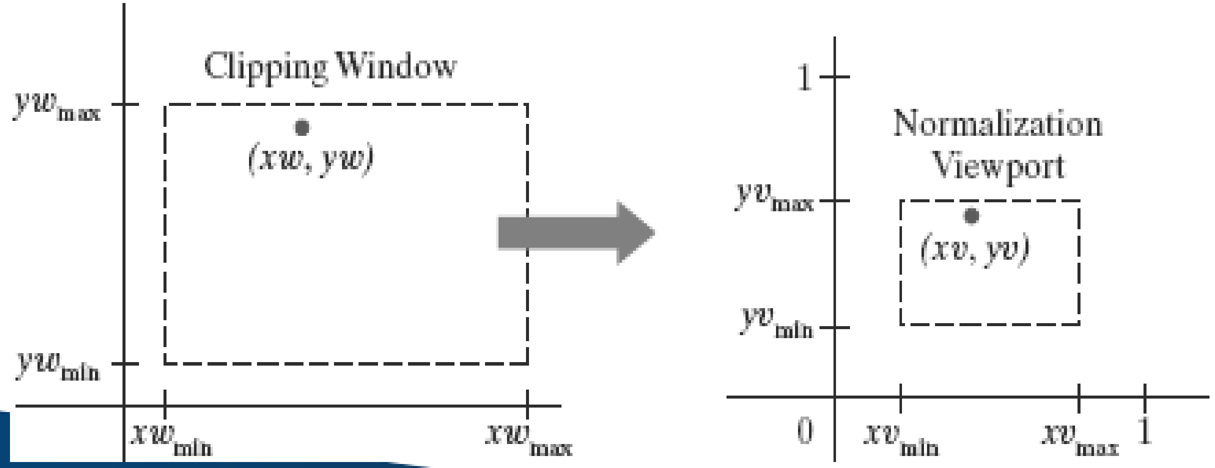
3. Normalization and Viewport Transformations

The viewport coordinates are often given in the range from 0 to 1 so that the viewport is positioned within a unit square.

After clipping, the unit square containing the viewport is mapped to the output display device.

3.1 Mapping the Clipping Window into a Normalized Viewport

- First consider a viewport defined with normalized coordinate values between 0 and 1.
- Object descriptions are transferred to this normalized space using a transformation that maintains the same relative placement of a point in the viewport as it had in the clipping window Position (*xw*, *yw*) in the clipping window is mapped to position (*xv*, *yv*) in the associated viewport.



- To transform the world-coordinate point into the same relative position within the viewport, we require that

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

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

- Solving these expressions for the viewport position (*xv*, *yv*), we have

  xv = sxxw + tx yv = syyw + ty

  Where the scaling factors are

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

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

and the translation factors are

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

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

- We could obtain the transformation from world coordinates to viewport coordinates with the following sequence:

  1. Scale the clipping window to the size of the viewport using a fixed-point position of (xw_{min}, yw_{min}).

  2. Translate (xw_{min}, yw_{min}) to (xv_{min}, yv_{min}).

- The scaling transformation in step (1) can be represented with the two dimensional Matrix

$$\mathbf{S} = \begin{bmatrix} s_x & 0 & xw_{min}(1 - s_x) \\ 0 & s_y & yw_{min}(1 - s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

- The two-dimensional matrix representation for the translation of the lower-left corner of the clipping window to the lower-left viewport corner is

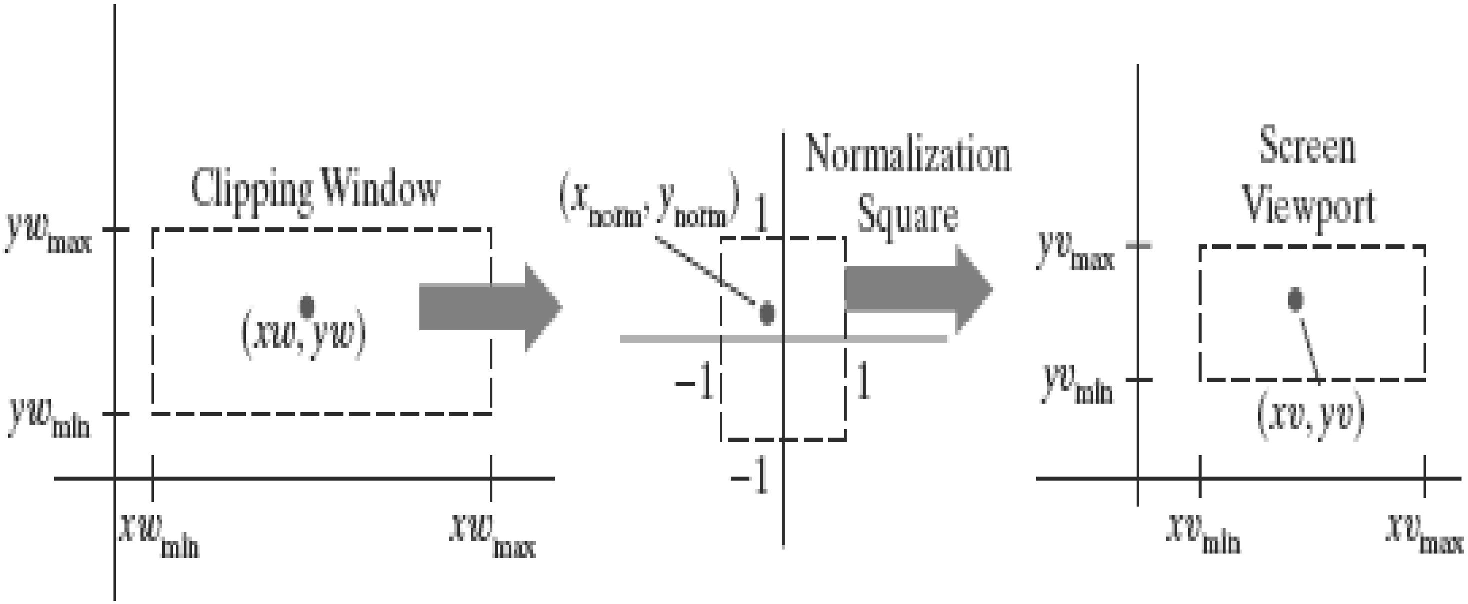
$$\mathbf{T} = \begin{bmatrix} 1 & 0 & xv_{min} - xw_{min} \\ 0 & 1 & yv_{min} - yw_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

- And the composite matrix representation for the transformation to the normalized viewport is

$$\mathbf{M}_{\text{window, normviewp}} = \mathbf{T} \cdot \mathbf{S} = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

2. Mapping the Clipping Window into a Normalized Square

- Another approach to two-dimensional viewing is to transform the clipping window into a normalized square, **clip in normalized coordinates, and then transfer the scene description to a viewport specified in screen coordinates.**
- This transformation is illustrated in Figure below with normalized coordinates in the range from −1 to 1



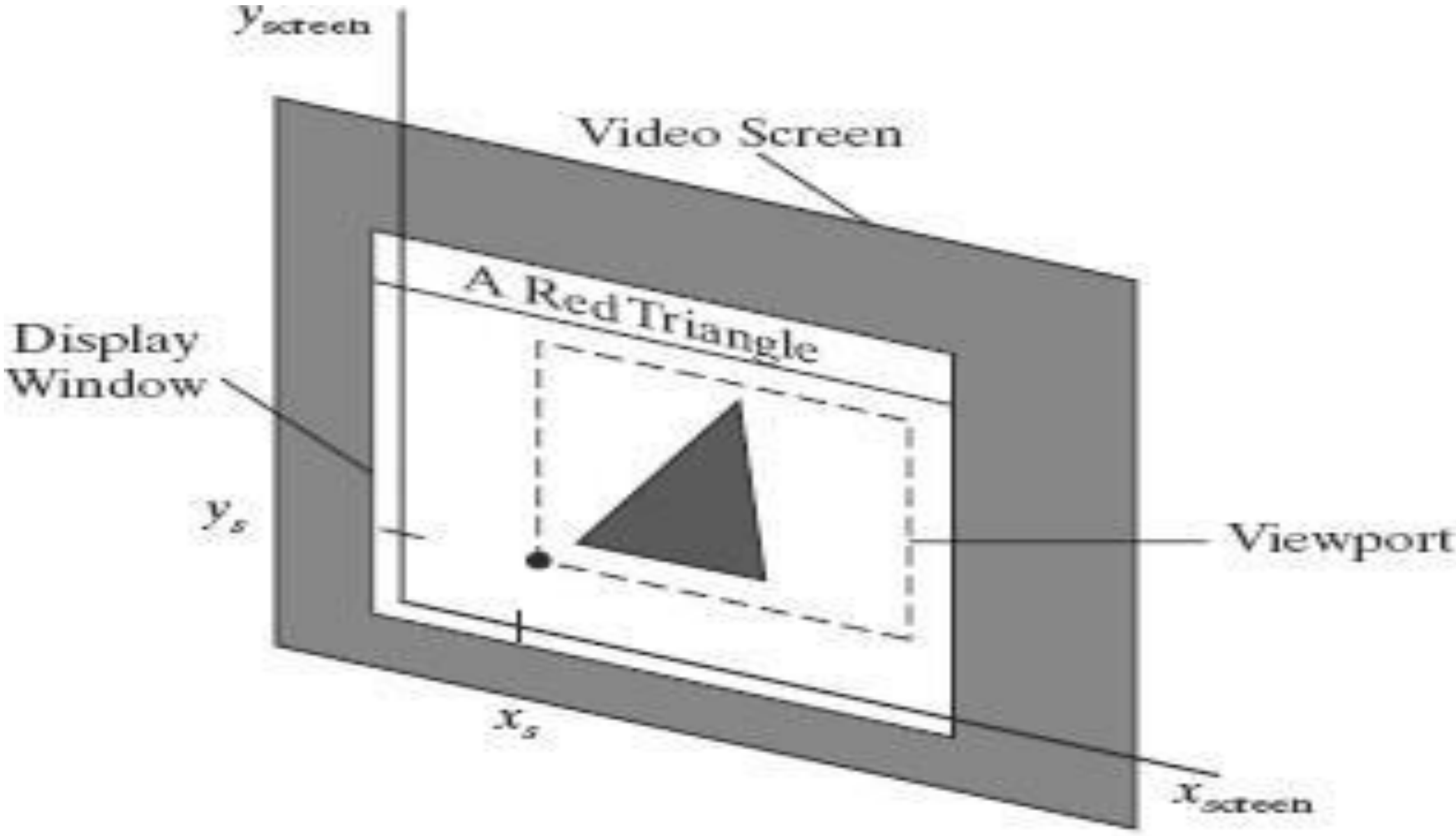
- The matrix for the normalization transformation is obtained by substituting -1 for xv_{min} and yv_{min} and substituting $+1$ for xv_{max} and yv_{max}.

$$\mathbf{M}_{\text{window, normsquare}} = \begin{bmatrix} \frac{2}{xw_{max} - xw_{min}} & 0 & -\frac{xw_{max} + xw_{min}}{xw_{max} - xw_{min}} \\ 0 & \frac{2}{yw_{max} - yw_{min}} & -\frac{yw_{max} + yw_{min}}{yw_{max} - yw_{min}} \\ 0 & 0 & 1 \end{bmatrix}$$

- Similarly, after the clipping algorithms have been applied, the normalized square with edge length equal to 2 is transformed into a specified viewport.
- This time, we get the transformation matrix by substituting -1 for xw_{min} and yw_{min} and substituting $+1$ for xw_{max} and yw_{max}

$$\mathbf{M}_{\text{normsquare, viewport}} = \begin{bmatrix} \frac{xv_{max} - xv_{min}}{2} & 0 & \frac{xv_{max} + xv_{min}}{2} \\ 0 & \frac{yv_{max} - yv_{min}}{2} & \frac{yv_{max} + yv_{min}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

- Typically, the lower-left corner of the viewport is placed at a coordinate position specified relative to the lower-left corner of the display window. Figure demonstrates the positioning of a viewport within a display window.


y_{screen}
Video Screen
A Red Triangle
Display Window
y_s
Viewport
x_s
x_{screen}