Assignment No. 4

Title	Basic 2-D Transformations.
Aim/Problem Statement	a) Write C++ program to draw 2-D object and perform following basic transformations: Scaling, Translation, Rotation. Apply the concept of operator overloading. **OR** b) Write C++ program to implement translation, rotation and scaling transformations on equilateral triangle and rhombus. Apply the concept of operator overloading.
CO Mapped	
Pre-requisite	1. Basic programming skills of C++ 2. 64-bit Open source Linux 3. Open Source C++ Programming tool like G++/GCC
Learning Objective	To learn and apply basic transformations on 2-D objects.

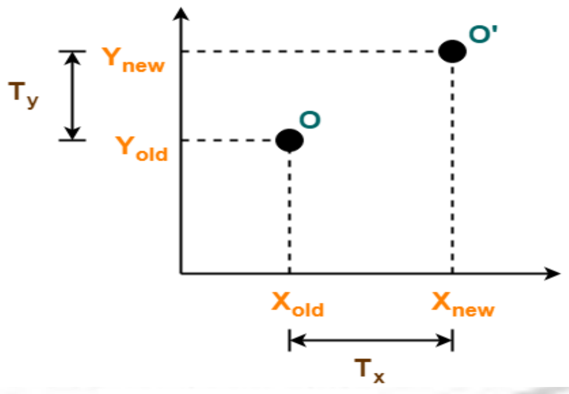
Theory:

Transformation means changing some graphics into something else by applying rules. We can have various types of transformations such as translation, scaling up or down, rotation, shearing, reflection etc. When a transformation takes place on a 2D plane, it is called 2D transformation. Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation. Translation, Scaling and Rotation are basic transformations.

1) Translation:

A translation moves an object to a different position on the screen. You can translate a point in 2D by adding translation coordinate or translation vector (T_x, T_y) to the original coordinates. Consider

- 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)



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-

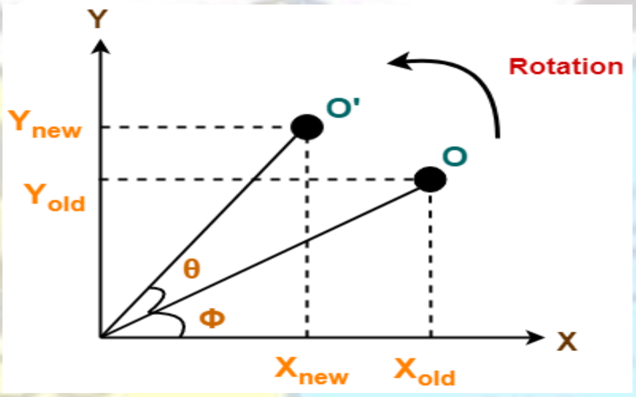
$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

Translation Matrix

2) Rotation:

In rotation, we rotate the object at particular angle θ (theta) from its original position. Consider

- 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})



This anti-clockwise 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-

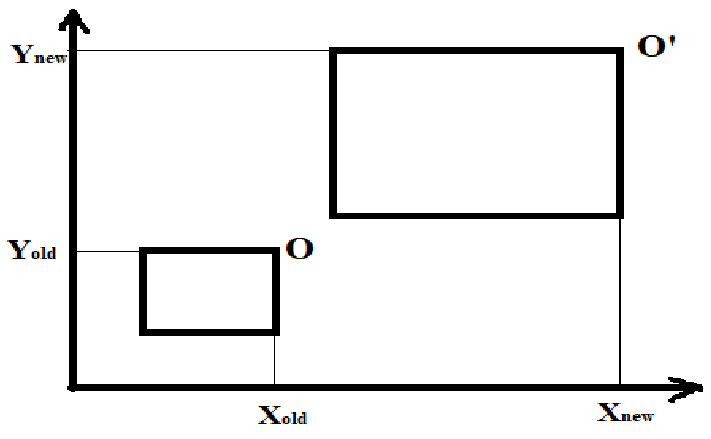
$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Rotation Matrix

3) Scaling:

Scaling transformation is used to change the size of an object. In the scaling process, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor (S_x, S_y). If scaling factor > 1, then the object size is increased. If scaling factor < 1, then the object size is reduced. Consider

- 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

Homogeneous Coordinates:

Matrix multiplication is easier to implement in hardware and software as compared to matrix addition. Hence we want to replace matrix addition by multiplication while performing transformation operations. So the solution is **homogeneous coordinates**, which allows us to express all transformations (including translation) as matrix multiplications.
To obtain homogeneous coordinates we have to represent transformation matrices in 3x3 matrices instead of 2x2. For this we add dummy coordinate. Each 2 dimensional position (x,y) can be represented by homogeneous coordinate as (x,y,1).

Translation Matrix (Homogeneous Coordinates representation)

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

Rotation Matrix (Homogeneous Coordinates representation)

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

Scaling Matrix (Homogeneous Coordinates representation)

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

Applying transformations on equilateral triangle:

Consider that coordinates of vertices of equilateral triangle are (X1,Y1), (X2,Y2) and (X3,Y3). After applying basic transformations, we will get corresponding coordinates as (X1',Y1'), (X2',Y2') and (X3',Y3') respectively. Following multiplication will give the translation on this equilateral triangle:

$$\begin{bmatrix} X1' & X2' & X3' \\ Y1' & Y2' & Y3' \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X1 & X2 & X3 \\ Y1 & Y2 & Y3 \\ 1 & 1 & 1 \end{bmatrix}$$

Similarly we can apply rotation and scaling on equilateral triangle.

Applying transformations on rhombus:

Consider that coordinates of vertices of rhombus are (X1,Y1), (X2,Y2), (X3,Y3) and (X4,Y4) applying basic transformations, we will get corresponding coordinates as (X1',Y1'), (X2',Y2'), (X3',Y3') and (X4',Y4') respectively. Following multiplication will give the translation on this rhombus:

$$\begin{bmatrix} X1' & X2' & X3' & X4' \\ Y1' & Y2' & Y3' & Y4' \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X1 & X2 & X3 & X4 \\ Y1 & Y2 & Y3 & Y4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Similarly we can apply rotation and scaling on rhombus.

Conclusion:

Questions:

1. How to rotate any 2-D object about an arbitrary point? Explain in brief.
2. Explain the concept of operator overloading with example.