Assignment No - 9

Name :- Sameer Manoj Bramhecha
Roll No : - 21115
Batch :- E-1
Date of performance :- 26/11/2021
Date of Submission :- 30/11/2021

Title :→ Basic 2-D Transformation.

Problem Statement :→
Write a C++ program to draw 2-D object and perform following basic transformations, (a) Scaling, (b) Translation, (c) Rotation. Apply the concept of operator overloading.

Learning Objectives :-
To learn and apply basic transformations on 2-D objects.

Learning Outcomes :→ After the completion of this assignment students will be able to implement basic 2-D transformations.

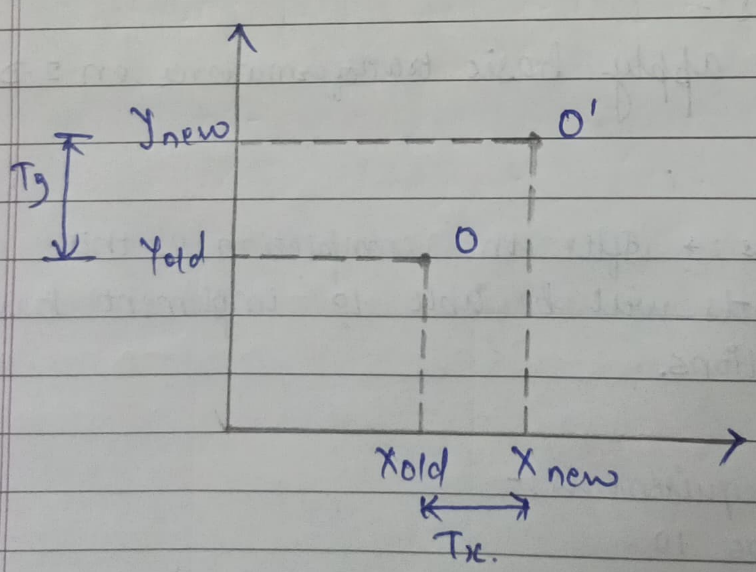
S/W and H/W requirements :-
1> 64-bit Windows 10.
2> Open-source C++ programming tool like G++/GCC
3> Qt Creator

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 :-

- 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 ~~the~~ adding the translation coordinates to the old coordinates of the object as →

$X_{new} = X_{old} + T_x$.

$Y_{new} = Y_{old} + T_y$

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

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

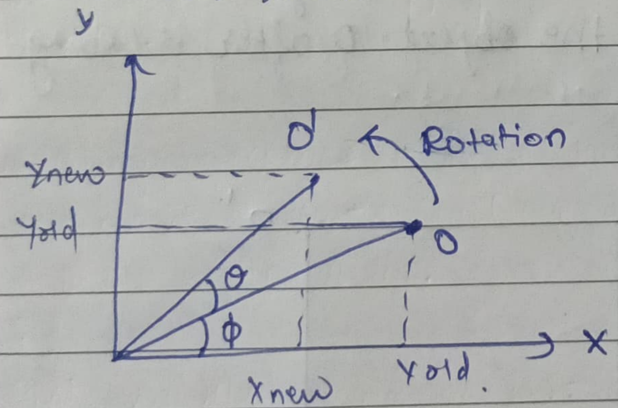
2.] Rotation :→

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 -

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

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

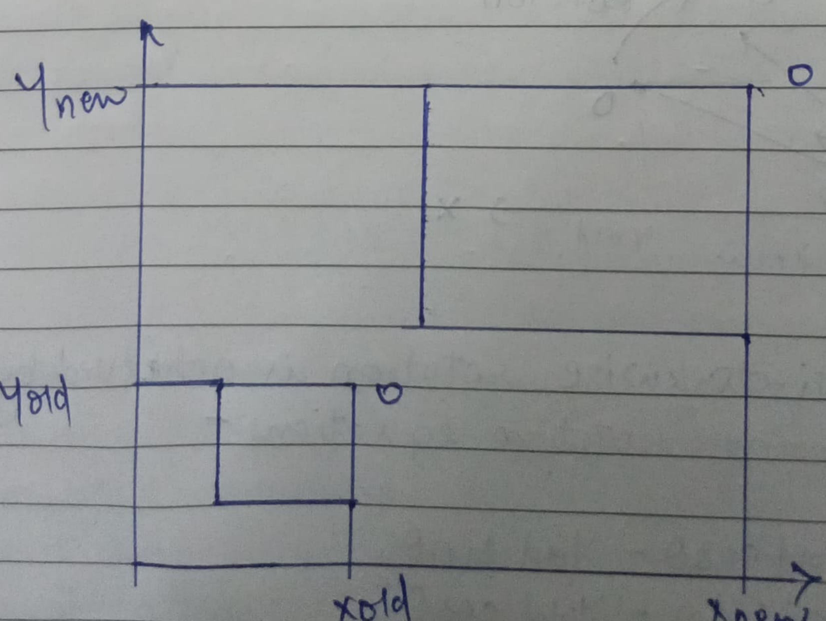
In Matrix form, the above rotation equations may be represented as —

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

8.] Scaling :→

Scaling transformation is used to change the size of an object. In the scaling factor, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with 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 equation -

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

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

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

Algorithm:-
1.) Translation

1.) Start
2.) Accept x_1, y_1 (initial coordinates), length 'l' and breadth 'b' of rectangle from user
3.) Accept T_x and T_y from user.
4.) Shift all vertices of rectangle by factor T_x in x-direction and T_y in y-direction.
5.) Display the translated object.
6.) Stop.

2) Rotation:-

1.) Start
2.) Read ~~verti~~ intial vertices, length 'l' and breadth 'b' from user.
3.) Read rotation angle 'θ' from user.
4.) Convert 'θ' from degrees to radians.
5.) ~~Shift~~ Rotate all ~~vertices~~ edges of rectangle by θ angle.

6.) Display the rotated ~~angle~~ object
7.) Stop.

8.) Scaling:-

1.) Start
2.) Accept initial coordinates, length 'l' & breadth 'b' from user.
3.) Read sx and sy from user.
4.) Define matrix 'a' = $\begin{bmatrix} sx & 0 \\ 0 & sy \end{bmatrix}$

5.) Define matrix 'c' = $\begin{bmatrix} l \\ b \end{bmatrix}$

6.) Using overloaded operator *, multiply a and c.

7.) Display the scaled object.
8.) Stop.

Conclusion:- We have successfully implemented basic 2-D transformations.