Assignment No - 7

Title :- 2-D transformation

Problem statement :-
write C++ / Java program to draw 2-D object & perform following basic transformation
a) Scaling
b) Translation
c) Rotation

Objective :- To implement the 2-D transformation with translation Scaling & rotation of 2-D object

S/w used :- cpp, Qt Creator

Theory :-

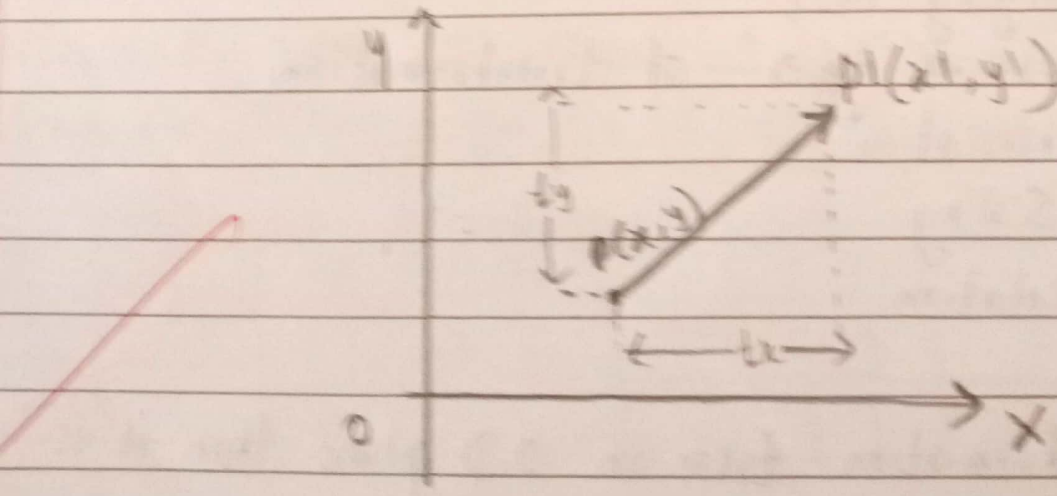
- transformation means changing some graphics into something else by applying rules.
- we have various types of transformation
  i) translation
  ii) Scaling
  iii) Rotation.
- when transformation take on 2-D plane then it is called 2-D transformation.

• Homogeneous coordinates :-

- To perform a sequence of transformation such as translation followed by rotation & scaling
- To shorten this process, we have to use 3×3 transformation matrix instead of 2×2 transformation.
- To convert to 2×2 into 3×3 matrix we need to add extra dummy coordinate W.
- In this way, we can represent the coordinate point by 3 number instead of 2 which is Homogeneous coordinate system.

• translation :-

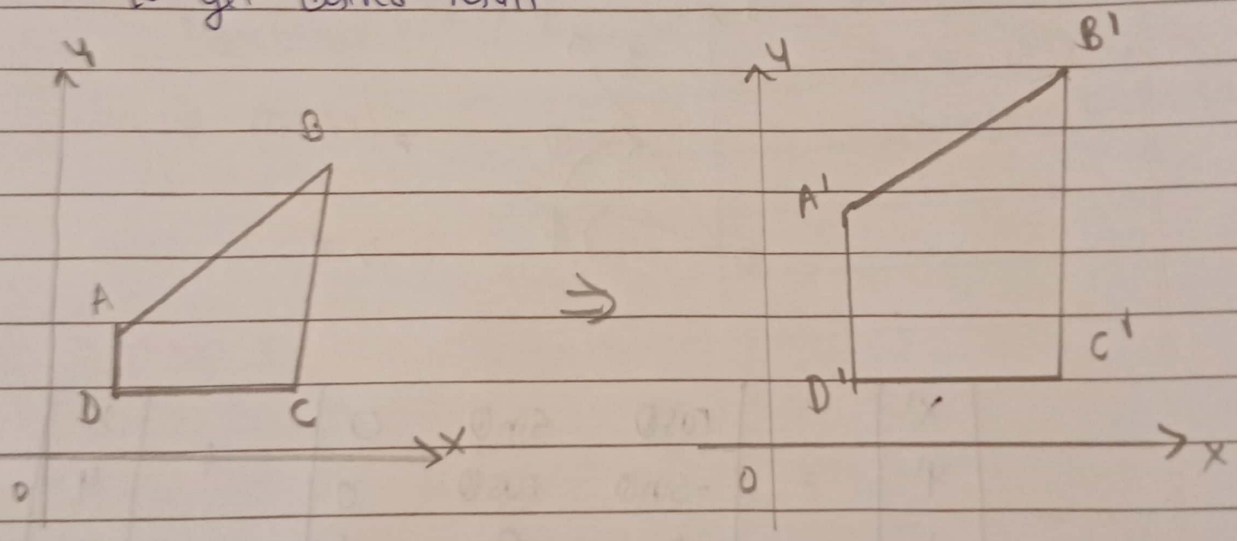
A translation moves an object to a different position on the screen. you can translate a point in 2D by adding translation coordinate (tx, ty) to original coordinate X, Y to X', Y'



$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

- **Scaling :-**

To change the size of an object, scaling transformation is used. In scaling process, you either expand or compress the dimension of the object. Scaling can be achieved by multiplying the original coordinates of the object with scaling factor to get desired result.



If we provide value less than 1 to the Scaling fa
then we can reduce the size of object & vice versa.

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

- Pseudo code for multiplication of matrices :-

```
procedure MatrixMulti ( coordinateMatrix , transformationMatrix )
    input coordinateMatrix , transformationMatrix
    output transformedMatrix

  for (i=0; i<n; i++)
    for (j=0 ; j<n; j++)
      transformedMatrix[i][j]:=0
        for (k=0 ; k< n ; k++)

  transformedMatrix[i][j]= transformedMatrix[i][j] +
                           coordinateMatrix[i][k] *
                           transformationMatrix[

        end for
    end for
  end for
end MatrixMulti
```

conclusion :-

we have learn & implement 2-D transformation with scaling, Rotation & translation on 2-D ob