**Aim:**  To implement 2D Transformations: Translation, Scaling, Rotation.

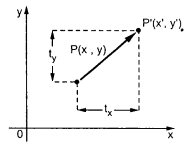
**Objective:**

To understand the concept of transformation, identify the process of transformation and application of these methods to different object and noting the difference between these transformations.

**Theory:**

**1) Translation –**

Translation is defined as moving the object from one position to another position along straight line path. We can move the objects based on translation distances along x and y axis. tx denotes translation distance along x-axis and ty denotes translation distance along y axis.

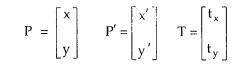


Consider (x,y) are old coordinates of a point. Then the new coordinates of that same point (x’,y’) can be obtained as follows:

**x’ = x + tx**

**y’ = y + ty**

We denote translation transformation as P. we express above equations in matrix form as:  
P’ = P + T , where



**Program:**

**#include<stdio.h>**

**#include<conio.h>**

**#include<graphics.h>**

**#include<math.h>**

**void main()**

**{**

**int gd =DETECT,gm,ch,sx,sy,tx,ty,nx1,nx2,ny1,ny2;**

**double r,t;**

**initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");**

**line(100,100,200,100);**

**printf("Transition ");**

**printf("enter trans factor \n");**

**scanf("%d%d",&tx,&ty);**

**nx1=100+tx;**

**ny1=100+ty;**

**nx2=200+tx;**

**ny2=100+ty;**

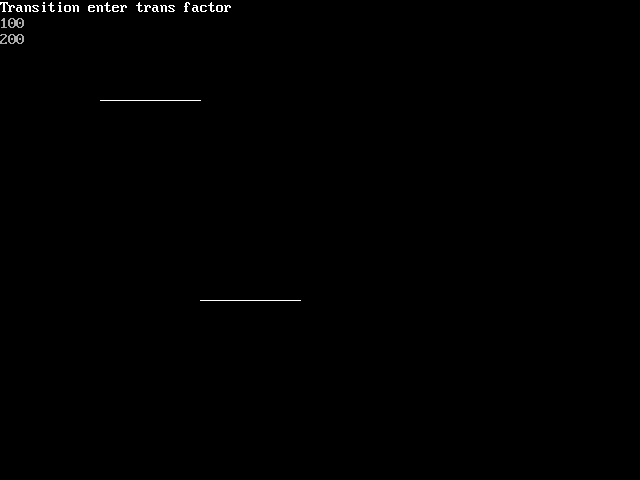
**line(nx1,ny1,nx2,ny2);**

**getch();**

**closegraph();**

**}**

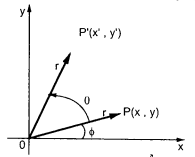
**Output –**



**2) Rotation –**

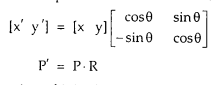
A rotation repositions all points in an object along a circular path in the plane centered at the

pivot point. We rotate an object by an angle theta. New coordinates after rotation depend on both x and y.





The above equations can be represented in the matrix form as given below



where R is the rotation matrix and it is given as



**Program:**

**#include<stdio.h>**

**#include<conio.h>**

**#include<graphics.h>**

**#include<math.h>**

**void main()**

**{**

**int gd =DETECT,gm,ch,sx,sy,tx,ty,nx1,nx2,ny1,ny2;**

**double r,t;**

**initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");**

**line(100,100,200,100);**

**printf("Rotation ");**

**printf("enter angle");**

**scanf("%lf",&r);**

**t=(3.14\*r)/180;**

**nx1=(int)(100+(200-100)\*cos(t)-(100-100)\*sin(t));**

**ny1=(int)(100+(200-100)\*sin(t)+(100-100)\*cos(t));**

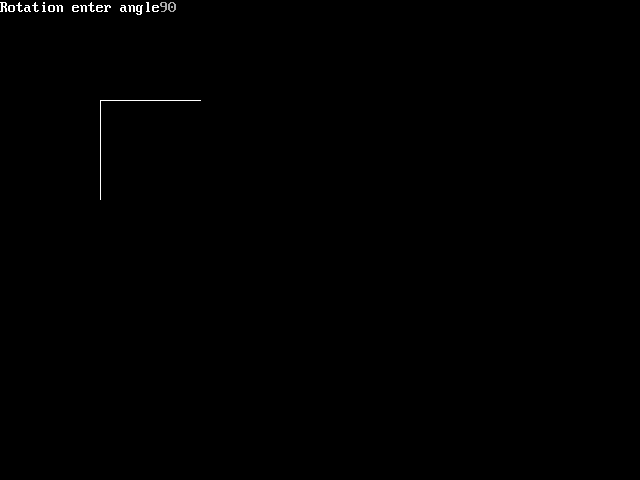
**line(100,100,nx1,ny1);**

**getch();**

**closegraph();**

**}**

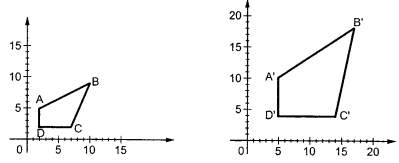
**Output:**



**3) Scaling -**

scaling refers to changing the size of the object either by increasing or decreasing. We will

increase or decrease the size of the object based on scaling factors along x and y-axis.



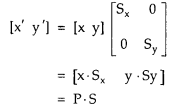
If (x, y) are old coordinates of object, then new coordinates of object after applying scaling

transformation are obtained as:

x’ = x \* Sx

y’ = y \* Sy

Sx and Sy are scaling factors along x-axis and y-axis. we express the above equations in matrix form as:



**Program:**

**#include<stdio.h>**

**#include<conio.h>**

**#include<graphics.h>**

**#include<math.h>**

**void main()**

**{**

**int gd =DETECT,gm,ch,sx,sy,tx,ty,nx1,nx2,ny1,ny2;**

**double r,t;**

**initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");**

**line(100,100,200,100);**

**printf("Scaling ");**

**printf("enter scaling factor \n");**

**scanf("%d%d",&sx,&sy);**

**nx1=100+sx;**

**ny1=100+sy;**

**nx2=200+sx;**

**ny2=100+sy;**

**line(nx1,ny1,nx2,ny2);**

**getch();**

**closegraph();**

**}**

**Output –**



**Conclusion:** Comment on :

1. **Application of Transformations**:
   * 2D transformations are fundamental in computer graphics and image processing.
   * **Translation**: Used for moving objects within an image or on a screen, such as dragging and dropping icons.
   * **Scaling**: Applied for resizing objects, zooming in/out, or adjusting the size of elements.
   * **Rotation**: Essential for tasks like rotating images, elements, or shapes.
2. **Difference Noted Between Methods**:
   * **Translation**: Involves changing the coordinates of an object by adding/subtracting values to its x and y coordinates.
   * **Scaling**: Adjusts the size of an object by multiplying its coordinates by scale factors.
   * **Rotation**: Rotates objects around a specified point (often the origin) by changing their angles.
3. **Applications to Different Objects**:
   * **Translation**: Useful for moving text, images, or any graphical element on a screen.
   * **Scaling**: Applied to images, fonts, and icons to control their size.
   * **Rotation**: Commonly used for rotating images, graphics, and shapes to achieve desired orientations.