Program 8

Aim: Perform the following 2D Transformation operation:

a) Translation

b) Rotation

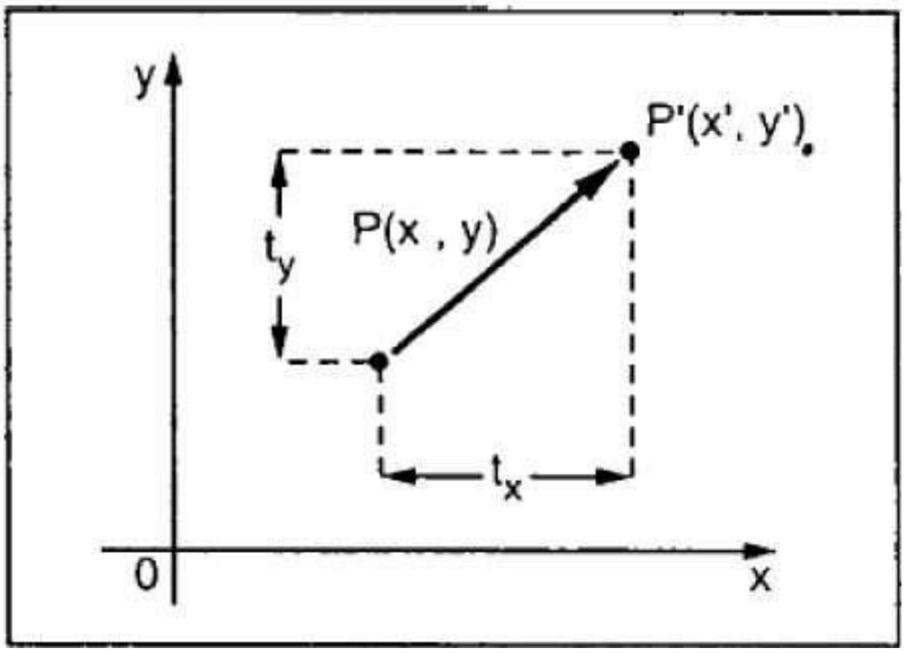
c) Scaling

d) Sheering

Theory:

Translation

A translation moves an object to a different position on the screen. A point in 2D can be translated by adding translation coordinate (tx, ty) to the original coordinate (X, Y) to get the new coordinate (X’, Y’).



From the above figure,

X’ = X + tx

Y’ = Y + ty

The pair (tx, ty) is called the translation vector or shift vector. The above equations can also be represented using a translation matrix.

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

In this experiment, the 3 coordinated of a triangle are given along with the translation factor. We need to find the new coordinates and plot the translated triangle with respect to the original triangle.

Rotation

It is a process of changing the angle of the object. Rotation can be clockwise or anticlockwise. For rotation, we have to specify the angle of rotation and rotation point. Rotation point is also called a pivot point. It is print about which object is rotated. The positive value of the pivot point (rotation angle) rotates an object in a counter-clockwise (anti-clockwise) direction. The negative value of the pivot point (rotation angle) rotates an object in a clockwise direction. When the object is rotated, then every point of the object is rotated by the same angle.

Scaling

Transformation is a process of converting the original picture coordinates into a different set of picture coordinates by applying certain rules to change their position and/or structure. When a transformation takes place on a 2D plane, it is called 2D transformation. Scaling is a type of linear transformation which is used to change the size of an object. In the scaling process, the dimensions of the object are either expanded or compressed. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result. When (Sx, Sy) <1, the image is compressed, else it is enlarged.

Let us assume that in a two-dimensional system, the original coordinates are (X, Y), the scaling factors are (Sx, Sy), and the produced coordinates are (X’, Y’).

This can be mathematically represented as shown below:

$$\mathbf{X' = X.\ Sx \text{ and } Y' = Y.\ Sy}$$

The scaling factor Sx, Sy scales the object in X and Y direction respectively. The above equations can also be represented in matrix form as below –

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shearing

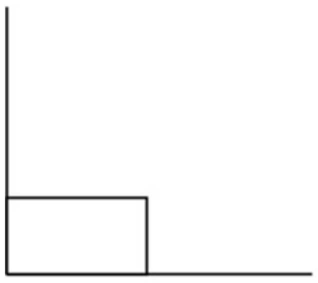
It is transformation which changes the shape of object. The sliding of layers of object occurs. The shear can be in one direction or in two directions.

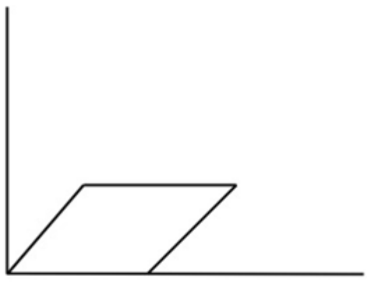
Shearing in the X-direction: In this horizontal shearing sliding of layers occur. The homogeneous matrix for shearing in the x-direction is shown below:

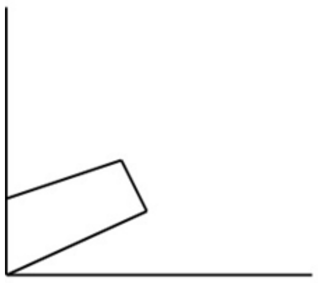
$$\begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

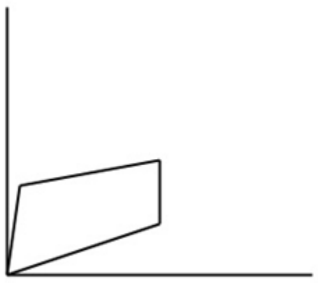
Shearing in the Y-direction: Here shearing is done by sliding along vertical or y-axis.

$$\begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$


Original Object


Shear in X direction


Shear in Y direction


Shear in both directions

Algorithm:

Translation

Step 1 - Start

Step 2 - Initialize the graphics mode.

Step 3 - Construct the Triangle using the three coordinates to draw three lines.

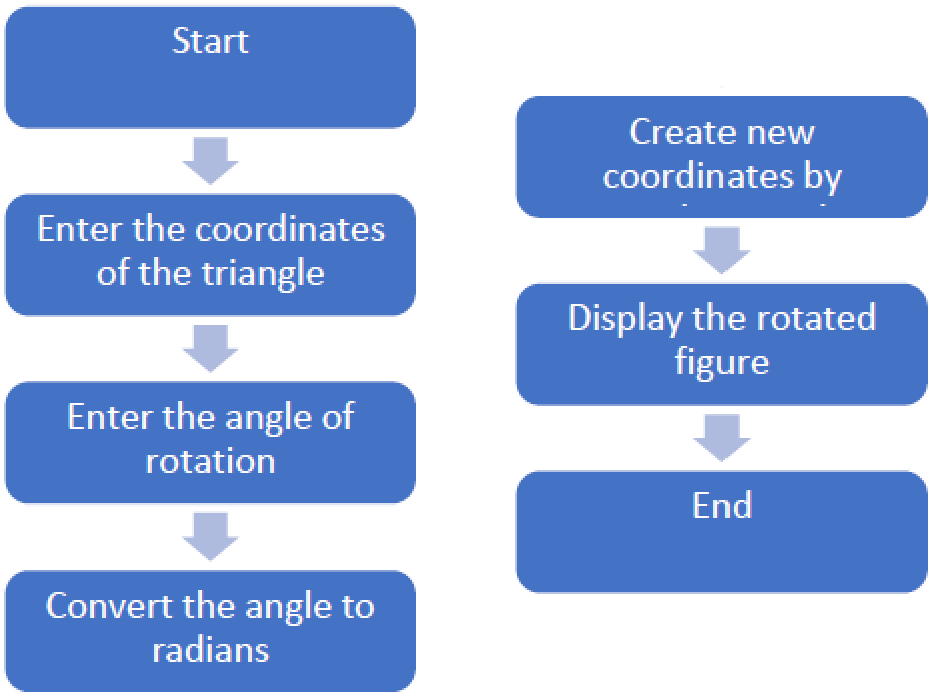
Step 4 - Translation

a) Get the translation value tx, ty

b) Move the 2d object with tx, ty (x'=x+tx,y'=y+ty for all the three points).

c)Plot the new three points, i.e. (x',y')

Rotation


Start
Enter the coordinates of the triangle
Enter the angle of rotation
Convert the angle to radians
Create new coordinates by
Display the rotated figure
End

Scaling

Step 1 - Start

Step 2 - Initialize the graphics mode.

Step 3 - Plot a 2D object using given coordinates (x,y)

Step 4 - Get the scaling value Sx,Sy

Step 5 - Resize the object with Sx,Sy (x’=x*Sx,y’=y*Sy)

Step 6 - Plot the new object using new coordinates (x’,y’)

Code:

Translation

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>

int main(){
    int gm, gd = DETECT;
    initgraph(&gd, &gm, "C:\\TurboC3\\BGI");

    int x1, x2, x3, y1, y2, y3;
    printf("2D Transformation operation\nTranslation\nEnter the
 points of triangle\n");
    scanf("%d%d%d%d%d%d", &x1, &y1, &x2, &y2, &x3, &y3);
    printf("Enter translation factor\n");
    int tx, ty;
    scanf("%d %d", &tx, &ty);

    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x1, y1, x3, y3);

    int nx1, nx2, nx3, ny1, ny2, ny3;
    nx1 = x1 + tx;
    nx2 = x2 + tx;
    nx3 = x3 + tx;
    ny1 = y1 + ty;
    ny2 = y2 + ty;
    ny3 = y3 + ty;
    line(nx1, ny1, nx2, ny2);
    line(nx2, ny2, nx3, ny3);
    line(nx3, ny3, nx1, ny1);

    getch();
    closegraph();
    return 0;
}
```

Rotation

```
#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <math.h>

int main(){
    int gm, gd = DETECT;
    initgraph(&gd, &gm, "C:\\TurboC3\\BGI");

    int x1, x2, x3, y1, y2, y3;
    printf("2D Transformation operation\nRotation\nEnter the po
ints of triangle\n");
    scanf("%d%d%d%d%d%d", &x1, &y1, &x2, &y2, &x3, &y3);
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x1, y1, x3, y3);

    printf("Enter the angle of rotation:\n");
    int rq;
    scanf("%d", &rq);

    rq = 3.14 * rq / 180;
    int nx1, nx2, nx3, ny1, ny2, ny3;
    nx1 = abs(x1 * cos(rq) + y1 * sin(rq));
    ny1 = abs(x1 * sin(rq) + y1 * cos(rq));
    nx3 = abs(x3 * cos(rq) + y3 * sin(rq));
    ny2 = abs(x2 * sin(rq) + y2 * cos(rq));
    nx2 = abs(x2 * cos(rq) + y2 * sin(rq));
    ny3 = abs(x3 * sin(rq) + y3 * cos(rq));

    line(nx1, ny1, nx2, ny2);
    line(nx2, ny2, nx3, ny3);
    line(nx3, ny3, nx1, ny1);

    getch();
    closegraph();
    return 0;
}
```

Scaling

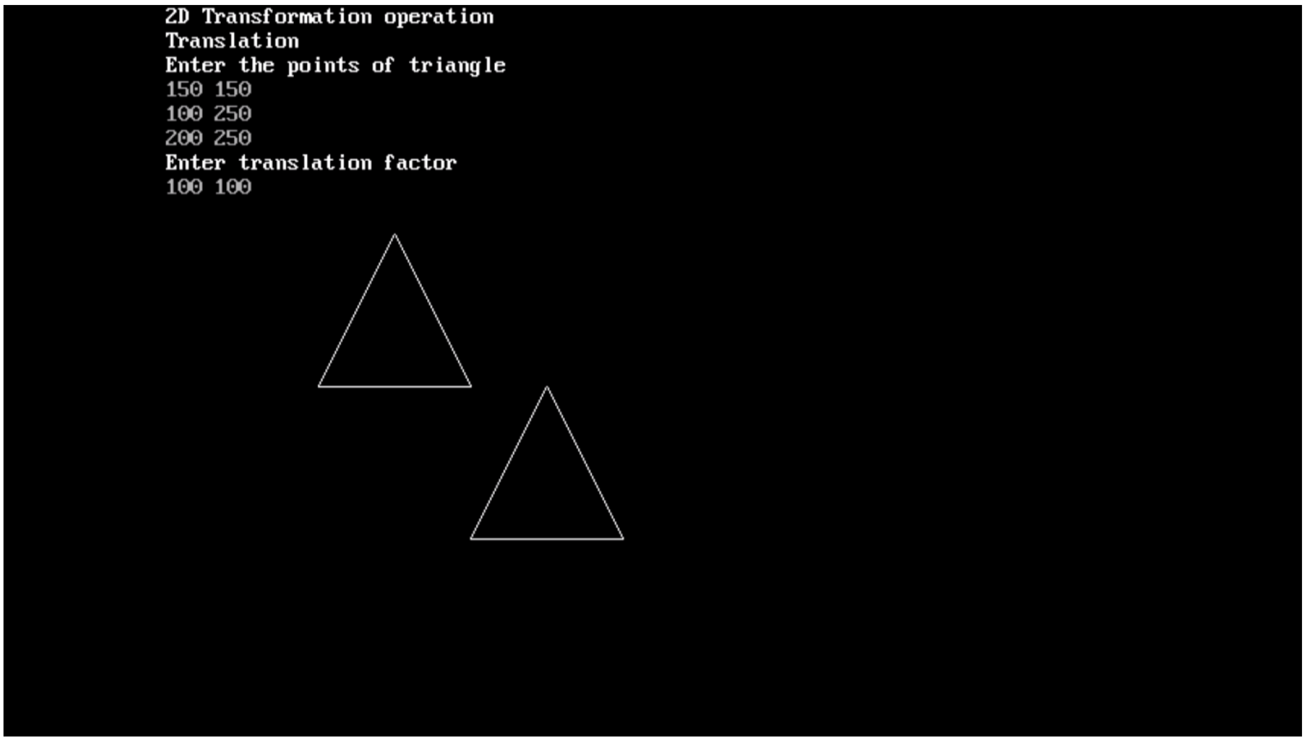
```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>

int PolygonPoints[4][2] = {{10, 10}, {10, 100}, {100, 100}, {10
0, 10}};
float Sx = 0.5;
float Sy = 2.0;

void PolyLine(){
    cleardevice();
    line(0, 240, 640, 240);
    line(320, 0, 320, 480);
    for(int itr=0; itr<4; itr++){
        line(320 + PolygonPoints[itr][0], 240 - PolygonPoints[i
tr][1], 320 + PolygonPoints[(itr + 1) % 4][0], 240 - PolygonPoi
nts[(itr + 1) % 4][1]);
    }
}
void Scale(){
    int itr;
    int Tx, Ty;
    cout<<endl;
    for(itr=0; itr<4; itr++){
        PolygonPoints[itr][0] *= Sx;
        PolygonPoints[itr][1] *= Sy;
    }
}

void main(){
    int gDriver = DETECT, gMode;
    int itr;
    initgraph(&gDriver, &gMode, "C:\\TURBOC3\\BGI");
    cout<<"2D Transformation operation\nScaling\n";
    PolyLine();
    getch();
    Scale();
    PolyLine();
    getch();
}
```

Shearing

```
#include<iostream.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<dos.h>

void mul(int mat[3][3],int points[10][3],int n);
void shear(int points[10][3],int n);
void init(int points[10][3],int n);

int main(){
    int i,x,y;
    int points[10][3],n;
    clrscr();
    cout<<"2D Transformation operation\nShearing\nEnter the no.
 of points : ";
    cin>>n;
    for(i=0;i<n;i++){
        cin>>x>>y;
        points[i][0]=x;
        points[i][1]=y;
        points[i][2]=1;
    }
    shear(points,n);
    getch();
    return 0;
}

void init(int points[10][3],int n){
    int gd=DETECT,gm,i;
    initgraph(&gd, &gm, "C:\\TurboC3\\BGI");
    setcolor(10);
    line(0,240,640,240);
    line(320,0,320,480);
    setcolor(3);
    for(i=0;i<n-1;i++){
        line(320+points[i][0],240-
points[i][1],320+points[i+1][0],240-points[i+1][1]);
    }
    line(320+points[n-1][0],240-points[n-
1][1],320+points[0][0],240-points[0][1]);
}

void mul(int mat[3][3],int points[10][3],int n){
    int i,j,k;
    int res[10][3];
    for(i=0;i<n;i++){
        for(j=0;j<3;j++){
            res[i][j]=0;
            for(k=0;k<3;k++){
                res[i][j] = res[i][j] + points[i][k]*mat[k][j];
            }
        }
    }
    setcolor(15);
    for(i=0;i<n-1;i++){
       line(320+res[i][0],240-res[i][1],320+res[i+1][0],240-
res[i+1][1]);
    }
    line(320+res[n-1][0],240-res[n-1][1],320+res[0][0],240-
res[0][1]);
}
void shear(int points[10][3],int n){
    int opt, arr_shear[3][3];
    cout<<"\n1.X-shear\n2.Y-shear\nYour Choice: ";
    cin>>opt;
    switch(opt){
        case 1:
            int xsh;
            cout<<"\nEnter the x shear : ";
            cin>>xsh;
            arr_shear[0][0]=1;
            arr_shear[1][0]=xsh;
            arr_shear[2][0]=0;
            arr_shear[0][1]=0;
            arr_shear[1][1]=1;
            arr_shear[2][1]=0;
            arr_shear[0][2]=0;
            arr_shear[1][2]=0;
            arr_shear[2][2]=1;
            init(points,n);
            mul(arr_shear,points,n);
        break;
        case 2:
            int ysh;
            cout<<"\nEnter the y shear : ";
            cin>>ysh;
            arr_shear[0][0]=1;
            arr_shear[1][0]=0;
            arr_shear[2][0]=0;
            arr_shear[0][1]=ysh;
            arr_shear[1][1]=1;
            arr_shear[2][1]=0;
            arr_shear[0][2]=0;
            arr_shear[1][2]=0;
            arr_shear[2][2]=1;
            init(points,n);
            mul(arr_shear,points,n);
        break;
    }
}
```

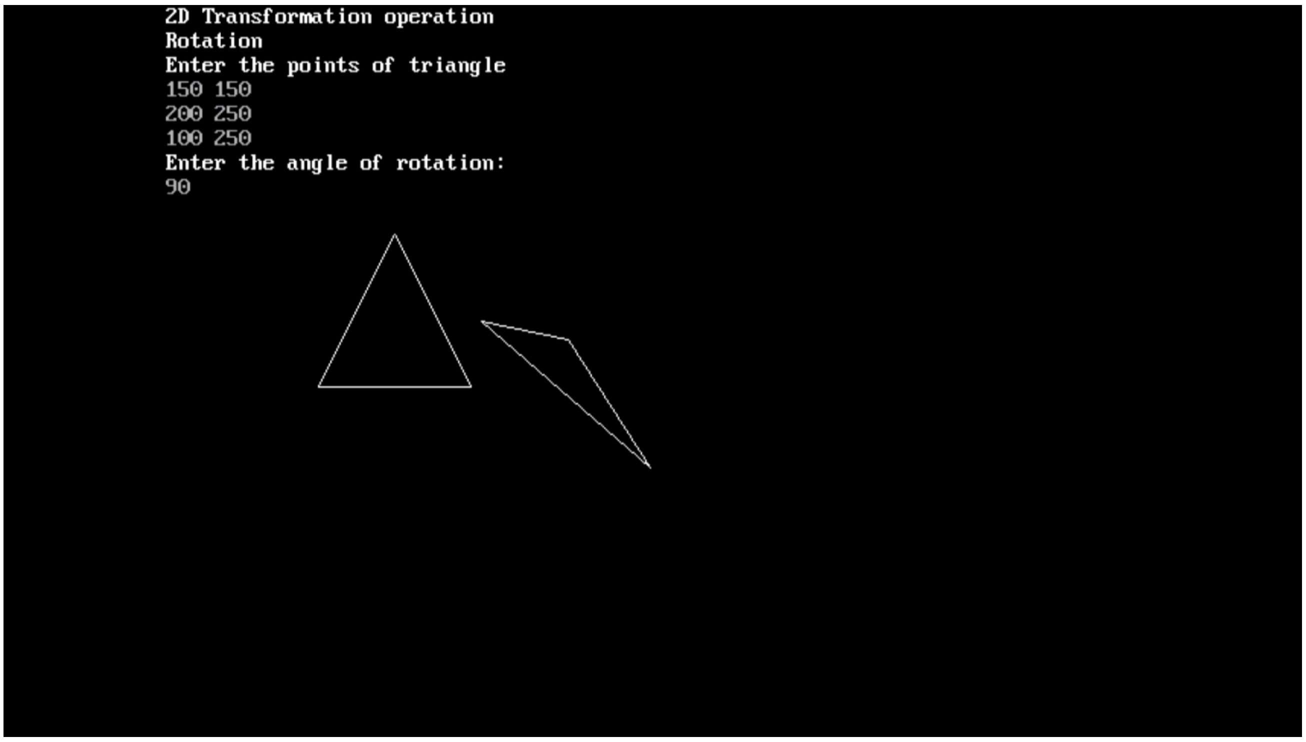
Output:

Translation



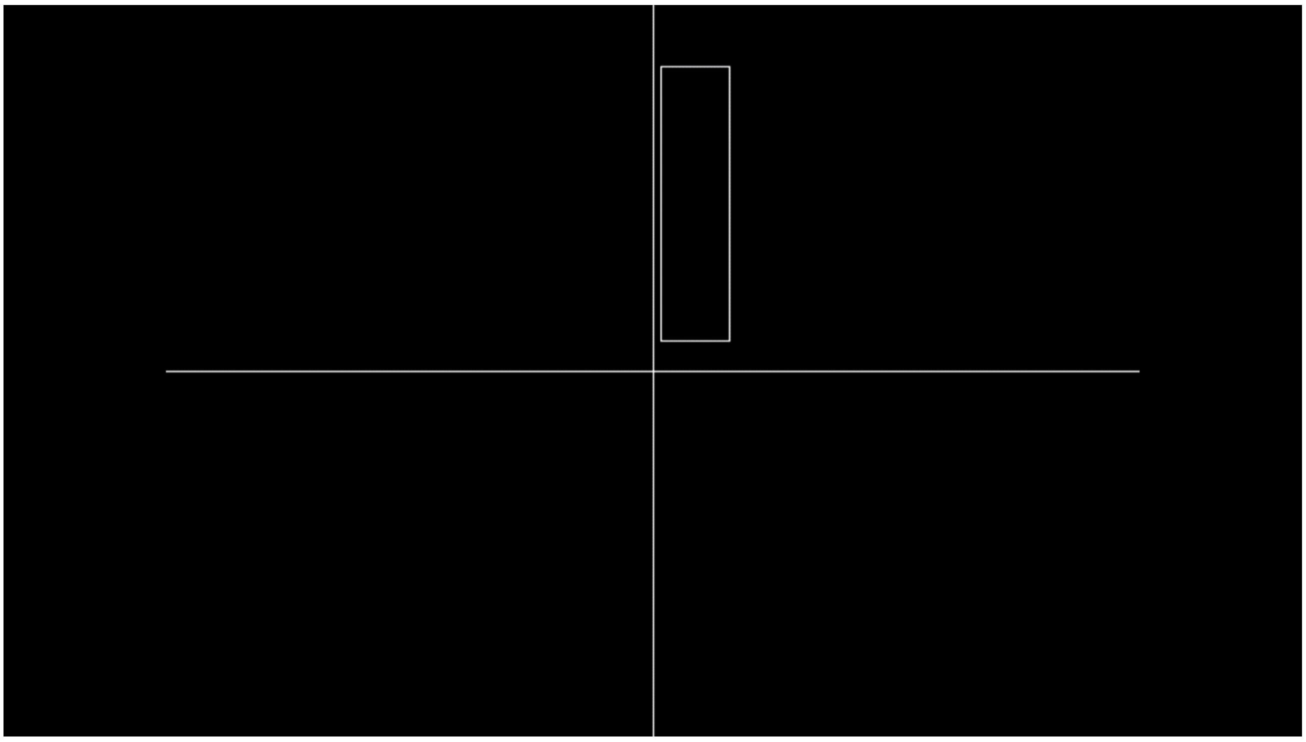
```
2D Transformation operation
Translation
Enter the points of triangle
150 150
100 250
200 250
Enter translation factor
100 100
```

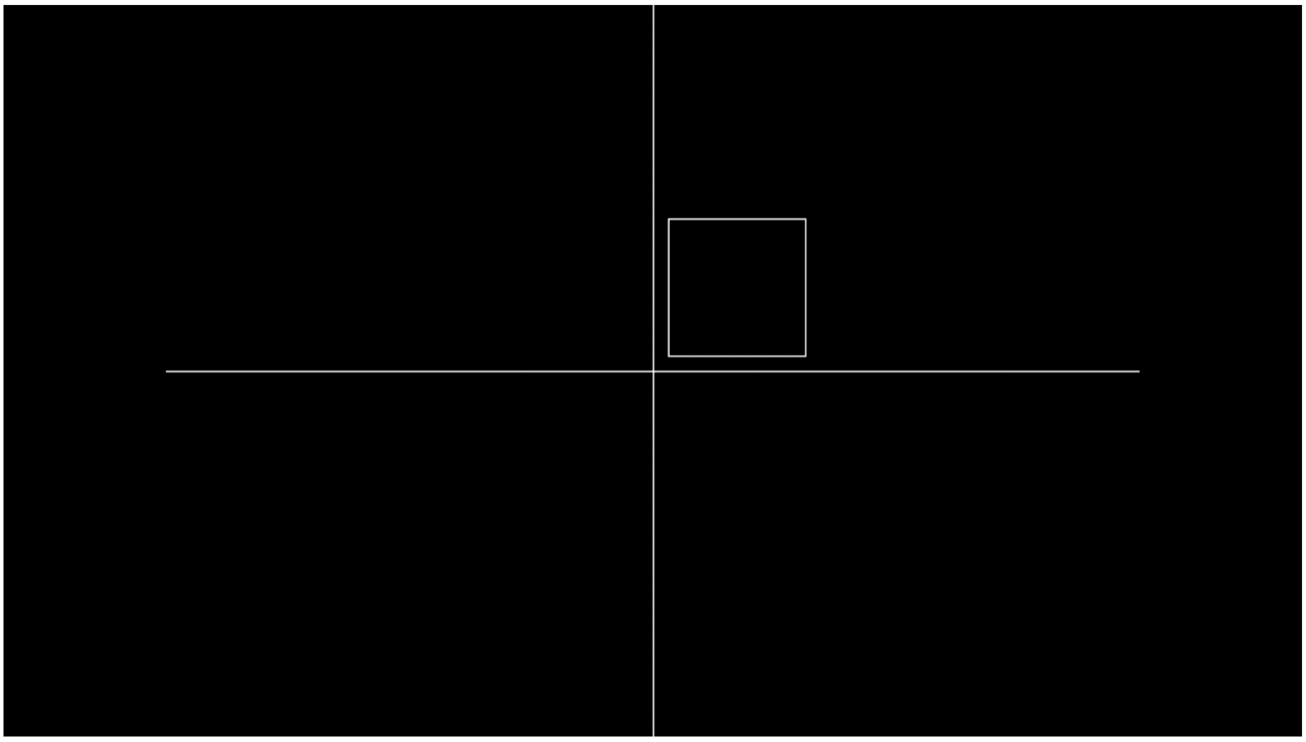
Rotation



```
2D Transformation operation
Rotation
Enter the points of triangle
150 150
200 250
100 250
Enter the angle of rotation:
90
```

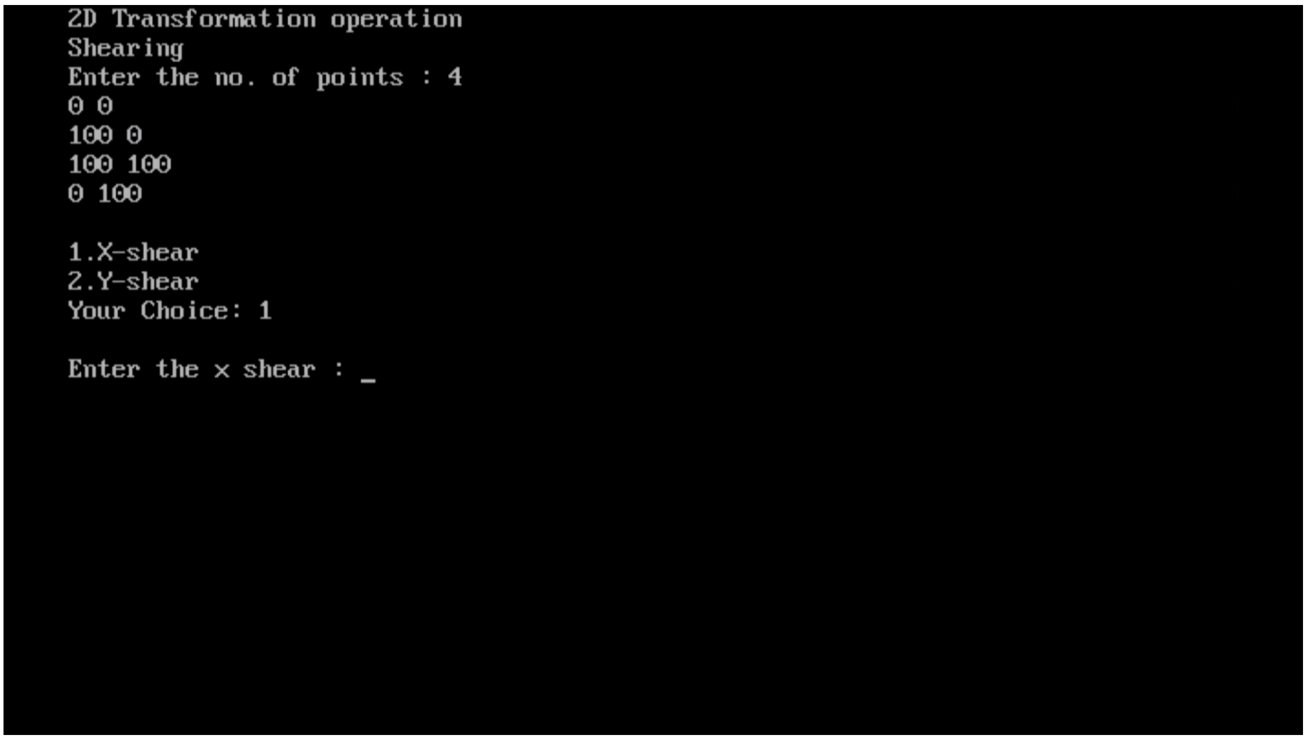
Scaling



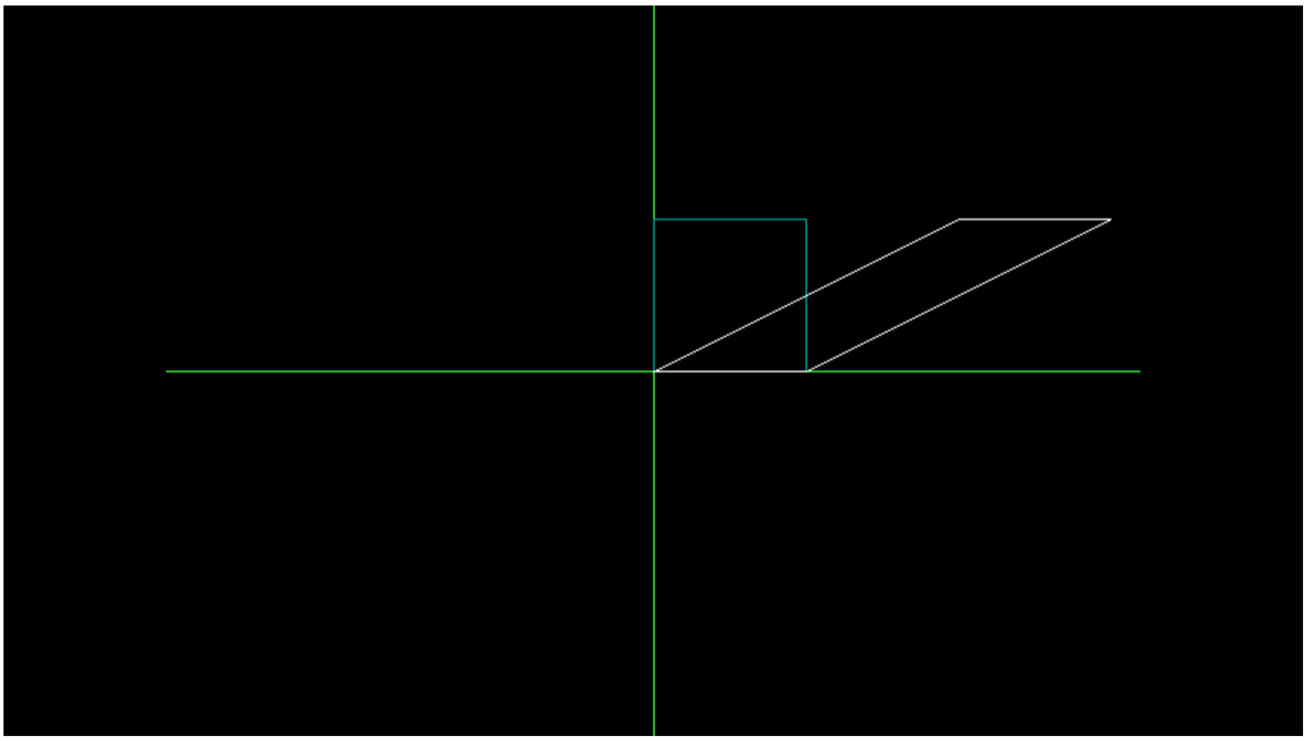


Shearing

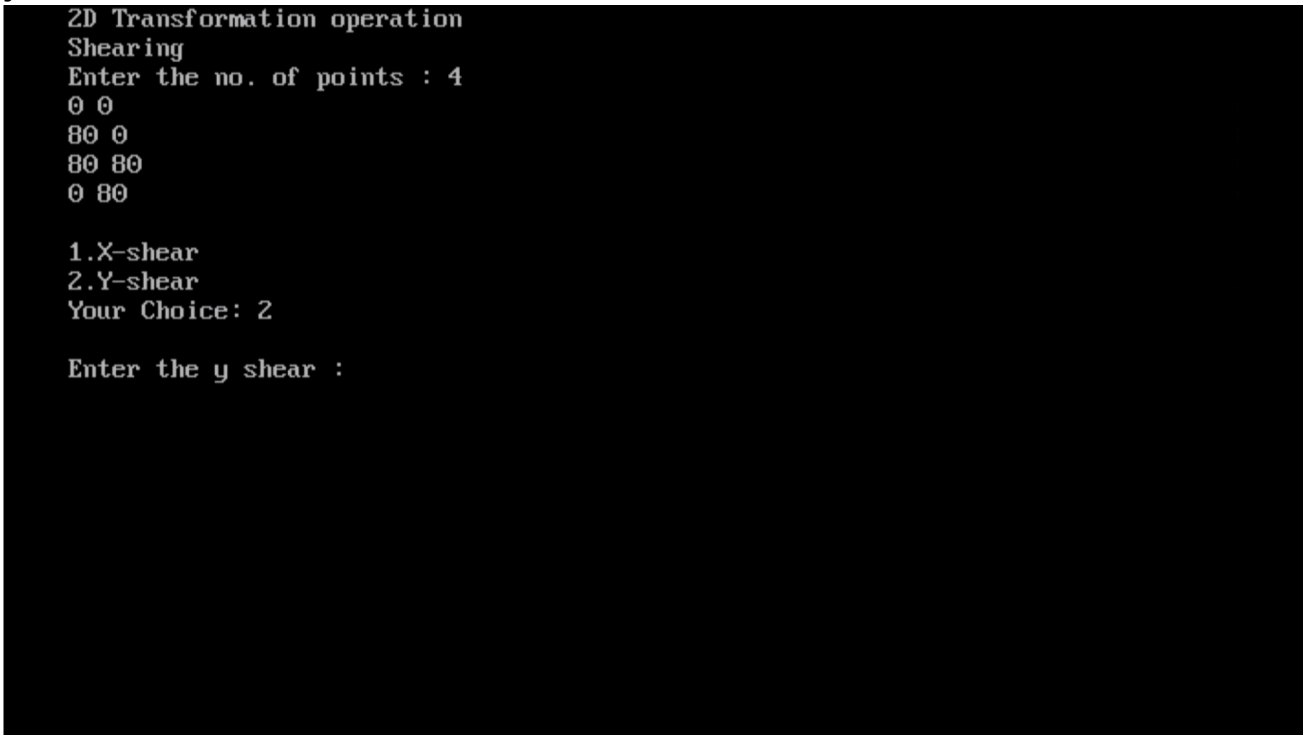
x-shear



```
2D Transformation operation
Shearing
Enter the no. of points : 4
0 0
100 0
100 100
0 100

1.X-shear
2.Y-shear
Your Choice: 1

Enter the x shear : _
```



y-shear



```
2D Transformation operation
Shearing
Enter the no. of points : 4
0 0
80 0
80 80
0 80

1.X-shear
2.Y-shear
Your Choice: 2

Enter the y shear :
```