EXPERIMENT 6

COMPUTER GRAPICS AND MULTIMEDIA

Aim

Write C Programs for the implementation of 2D transformations.

Syeda Reeha Quasar
14114802719
3C7

EXPERIMENT 6

AIM:

Write C Programs for the implementation of 2D transformations.

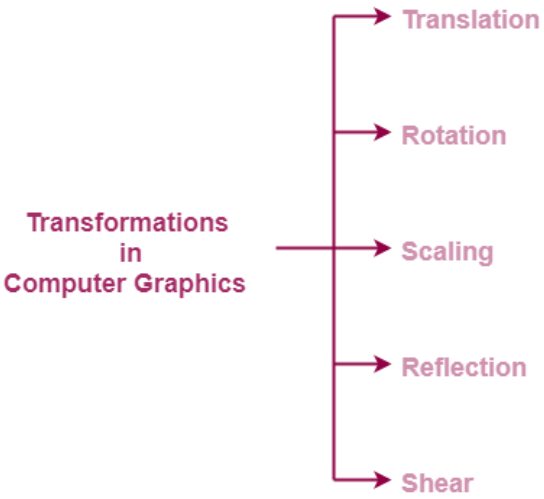
THEORY:

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, 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.

In computer graphics, various transformation techniques are-

1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shear



In this experiment, we will discuss about 2D Translation, Rotation and Scaling in Computer Graphics.

2D Translation in Computer Graphics-

In Computer graphics,

2D Translation is a process of moving an object from one position to another in a 2-dimensional plane.

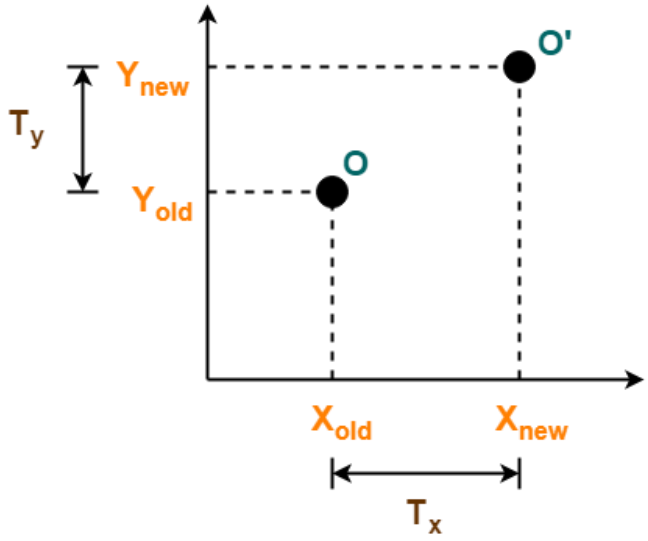
Consider a point object O has to be moved from one position to another in a 2D plane.

Let-

- 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)

Given a Translation vector (T_x, T_y)-

- T_x defines the distance the X_{old} coordinate has to be moved.
- T_y defines the distance the Y_{old} coordinate has to be moved.



2D Translation in Computer Graphics

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

- The homogeneous coordinates representation of (X, Y) is (X, Y, 1).
- Through this representation, all the transformations can be performed using matrix / vector multiplications.

The above translation matrix may be represented as a 3 x 3 matrix as-

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

Translation Matrix

(Homogeneous Coordinates Representation)

2D Rotation in Computer Graphics-

In Computer graphics,

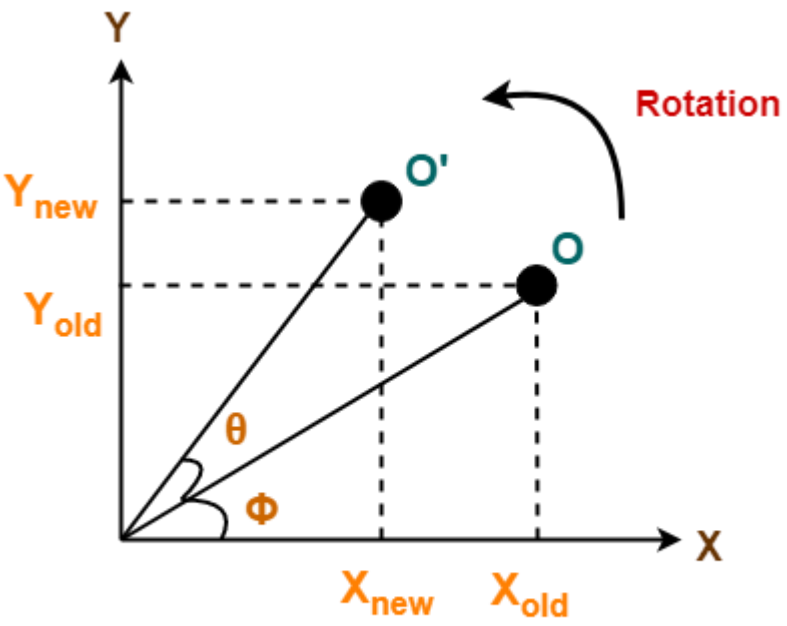
2D Rotation is a process of rotating an object with respect to an angle in a two dimensional plane.

Consider a point object O has to be rotated from one angle to another in a 2D plane.

Let-

- 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})



2D Rotation in Computer Graphics

This 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

For homogeneous coordinates, the above rotation matrix may be represented as a 3 x 3 matrix as-

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

Rotation Matrix

(Homogeneous Coordinates Representation)

2D Scaling in Computer Graphics-

In computer graphics, scaling is a process of modifying or altering the size of objects.

- Scaling may be used to increase or reduce the size of object.
- Scaling subjects the coordinate points of the original object to change.
- Scaling factor determines whether the object size is to be increased or reduced.
- If scaling factor > 1, then the object size is increased.
- If scaling factor < 1, then the object size is reduced.

Consider a point object O has to be scaled in a 2D plane.

Let-

- 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

For homogeneous coordinates, the above scaling matrix may be represented as a 3 x 3 matrix as-

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

Scaling Matrix

(Homogeneous Coordinates Representation)

SOURCE CODE:

```
#include <stdio.h>

#include <graphics.h>

#include <conio.h>

#include <math.h>


void smiley(int x, int y, int r)

{

        setcolor(YELLOW);


    // creating circle and fill it with

    // yellow color using floodfill.

    circle(x, y, r);

    setfillstyle(SOLID_FILL, YELLOW);

    floodfill(x, y, YELLOW);


    // Set color of background to black

    setcolor(BLACK);

    setfillstyle(SOLID_FILL, BLACK);


    // Use fill ellipse for creating eyes

    fillellipse(x + 10, y - 15, 2, 6);
```

```
    fillellipse(x - 10, y - 15, 2, 6);


    // Use ellipse for creating mouth
    ellipse(x, y, y + 105, x + 35, r - 20, r - 31);
    ellipse(x, y, y + 105, x + 35, r - 20, r - 30);
    ellipse(x, y, y + 105, x + 35, r - 20, r - 29);
}


int rotation(int x, int y, int r, int flag)
{
        int p, q;
        p = abs(x * cos(r) - y * sin(r));
        q = abs(y * cos(r) + x * sin(r));
        if (flag == 1)
                return p;
        return q;
}


void rotate(float angle){
int x1,x2,x3,x4;
 int y1,y2,y3,y4;
 int refx,refy;
```

```
int ax1,ax2,ax3,ax4,ay1,ay2,ay3,ay4;

angle=angle*(3.14/180);

refx=100;

refy=100;

x1=100;

y1=100;

x2=150;

y2=100;

x3=150;

y3=150;

x4=100;

y4=150;

ax1=refy+(x1-refx)*cos(angle)-(y1-refy)*sin(angle);

ay1=refy+(x1-refx)*sin(angle)+(y1-refy)*cos(angle);

ax2=refy+(x2-refx)*cos(angle)-(y2-refy)*sin(angle);

ay2=refy+(x2-refx)*sin(angle)+(y2-refy)*cos(angle);

ax3=refy+(x3-refx)*cos(angle)-(y3-refy)*sin(angle);

ay3=refy+(x3-refx)*sin(angle)+(y3-refy)*cos(angle);

ax4=refy+(x4-refx)*cos(angle)-(y4-refy)*sin(angle);

ay4=refy+(x4-refx)*sin(angle)+(y4-refy)*cos(angle);

setcolor(4);

rectangle(100,150,150,100);
```

```
 line(ax1,ay1,ax2,ay2);
 line(ax2,ay2,ax3,ay3);
 line(ax3,ay3,ax4,ay4);
 line(ax4,ay4,ax1,ay1);
}

main()
{       initwindow(800, 800);
        rectangle(150, 200, 300, 400);
        circle(200, 200, 100);
        smiley(300, 100, 100);

        // translation
        printf("\n TRANSLATION \n\n");
        printf("Enter the value you want the x-coordinate to: \n");
        int x, y;
        scanf("%d", &x);
        printf("Enter the value you want the y-coordinate to: \n");
        scanf("%d", &y);
        rectangle(150 + x, 200 + y, 300 + x, 400 + y);
        circle(200 + x, 200 + y, 100);
        smiley(300 + x, 100 + y, 100);
```

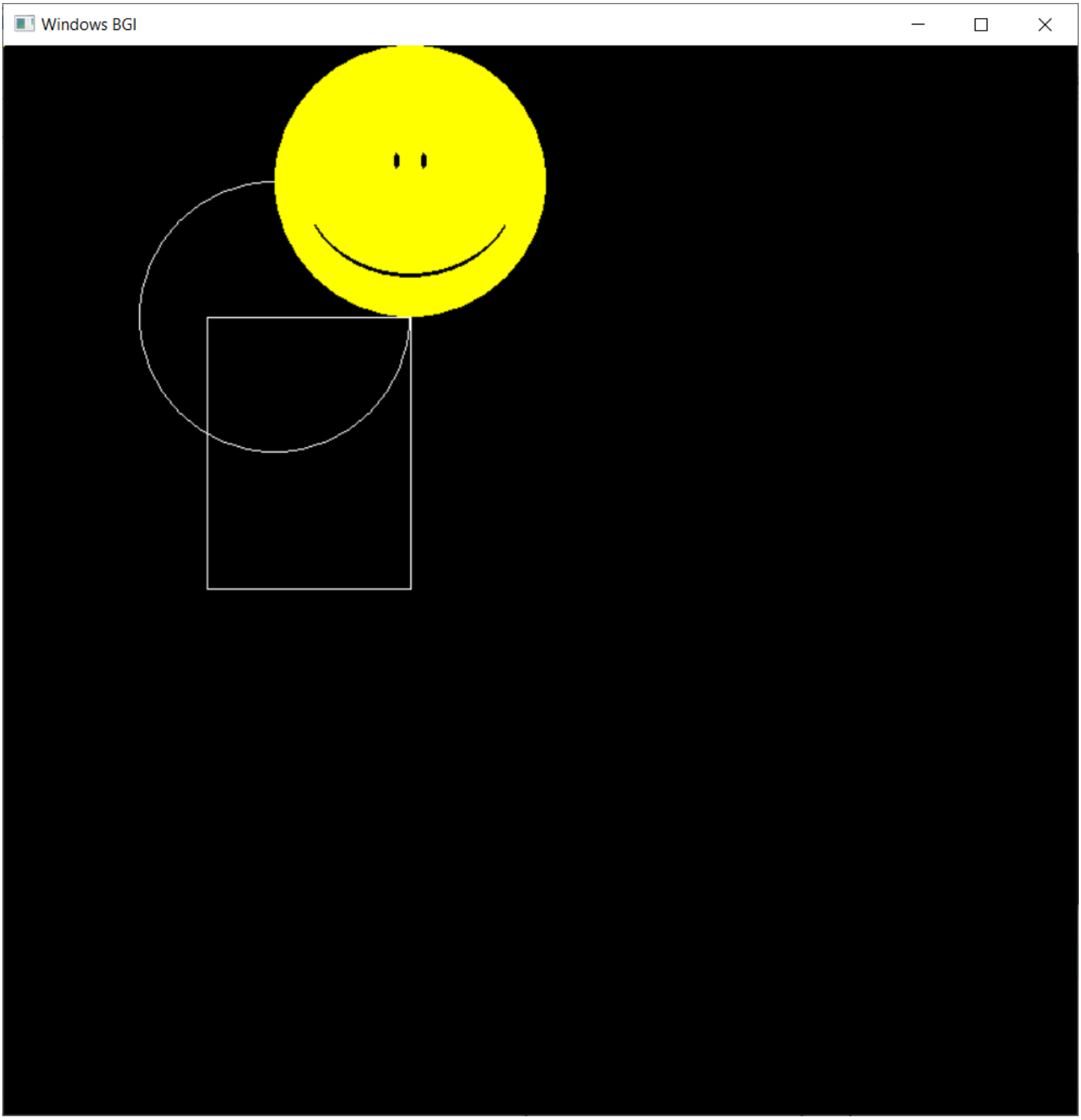
```
        // scaling
        printf("\n\n scaling \n\n");
        printf("Enter the scaling factor\n");
        int d;
        scanf("%d", &d);
        rectangle(150 * d , 200 * d, 300 * d, 400 * d);
        circle(200, 200 , 100 * d);

        // rotation
        printf("\n\n ROTATION \n\n");
        printf("Enter the angle\n");
        int r;
        scanf("%d", &r);
        // 2 ways to print:
//      rectangle(rotation(15, 20, r, 1), rotation(15, 20, r, 0), rotation(30, 40, r, 1), rotation(30, 40, r, 0));
        rotate(r);
        getch();
        closegraph();
}
```

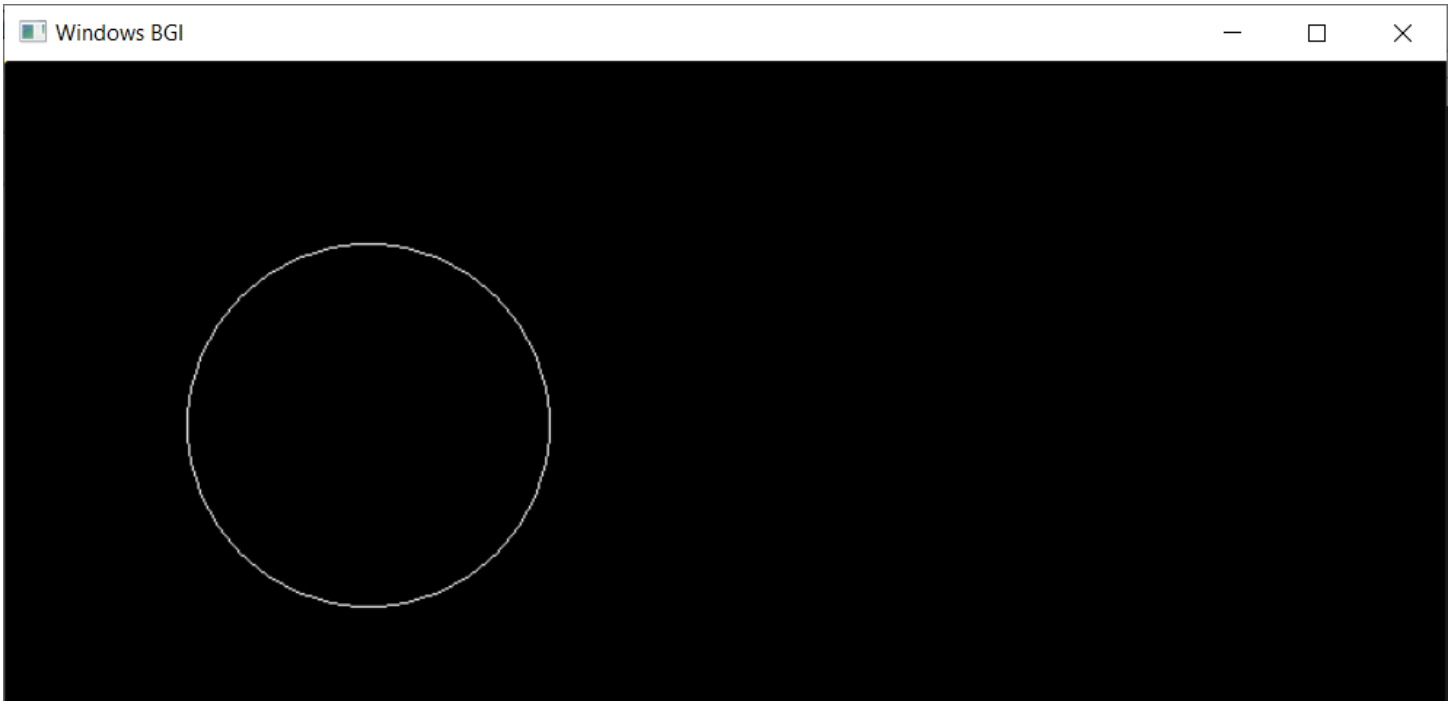
OUTPUT:

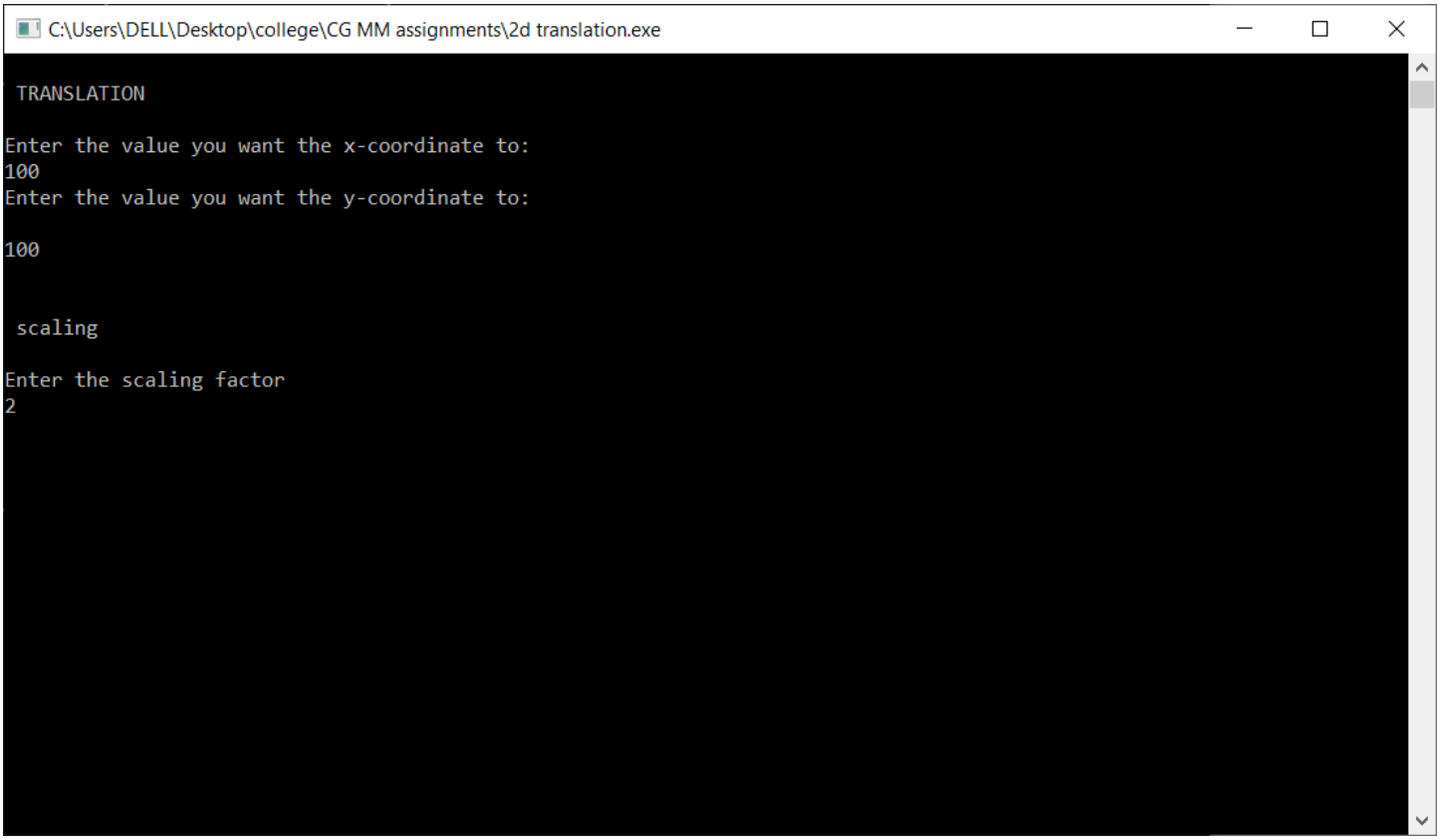
SHAPES GOING TO USE:

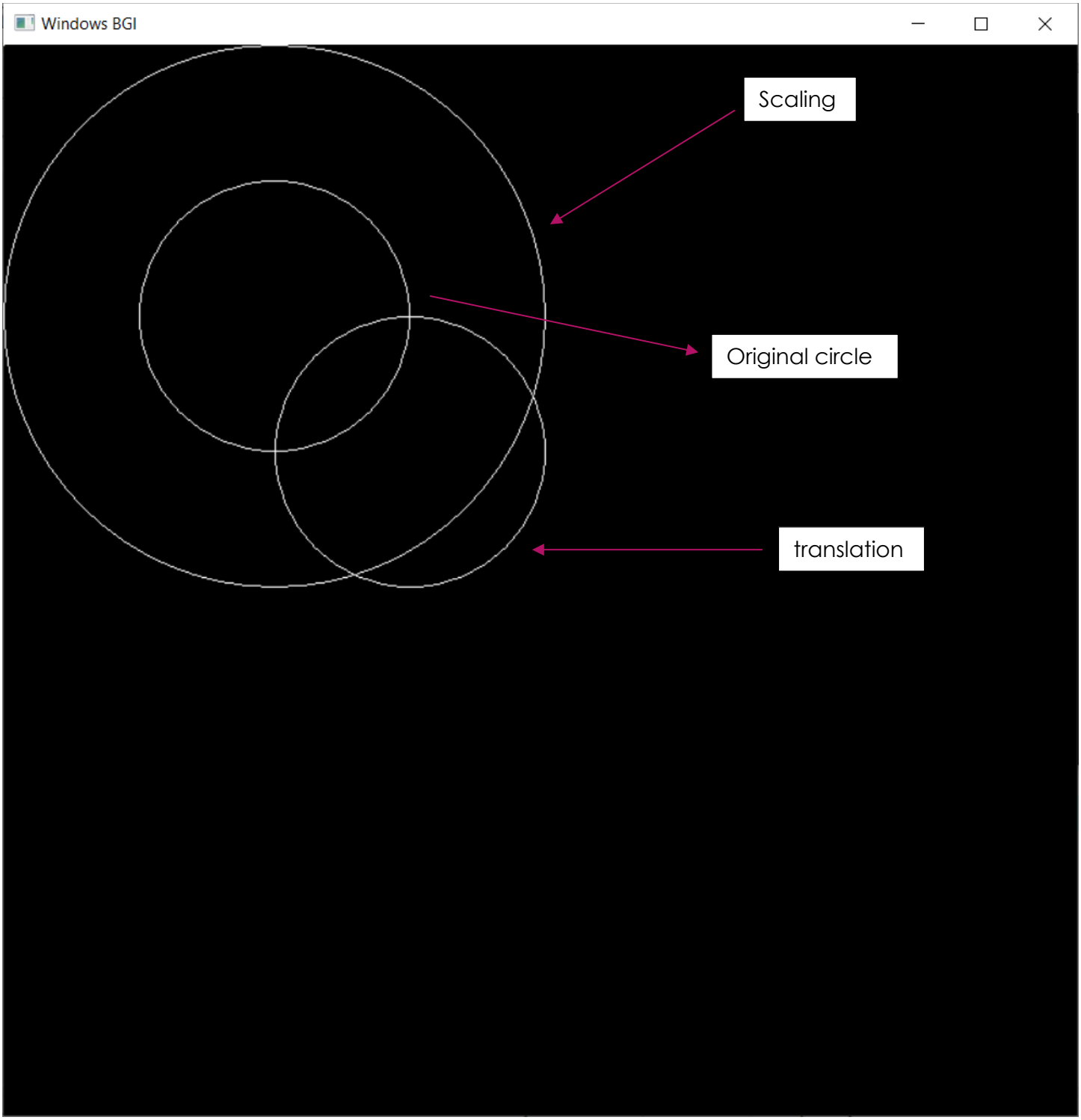
- Square
- Circle



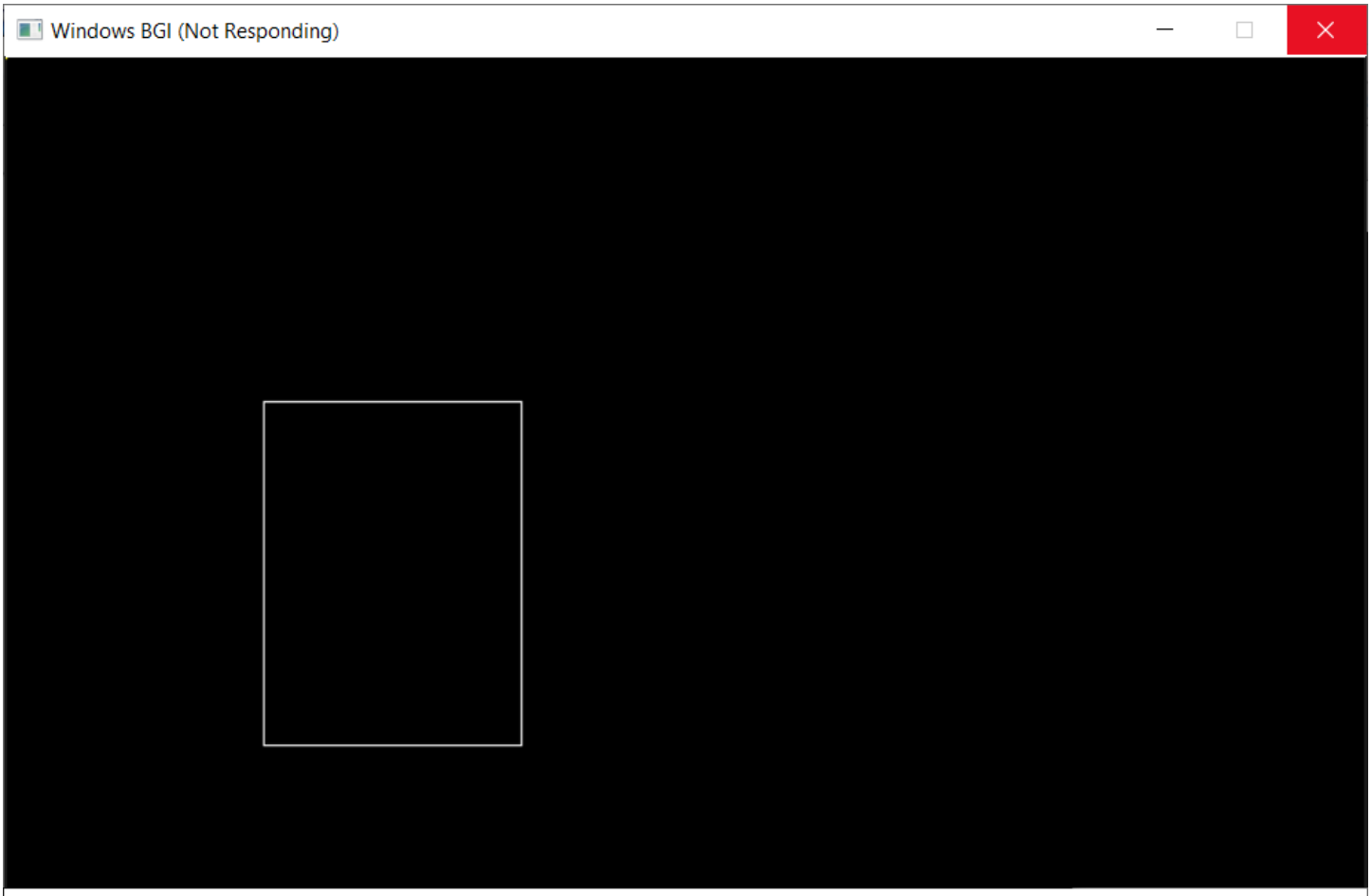
All transformations on circle:

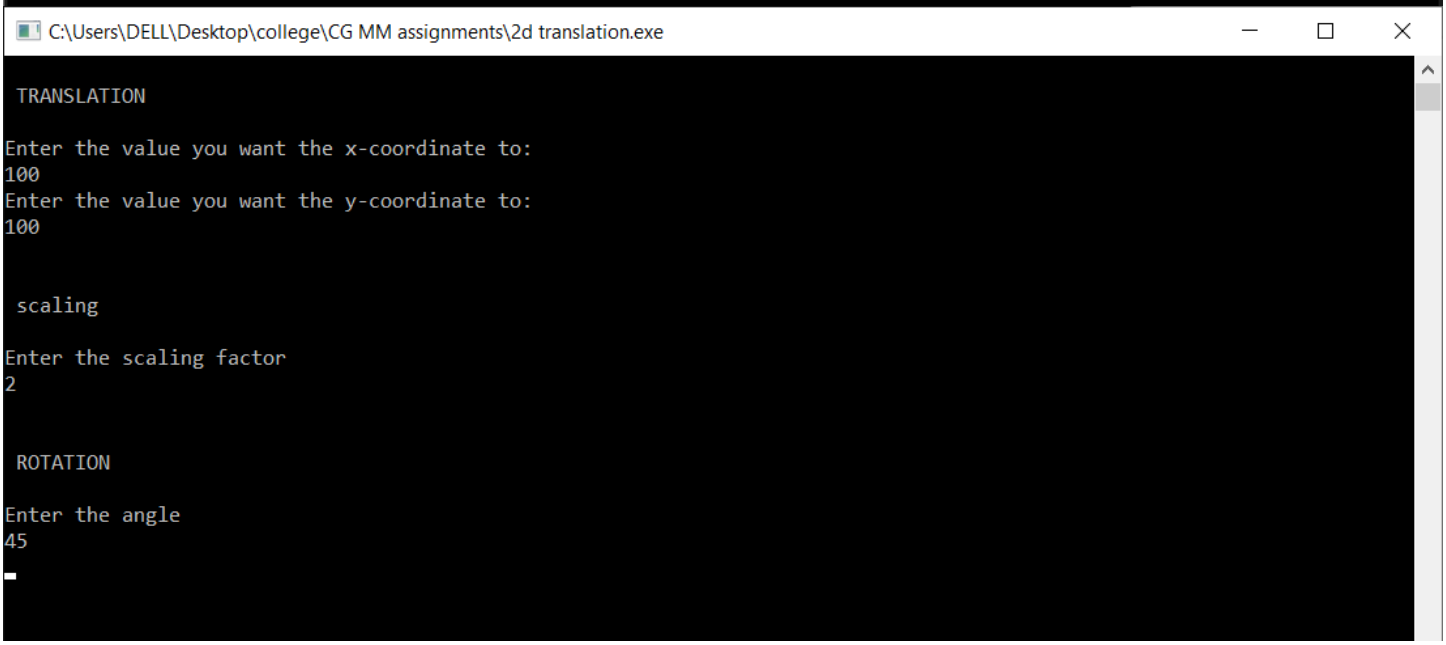


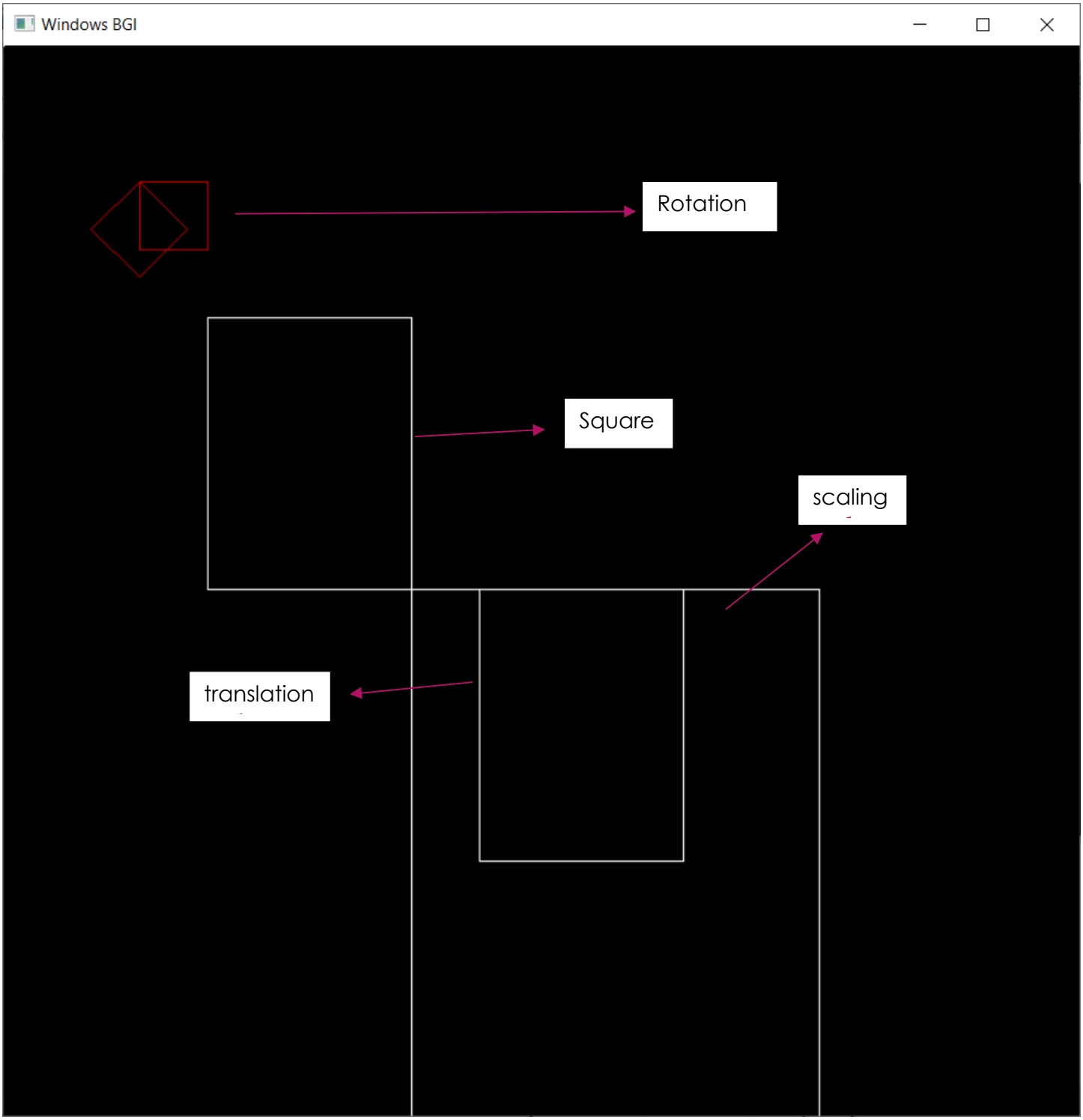



Windows BGI
Scaling
Original circle
translation

ALL TRANSFORMATIONS ON SQUARE:






Windows BGI
Rotation
Square
scaling
translation

VIVA-VOCE:

Q1. Given a circle C with radius 10 and center coordinates (1, 4). Apply the translation with distance 5 towards X axis and 1 towards Y axis. Obtain the new coordinates of C without changing its radius?

Ans.

Old center coordinates of C = (Xold, Yold) = (1, 4)

Translation vector = (Tx, Ty) = (5, 1)

Applying the translation equations, we have-

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

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix} + \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \end{bmatrix}$$

- Xnew = Xold + Tx = 1 + 5 = 6
- Ynew = Yold + Ty = 4 + 1 = 5

New center coordinates of C = (6, 5).

Q2. Given a line segment with starting point as (0, 0) and ending point as (4, 4). Apply 30 degree rotation anticlockwise direction on the line segment and find out the new coordinates of the line.

Ans.

Old ending coordinates of the line = (Xold, Yold) = (4, 4)

Rotation angle = θ = 30°

new ending coordinates of the line after rotation = (Xnew, Ynew).

Xnew

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

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos 30 & -\sin 30 \\ \sin 30 & \cos 30 \end{bmatrix} \times \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 4 \times \cos 30 - 4 \times \sin 30 \\ 4 \times \sin 30 + 4 \times \cos 30 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 4 \times \cos 30 - 4 \times \sin 30 \\ 4 \times \sin 30 + 4 \times \cos 30 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1.46 \\ 5.46 \end{bmatrix}$$

= Xold x cosθ – Yold x sinθ

= 4 x cos30° – 4 x sin30°

= 4 x (√3 / 2) – 4 x (1 / 2)

= 2√3 – 2

= 2(√3 – 1)

= 2(1.73 – 1)

= 1.46

Ynew

= Xold x sinθ + Yold x cosθ

= 4 x sin30° + 4 x cos30°

= 4 x (1 / 2) + 4 x (√3 / 2)

= 2 + 2√3

= 2(1 + √3)

= 2(1 + 1.73)

= 5.46

New ending coordinates of the line after

rotation = (1.46, 5.46).

Q3. Given a square object with coordinate points A(0, 3), B(3, 3), C(3, 0), D(0, 0). Apply the scaling parameter 2 towards X axis and 3 towards Y axis and obtain the new coordinates of the object.

Ans.

Old corner coordinates of the square = A (0, 3), B(3, 3), C(3, 0), D(0, 0)

Scaling factor along X axis = 2

Scaling factor along Y axis = 3

For Coordinates A(0, 3)

•Xnew = Xold x Sx = 0 x 2 = 0

•Ynew = Yold x Sy = 3 x 3 = 9

For Coordinates B(3, 3)

•Xnew = Xold x Sx = 3 x 2 = 6

•Ynew = Yold x Sy = 3 x 3 = 9

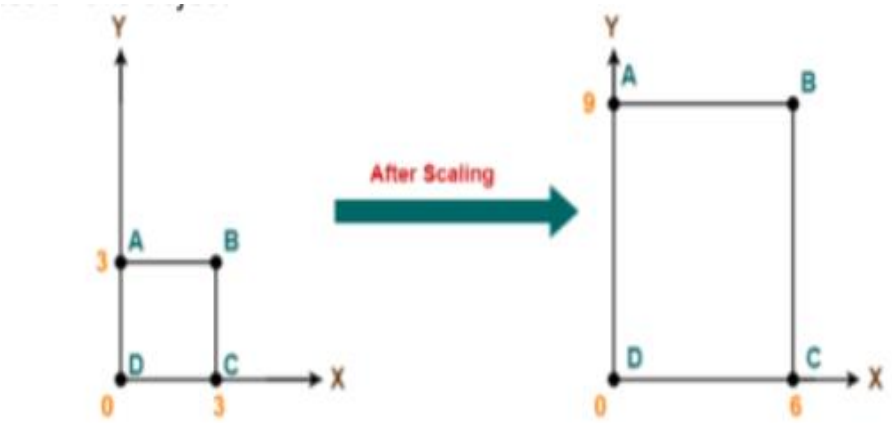
For Coordinates C(3, 0)

•Xnew = Xold x Sx = 3 x 2 = 6

•Ynew = Yold x Sy = 0 x 3 = 0

For Coordinates D(0, 0)

•Xnew = Xold x Sx = 0 x 2 = 0



•Ynew = Yold x Sy = 0 x 3 = 0

Q4. Given a triangle with coordinate points A(3, 4), B(6, 4), C(5, 6). Apply the reflection on the X axis and obtain the new coordinates of the object.

Ans.

For Coordinates A(3, 4)

•Xnew = Xold = 3

•Ynew = -Yold = -4

For Coordinates B(6, 4)

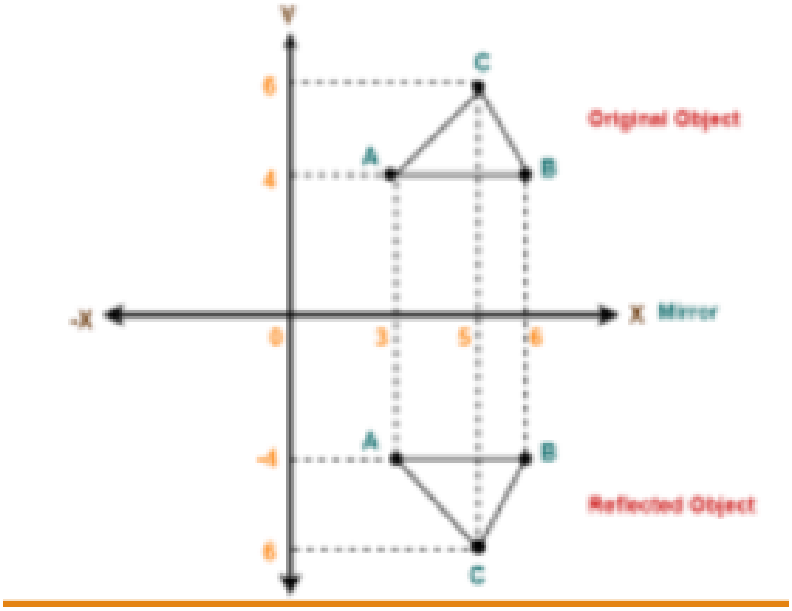
•Xnew = Xold = 6

•Ynew = -Yold = -4

For Coordinates C(5, 6)

•Xnew = Xold = 5

•Ynew = -Yold = -6



Q5. Given a triangle with points (1, 1), (0, 0) and (1, 0). Apply shear parameter 2 on X axis and 2 on Y axis and find out the new coordinates of the object.

Ans.

- Old corner coordinates of the triangle = A (1, 1), B(0, 0), C(1, 0)
- Shearing parameter towards X direction (Shx) = 2
- Shearing parameter towards Y direction (Shy) = 2

Shearing in X AxisFor Coordinates A(1, 1)

Xnew = Xold + Shx x

Yold = 1 + 2 x 1 = 3

Ynew = Yold = 1

For Coordinates B(0, 0)

Xnew = Xold + Shx x Yold = 0 + 2 x 0 = 0

Ynew = Yold = 0

Shearing in X AxisFor Coordinates C(1, 0)

- Xnew = Xold + Shx x Yold = 1 + 2 x 0 = 1
- Ynew = Yold = 0

New coordinates of the triangle after shearing in X axis = A (3, 1), B(0, 0), C(1, 0)