Experiment – 9

Aim:

Write a program for 2D transformation Including (Translation, rotation, scaling, reflection)

Theory :

2D transformation refers to the process of manipulating the position, size, orientation, and shape of 2D objects in a 2D space. It involves applying mathematical operations to the coordinates of the vertices that define the object to change its appearance.

There are four types of 2D transformations: translation, rotation, scaling, and reflection.

For Translation:

Translation involves moving the object from one position to another in the same direction. This is done by adding or subtracting values to the x and y coordinates of the vertices.

Here we have original square in “Red” and Translated square in “Yellow”

For Rotation:

Rotation involves rotating the object around a fixed point. This is done by applying trigonometric functions to the coordinates of the vertices.

Here we have original square in “Red” and Rotated square in “Green”

For Scaling:

Scaling involves increasing or decreasing the size of the object. This is done by multiplying or dividing the x and y coordinates of the vertices by a scaling factor.

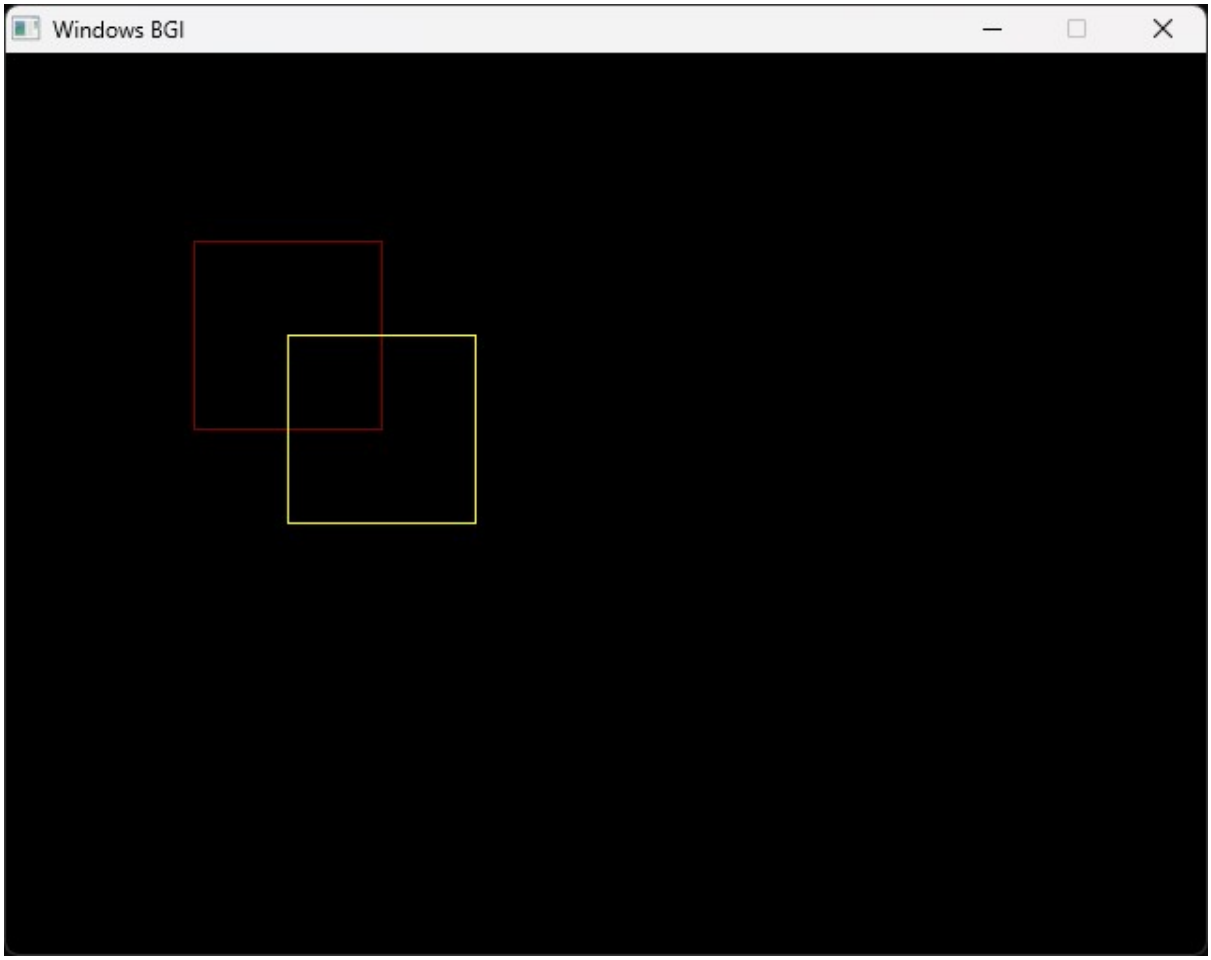
Here we have original square in “Red” and Scaled square in “Green”

For Reflection:

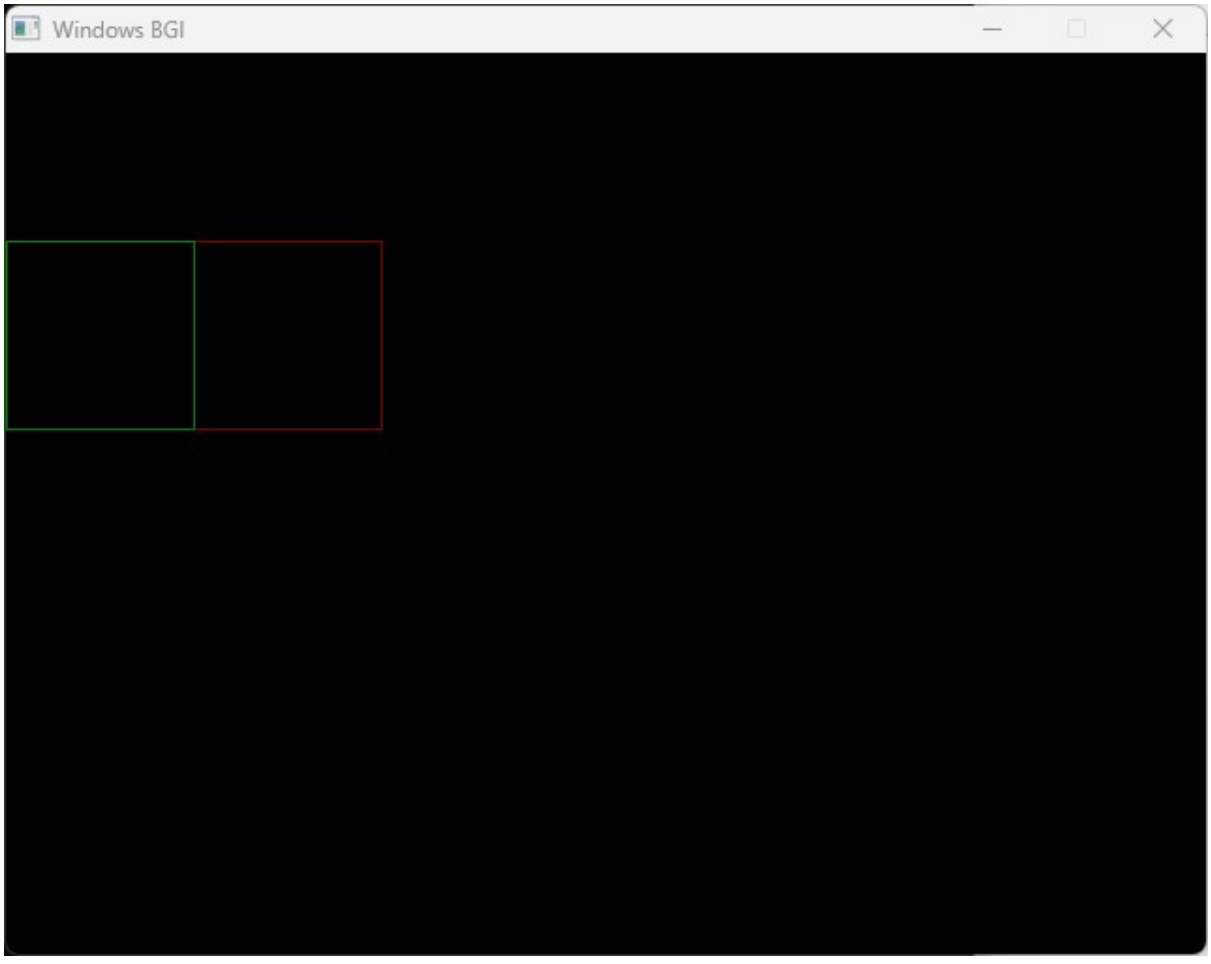
Reflection involves flipping the object over a fixed line. This is done by multiplying the x or y coordinates of the vertices by -1.

Here we have original square in “Red” and Reflected square in “Yellow”

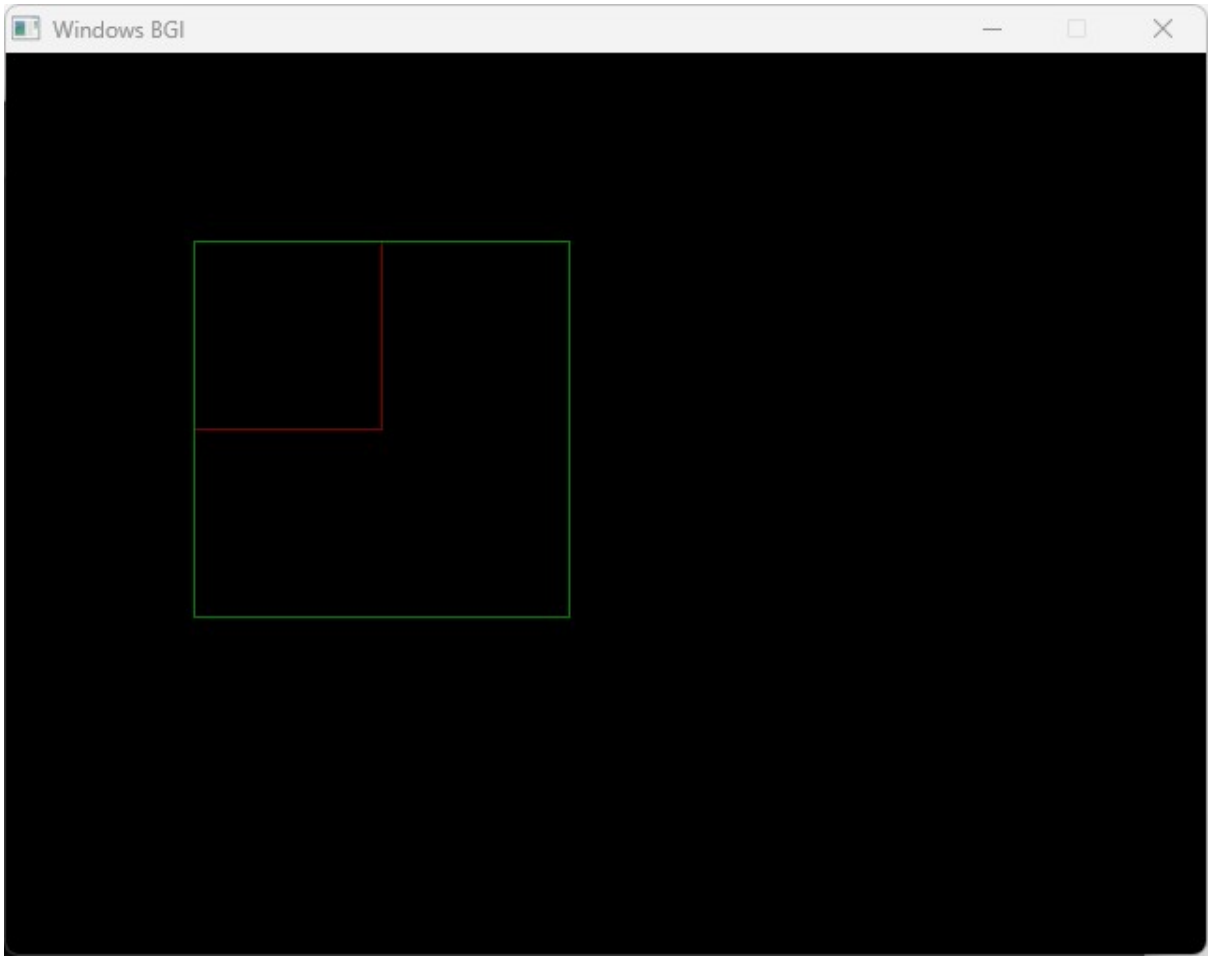
```cpp
#include <graphics.h>
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    int x[4] = {100, 200, 200, 100};
    int y[4] = {100, 100, 200, 200};
    int tx = 50, ty = 50;
    float theta = 45;
    float sx = 2, sy = 2;
    int xf = 100, yf = 100;

    // Original square
    setcolor(RED);
    rectangle(x[0], y[0], x[2], y[2]);

    // Translation
    setcolor(YELLOW);
    for (int i = 0; i < 4; i++)
    {
        x[i] += tx;
        y[i] += ty;
    }
    rectangle(x[0], y[0], x[2], y[2]);

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



```cpp
#include <graphics.h>
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    int x[4] = {100, 200, 200, 100};
    int y[4] = {100, 100, 200, 200};
    int tx = 50, ty = 50;
    float theta = 45;
    float sx = 2, sy = 2;
    int xf = 100, yf = 100;

    // Original square
    setcolor(RED);
    rectangle(x[0], y[0], x[2], y[2]);

    // Rotation
    setcolor(GREEN);
    for (int i = 0; i < 4; i++)
    {
        float xr = (x[i] - xf) * cos(theta * M_PI / 90) - (y[i] - yf) * sin(theta * M_PI / 90) + xf;
        float yr = (x[i] - xf) * sin(theta * M_PI / 90) + (y[i] - yf) * cos(theta * M_PI / 90) + yf;
        x[i] = (int) xr;
        y[i] = (int) yr;
    }
    rectangle(x[0], y[0], x[2], y[2]);

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



```cpp
#include <graphics.h>
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    int x[4] = {100, 200, 200, 100};
    int y[4] = {100, 100, 200, 200};
    int tx = 50, ty = 50;
    float theta = 45;
    float sx = 2, sy = 2;
    int xf = 100, yf = 100;

    // Original square
    setcolor(RED);
    rectangle(x[0], y[0], x[2], y[2]);

    // Scaling
    setcolor(GREEN);
    for (int i = 0; i < 4; i++)
    {
        x[i] = (int) (xf + sx * (x[i] - xf));
        y[i] = (int) (yf + sy * (y[i] - yf));
    }
    rectangle(x[0], y[0], x[2], y[2]);

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



```cpp
#include <graphics.h>
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    int x[4] = {100, 200, 200, 100};
    int y[4] = {100, 100, 200, 200};
    int tx = 50, ty = 50;
    float theta = 45;
    float sx = 2, sy = 2;
    int xf = 100, yf = 100;

    // Original square
    setcolor(RED);
    rectangle(x[0], y[0], x[2], y[2]);

    // Reflection
    setcolor(YELLOW);
    for (int i = 0; i < 4; i++)
    {
        x[i] = 2 * xf - x[i];
    }
    rectangle(x[0], y[0], x[2], y[2]);


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