SSN COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1712 – GRAPHICS AND MULTIMEDIA LAB

Name : Rahul Ram M
Reg .No : 185001121
Date : 30/08/2021

EX NO: 5a

2D Transformations – Translation, Rotation and Scaling

Write a C++ menu-driven program using OPENGL to perform 2D transformations – translation, rotation, scaling for line and polygon.

Aim:

To write a C++ menu-driven program using OPENGL to perform 2D transformations – translation, rotation, scaling for line and polygon.

Algorithm:

1. Create functions for drawing lines and quads using GL_LINES and GL_QUADS.
2. drawPolygon() - Draw quads without modifying any coordinates.
3. drawLine() - Draw line without modifying any coordinates.
4. translateLine(int x, int y) - Add x and y to each coordinate and draw the Line.
5. translatePolygon(int x, int y) - Add x and y to each coordinate and draw the polygon.
6. scaleLine(int x, int y) - Multiply scaling factor x and y to each coordinate and draw the Line.
7. scalePolygon(int x, int y) - Multiply scaling factor x and y to each coordinate and draw the polygon.
8. rotateLine(double angle) - Make x coordinate as (x*cos(angle)) - (y*sin(angle)) and y coordinate as (x*sin(angle) + y*cos(angle)) and draw the line.
9. rotatePolygon(double angle) - Make x coordinate as (x*cos(angle)) - (y*sin(angle)) and y coordinate as (x*sin(angle) + y*cos(angle)) and draw the polygon.
10. Read the number of edges from the user and read the choice as 1. Translation, 2. Rotation, 3.Scaling
11. Call the required function to draw lines and polygons.

Code:

```
#include <windows.h>
#include <gl/glut.h>
#include <math.h>
#include <iostream>
#include <vector>
using namespace std;

double degree;
vector<int> pntX;
vector<int> pntY;

vector<int> lpntX;
vector<int> lpntY;

int choice, vertices;

void myInit()
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glPointSize(1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-1000, 1000, -1000, 1000);
}


void drawLine()
{
    glBegin(GL_LINES);
    glColor3f(1.0, 0.0, 0.0);
    for (int i = 0; i < vertices; i++)
    {
        glVertex2i(lpntX[i], lpntY[i]);
    }
    glEnd();
}

void translateLine(int x, int y)
{
    glBegin(GL_LINES);
    glColor3f(0.0, 1.0, 0.0);
```

```
    for (int i = 0; i < vertices; i++)
    {
        glVertex2i(lpntX[i] + x, lpntY[i] + y);
    }
    glEnd();
}

void scaleLine(double x, double y)
{
    glBegin(GL_LINES);
    glColor3f(1.0, 1.0, 0.0);
    for (int i = 0; i < vertices; i++)
    {
        glVertex2i(round(lpntX[i] * x), round(lpntY[i] * y));
    }
    glEnd();
}

void rotateLine(double angleRad)
{
    glBegin(GL_LINES);
    glColor3f(0.0, 0.0, 1.0);
    for (int i = 0; i < vertices; i++)
    {
        glVertex2i(round((lpntX[i] * cos(angleRad)) - (lpntY[i] * sin(angleRad))) ,
round((lpntX[i] * sin(angleRad)) + (lpntY[i] * cos(angleRad))));
    }
    glEnd();
}

void drawPolygon()
{
    glBegin(GL_QUADS);
    glColor3f(1.0, 0.0, 0.0);
    for (int i = 0; i < vertices; i++)
    {
        glVertex2i(pntX[i], pntY[i]);
    }
    glEnd();
}

void translatePolygon(int x, int y)
{
```

```
    glBegin(GL_QUADS);
    glColor3f(0.0, 1.0, 0.0);
    for (int i = 0; i < vertices; i++)
    {
        glVertex2i(pntX[i] + x, pntY[i] + y);
    }
    glEnd();
}

void rotatePolygon(double angleRad)
{
    glBegin(GL_QUADS);
    glColor3f(0.0, 0.0, 1.0);
    for (int i = 0; i < vertices; i++)
    {
        glVertex2i(round((pntX[i] * cos(angleRad)) - (pntY[i] * sin(angleRad))),
round((pntX[i] * sin(angleRad)) + (pntY[i] * cos(angleRad))));
    }
    glEnd();
}

void scalePolygon(double x, double y)
{
    glBegin(GL_QUADS);
    glColor3f(1.0, 1.0, 0.0);
    for (int i = 0; i < vertices; i++)
    {
        glVertex2i(round(pntX[i] * x), round(pntY[i] * y));
    }
    glEnd();
}

void printMenu()
{
    cout << "1. Translation" << "\n";
    cout << "2. Rotation" << "\n";
    cout << "3. Scaling" << "\n";
    cout << "-1. exit" << "\n";
    cout << "Choose : " << "\n";
}

void display(void)
{
```

```
    glClear(GL_COLOR_BUFFER_BIT);
    int x, y;

    cout << "1. Line\n";
    cout << "2. Polygon\n";
    cout << "Enter your choice: ";
    cin >> choice;
    if (choice == 1)
    {
        vertices = 2;
        for (int i = 0; i < vertices; i++)
        {
            cout << "x coordinate : ";
            cin >> x;
            cout << "y coordinate : ";
            cin >> y;
            lpntX.push_back(x);
            lpntY.push_back(y);
        }
        printMenu();
        cin >> choice;
        while (choice != -1)
        {
            switch (choice)
            {
            case 1:
            {
                cout << "Translation\n";
                cout << "new x coordinate : ";
                cin >> x;
                cout << "new y coordinate : ";
                cin >> y;
                translateLine(x, y);
                break;
            }
            case 2:
            {
                cout << "Rotation\n";
                cout << "Degree : ";
                cin >> degree;
                rotateLine(degree * 3.1416 / 180);
                break;
            }
```

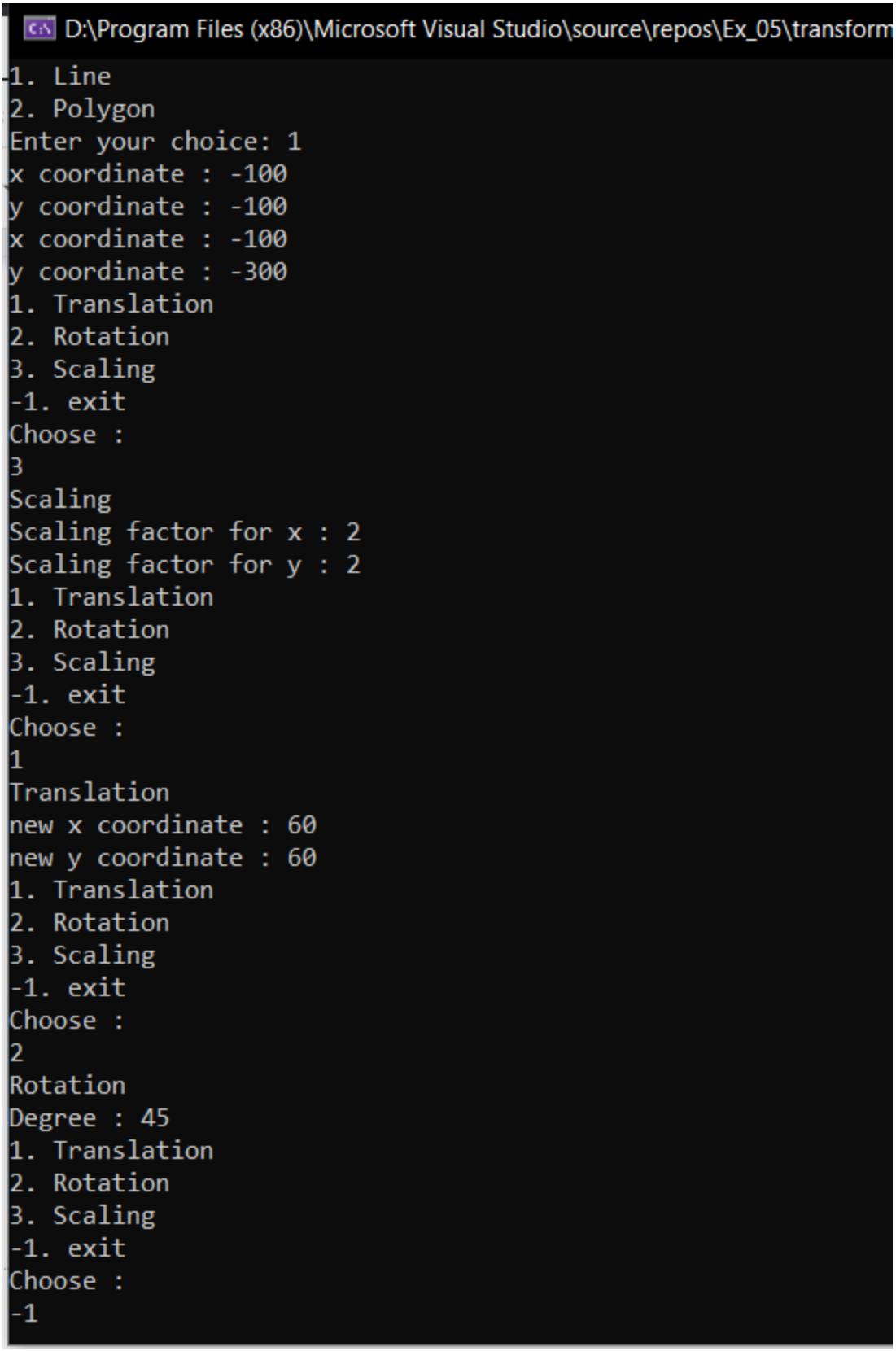
```
            case 3:
            {
                cout << "Scaling\n";
                cout << "Scaling factor for x : ";
                cin >> x;
                cout << "Scaling factor for y : ";
                cin >> y;
                scaleLine(x, y);
                break;
            }
            }
            printMenu();
            cin >> choice;
        }
        drawLine();
    }
    else
    {
        cout << "Number of Edges: ";
        cin >> vertices;
        for (int i = 0; i < vertices; i++)
        {
            cout << "x coordinate : ";
            cin >> x;
            cout << "y coordinate : ";
            cin >> y;
            pntX.push_back(x);
            pntY.push_back(y);
        }
        printMenu();
        cin >> choice;
        while (choice != -1)
        {
            switch (choice)
            {
            case 1:
            {
                cout << "Translation\n";
                cout << "new x coordinate : ";
                cin >> x;
                cout << "new y coordinate : ";
                cin >> y;
                translatePolygon(x, y);
```

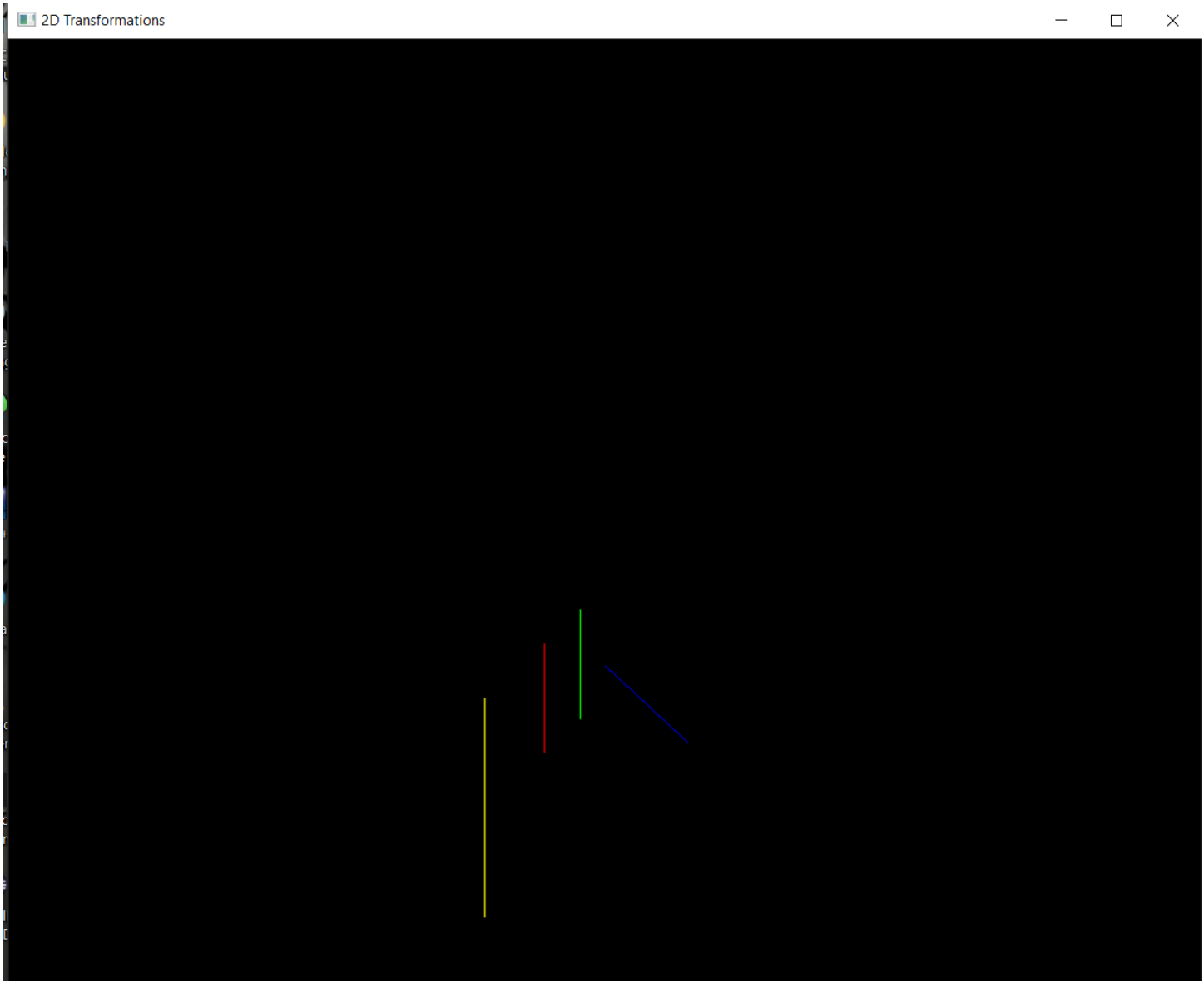
```
                break;
            }
            case 2:
            {
                cout << "Rotation\n";
                cout << "Degree : ";
                cin >> degree;
                rotatePolygon(degree * 3.1416 / 180);
                break;
            }
            case 3:
            {
                cout << "Scaling\n";
                cout << "Scaling factor for x : ";
                cin >> x;
                cout << "Scaling factor for y : ";
                cin >> y;
                scalePolygon(x, y);
                break;
            }
            }
            printMenu();
            cin >> choice;
        }
        drawPolygon();
    }

    glFlush();
}

int main(int argc, char** argv)
{

    glutInit(&argc, argv);                 // Initialize GLUT
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

    glutInitWindowSize(1000, 1000);   // Set the window's initial width & height
    glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
    glutCreateWindow("2D Transformations"); // Create a window with the given title

    myInit();
    glutDisplayFunc(display); // Register display callback handler for window re-paint
    glutMainLoop();           // Enter the infinitely event-processing loop
```

```
    return 0;
}

// 0 0 400 0 400 200 0 200 2 2 50 50 120 -100 -100 -100 -300 2 2 60 60 45
```

Input/Output Screenshot:

Line:

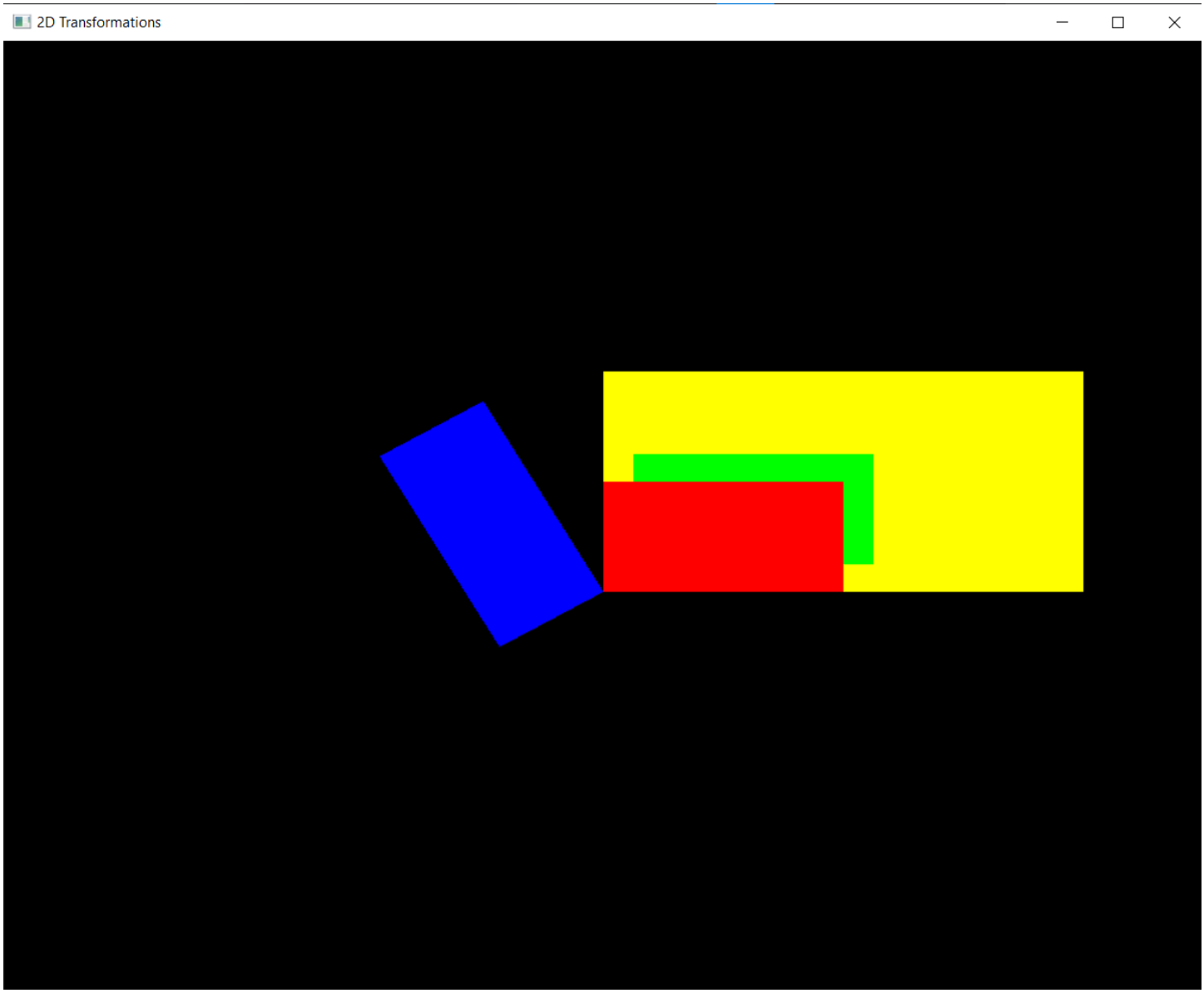

D:\Program Files (x86)\Microsoft Visual Studio\source\repos\Ex_05\transform
1. Line
2. Polygon
Enter your choice: 1
x coordinate : -100
y coordinate : -100
x coordinate : -100
y coordinate : -300
1. Translation
2. Rotation
3. Scaling
-1. exit
Choose :
3
Scaling
Scaling factor for x : 2
Scaling factor for y : 2
1. Translation
2. Rotation
3. Scaling
-1. exit
Choose :
1
Translation
new x coordinate : 60
new y coordinate : 60
1. Translation
2. Rotation
3. Scaling
-1. exit
Choose :
2
Rotation
Degree : 45
1. Translation
2. Rotation
3. Scaling
-1. exit
Choose :
-1



Red - normal line.
Green - translated on both x and y axis by 60 each.
Yellow - scaled on both axes by a factor of 2.
Blue - Rotated about 45 degrees in anti-clockwise direction.

Polygon:

```
D:\Program Files (x86)\Microsoft Visual Studio\source\repos\Ex_05\transformatio
1. Line
2. Polygon
Enter your choice: 2
Number of Edges: 4
x coordinate : 0
y coordinate : 0
x coordinate : 400
y coordinate : 0
x coordinate : 400
y coordinate : 200
x coordinate : 0
y coordinate : 200
1. Translation
2. Rotation
3. Scaling
-1. exit
Choose :
3
Scaling
Scaling factor for x : 2
Scaling factor for y : 2
1. Translation
2. Rotation
3. Scaling
-1. exit
Choose :
1
Translation
new x coordinate : 50
new y coordinate : 50
1. Translation
2. Rotation
3. Scaling
-1. exit
Choose :
2
Rotation
Degree : 120
1. Translation
2. Rotation
3. Scaling
-1. exit
Choose :
-1
```



Red - normal rectangle.
Green - translated on both x and y axis by 50 each.
Yellow - scaled on both axes by a factor of 2.
Blue - Rotated about 120 degrees in anti-clockwise direction.

Result:

Thus a menu driven program is created using c++ with opengl to make 2d transformations on lines and polygons.