SSN COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1712 – GRAPHICS AND MULTIMEDIA LAB

Name : Rahul Ram M
Reg .No : 185001121
Date : 04/09/2021

EX NO: 5b

2D Transformations – Reflection and Shearing

Write a C++ menu-driven program using OPENGL to perform 2D transformations – reflection and shearing for polygons

Aim:

To write a C++ menu-driven program using OPENGL to perform 2D transformations – reflection and shearing for polygons.

Algorithm:

1. Reflection along X axis:
    a. Make y = -y and using GL_QUADS draw the polygon.
2. Reflection alongY axis:
    a. Make x = -x and using GL_QUADS draw the polygon.
3. Reflection along Origin:
    a. Make x = -x and y = -y and draw the polygon.
4. Reflection along X=Y:
    a. Make x=y and y = x and draw the polygon.
5. Shearing along the X axis:
    a. Read Shearing parameter from the user.
    b. Add the shearing parameter to the x coordinate for the above two points and draw the polygon.
6. Shearing along the Y axis:
    a. Read Shearing parameter from the user.
    b. Add the shearing parameter to the y coordinate for the above two points and draw the polygon.

Code:

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

vector<int> pntX;
vector<int> pntY;

int choice, vertices;

void myInit()
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glPointSize(1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glEnable(GL_BLEND); //Enable blending.
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA); //Set blending function.
    gluOrtho2D(-1000, 1000, -1000, 1000);
}

void drawPolygon()
{
    glBegin(GL_QUADS);
    glColor4f(1.0f, 0.0f, 0.0f, 0.3f);
    for (int i = 0; i < vertices; i++)
    {
        glVertex2i(pntX[i], pntY[i]);
    }
    glEnd();
}

void refectionAlongX()
{
    glBegin(GL_QUADS);
    glColor3f(0.0f, 1.0f, 0.0f);
    for (int i = 0; i < vertices; i++)
    {
        glVertex2i(pntX[i], -1 * pntY[i]);
```

```
    }
    glEnd();
}

void refectionAlongY()
{
    glBegin(GL_QUADS);
    glColor3f(0.0f, 0.0f, 1.0f);
    for (int i = 0; i < vertices; i++)
    {
        glVertex2i(-1 * pntX[i], pntY[i]);
    }
    glEnd();
}

void reflectionAlongAxis()
{
    glBegin(GL_QUADS);
    glColor3f(0.5f, 0.5f, 1.0f);
    for (int i = 0; i < vertices; i++)
    {
        glVertex2i(-1 * pntX[i], -1 * pntY[i]);
    }
    glEnd();
}

void reflectionAlongXY()
{
    glBegin(GL_QUADS);
    glColor3f(1.0f, 0.5f, 0.5f);
    for (int i = 0; i < vertices; i++)
    {
        glVertex2i(pntY[i], pntX[i]);
    }
    glEnd();
}

void shearingAlongX(int x)
{
    glBegin(GL_QUADS);

    glColor3f(0.0f, 1.0f, 1.0f);
    for (int i = 0; i < vertices; i++)
```

```
    {
        if (i >= 2)
        {
            glVertex2i(pntX[i] + x, pntY[i]);
        }
        else
        {
            glVertex2i(pntX[i], pntY[i]);
        }
    }
    glEnd();
}

void shearingAlongY(int y)
{
    glBegin(GL_QUADS);
    glColor4f(1.0f, 0.0f, 1.0f, 0.7f);
    for (int i = 0; i < vertices; i++)
    {
        if (i >= 1 && i < 3)
        {
            glVertex2i(pntX[i], pntY[i] + y);
        }
        else
        {
            glVertex2i(pntX[i], pntY[i]);
        }
    }
    glEnd();
}

void printMenu()
{
    cout << "\n1. Reflection along X-axis" << "\n";
    cout << "2. Reflection along Y-axis" << "\n";
    cout << "3. Reflection along Origin" << "\n";
    cout << "4. Reflection along x=y" << "\n";
    cout << "5. Shearing along X-axis" << "\n";
    cout << "6. Shearing along Y-axis" << "\n";
    cout << "-1. exit" << "\n";
    cout << "Choose : " << "\n";
}
```

```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    int x, y;

    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);
    }

    //drawPolygon();

    printMenu();
    cin >> choice;
    while (choice != -1)
    {
        switch (choice)
        {
            case 1:
            {
                refectionAlongX();
                break;
            }
            case 2:
            {
                refectionAlongY();
                break;
            }
            case 3:
            {
                reflectionAlongAxis();
                break;
            }
            case 4:
            {
```

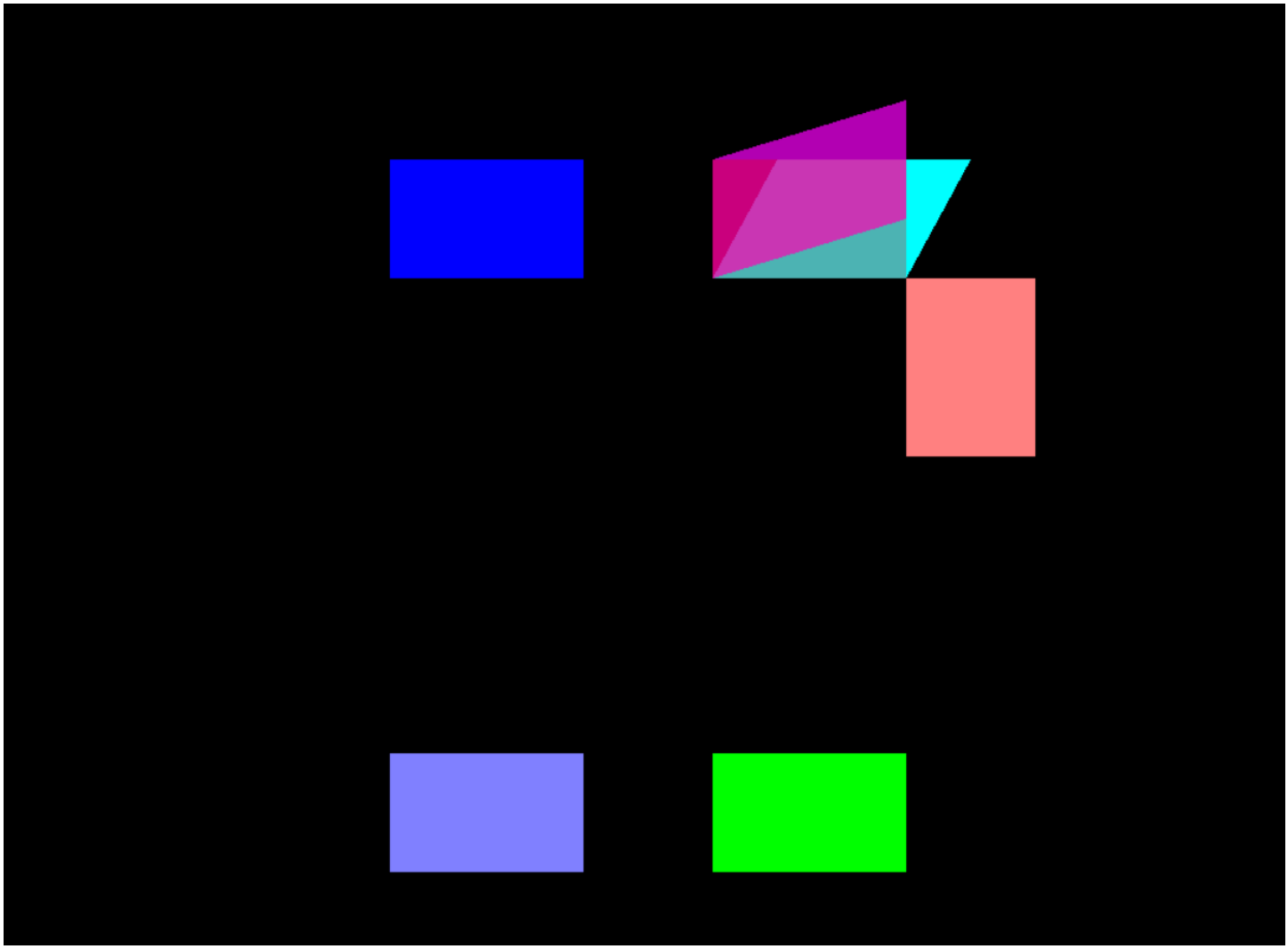
```
                reflectionAlongXY();
                break;
            }
            case 5:
            {
                cout << "Shearing parameter along X-axis: ";
                cin >> x;
                shearingAlongX(x);
                break;
            }
            case 6:
            {
                cout << "Shearing parameter along Y-axis: ";
                cin >> y;
                shearingAlongY(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;
}

// 4 100 100 400 100 400 300 100 300 1 2 3 4 5 100 6 100 -1

// 4 250 100 400 250 250 400 100 250 1 2 3 4 5 100 6 100 -1
```

```
// 4 100 400 400 400 400 600 100 600 1 2 3 4 5 100 6 100 -1
```

Input/Output Screenshot:

```
Number of edges : 4
x coordinate : 100
y coordinate : 400
x coordinate : 400
y coordinate : 400
x coordinate : 400
y coordinate : 600
x coordinate : 100
y coordinate : 600

1. Reflection along X-axis
2. Reflection along Y-axis
3. Reflection along Origin
4. Reflection along x=y
5. Shearing along X-axis
6. Shearing along Y-axis
-1. exit
Choose :
1

1. Reflection along X-axis
2. Reflection along Y-axis
3. Reflection along Origin
4. Reflection along x=y
5. Shearing along X-axis
6. Shearing along Y-axis
-1. exit
Choose :
2

1. Reflection along X-axis
2. Reflection along Y-axis
3. Reflection along Origin
4. Reflection along x=y
5. Shearing along X-axis
6. Shearing along Y-axis
-1. exit
Choose :
3

1. Reflection along X-axis
2. Reflection along Y-axis
3. Reflection along Origin
4. Reflection along x=y
5. Shearing along X-axis
6. Shearing along Y-axis
-1. exit
Choose :
```

```
1. Reflection along X-axis
2. Reflection along Y-axis
3. Reflection along Origin
4. Reflection along x=y
5. Shearing along X-axis
6. Shearing along Y-axis
-1. exit
Choose :
4

1. Reflection along X-axis
2. Reflection along Y-axis
3. Reflection along Origin
4. Reflection along x=y
5. Shearing along X-axis
6. Shearing along Y-axis
-1. exit
Choose :
5
Shearing parameter along X-axis: 100

1. Reflection along X-axis
2. Reflection along Y-axis
3. Reflection along Origin
4. Reflection along x=y
5. Shearing along X-axis
6. Shearing along Y-axis
-1. exit
Choose :
6
Shearing parameter along Y-axis: 100

1. Reflection along X-axis
2. Reflection along Y-axis
3. Reflection along Origin
4. Reflection along x=y
5. Shearing along X-axis
6. Shearing along Y-axis
-1. exit
Choose :
-1
Number of edges :
```



Overlapping red rectangle - original
Overlapping pink - shearing along y axis
Overlapping light blue - shearing along x axis
Thick blue on left - reflection along y axis
Green bottom right - reflection along x axis
Violet bottom left - reflection along origin
Maroon top right - reflection along x=y

Result:

Thus a menu driven program is created using c++ with opengl to make 2d transformations(shearing, reflection) polygon.