UCS1712 - GRAPHICS AND MULTIMEDIA LAB

EX - 5b: REFLECTION AND SHEARING

NAME: KEERTHANA T
REGISTER NUMBER: 185001074
CLASS-SEC: CSE-B
DATE: 02/09/2021

AIM :

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

a. Reflection – Show reflection along the x-axis, y-axis, about the origin, about line x=y.

b. Shearing – Show shearing along the x-axis and y-axis

ALGORITHM :

- Start
- Import GL library as a header file
- Create a function void drawPolygon() to draw a polygon for the given vertices.
- Create a function void mutliplyMatrices() to assigned the new parameters of the polygon after Reflection and shearing.
- Create a function void drawReflection() to draw the reflected polygon based on the given line for reflection
- Create a function void drawShear() to draw the sheared polygon based on the sheared factor on the preferred axis.
- Create a function void drawLine() to draw the line for reflection.
- Create a function void myInit() for the initial configuration of the display window

- Create a function void myDisplay()
- Now create a menu-driven program for choosing the kind of operation to be done for the given polygon.
- If the choice is:
    1. Reflection about the origin
    2. Reflection about X-axis
    3. Reflection about Y-axis
    4. Reflection about X=Y
    5. Shearing along X-axis
    6. Shearing along Y-axis
    7. Exit
- Create the void main() function
- Get parameters for the initial polygon.
- Implement the menu-driven program and get the input choice for the operation to be executed.
- After choosing the operation take in the input parameters for the operation to be executed.
- End

CODE:

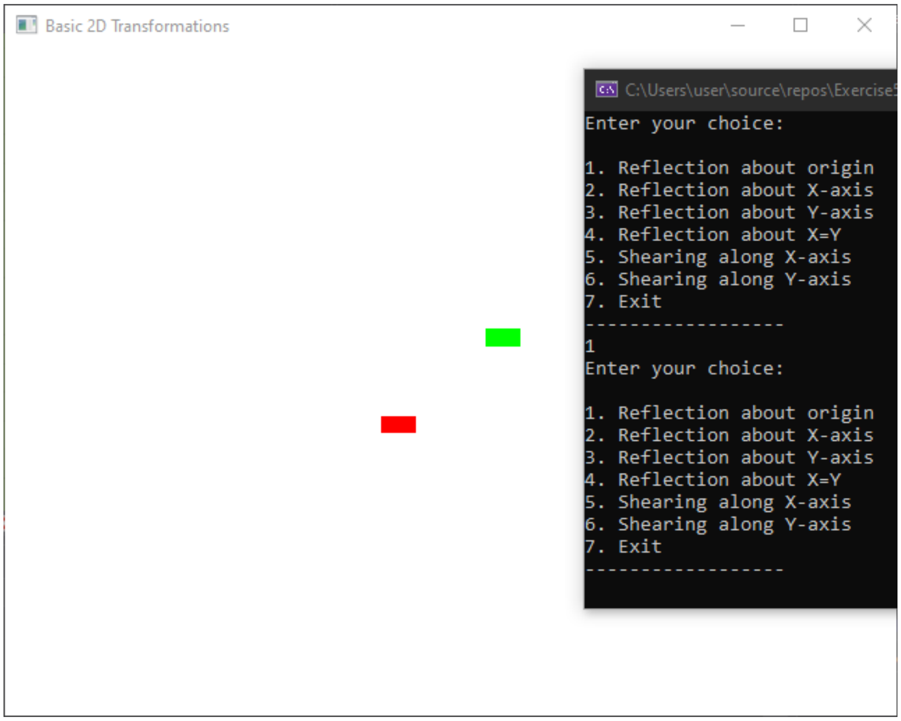
```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <iostream>
#include <vector>
#include <GL/glut.h>
using namespace std;

int pntX1, pntY1, choice = 0, edges;
vector<int> pntX;
vector<int> pntY;
int shx, shy;
int res[3][1] = { 0 };
```

```
const int REFLECTION_MATRIX[4][3][3] = { {{-1, 0, 0}, {0, -1, 0}, {0, 0, 1}}, {{1, 0, 0}, {0, -1, 0}, {0, 0, 1}}, {{-1, 0, 0}, {0, 1, 0}, {0, 0, 1}}, {{0, 1, 0}, {1, 0, 0}, {0, 0, 1}} };


enum LINE_TYPE {
    ORIGIN, X_AXIS, Y_AXIS, XY_LINE
};

void mutliplyMatrices(const int tr[3][3], const int pt[3][1]) {
    memset(res, 0, 3 * 1 * sizeof(int));

    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 1; j++)
            for (int k = 0; k < 3; k++)
                res[i][j] += tr[i][k] * pt[k][j];

}

void drawPolygon()
{
    glBegin(GL_QUADS);
    glColor3f(0.0, 1.0, 0.0);
    for (int i = 0; i < 4; i++)
    {
        glVertex2i(pntX[i], pntY[i]);
    }
    glEnd();
}
void drawReflection(LINE_TYPE lt) {
    glBegin(GL_QUADS);
    glColor3f(1.0, 0.0, 0.0);
    for (int i = 0; i < 4; i++)
    {
        int pt_arr[3][1] = { {pntX[i]}, {pntY[i]}, {1} };
        mutliplyMatrices(REFLECTION_MATRIX[lt], pt_arr);
        glVertex2i(res[0][0], res[1][0]);
    }
    glEnd();
}

void drawShear(LINE_TYPE lt) {
    glBegin(GL_QUADS);
    glColor3f(1.0, 0.0, 0.0);
    if (lt == X_AXIS) {
        int shear_matrix[3][3] = { {1, shx, 0}, {0, 1, 0}, {0, 0, 1} };
        for (int i = 0; i < 4; i++)
        {
            int pt_arr[3][1] = { {pntX[i]}, {pntY[i]}, {1} };
            mutliplyMatrices(shear_matrix, pt_arr);
            glVertex2i(res[0][0], res[1][0]);
        }
        glEnd();
    }
    else if (lt == Y_AXIS) {
        int shear_matrix[3][3] = { {1, 0, 0}, {shy, 1, 0}, {0, 0, 1} };
        for (int i = 0; i < 4; i++)
        {
            int pt_arr[3][1] = { {pntX[i]}, {pntY[i]}, {1} };
            mutliplyMatrices(shear_matrix, pt_arr);
            glVertex2i(res[0][0], res[1][0]);
        }
        glEnd();
    }

}

void drawLine(LINE_TYPE lt) {
    glPushAttrib(GL_ENABLE_BIT);
    glLineStipple(1, 0xAAAA);
    glLineWidth(2);
    glEnable(GL_LINE_STIPPLE);
    glBegin(GL_LINES);
    glColor3f(0.0, 0.0, 1.0);

    switch (lt) {
    case X_AXIS: {
        glVertex2i(-700, 0);
        glVertex2i(700, 0);
    }
               break;
    case Y_AXIS: {
        glVertex2i(0, -700);
        glVertex2i(0, 700);
    }
               break;
    case XY_LINE: {
        glVertex2i(-700, -700);
        glVertex2i(700, 700);
    }
                break;

    }
    glPopAttrib();
    glEnd();

}

void myInit(void)
{
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glColor3f(0.0f, 0.0f, 0.0f);
    glPointSize(3.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-640.0, 640.0, -480.0, 480.0);
}

void myDisplay(void)
{
    // Initial Polygon
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0, 0.0, 0.0);
    drawPolygon();
    glFlush();
    while (true)
    {
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(0.0, 0.0, 0.0);
        cout << "Enter your choice:\n\n";
        cout << "1. Reflection about origin" << endl;
        cout << "2. Reflection about X-axis" << endl;
        cout << "3. Reflection about Y-axis" << endl;
        cout << "4. Reflection about X=Y" << endl;
        cout << "5. Shearing along X-axis" << endl;
        cout << "6. Shearing along Y-axis" << endl;
        cout << "7. Exit" << endl;

        cout << "------------------" << endl;
        cin >> choice;

        if (choice == 7)
        {
            return;
        }
        LINE_TYPE lt;
        switch (choice) {
        case 1: {
            drawPolygon();
            lt = ORIGIN;
            drawReflection(lt);
        }
              break;
        case 2: {
            drawPolygon();
            lt = X_AXIS;
            drawLine(lt);
            drawReflection(lt);
        }
              break;
        case 3: {
            drawPolygon();
            lt = Y_AXIS;
            drawLine(lt);
            drawReflection(lt);
        }
              break;
        case 4: {
            drawPolygon();
            lt = XY_LINE;
            drawLine(lt);
            drawReflection(lt);
        }
              break;
        case 5: {
            cout << "Enter the shearing factor along x axis: " << endl;
            cin >> shx;
            drawPolygon();
            lt = X_AXIS;
            drawShear(lt);
        }
              break;
        case 6: {
            cout << "Enter the shearing factor along y axis: " << endl;
```

```
            cin >> shy;
            drawPolygon();
            lt = Y_AXIS;
            drawShear(lt);
        }
        }
        glFlush();
    }

}

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

    cout << "Enter vertices\n";
    for (int i = 0; i < 4; i++)
    {
        cout << "Enter co-ordinates for vertex  " << i + 1 << "(X, Y): ";
        cin >> pntX1 >> pntY1;
        pntX.push_back(pntX1);
        pntY.push_back(pntY1);
    }
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(100, 150);
    glutCreateWindow("Basic 2D Transformations");
    glutDisplayFunc(myDisplay);
    myInit();
    glutMainLoop();
}
```

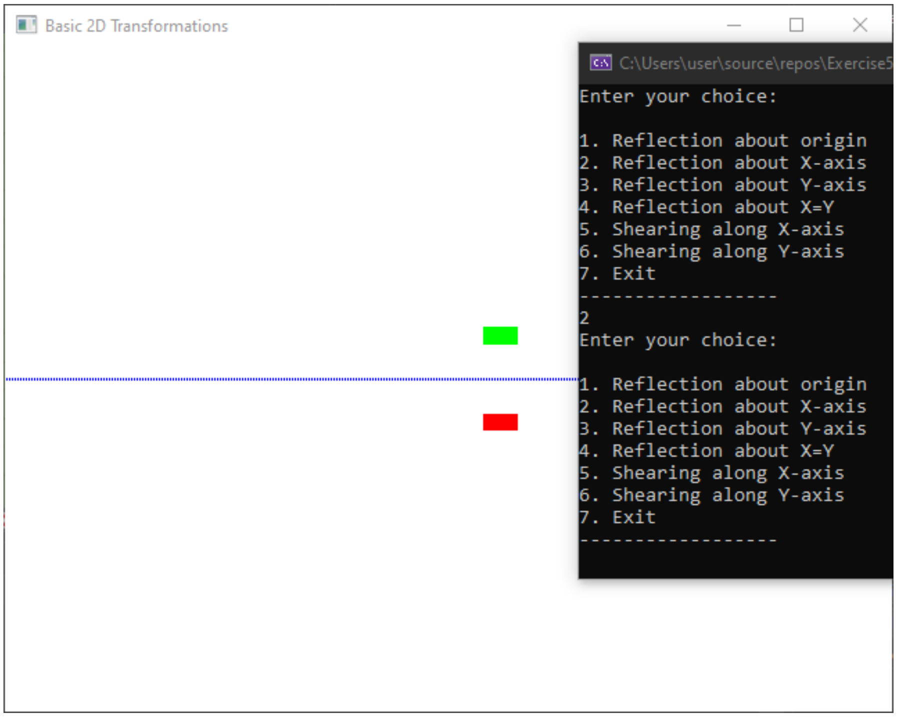
OUTPUT:

1. Reflection about the origin



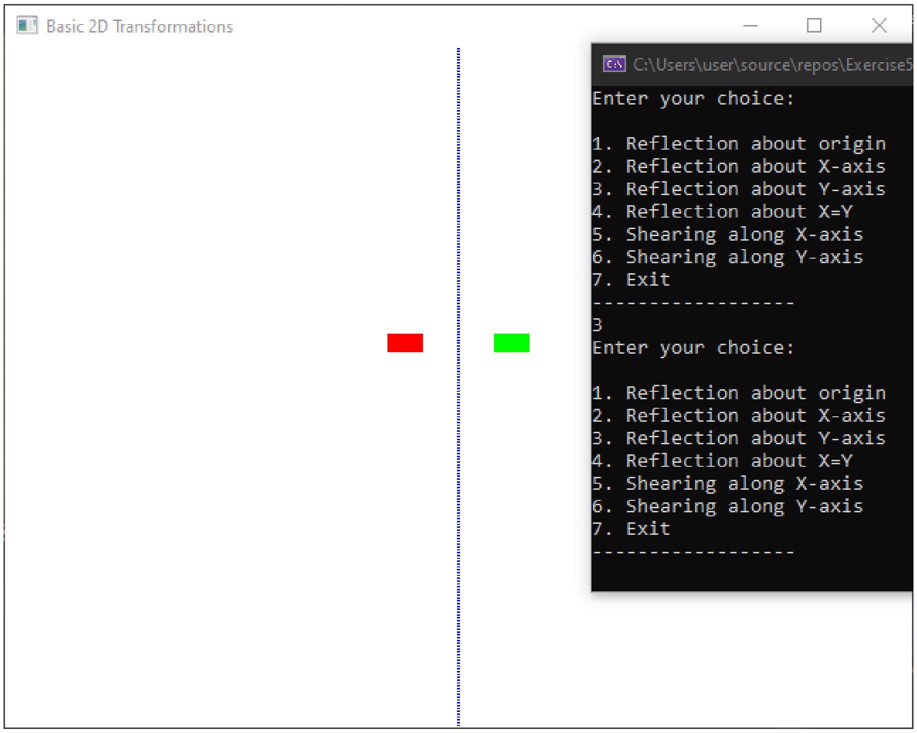
```
Enter your choice:

1. Reflection about origin
2. Reflection about X-axis
3. Reflection about Y-axis
4. Reflection about X=Y
5. Shearing along X-axis
6. Shearing along Y-axis
7. Exit
------------------
1
Enter your choice:

1. Reflection about origin
2. Reflection about X-axis
3. Reflection about Y-axis
4. Reflection about X=Y
5. Shearing along X-axis
6. Shearing along Y-axis
7. Exit
------------------
```

2. Reflection about X-axis:



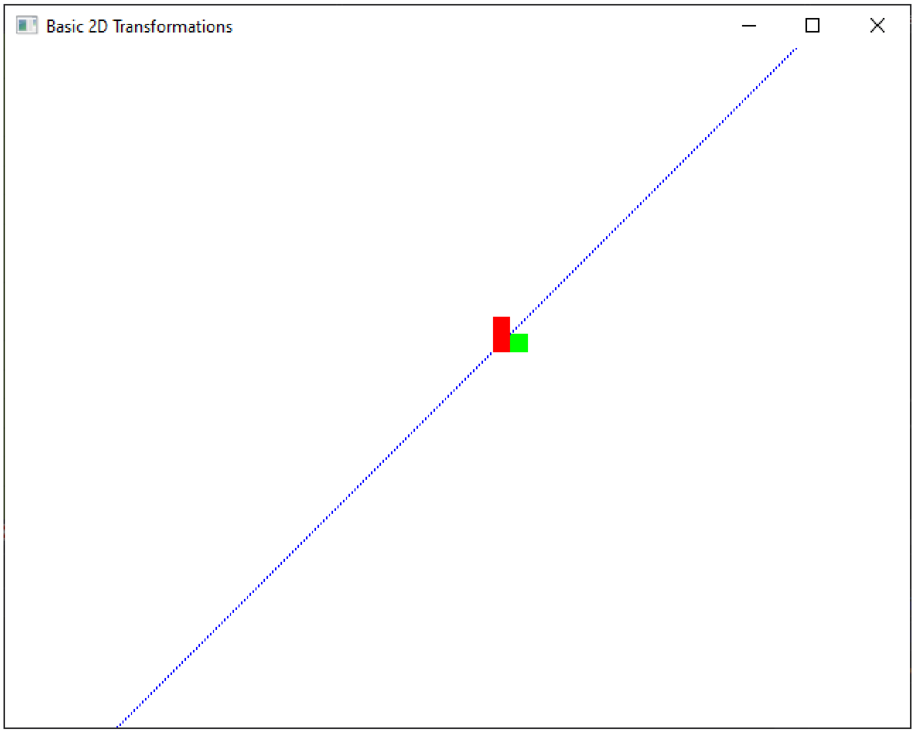
```
Enter your choice:

1. Reflection about origin
2. Reflection about X-axis
3. Reflection about Y-axis
4. Reflection about X=Y
5. Shearing along X-axis
6. Shearing along Y-axis
7. Exit
------------------
2
Enter your choice:

1. Reflection about origin
2. Reflection about X-axis
3. Reflection about Y-axis
4. Reflection about X=Y
5. Shearing along X-axis
6. Shearing along Y-axis
7. Exit
------------------
```

3. Reflection about Y-axis:

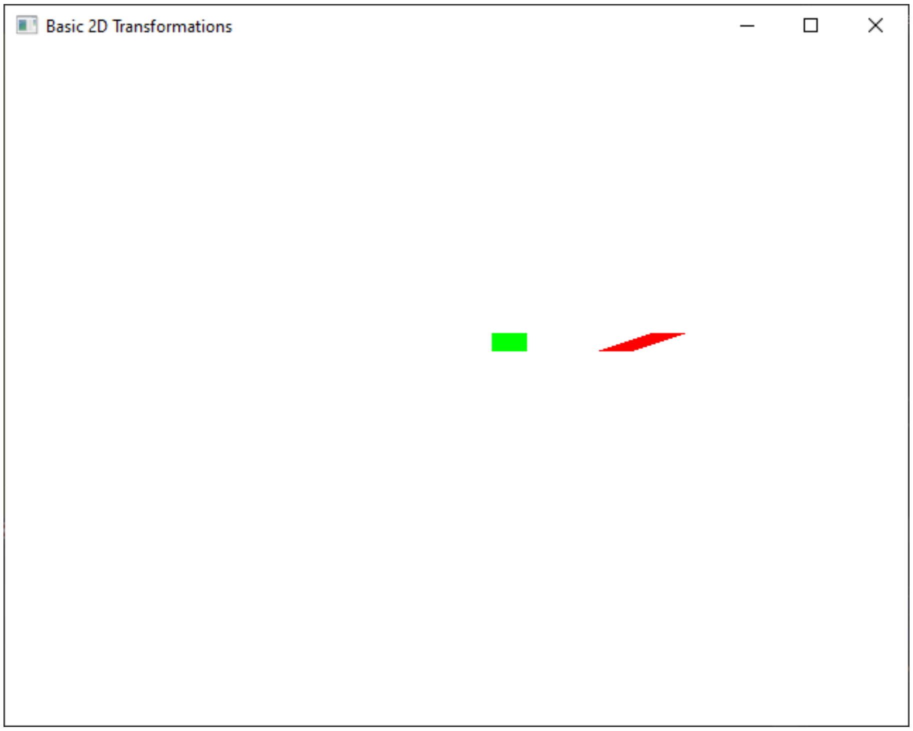


```
Enter your choice:

1. Reflection about origin
2. Reflection about X-axis
3. Reflection about Y-axis
4. Reflection about X=Y
5. Shearing along X-axis
6. Shearing along Y-axis
7. Exit
------------------
3
Enter your choice:

1. Reflection about origin
2. Reflection about X-axis
3. Reflection about Y-axis
4. Reflection about X=Y
5. Shearing along X-axis
6. Shearing along Y-axis
7. Exit
------------------
```

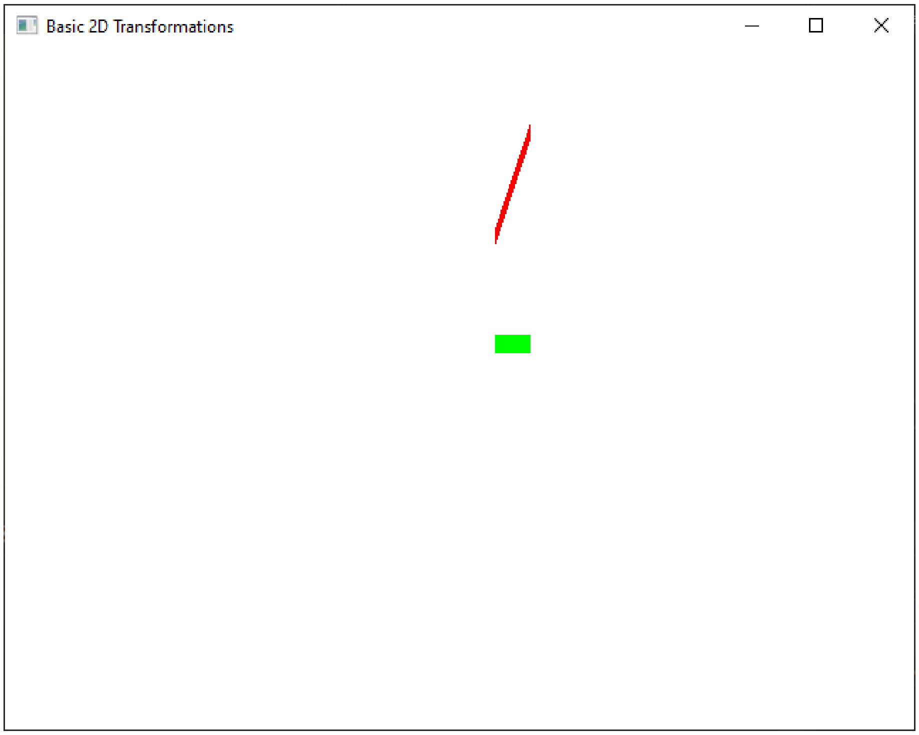
4. Reflection about x=y line



5. Shearing along X-axis with factor = 3:



6. Shearing along Y-axis with shearing factor:3



RESULT:

Thus we have successfully implemented the extended 2D transformations such as Reflection and shearing for the given polygon.