Graphics and Multimedia Lab
Ex5
2D Transformations in C++ using OpenGL

Sai Shashaank R
205001086

Question:

To apply the following 2D transformations on objects and to render the final output along with the original object.

1) Translation

2) Rotation

   a) about origin

   b) with respect to a fixed point (xr,yr)

3) Scaling with respect to

   a) origin - Uniform Vs Differential Scaling

   b) fixed point (xf,yf)

4) Reflection with respect to

   a) x-axis

   b) y-axis

   c) origin

   d) the line x=y

5) Shearing

   a) x-direction shear

   b) y-direction shear

Note: Use Homogeneous coordinate representations and matrix multiplication to perform transformations. Divide the output window into four quadrants. (Use LINES primitive to draw x and y axis.

Code:

```
#include <GLUT/GLUT.h>
#include <iostream>
#include <vector>
#include <math.h>
#define vvd vector<vector<double>>
#define vd vector<double>
using namespace std;
```

```
vvd points,points2;
void myInit(void){
    glClearColor(1,1,1,1);
    glMatrixMode(GL_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-500, 500, -500, 500);;
}
vvd multiply(vvd a,vvd b){
    int r1 = a.size();
    int c1 = a[0].size();

    int r2 = b.size();
    int c2 = b[0].size();

    vvd c(r1,vd(c2,0));

    for(int i=0;i<r1;i++){
        for(int j=0;j<c2;j++){
            for(int k=0;k<r2;k++){
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }

    return c;
}
vvd translate(vvd points, int tx, int ty){
    vvd T(3,vd(3,0));

    for(int i=0;i<3;i++){
        T[i][i] = 1;
    }

    T[0][2] = tx;
    T[1][2] = ty;

    return multiply(T, points);
}
vvd scale(vvd points,int sx, int sy){
    vvd S(3,vd(3,0));
```

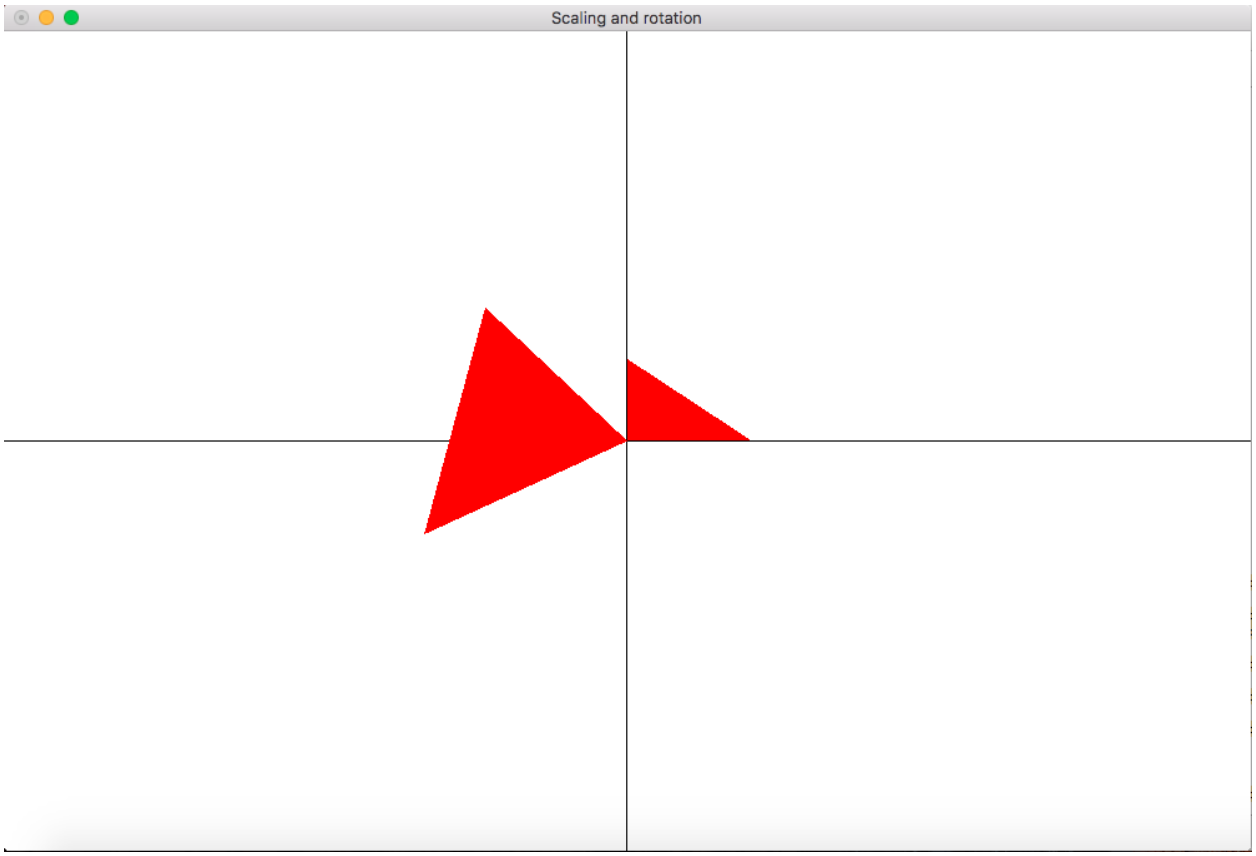
```
    S[0][0] = sx;
    S[1][1] = sy;
    S[2][2] = 1;

    return multiply(S,points);
}
vvd rotate(vvd P , double deg){

    double rad = (M_PI/180) * deg;

    vvd R(3,vd(3,0));

    R[0][0] = R[1][1] = cos(rad);

    R[2][2] = 1;

    R[0][1] = -1*sin(rad);

    R[1][0] = sin(rad);



    return multiply(R, P);

}

vvd shear(vvd& P , double shx,double shy){
    vvd SH(3,vd(3,0));
    SH[0][1] = shx;
    SH[1][0] = shy;
    for(int i=0;i<3;i++)SH[i][i] = 1;

    return multiply(SH, P);
}


void display_mat(vvd res){
    for(int i=0;i<res.size();i++){
        for(int j=0;j<res[0].size();j++)
            cout<<res[i][j]<<" ";
        cout<<endl;
    }
```

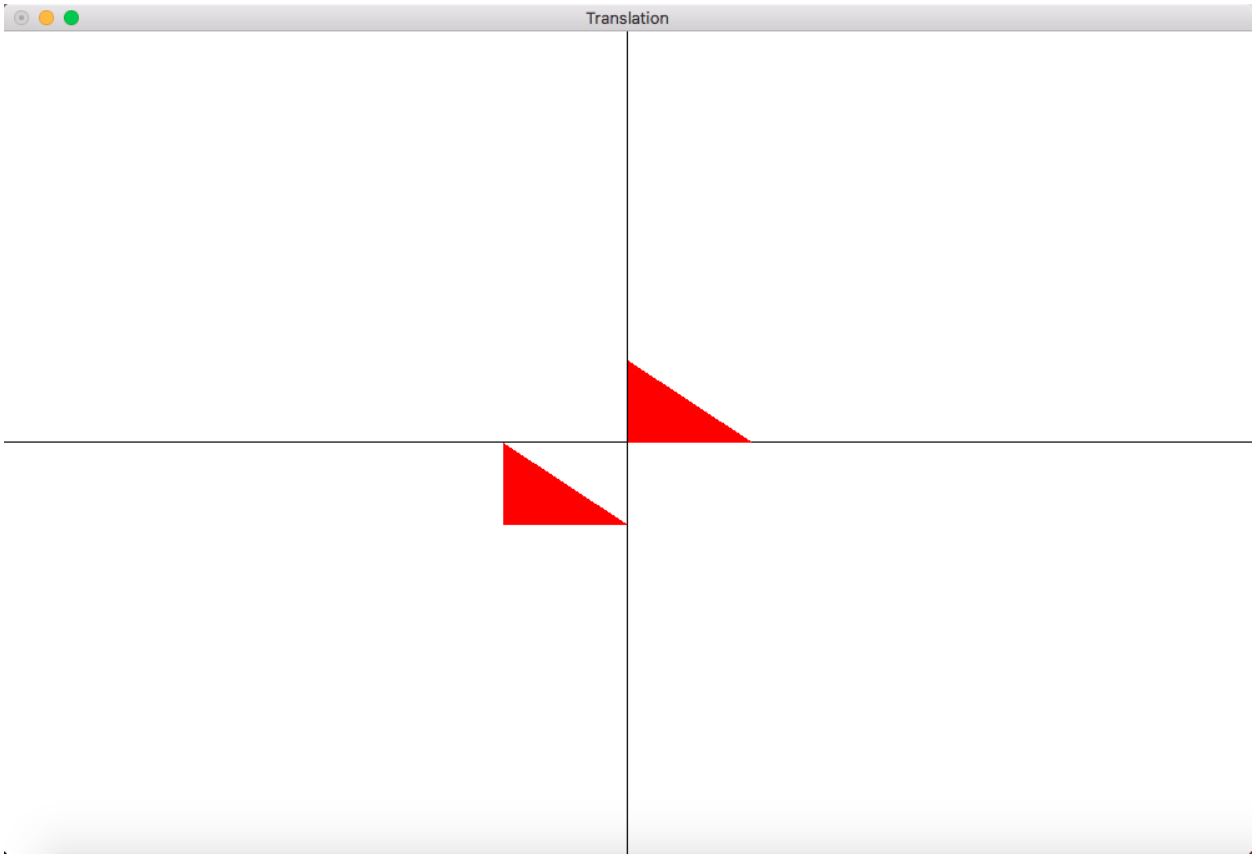
```
}
void plot(vvd fig1, vvd fig2){
    glColor3f(0,0,0);
    //Drawing the axes

    glBegin(GL_LINES);
    glVertex2i(-500,0);
    glVertex2i(500,0);
    glEnd();

    glBegin(GL_LINES);
    glVertex2i(0,-500);
    glVertex2i(0,500);
    glEnd();

    //cout<<"I am here.";

    /*cout<<"Figures";

    display_mat(fig1);
    display_mat(fig2);*/

    glColor3f(0,1,0);
    glEnable(GL_BLEND);

    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

    glBegin(GL_POLYGON);
    for(int i=0;i<fig1[0].size();i++)
        glVertex2i(fig1[0][i],fig1[1][i]);

    glEnd();

    glBegin(GL_POLYGON);
    for(int i=0;i<fig2[0].size();i++)
        glVertex2i(fig2[0][i],fig2[1][i]);

    glEnd();
}
void myDisplay(){
    glClear(GL_COLOR_BUFFER_BIT);
    plot(points,points2);
```

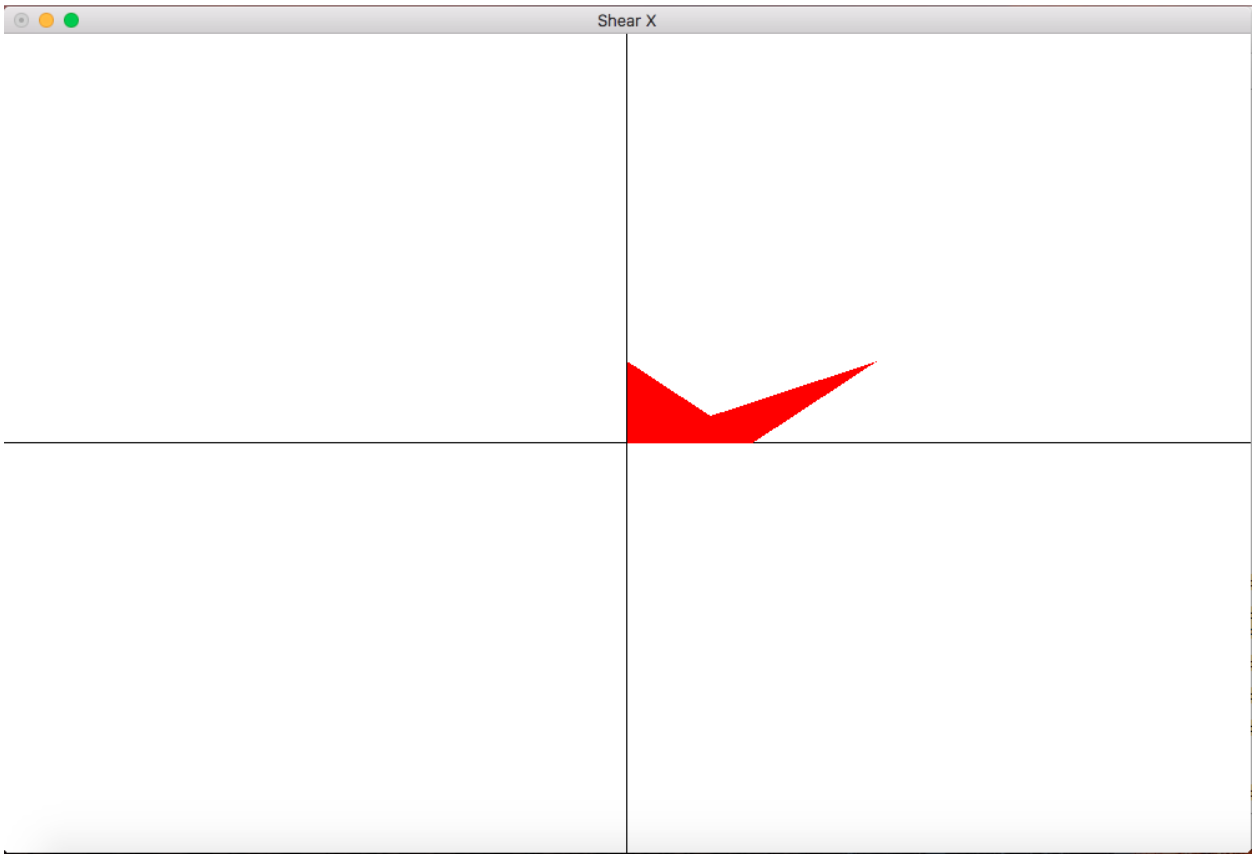
```
    glFlush();
}
int main(int argc, char * argv[]) {

    vvd a = {
        {1,0,1},
        {0,1,2},
        {0,0,1}
    };

    vvd b = {
        {1,3,3},
        {4,5,6},
        {7,8,9}
    };

    points = {
        {0,0,100},
        {0,100,0},
        {1,1,1}
    };

    points2 = rotate(scale(points,2,2),90+35);

    //display_mat(scale(points,2,2));

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(1000, 1000);

    glutInitWindowPosition(0, 0);

    glutCreateWindow("Transformation");

    glutDisplayFunc(myDisplay);

    myInit();

    glutMainLoop();

    return 0;
```

}

Output:




Translation



Learning Outcomes:

I learnt how to implement various 2D transformations using OpenGL in C++.