CS 352
Computer Graphics & Visualization
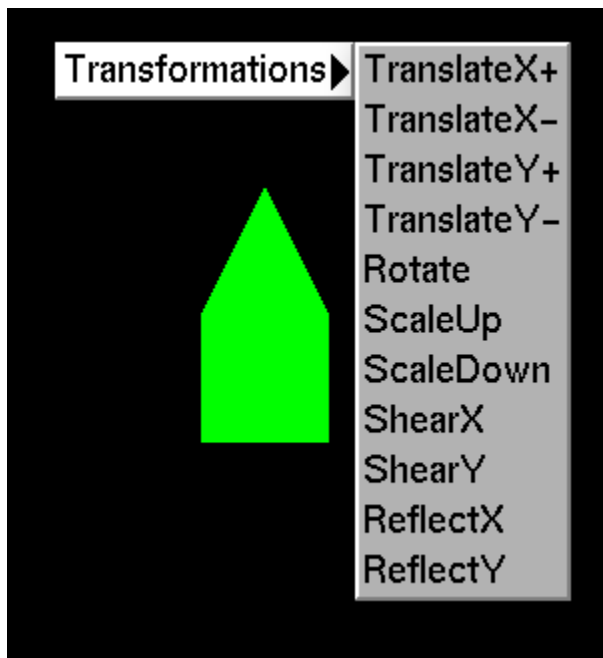# Assignment - 6
Name - Sairaj Loke
Roll No - 210001035

Index
1. Transforms

```cpp
#include <GL/glut.h>
#include <iostream>
#include<cmath>
using namespace std;


/*
Sairaj R. Loke 210001035
CS352 Lab6 Transforms

Note:
Note both run.sh and Makefile not needed but i have kept them as it is as I was trying both of
them to compare the use
I have added some simple modern c++ features and constructors as this assignment was very
simple and very low differentiating factor
*/

struct Color{
    GLfloat r = 0.0;
    GLfloat g = 0.0;
    GLfloat b = 0.0;

    Color(GLfloat r, GLfloat g, GLfloat b): r(r),g(g), b(b){}
};

//static just a good practice while working with larger system where i dont want this variable to
go out of this translational unit
static GLfloat polygon_vertices[][2] = {{-1,0}, {1, 0},{1, 2},{0,4},{-1,2}};  //initialoizing the pentagon
vertices
GLfloat PI_ = 3.14159 ;
static int polygon_sides = 5; // say a pentagon
static Color colorPoly(0.0,1.0,0.0) ;



void drawPolygon() {
    glBegin(GL_POLYGON);
    for(auto& val: polygon_vertices){  //using modern c++ features like range based for loops
        glVertex2fv(val);
    }
    glEnd();
}

void display() {
```

```cpp
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(colorPoly.r, colorPoly.g , colorPoly.b);
    drawPolygon();
    glFlush();
}

void translate(float dx, float dy) {
    for (auto& vertex : polygon_vertices) {
        vertex[0] += dx;
        vertex[1] += dy;
        cout<<vertex[0]<<" "<<vertex[1]<<endl;
    }
    glutPostRedisplay();
}

void rotate(float angle) {
    float theta = angle*(PI_ / 180.0);
    float tempX, tempY;
    for (auto&vertex: polygon_vertices) {
        tempX = vertex[0];
        tempY = vertex[1];
        vertex[0] = tempX*cos(theta) - tempY*sin(theta);
        vertex[1] = tempX*sin(theta) + tempY*cos(theta);
        cout<<vertex[0]<<" "<<vertex[1]<<endl;
    }
    glutPostRedisplay();  //https://www.opengl.org/resources/libraries/glut/spec3/node20.html  ,.
    // The next iteration through glutMainLoop, the window's display callback will be called to
redisplay the window's normal plane.
}

void scale(float sx, float sy) {
    for (auto& vertex: polygon_vertices) {
        vertex[0] *= sx;
        vertex[1] *= sy;
        cout<<vertex[0]<<" "<<vertex[1]<<endl;
    }
    glutPostRedisplay();
}

void shear(float shx, float shy) {
    for (auto& vertex: polygon_vertices) {
        vertex[0] += shx*vertex[1];
        vertex[1] += shy*vertex[0];
        cout<<vertex[0]<<" "<<vertex[1]<<endl;
    }
```

```cpp
    }
    glutPostRedisplay();
}

void reflect(char dir) {
    if (dir == 'y'){
        for (auto& vertex: polygon_vertices)
        vertex[0] = -vertex[0];// y axis
    }
    else if (dir == 'x'){
        for (auto& vertex: polygon_vertices)
        vertex[1] = -vertex[1];// x axis
    }


    glutPostRedisplay();
}

void menu(int choice) {
    switch (choice) {
        case 1:
            translate(5, 0);
            break;
        case 2:
            translate(-5, 0);
            break;
        case 3:
            translate(0, 5);
            break;
        case 4:
            translate(0, -5);
            break;
        case 5:
            rotate(20);
            break;
        case 6:
            scale(2, 2);
            break;
        case 7:
            scale(0.5, 0.5);
            break;
        case 8:
            shear(0.5, 0);
            break;
```

```cpp
        case 9:
            shear(0, 0.5);
            break;
        case 10:

            reflect('x');
            break;
        case 11:
            cout<<"Y";
            reflect('y');
            break;
        case 12:
            exit(0);
            break;


    }
}




int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800, 800);
    glutCreateWindow("Sairaj's 2D Transformations");
    gluOrtho2D(-200, 200, -200, 200);  //n sets up a two-dimensional orthographic viewing
region...so -200 to +200 in x, -200 to 200 in y
    glutDisplayFunc(display);

    int submenu = glutCreateMenu(menu);
    glutAddMenuEntry("TranslateX+", 1);
    glutAddMenuEntry("TranslateX-", 2);
    glutAddMenuEntry("TranslateY+", 3);
    glutAddMenuEntry("TranslateY-", 4);
    glutAddMenuEntry("Rotate", 5);
    glutAddMenuEntry("ScaleUp", 6);
    glutAddMenuEntry("ScaleDown", 7);
    glutAddMenuEntry("ShearX", 8);
    glutAddMenuEntry("ShearY", 9);
    glutAddMenuEntry("ReflectX", 10);
    glutAddMenuEntry("ReflectY", 11);
```
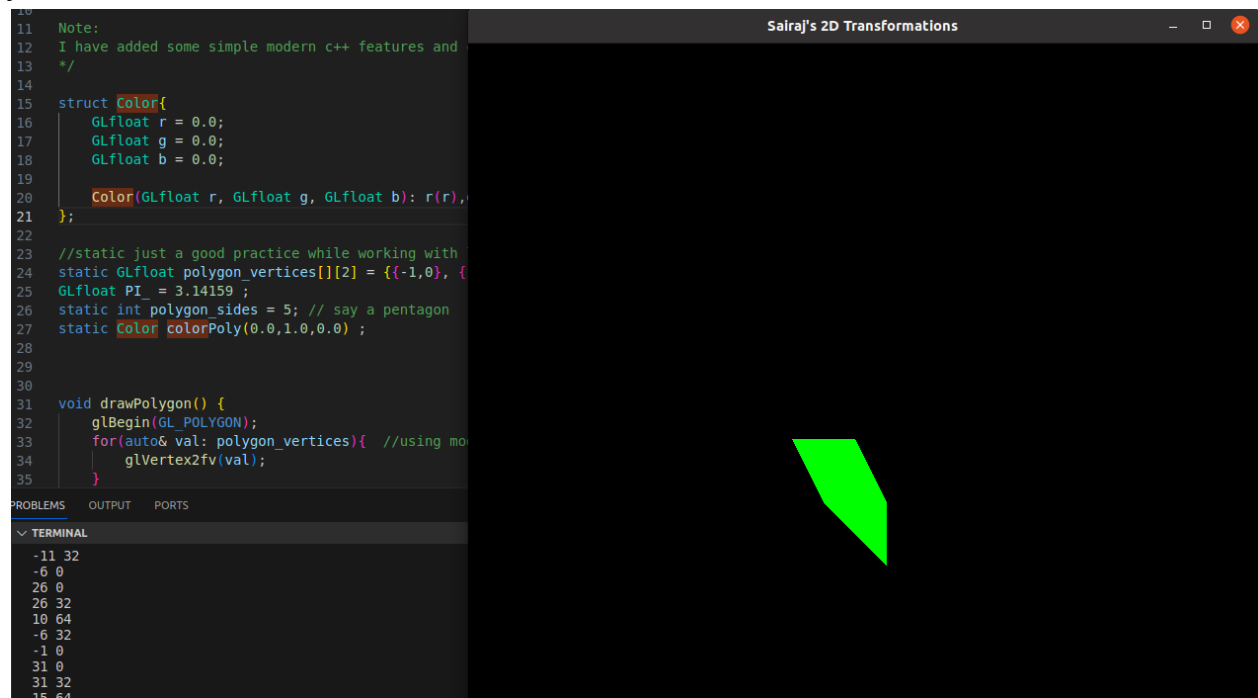
glutCreateMenu(menu);
glutAddSubMenu("Transformations", submenu);
// glutAddMenuEntry("Exit", 6);
glutAttachMenu(GLUT_RIGHT_BUTTON);

glutMainLoop();

return 0;
}

```cpp
11    Note:
12    I have added some simple modern c++ features and
13    */
14
15    struct Color{
16        GLfloat r = 0.0;
17        GLfloat g = 0.0;
18        GLfloat b = 0.0;
19
20        Color(GLfloat r, GLfloat g, GLfloat b): r(r),
21    };
22
23    //static just a good practice while working with
24    static GLfloat polygon_vertices[][2] = {{-1,0}, {
25    GLfloat PI_ = 3.14159 ;
26    static int polygon_sides = 5; // say a pentagon
27    static Color colorPoly(0.0,1.0,0.0) ;
28
29
30
31    void drawPolygon() {
32        glBegin(GL_POLYGON);
33        for(auto& val: polygon_vertices){  //using mo
34            glVertex2fv(val);
35        }
```

PROBLEMS   OUTPUT   PORTS

∨ TERMINAL
```
-11 32
-6 0
26 0
26 32
10 64
-6 32
-1 0
31 0
31 32
15 64
```

Sairaj's 2D Transformations

Sairaj's 2D Transformations

Transformations ▶

TranslateX+
TranslateX−
TranslateY+
TranslateY−
Rotate
ScaleUp
ScaleDown
ShearX
ShearY
ReflectX
ReflectY