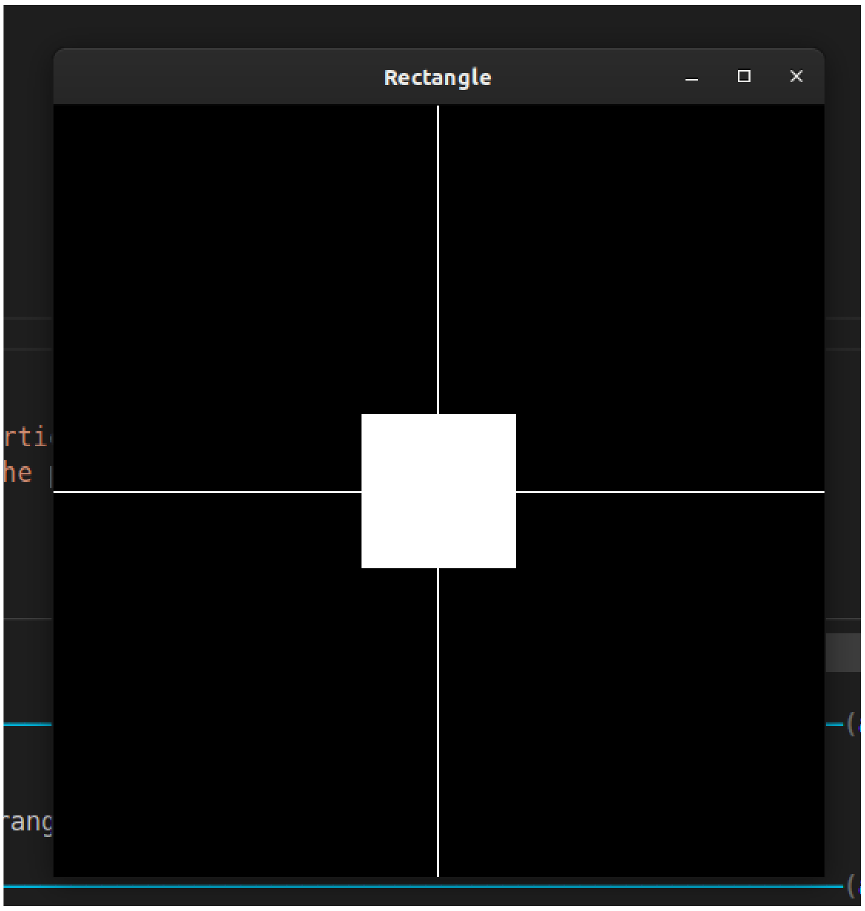
CS352 Assignment 2
openGL Rotate Translate scale

Ayush Agrawal
180001011
01-04-2021

Q1. Make rectangle using inbuild function: Code -

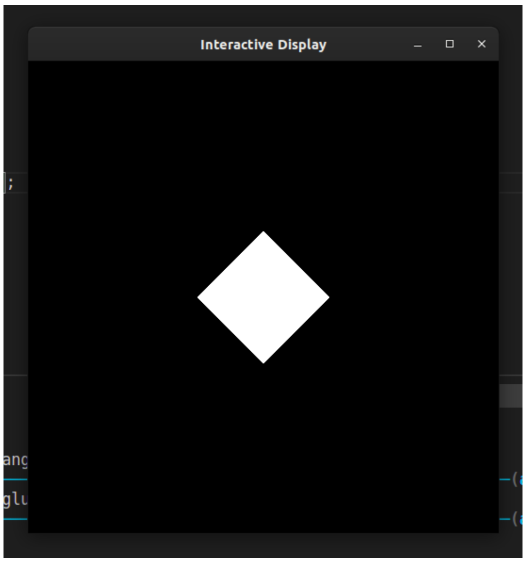
```
#include <iostream>
#include <GL/glut.h>
#include <vector>

using namespace std;

// Set window height and width.

const double WIDTH = 500;
const double HEIGHT = 500;

int x0,yy0,x2,y2;

// take user input
void initPoints(){
    cout<<"Enter the coordinates of the vertices in order.\n";
    cout<<"For the sake of viewing, keep the points in the range -600,600\n";
    cin>>x0>>yy0>>x2>>y2;
}

// main drawing function for glut
void drawObject(){
    glClear(GL_COLOR_BUFFER_BIT);

    // Draw the X-axis
    glBegin(GL_LINES);
        glVertex2f(-50, 0);
        glVertex2f(50, 0);
    glEnd();

    // Draw the y-axis
    glBegin(GL_LINES);
        glVertex2f(0, -50);
        glVertex2f(0, 50);
    glEnd();

    // draw rectangle
    glRecti(x0, yy0, x2, y2);
    glFlush();
}

void myInit (void){
    // Reset background color with black (since all three argument is 0.0)
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(0.0, 0.0, 0.0, 1.0);

    // Set width of point to one unit
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // Set window size in X- and Y- direction
    gluOrtho2D(-50, 50, -50, 50);
}

int main(int argc, char** argv){
    initPoints();
    // GLute init and create window
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(WIDTH,HEIGHT);
    glutInitWindowPosition(600,100);
    glutCreateWindow("Rectangle");

    myInit();
    glutDisplayFunc(drawObject);
    glutMainLoop();
}
```

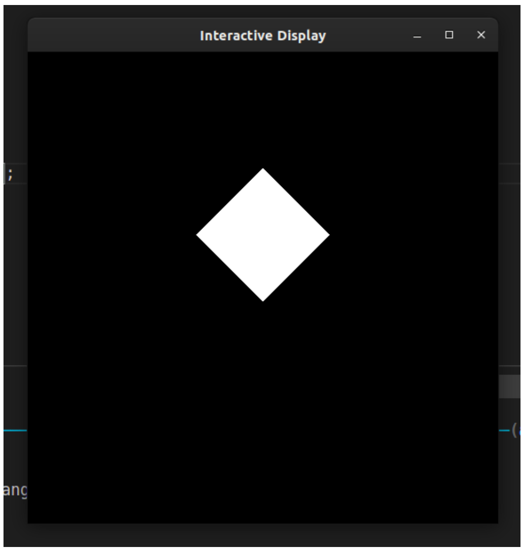


Q2. Scale Rotate Translate Sheer the polygon while running.

```
#include <iostream>
#include <GL/glut.h>
#include <vector>
#include <unistd.h>
// #include <utility>
using namespace std;

const double WIDTH = 500;
const double HEIGHT = 500;

int x0,yy0,x2,y2;

bool wantTranslate = false;
bool wantScale = false;
bool wantRotate = false;
bool wantSheer = false;

char inp;

// key inputs for transformation;
char TRANSLATE = 't';
char SCALE = 's';
char ROTATE = 'r';
char SHEER = 'x';


// take input from user;
void initPoints(){
   cout<<"Enter the coordinates of the vertices in order.\n";
   cout<<"For the sake of viewing, keep the points in the range -50,50\n";
   cin>>x0>>yy0>>x2>>y2;
}

// main drawing loop;
void drawObject(){
      glClear(GL_COLOR_BUFFER_BIT);

      // use by default functions to scale, rotate, translate

      if(wantTranslate){
         cout<<"wantTranslate: "<<wantTranslate<<"\n";
         glTranslatef(2, 2, 0);
      }

      // if scaled was pressed then scale.
      if(wantScale){
         cout<<"wantScale: "<<wantScale<<"\n";
         glScalef(0.5, 0.5, 1);
      } else {
         // glPopMatrix();
```

```
        }
        if(wantRotate){
            glRotatef(45, 0, 0, 1);
            cout<<"wantRotate: "<<wantRotate<<"\n";
        }
        if(wantSheer){
            // not available by default so using matrix multiplication

            GLfloat m[16] = {
                1.0f, 0.0f, 0.0f, 0.0f,
                2.0f, 1.0f, 0.0f, 0.0f,
                0.0f, 0.0f, 1.0f, 0.0f,
                0.0f, 0.0f, 0.0f, 1.0f
            };
            glMultMatrixf(m);
            cout<<"wantSheer: "<<wantSheer<<"\n";
        }

        glRecti(x0, yy0, x2, y2);

        glFlush();
}

void parseInput(unsigned char inp, int x, int y){
    wantTranslate = (inp == TRANSLATE);
    wantRotate = (inp == ROTATE);
    wantScale = (inp == SCALE);
    wantSheer = (inp == SHEER);

    drawObject();
}


void myInit (void){
    // Reset background color with black (since all three argument is 0.0)
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(0.0, 0.0, 0.0, 1.0);

    // Set width of point to one unit
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // Set window size in X- and Y- direction
    gluOrtho2D(-50, 50, -50, 50);
}

void reshape(int w,int h){
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(0.0, 0.0, 0.0, 1.0);

    // Set width of point to one unit
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // Set window size in X- and Y- direction
    gluOrtho2D(-50, 50, -50, 50);
}


int main(int argc, char** argv){
    initPoints();
    // GLute init and create window
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(WIDTH,HEIGHT);
    glutInitWindowPosition(600,100);
    glutCreateWindow("Interactive Display");
    myInit();

    // Register display callback
    glutDisplayFunc(drawObject);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(parseInput);

    glutMainLoop();
}
```



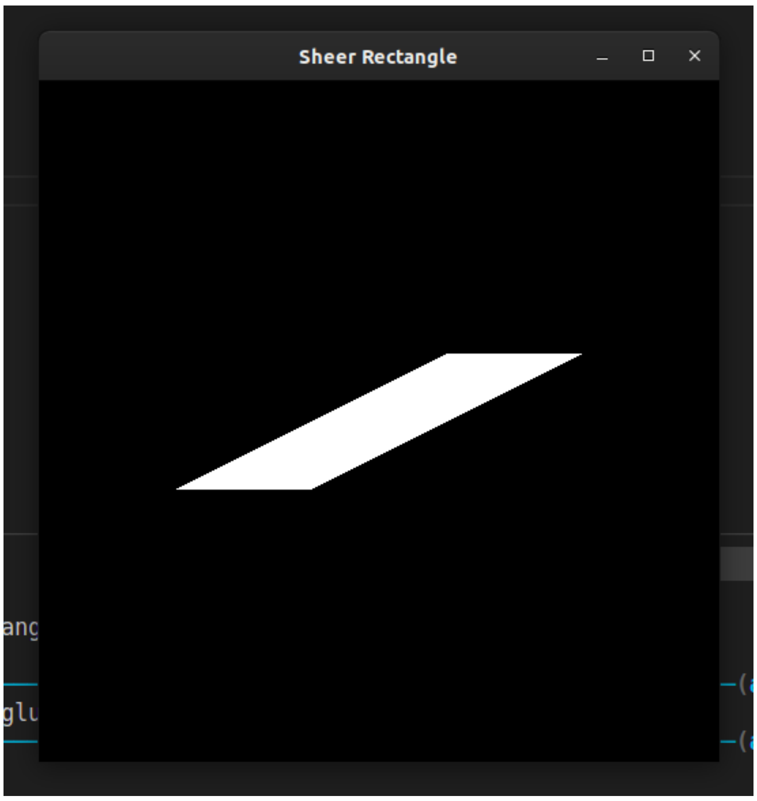


A rectangle was first rotated(1) then scaled + translated(2)

Q3. Apply Sheer on the polygon. - Code

```
#include <iostream>
#include <GL/glut.h>
#include <vector>
#include <unistd.h>
// #include <utility>
using namespace std;

const double WIDTH = 500;
const double HEIGHT = 500;

int x0,yy0,x2,y2;

char inp;


void initPoints(){
   cout<<"Enter the coordinates of the vertices in order.\n";
   cout<<"For the sake of viewing, keep the points in the range -40,40\n";

   cin>>x0>>yy0>>x2>>y2;
}

void drawObject(){
   glClear(GL_COLOR_BUFFER_BIT);

   // not available by default so using matrix multiplication

   GLfloat m[16] = {
      1.0f, 0.0f, 0.0f, 0.0f,
      2.0f, 1.0f, 0.0f, 0.0f,
      0.0f, 0.0f, 1.0f, 0.0f,
      0.0f, 0.0f, 0.0f, 1.0f
   };
   glMultMatrixf(m);

   glRecti(x0, yy0, x2, y2);

   glFlush();
}


void myInit (void){
   // Reset background color with black (since all three argument is 0.0)
   glClear(GL_COLOR_BUFFER_BIT);
   glClearColor(0.0, 0.0, 0.0, 1.0);

   // Set width of point to one unit
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
```

```
    // Set window size in X- and Y- direction
    gluOrtho2D(-50, 50, -50, 50);
}


int main(int argc, char** argv){
    initPoints();
    // GLute init and create window
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(WIDTH,HEIGHT);
    glutInitWindowPosition(600,100);
    glutCreateWindow("Sheer Rectangle");
    myInit();

    // Register display callback
    glutDisplayFunc(drawObject);

    glutMainLoop();
}
```


Sheer Rectangle