#include <windows.h>  
#include <GL/glut.h>  
#include <cmath>  
#include <iostream>  
using namespace std;  
const double PI=3.14159265358979323846;  
float theta = 0.01 \* (PI / 180);  
int sides;  
  
void display() {  
    glClear(GL\_COLOR\_BUFFER\_BIT);  
  
  
    double r=50;  
  
    GLfloat rt[] = {  
                    cos(theta), sin(theta), 0, 0,  
                    -sin(theta), cos(theta), 0, 0,  
                    0, 0, 1, 0,  
                    0, 0, 0, 1  
    };  
    GLfloat sc[] = {  
                    1.00001, 0, 0, 0,  
                    0, 1.00001, 0, 0,  
                    0, 0, 1.00001, 0,  
                    0, 0, 0, 1  
    };  
    GLfloat tr[] = {  
                    1, 0, 0, 0,  
                    0, 1, 0, 0,  
                    0, 0, 1, 0,  
                    0.00001, 0, 0, 1  
    };  
    glMultMatrixf(tr);  
    glMultMatrixf(sc);  
    glMultMatrixf(rt);  
  
    glColor3f(1.0, 1.0, 1.0);  
    glBegin(GL\_POLYGON);  
    for(int i=0;i<sides;i++){  
        double angle=i\*2\*PI/sides;  
        glVertex2d(r\*cos(angle),r\*sin(angle));  
    }  
    glVertex2d(r, 0);  
    glEnd();  
    glutSwapBuffers();  
}  
  
void displayCube() {  
    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); // Clear color and depth buffers  
    glMatrixMode(GL\_MODELVIEW);     // To operate on model-view matrix  
  
    // Render a color-cube consisting of 6 quads with different colors  
    glLoadIdentity();                 // Reset the model-view matrix  
    glTranslatef(1.5f, 0.0f, -7.0f);  // Move right and into the screen  
  
    glBegin(GL\_QUADS);                // Begin drawing the color cube with 6 quads  
      // Top face (y = 1.0f)  
      // Define vertices in counter-clockwise (CCW) order with normal pointing out  
      glColor3f(0.0f, 1.0f, 0.0f);     // Green  
      glVertex3f( 1.0f, 1.0f, -1.0f);  
      glVertex3f(-1.0f, 1.0f, -1.0f);  
      glVertex3f(-1.0f, 1.0f,  1.0f);  
      glVertex3f( 1.0f, 1.0f,  1.0f);  
  
      // Bottom face (y = -1.0f)  
      glColor3f(1.0f, 0.5f, 0.0f);     // Orange  
      glVertex3f( 1.0f, -1.0f,  1.0f);  
      glVertex3f(-1.0f, -1.0f,  1.0f);  
      glVertex3f(-1.0f, -1.0f, -1.0f);  
      glVertex3f( 1.0f, -1.0f, -1.0f);  
  
      // Front face  (z = 1.0f)  
      glColor3f(1.0f, 0.0f, 0.0f);     // Red  
      glVertex3f( 1.0f,  1.0f, 1.0f);  
      glVertex3f(-1.0f,  1.0f, 1.0f);  
      glVertex3f(-1.0f, -1.0f, 1.0f);  
      glVertex3f( 1.0f, -1.0f, 1.0f);  
  
      // Back face (z = -1.0f)  
      glColor3f(1.0f, 1.0f, 0.0f);     // Yellow  
      glVertex3f( 1.0f, -1.0f, -1.0f);  
      glVertex3f(-1.0f, -1.0f, -1.0f);  
      glVertex3f(-1.0f,  1.0f, -1.0f);  
      glVertex3f( 1.0f,  1.0f, -1.0f);  
  
      // Left face (x = -1.0f)  
      glColor3f(0.0f, 0.0f, 1.0f);     // Blue  
      glVertex3f(-1.0f,  1.0f,  1.0f);  
      glVertex3f(-1.0f,  1.0f, -1.0f);  
      glVertex3f(-1.0f, -1.0f, -1.0f);  
      glVertex3f(-1.0f, -1.0f,  1.0f);  
  
      // Right face (x = 1.0f)  
      glColor3f(1.0f, 0.0f, 1.0f);     // Magenta  
      glVertex3f(1.0f,  1.0f, -1.0f);  
      glVertex3f(1.0f,  1.0f,  1.0f);  
      glVertex3f(1.0f, -1.0f,  1.0f);  
      glVertex3f(1.0f, -1.0f, -1.0f);  
    glEnd();  // End of drawing color-cube  
    glutSwapBuffers();  
}  
  
void reshape(int width, int height) {  
    if (height == 0) height = 1;  
    GLfloat aspect = (GLfloat)width / (GLfloat)height;  
  
    glViewport(0, 0, width, height);  
  
     //glMatrixMode(GL\_PROJECTION);  
 //glLoadIdentity();  
     //gluPerspective(45.0f, aspect, 0.1f, 100.0f);  
}  
void intiopenGL() {  
    glClearColor(0.0, 0.0, 0.0, 0.0);  
    glMatrixMode(GL\_MODELVIEW);  
    gluOrtho2D(-250.0, 250.0, -250, 250.0);  
}  
  
void init() {  
   glClearColor(0.0f, 0.0f, 0.0f, 1.0f);  
   glClearDepth(1.0f);  
   glEnable(GL\_DEPTH\_TEST);  
   glDepthFunc(GL\_LEQUAL);  
   glShadeModel(GL\_SMOOTH);  
   glHint(GL\_PERSPECTIVE\_CORRECTION\_HINT, GL\_NICEST);  
}  
  
int main(int argc, char\*\* argv) {  
    cout<<"Enter Number of edges: ";  
    cin>>sides;  
    glutInit(&argc, argv);  
    glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);  
    glutInitWindowSize(640, 480);  
    glutInitWindowPosition(50, 50);  
    glutCreateWindow("Transformations");  
   //init();  
    intiopenGL();  
    glutDisplayFunc(display);  
    glutIdleFunc(display);  
    glutReshapeFunc(reshape);  
    glutMainLoop();  
    return 0;  
}