#include <stdio.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 transX, transY;

double scaleX, scaleY;

double angle, angleRad;

char reflectionAxis, shearingAxis;

int shearingX, shearingY;

double round(double d){

return floor(d + 0.5);}

void drawPolygon(){

glBegin(GL\_POLYGON);

glColor3f(1.0, 0.0, 0.0);

for (int i = 0; i < edges; i++){

glVertex2i(pntX[i], pntY[i]);}

glEnd();}

void drawPolygonMirrorReflection(char reflectionAxis){

glBegin(GL\_POLYGON);

glColor3f(0.0, 0.0, 1.0);

if (reflectionAxis == 'x' || reflectionAxis == 'X'){

for (int i = 0; i < edges; i++){

glVertex2i(round(pntX[i]), round(pntY[i] \* -1));}}

else if (reflectionAxis == 'y' || reflectionAxis == 'Y'){

for (int i = 0; i < edges; i++){

glVertex2i(round(pntX[i] \* -1), round(pntY[i]));}}

glEnd();}

void myInit(void){

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0f, 0.0f, 0.0f);

glPointSize(4.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-640.0, 640.0, -480.0, 480.0);}

void myDisplay(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0, 0.0, 0.0);

if (choice == 4{

drawPolygon();

drawPolygonMirrorReflection(reflectionAxis);}

glFlush();}

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

cin >> choice;

if (choice == 6) {

return 0;}

cout << "\n\nFor Polygon:\n" << endl;

cout << "Enter no of edges: "; cin >> edges;

for (int i = 0; i < edges; i++){

cout << "Enter co-ordinates for vertex " << i + 1 << " : "; 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("2d- Reflection");

glutDisplayFunc(myDisplay);

myInit();

glutMainLoop();}

