Name : : SAMRUDDHI NEVAGI

Roll no : 23540

**CG ASSIGNMENT – 6**

**CODE:**

#include <GL/glut.h>

#include <iostream>

#include <math.h>

#include <vector>

#include <algorithm>

using namespace std;

int ch = 0;

vector<int> arr;

int ct = 0;

float colorarr[] = {1.0,0.0,0.0};

float flc[] = {};

float neg[] = {0.0,1.0,1.0};

void copyarr(float\* arr1){

for(int i=0; i<3; i++){

flc[i] = arr1[i];

if(arr1[i] == 0.0f){

neg[i] = 1.0f;

}

if(arr1[i] == 1.0f){

neg[i] = 0.0f;

}

}

}

void drawTriangle(int x1, int y1, int x2, int y2, int x3, int y3, float\* flc){

glColor3f(flc[0],flc[1],flc[2]);

glLineWidth(3);

glBegin(GL\_LINE\_LOOP);

glVertex2i(x1,y1);

glVertex2i(x2,y2);

glVertex2i(x3,y3);

glEnd();

glLineWidth(1);

glFlush();

}

void drawQuadritaleral(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4, float\* flc){

glColor3f(flc[0],flc[1],flc[2]);

glLineWidth(3);

glBegin(GL\_LINE\_LOOP);

glVertex2i(x1,y1);

glVertex2i(x2,y2);

glVertex2i(x3,y3);

glVertex2i(x4,y4);

glEnd();

glLineWidth(1);

glFlush();

}

void TrnsScaling(vector<int> arr){

vector<int> sarr;

int sz = arr.size();

float Sx, Sy;

cout << "Enter Sx scaling factor: ";

cin >> Sx;

cout << "Enter Sy scaling factor: ";

cin >> Sy;

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

if(i % 2 == 0){

sarr.push\_back(arr.at(i)\*Sx);

}

if(i % 2 == 1){

sarr.push\_back(arr.at(i)\*Sy);

}

}

if (sz == 6){

drawTriangle(sarr.at(0),sarr.at(1),sarr.at(2),sarr.at(3),sarr.at(4),sarr.at(5),neg);

}

else if(sz == 7){

drawQuadritaleral(sarr.at(0),sarr.at(1),sarr.at(2),sarr.at(3),sarr.at(4),sarr.at(5),sarr.at(6),sarr.at(7),neg);

}

sarr.clear();

}

void Rotationabtpt(vector<int> arr, int x, int y){

int xr = x;

int yr = y;

int angle;

cout << "Enter angle in degree's to rotate: ";

cin >> angle;

float theta = angle \* M\_PI / 180;

vector<int> sarr;

int sz = arr.size();

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

if(i % 2 == 0){

int nx = xr + ((arr.at(i) - xr)\*cos(theta)) - ((arr.at(i+1) - yr)\*sin(theta));

sarr.push\_back(nx);

}

if(i % 2 == 1){

int ny = yr + ((arr.at(i-1) - xr)\*sin(theta)) + ((arr.at(i) - yr)\*cos(theta));

sarr.push\_back(ny);

}

}

if (sz == 6){

drawTriangle(sarr.at(0),sarr.at(1),sarr.at(2),sarr.at(3),sarr.at(4),sarr.at(5),neg);

}

else if(sz == 8){

drawQuadritaleral(sarr.at(0),sarr.at(1),sarr.at(2),sarr.at(3),sarr.at(4),sarr.at(5),sarr.at(6),sarr.at(7),neg);

}

sarr.clear();

}

void mouse(int button, int state, int x, int y)

{

static int xx, yy;

xx = x - 300;

yy = 300 - y;

int sz = arr.size();

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

if (ch == 1)

{

if (sz < 6){

arr.push\_back(xx);

arr.push\_back(yy);

}

sz = arr.size();

if (sz == 6){

drawTriangle(arr.at(0),arr.at(1),arr.at(2),arr.at(3),arr.at(4),arr.at(5),flc);

}

}

if (ch == 2)

{

if (sz < 4){

arr.push\_back(xx);

arr.push\_back(yy);

}

sz = arr.size();

}

if (ch == 3)

{

if (sz < 8){

arr.push\_back(xx);

arr.push\_back(yy);

}

sz = arr.size();

if (sz == 8){

drawQuadritaleral(arr.at(0),arr.at(1),arr.at(2),arr.at(3),arr.at(4),arr.at(5),arr.at(6),arr.at(7),flc);

}

}

if (ch == 4)

{

Rotationabtpt(arr, xx, yy);

}

}

else if (button == GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN)

{

if(ct % 3 == 0){

colorarr[0] = 1.0;

colorarr[1] = 0.0;

colorarr[2] = 0.0;

cout << "Red color is choosen" << endl;

}

else if(ct % 3 == 1){

colorarr[0] = 0.0;

colorarr[1] = 1.0;

colorarr[2] = 0.0;

cout << "Green color is choosen" << endl;

}

else if(ct % 3 == 2){

colorarr[0] = 0.0;

colorarr[1] = 0.0;

colorarr[2] = 1.0;

cout << "Blue color is choosen" << endl;

}

ct++;

}

glFlush();

}

void keyboard(unsigned char key, int x, int y)

{

switch (key)

{

case 't':

{

ch = 1;

copyarr(colorarr);

cout << "Triangle is opted" << endl;

glutMouseFunc(mouse);

break;

}

case 'c':

{

ch = 2;

copyarr(colorarr);

cout << "Circle is opted" << endl;

glutMouseFunc(mouse);

break;

}

case 'q':

{

ch = 3;

copyarr(colorarr);

cout << "Quadrilateral is opted" << endl;

glutMouseFunc(mouse);

break;

}

case 's':

{

copyarr(colorarr);

cout << "Scaling Transformation is opted" << endl;

TrnsScaling(arr);

break;

}

case 'R':

{

ch = 4;

copyarr(colorarr);

cout << "Rotation about any arbitary point is opted" << endl;

glutMouseFunc(mouse);

cout << "Click on the arbitary point" << endl;

break;

}

case 'x':

{

arr.clear();

cout << "Queue is cleared" << endl;

break;

}

case 'X':

{

arr.clear();

glClearColor(1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

cout << "Screen is cleared" << endl;

break;

}

}

glutPostRedisplay();

}

void initialize()

{

glClearColor(1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

gluOrtho2D(-300, 300, -300, 300);

}

void initialaxis(){

glColor3f(0,0,0);

glLineWidth(2);

glBegin(GL\_LINES);

glVertex2i(-300,0);

glVertex2i(300,0);

glVertex2i(0,-300);

glVertex2i(0,300);

glEnd();

glFlush();

glutKeyboardFunc(keyboard);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(600, 600);

glutInitWindowPosition(800, 100);

glutCreateWindow("Filling Algorithm");

initialize();

cout << "Choose your Line type: " << endl;

cout << "--------------------------------------------" << endl;

cout << "t => Triangle" << endl;

cout << "c => Circle" << endl;

cout << "q => Quadrilateral" << endl;

cout << "--------------------------------------------" << endl;

cout << "s => Scaling" << endl;

cout << "r => Rotation about center" << endl;

cout << "R => Rotation about point" << endl;

cout << "--------------------------------------------" << endl;

cout << "x => clear the queue" << endl;

cout << "X => clear the screen" << endl;

cout << "Right Click to change the color" << endl;

cout << "--------------------------------------------" << endl;

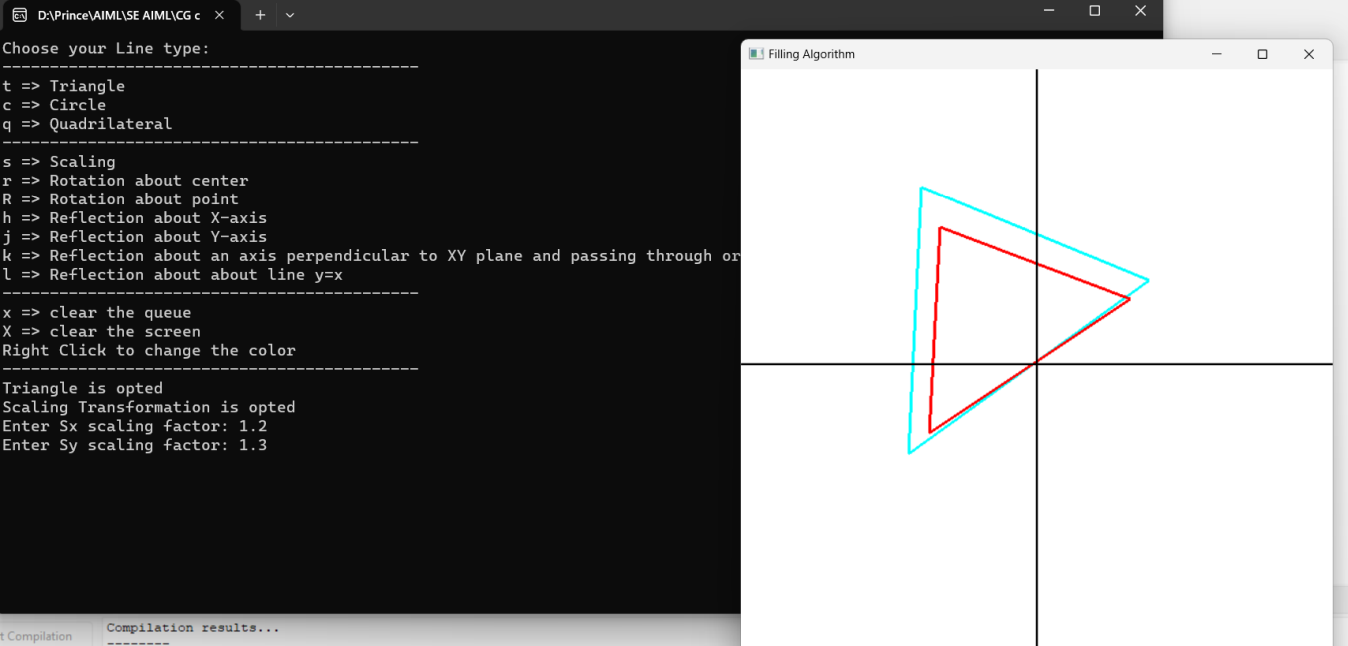
glutDisplayFunc(initialaxis);

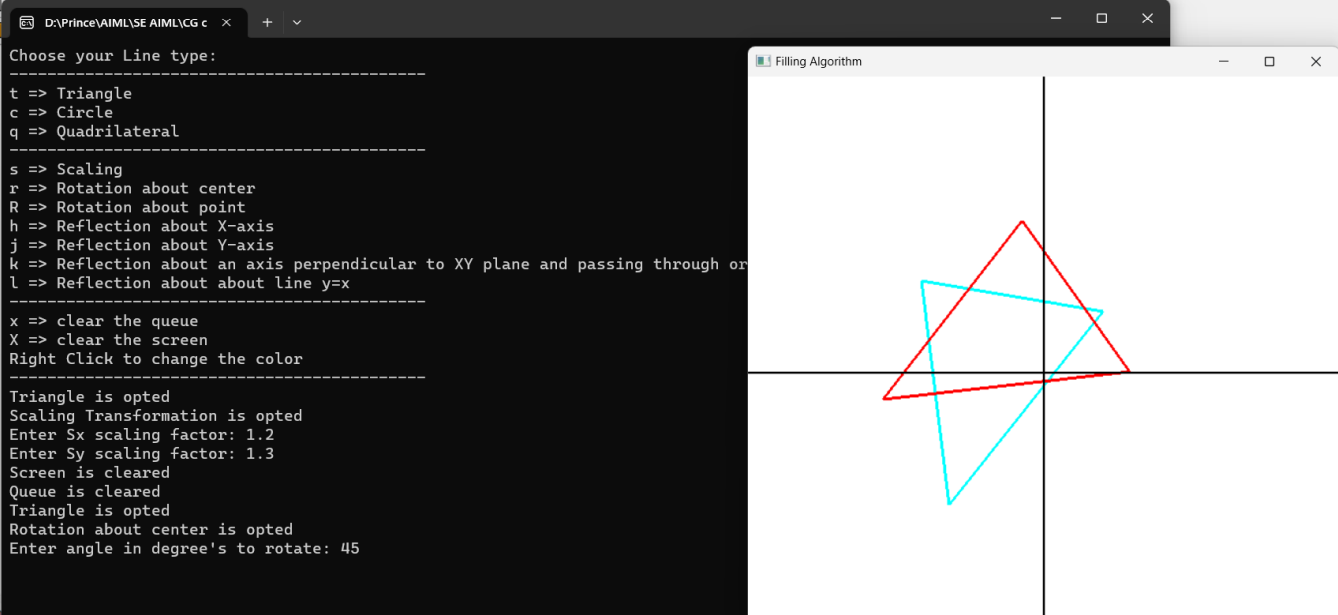
glutMainLoop();

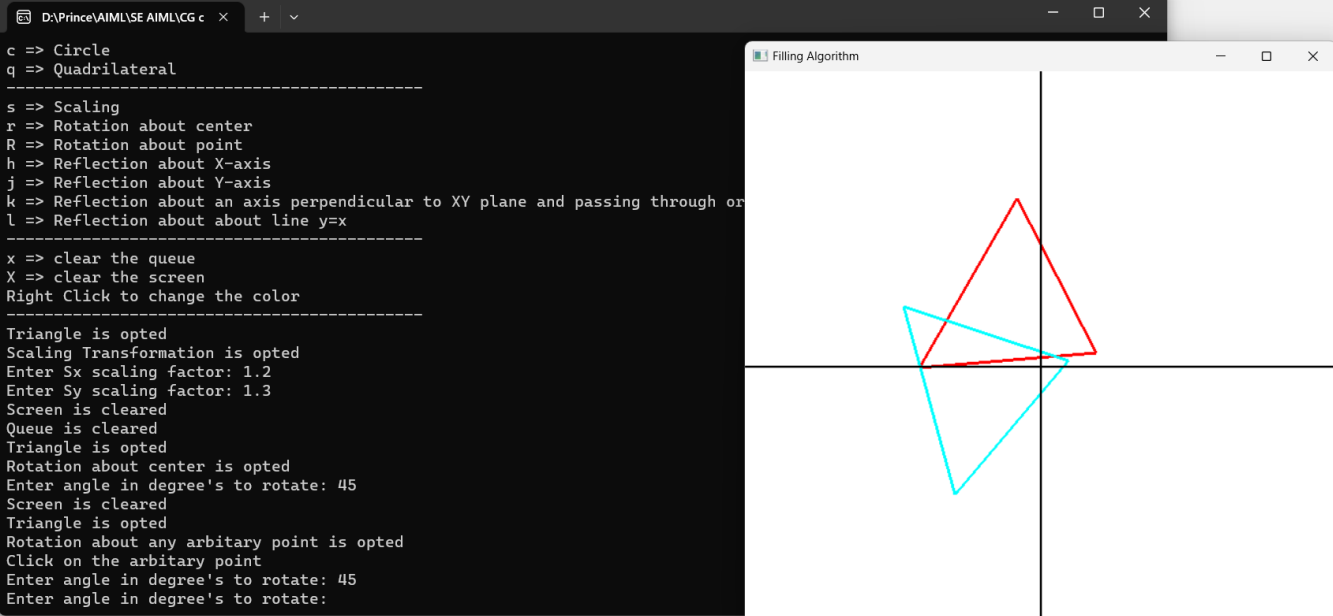
return 0;

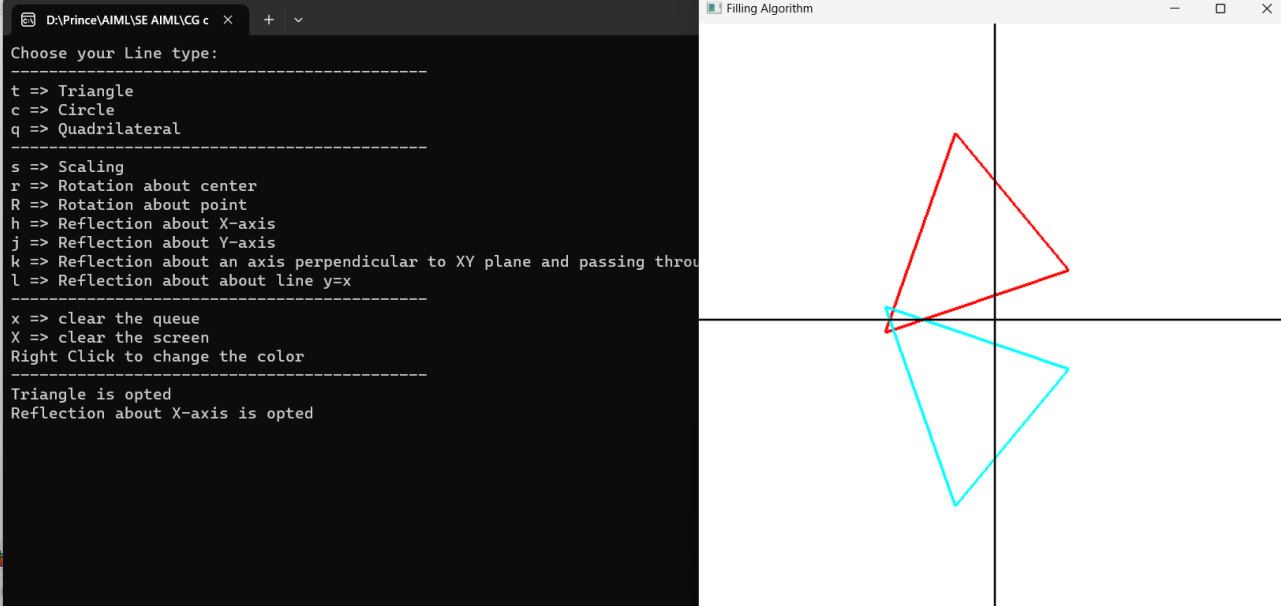
}

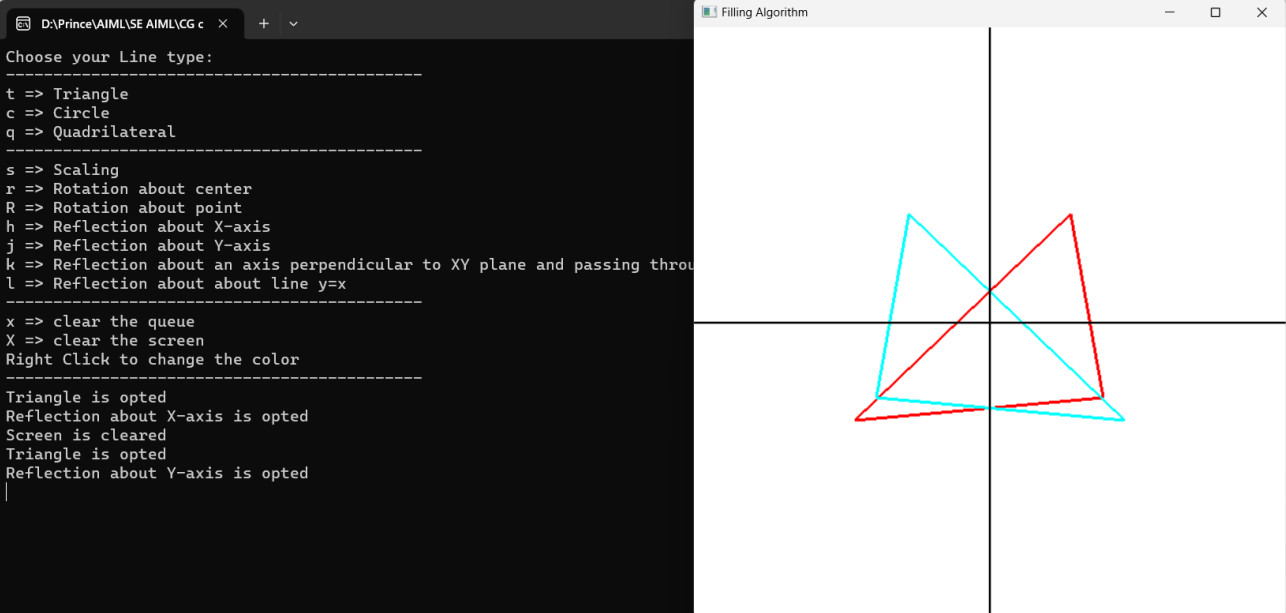
**OUTPUT:**

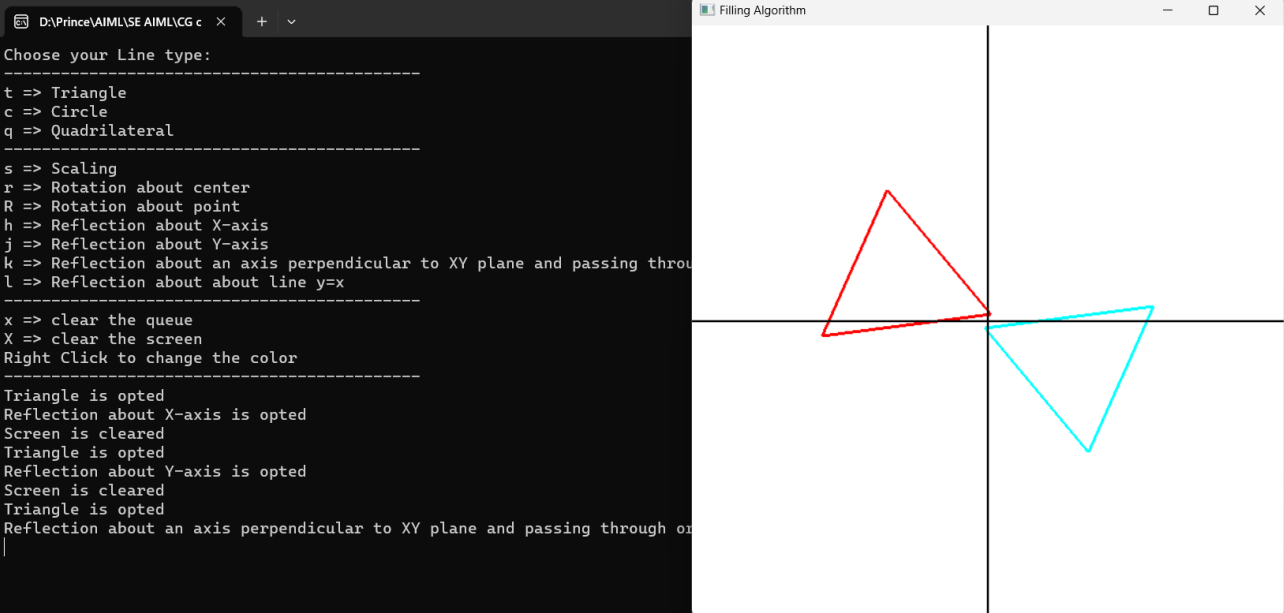
****

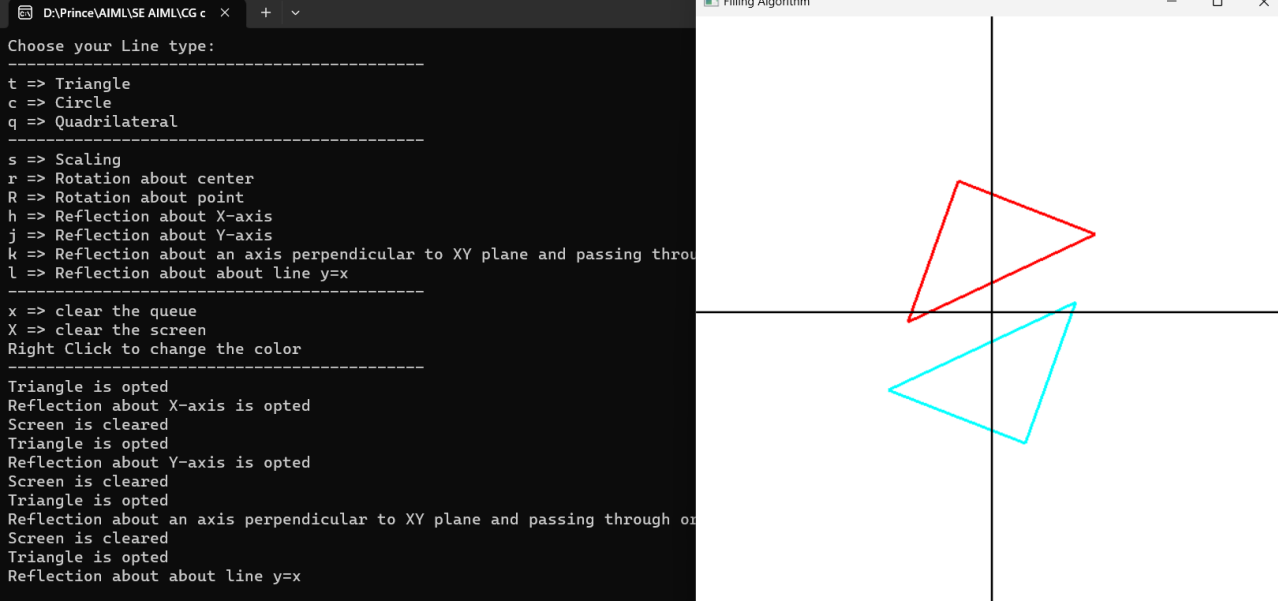
****

****

****

****

****

****