Name : : SAMRUDDHI NEVAGI

Roll no : 23540

**CG ASSIGNMENT – 4.2**

**CODE:**

#include <iostream>

#include <math.h>

#include <GL/glut.h>

#include <list>

using namespace std;

void init(){

glClearColor(1.0,1.0,1.0,1.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,640,0,480);

}

int enter = 1, sz, gymin=480, gymax=0, parity=0;

float\*\* pts;

float R = 1.0, G = 0.0, B = 0.0;

class points{

int x;

int y;

public:

points(int x,int y){

this->x = x;

this->y = y;

}

int getx(){

return x;

}

int gety(){

return y;

}

};

list <points\*> in;

void flip(){

if(parity==0)

parity = 1;

else

parity = 0;

}

class lines{

int ymin;

int ymax;

float m;

int xmin;

int c;

public:

lines(int x1, int x2, float x3, int x4, int x5){

ymin = x1;

ymax = x2;

m = x3;

xmin = x4;

c = x5;

}

int getymin(){return ymin;}

int getymax(){return ymax;}

float getm(){return m;}

int getxmin(){return xmin;}

int getc(){return c;}

};

lines\*\* line;

void fillStrip(int y, int xstart, int xend){

glColor3f(R,G,B);

glBegin(GL\_POINTS);

glVertex2i(xstart,y);

glVertex2i(xend,y);

glEnd();

glFlush();

//glColor3f(R,G,B);

if(parity==1){

glBegin(GL\_POINTS);

for(int i=xstart+1;i<xend;i++)

glVertex2i(i,y);

glEnd();

glFlush();

}

}

void scanFill(){

int sline = gymin;

int plausibleLines = 0;

while(sline<=gymax){

lines\*\* temp = new lines\*[sz];

// find plausible lines

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

if(sline<=line[i]->getymax() && sline>=line[i]->getymin())

{

temp[plausibleLines] = line[i];

plausibleLines++;

}

}

// kill if only one line

if(plausibleLines==1)

{

cout<<"ERR::Require a Closed Figure";

exit(0);

}

// find all intersection points

list <int> intersect;

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

int xintersect;

if(sline==temp[i]->getymax())

continue;

xintersect = (sline-temp[i]->getc())/temp[i]->getm();

intersect.push\_front(xintersect);

}

// sort

intersect.sort();

// plot the points

while(intersect.size()>0){

int start = intersect.front();

flip();

intersect.pop\_front();

int end = intersect.front();

fillStrip(sline,start,end);

}

plausibleLines = 0;

sline++;

}

}

void computeGlobalVars(){

//compute gloabl vars

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

if(gymin>line[i]->getymin())

gymin = line[i]->getymin();

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

if(gymax<line[i]->getymax())

gymax = line[i]->getymax();

}

void formLines(){

//create line from the points and omit horizontal lines

line = new lines\*[sz];

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

{

int q1[2],q2[2];

float mm;

if(i!=sz-1){

q1[0] = pts[i][0]; q1[1]=pts[i][1];

q2[0] = pts[i+1][0]; q2[1]=pts[i+1][1];

mm = (pts[i+1][1]-pts[i][1])/(pts[i+1][0]-pts[i][0]);

}

else{

q1[0] = pts[i][0]; q1[1]=pts[i][1];

q2[0] = pts[0][0]; q2[1]=pts[0][1];

mm = (pts[0][1]-pts[i][1])/(pts[0][0]-pts[i][0]);

}

if(q1[0]==q2[0] && q2[1]==q1[1])

{

cout<<"ERR::SAME POINTS";

exit(0);

}

int ymn,ymx,xmn,cc;

if(q2[1]<q1[1]){

ymn = q2[1];

ymx = q1[1];

}

else{

ymn = q1[1];

ymx = q2[1];

}

if(q1[0]<q2[0])

xmn = q1[0];

else

xmn = q2[0];

cc = q1[1] - (mm\*q1[0]);

if(mm!=0)

line[i] = new lines(ymn,ymx,mm,xmn,cc);

//cout<<"LINE "<<i<<" ymin "<<line[i]->getymin()<<" ymax "<<line[i]->getymax()<<" slope "<<line[i]->getm()<<" xmin "<<line[i]->getxmin()<<" c "<<line[i]->getc()<<endl;

}

computeGlobalVars();

}

void drawPolygon(){

//draw polygon and create the points array

glBegin(GL\_LINE\_LOOP);

pts = new float\*[in.size()];

for(int i=0; i<in.size(); i++){

pts[i] = new float[2];

}

sz = in.size();

while(in.size()>0){

points\* temp = in.front();

pts[in.size()-1][0] = temp->getx();

pts[in.size()-1][1] = temp->gety();

glVertex2i(temp->getx(),temp->gety());

in.pop\_front();

}

glEnd();

glFlush();

formLines();

}

void rgb(float r, float g, float b){

R = (r \* 3.92)/1000;

G = (g \* 3.92)/1000;

B = (b \* 3.92)/1000;

glColor3f(R,G,B);

glBegin(GL\_POLYGON);

glVertex2i(0,0);

glVertex2i(20,0);

glVertex2i(20,20);

glVertex2i(0,20);

glEnd();

glFlush();

}

void key(unsigned char key\_t, int x, int y){

if(key\_t=='d'){

enter = 0;

drawPolygon();

}

if(key\_t<=57 && key\_t>=48){

int val = key\_t - 48;

switch(val){

case 0: rgb(26.0, 188.0, 156.0);break;

case 1: rgb(46.0, 204.0, 113.0);break;

case 2: rgb(52.0, 152.0, 219.0);break;

case 3: rgb(155.0, 89.0, 182.0);break;

case 4: rgb(52.0, 73.0, 94.0);break;

case 5: rgb(255.0, 255.0, 255.0);break;

case 6: rgb(255.0, 0.0, 0.0);break;

case 7: rgb(231.0, 76.0, 60.0);break;

case 8: rgb(236.0, 240.0, 241.0);break;

case 9: rgb(149.0, 165.0, 166.0);break;

}

}

if(key\_t=='s'){

enter = 1;

scanFill();

in.empty();

}

}

void drag\_start(GLint x, GLint y){

y = 480-y;

glPointSize(4);

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

glFlush();

glPointSize(2);

}

void mouse(int btn, int state, int x, int y){

y = 480-y;

if(btn==GLUT\_LEFT\_BUTTON)

{

if(state==GLUT\_DOWN)

{

if(enter){

points\* temp = new points(x,y);

in.push\_front(temp);

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

glFlush();

}

}

}

}

void world(){

glPointSize(2);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1,0,0);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(640,480);

glutInitWindowPosition(200,200);

glutCreateWindow("FloodFill Algorithm");

glutDisplayFunc(world);

glutMouseFunc(mouse);

glutMotionFunc(drag\_start);

glutKeyboardFunc(key);

init();

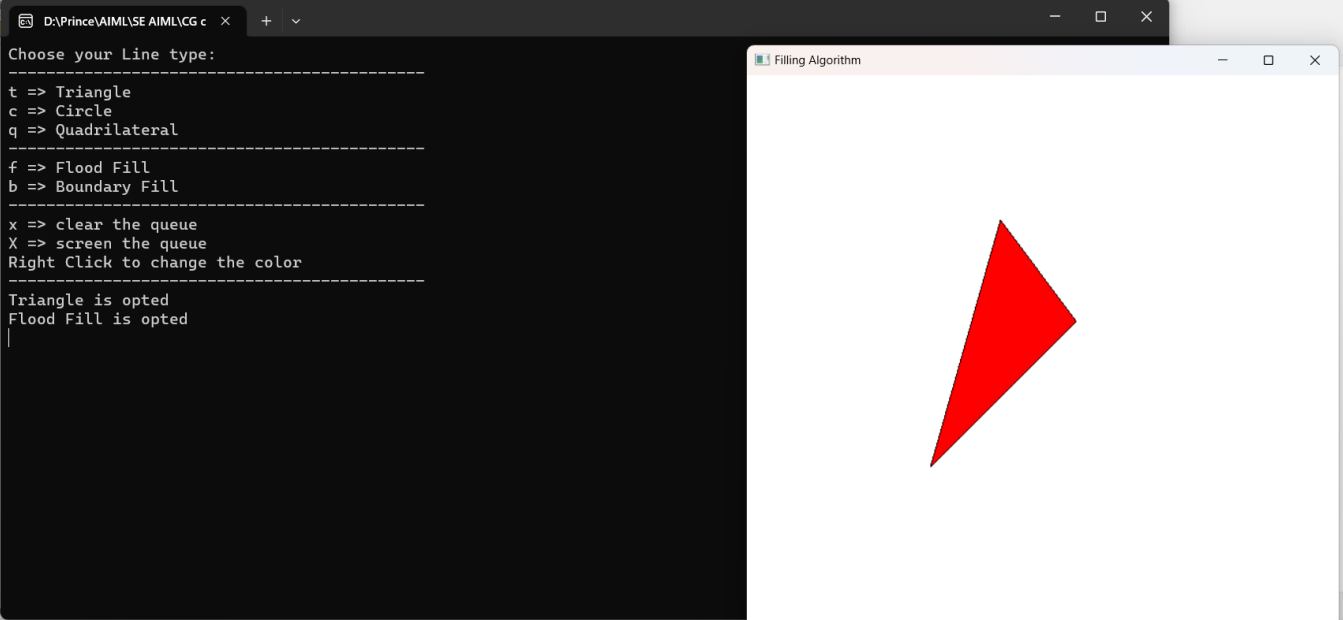
glutMainLoop();

return 0;

}

//First s then d

**OUTPUT:**

****