**Name: Manthan Khawse**

**Class: CS SY Div: C**

**Roll no.: 3**

**PRN: 12310137**

**CGAVR Assignment 4**

1. **Boundary Fill**

**Code:**

#include <GL/glut.h>

#include <set>

#include <stack>

#include <utility>

#include <vector>

#include <iostream>

using namespace std;

set<pair<int, int>> boundaryPoints;

set<pair<int, int>> filledPoints;

vector<pair<int, int>> vertices;

void drawLine(int x1, int y1, int x2, int y2) {

int dx = abs(x2 - x1), dy = abs(y2 - y1);

int sx = (x1 < x2) ? 1 : -1;

int sy = (y1 < y2) ? 1 : -1;

int err = dx - dy;

while (true) {

glBegin(GL\_POINTS);

glVertex2i(x1, y1);

glEnd();

boundaryPoints.insert({x1, y1});

if (x1 == x2 && y1 == y2) break;

int e2 = err \* 2;

if (e2 > -dy) {

err -= dy;

x1 += sx;

}

if (e2 < dx) {

err += dx;

y1 += sy;

}

}

}

bool isPointInsidePolygon(int x, int y) {

int count = 0;

int n = vertices.size();

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

int x1 = vertices[i].first, y1 = vertices[i].second;

int x2 = vertices[(i + 1) % n].first, y2 = vertices[(i + 1) % n].second;

if ((y1 > y) != (y2 > y)) {

float slope = (float)(x2 - x1) / (y2 - y1);

float intersectX = x1 + slope \* (y - y1);

if (x < intersectX) {

count++;

}

}

}

return count % 2 == 1;

}

void fillPolygon(int startX, int startY) {

stack<pair<int, int>> pointStack;

pointStack.push({startX, startY});

while (!pointStack.empty()) {

pair<int, int> currentPoint = pointStack.top();

pointStack.pop();

int x = currentPoint.first;

int y = currentPoint.second;

if (boundaryPoints.find(currentPoint) != boundaryPoints.end() ||

filledPoints.find(currentPoint) != filledPoints.end() ||

!isPointInsidePolygon(x, y)) {

continue;

}

filledPoints.insert(currentPoint);

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

pointStack.push({x + 1, y});

pointStack.push({x - 1, y});

pointStack.push({x, y + 1});

pointStack.push({x, y - 1});

pointStack.push({x + 1, y + 1});

pointStack.push({x + 1, y - 1});

pointStack.push({x - 1, y + 1});

pointStack.push({x - 1, y - 1});

}

glFlush();

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

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

drawLine(vertices[i].first, vertices[i].second,

vertices[(i + 1) % vertices.size()].first,

vertices[(i + 1) % vertices.size()].second);

}

fillPolygon(0, 0);

glFlush();

}

void init() {

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glColor3f(1.0f, 0.0f, 0.0f);

glPointSize(3.0f);

gluOrtho2D(-100, 100, -100, 100);

}

void takeInput() {

int n;

cout << "Enter the number of vertices: ";

cin >> n;

cout << "Enter the vertices (x y) one by one:\n";

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

int x, y;

cin >> x >> y;

vertices.push\_back({x, y});

}

}

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

takeInput(); // Get vertex input from user

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Manual 8-Neighbor Polygon Fill with Point Inside Check");

init();

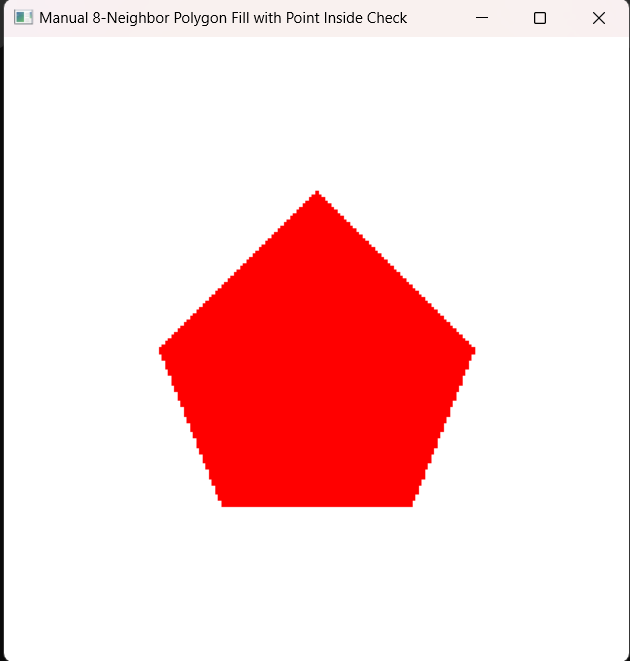
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**Output:**

****

1. **Flood Fill**

**Code:**

#include <GL/glut.h>

#include <stdio.h>

#include <stdlib.h>

// Window size

const int WINDOW\_WIDTH = 800;

const int WINDOW\_HEIGHT = 600;

typedef struct

{

float r, g, b;

} Color;

int numVertices;

int vertices[100][2];

Color getPixelColor(int x, int y)

{

Color color;

glReadPixels(x, y, 1, 1, GL\_RGB, GL\_FLOAT, &color);

return color;

}

void setPixelColor(int x, int y, Color color)

{

glBegin(GL\_POINTS);

glColor3f(color.r, color.g, color.b);

glVertex2i(x, y);

glEnd();

glFlush();

}

void floodFill(int x, int y, Color oldColor, Color newColor)

{

Color currentColor = getPixelColor(x, y);

if(currentColor.r == oldColor.r && currentColor.g == oldColor.g && currentColor.b == oldColor.b)

{

setPixelColor(x, y, newColor);

// (8-connected)

floodFill(x + 1, y, oldColor, newColor);

floodFill(x - 1, y, oldColor, newColor);

floodFill(x, y + 1, oldColor, newColor);

floodFill(x, y - 1, oldColor, newColor);

floodFill(x+1, y + 1, oldColor, newColor);

floodFill(x-1, y - 1, oldColor, newColor);

floodFill(x+1, y - 1, oldColor, newColor);

floodFill(x+1, y + 1, oldColor, newColor);

}

}

void getPolygonCoordinates()

{

printf("Enter the number of vertices: ");

scanf("%d", &numVertices);

if(numVertices < 3)

{

printf("A polygon must have at least 3 vertices!\n");

exit(1);

}

printf("Enter the coordinates of the polygon:\n");

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

{

printf("Vertex %d (x, y): ", i + 1);

scanf("%d %d", &vertices[i][0], &vertices[i][1]);

}

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

// Draw the polygon with a thicker boundary

glColor3f(1.0f, 0.0f, 0.0f);

glLineWidth(3.0f);

glBegin(GL\_LINE\_LOOP);

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

{

glVertex2i(vertices[i][0], vertices[i][1]);

}

glEnd();

glFlush();

// defining old and new colour

Color oldColor = { 1.0f, 1.0f, 1.0f }; // White

Color newColor = { 0.0f, 1.0f, 0.0f }; // Green

// Start filling from a point inside the polygon

int seedX = 0, seedY = 0;

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

seedX += vertices[i][0];

seedY += vertices[i][1];

}

seedX /= numVertices;

seedY /= numVertices;

floodFill(seedX, seedY, oldColor, newColor);

}

// Initialization function

void init()

{

glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // White background

//glColor3f(0.0f, 0.0f, 0.0f); // Black color for drawing

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, WINDOW\_WIDTH, 0.0, WINDOW\_HEIGHT);

}

// Main function

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

{

// Get the polygon coordinates from the user

getPolygonCoordinates();

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(WINDOW\_WIDTH, WINDOW\_HEIGHT);

glutCreateWindow("Flood Fill on Any Polygon");

init();

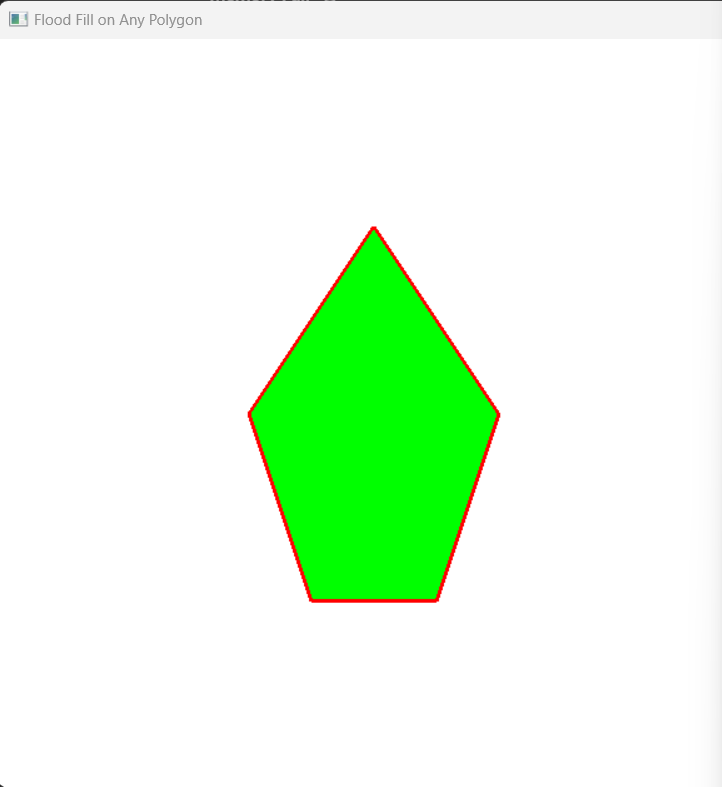
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**Output:**

****