Assignment No.2

# Name: Omkar Manohar Hepat

# Class: CS-B

# Roll No: 31 PRN No: 12211509

# Subject: CGAVR

## Q1. Polygon Filling using Boundary Fill

#include <GL/glut.h>

#include <stdio.h>

int width = 600, height = 600;

void drawPixel(int x, int y, float color[3]) {

glColor3fv(color);

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

glFlush();

}

void getPixelColor(int x, int y, float color[3]) {

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

}

void boundaryFill(int x, int y, float fillColor[3], float borderColor[3]) {

float color[3];

getPixelColor(x, y, color);

if ((color[0] != borderColor[0] || color[1] != borderColor[1] || color[2] != borderColor[2]) &&

(color[0] != fillColor[0] || color[1] != fillColor[1] || color[2] != fillColor[2])) {

drawPixel(x, y, fillColor);

boundaryFill(x + 1, y, fillColor, borderColor);

boundaryFill(x - 1, y, fillColor, borderColor);

boundaryFill(x, y + 1, fillColor, borderColor);

boundaryFill(x, y - 1, fillColor, borderColor);

}

}

void drawTriangle() {

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_POLYGON);

glVertex2i(300, 100);

glVertex2i(450, 300);

glVertex2i(200, 400);

glEnd();

}

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

if (btn == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) {

float borderColor[3] = { 0.0f, 0.0f, 0.0f }; //

float fillColor[3] = { 0.0f, 1.0f, 0.0f }; //

boundaryFill(x, height - y, fillColor, borderColor);

}

}

void init() {

glClearColor(1.0, 1.0, 1.0, 1.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0, width, 0, height);

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

drawTriangle();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(width, height);

glutCreateWindow("Boundary Fill");

glutDisplayFunc(display);

glutMouseFunc(mouse);

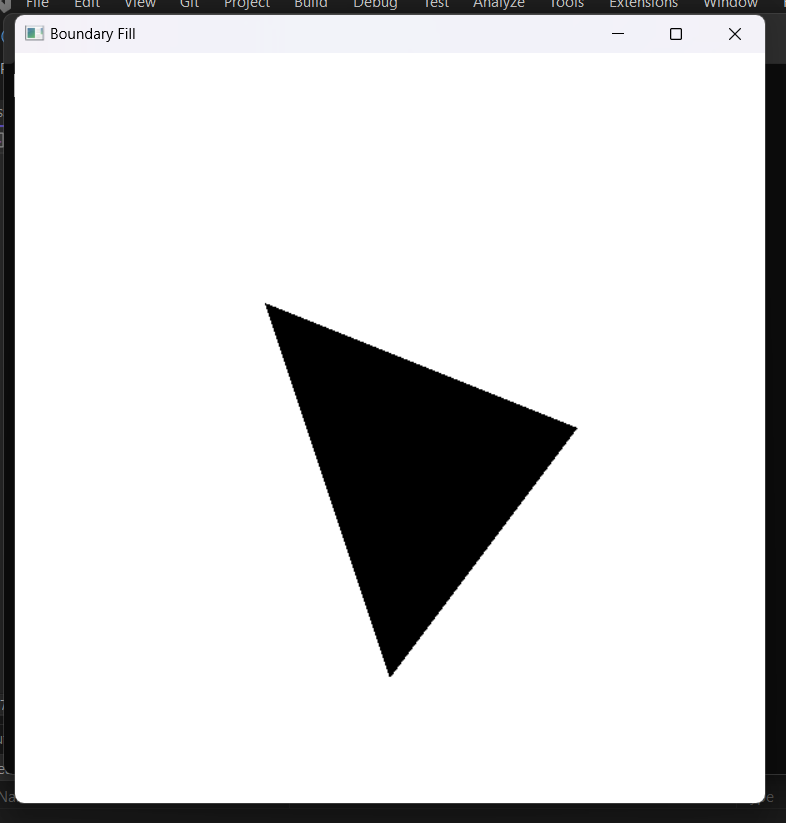
init();

glutMainLoop();

return 0;

}

Output:



## Q2.Polygon filling using flood fill.

#include <GL/glut.h>

// Function to initialize OpenGL

void init()

{

glClearColor(0.0, 0.0, 0.0, 1.0); // Set the background color to black

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, 500, 0, 500); // Set the 2D coordinate system

}

// Function to perform flood fill

void floodFill(int x, int y, float fillColor[3], float borderColor[3])

{

float interiorColor[3];

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

if (

(interiorColor[0] != borderColor[0] || interiorColor[1] != borderColor[1] || interiorColor[2] != borderColor[2]) &&

(interiorColor[0] != fillColor[0] || interiorColor[1] != fillColor[1] || interiorColor[2] != fillColor[2])

)

{

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

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

glFlush();

floodFill(x + 1, y, fillColor, borderColor);

floodFill(x - 1, y, fillColor, borderColor);

floodFill(x, y + 1, fillColor, borderColor);

floodFill(x, y - 1, fillColor, borderColor);

}

}

// Function to handle display

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

// Draw a rectangle

glColor3f(1.0, 1.0, 1.0);

glBegin(GL\_LINE\_LOOP);

glVertex2i(20, 20);

glVertex2i(80, 20);

glVertex2i(80, 80);

glVertex2i(20, 80);

glEnd();

glFlush();

// Perform flood fill

float fillColor[] = { 0.0, 1.0, 0.0 }; // Green color

float borderColor[] = { 1.0, 1.0, 1.0 }; // White color

floodFill(50, 50, fillColor, borderColor);

glFlush();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Flood Fill");

init();

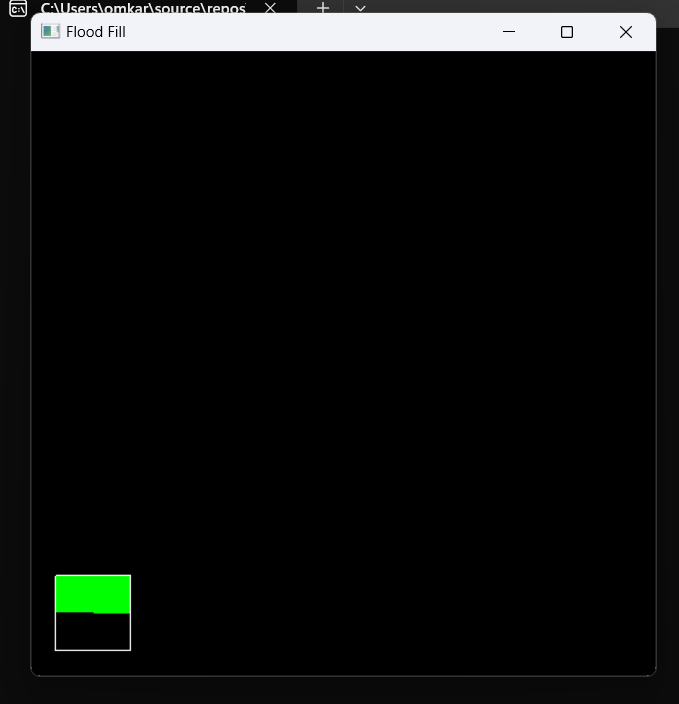
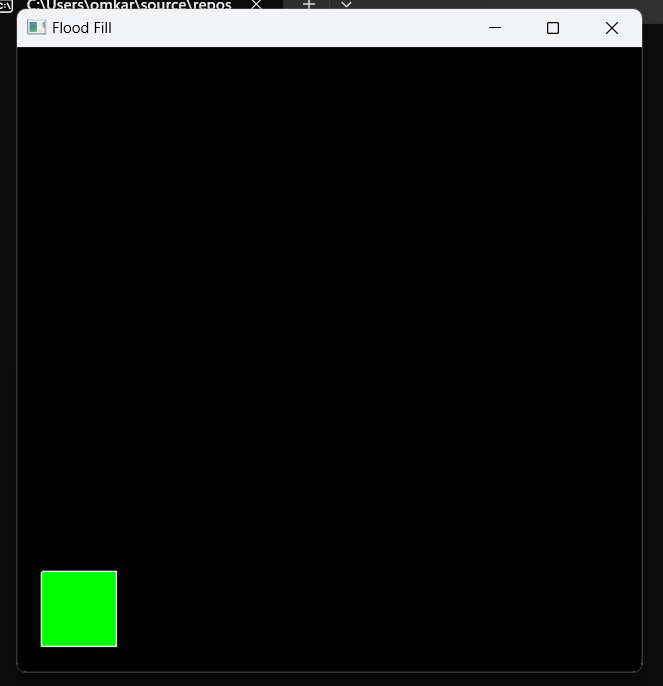
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**Output:**



**Q3. Polygon filling using Edge Fill**

#include<stdio.h>

#include<GL/glut.h>

#include<math.h>

int le[500], re[500], flag = 0, m;

int fillPolygon = 0;

void init()

{

gluOrtho2D(0, 500, 0, 500);

}

void edge(int x0, int y0, int x1, int y1)

{

if (y1 < y0)

{

int tmp;

tmp = y1;

y1 = y0;

y0 = tmp;

tmp = x1;

x1 = x0;

x0 = tmp;

}

int x = x0;

m = (y1 - y0) / (x1 - x0);

for (int i = y0; i < y1; i++)

{

if (x < le[i])

le[i] = x;

if (x > re[i])

re[i] = x;

x += (1 / m);

}

}

void drawPixel(int x, int y) {

glColor3f(1, 0, 0);

glBegin(GL\_POINTS);

glVertex2f(x, y);

glEnd();

}

void scanFill() {

for (int i = 0; i < 500; i++)

{

if (le[i] < re[i])

{

for (int j = le[i]; j < re[i]; j++)

{

drawPixel(j, i);

}

}

}

}

void display()

{

glClearColor(1, 1, 1, 1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0, 0, 1);

glBegin(GL\_LINE\_LOOP);

glVertex2f(200, 100);

glVertex2f(100, 200);

glVertex2f(200, 300);

glVertex2f(300, 200);

glEnd();

for (int i = 0; i < 500; i++)

{

le[i] = 500;

re[i] = 0;

}

edge(200, 100, 100, 200);

edge(100, 200, 200, 300);

edge(200, 300, 300, 200);

edge(300, 200, 200, 100);

if (fillPolygon == 1)

{

scanFill();

}

glFlush();

}

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

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

fillPolygon = 1;

glutPostRedisplay();

}

}

void ScanMenu(int id) {

if (id == 1)

fillPolygon = 1;

else if (id == 2)

fillPolygon = 0;

else

exit(0);

glutPostRedisplay();

}

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

{

glutInit(&argc, argv);

glutInitWindowPosition(100, 100);

glutInitWindowSize(500, 500);

glutCreateWindow("scan line");

init();

glutDisplayFunc(display);

glutMouseFunc(mouse);

glutCreateMenu(ScanMenu);

glutAddMenuEntry("Fill", 1);

glutAddMenuEntry("Clear", 2);

glutAddMenuEntry("Exit", 3);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutMainLoop();

return 0;

}

Output:

