Practical no:4

Title:- Implement the following polygon filling methods : ii)Flood fill; using mouse click, keyboard interface and menu driven programming.

Name:-Ujawala Sinha
Roll no:S554

Code:

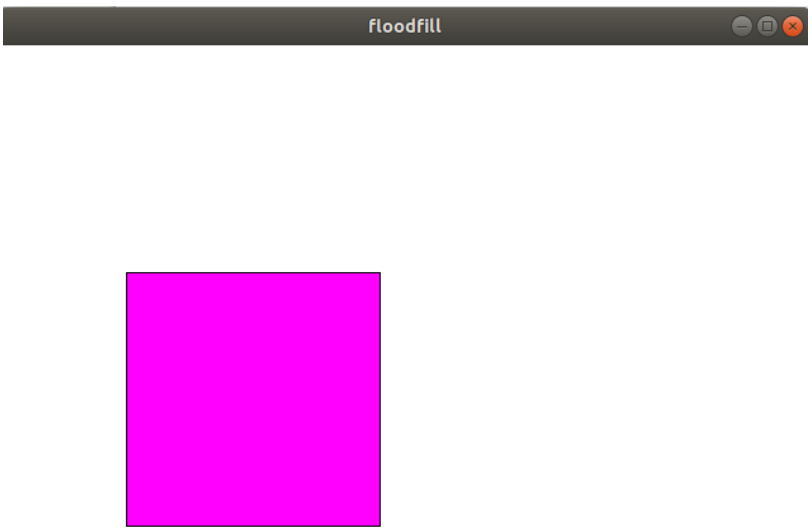
```
#include <GL/glut.h>
#include <cmath>
#include <iostream>

struct Point {
	GLint x;
	GLint y;
};

struct Color {
	GLfloat r;
	GLfloat g;
	GLfloat b;
};

// Function to draw a line using DDA algorithm
void draw_dda(Point p1, Point p2) {
	GLfloat dx = p2.x - p1.x;
	GLfloat dy = p2.y - p1.y;

	GLfloat x1 = p1.x;
	GLfloat y1 = p1.y;

	GLfloat step = 0;

	if(abs(dx) > abs(dy)) {
		step = abs(dx);
	} else {
		step = abs(dy);
	}

	GLfloat xInc = dx / step;
	GLfloat yInc = dy / step;

	for(float i = 1; i <= step; i++) {
		glVertex2i(x1, y1);
		x1 += xInc;
		y1 += yInc;
	}
}

// Initialization of OpenGL settings
```

```
void init() {
	glClearColor(1.0, 1.0, 1.0, 0.0);
	glColor3f(0.0, 0.0, 0.0);
	glPointSize(1.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0, 640, 0, 480);
}

// Function to get the color of a pixel
Color getPixelColor(GLint x, GLint y) {
	Color color;
	glReadPixels(x, y, 1, 1, GL_RGB, GL_FLOAT, &color);
	return color;
}

// Function to set the color of a pixel
void setPixelColor(GLint x, GLint y, Color color) {
	glColor3f(color.r, color.g, color.b);
	glBegin(GL_POINTS);
		glVertex2i(x, y);
	glEnd();
	glFlush();
}

// 8-connectivity flood fill algorithm
void floodFill(GLint x, GLint y, Color oldColor, Color newColor) {
	Color color = getPixelColor(x, y);

	// Check if the color of the pixel matches the old color
	if (color.r == oldColor.r && color.g == oldColor.g && color.b == oldColor.b) {
		setPixelColor(x, y, newColor);

		// Recursive calls for 8-connectivity (all 8 neighboring pixels)

		floodFill(x + 1, y - 1, oldColor, newColor); // Top-right
		floodFill(x - 1, y, oldColor, newColor);     // Left
		floodFill(x + 1, y, oldColor, newColor);     // Right
		floodFill(x - 1, y + 1, oldColor, newColor); // Bottom-left
		floodFill(x, y + 1, oldColor, newColor);     // Bottom
		floodFill(x - 1, y - 1, oldColor, newColor); // Top-left
		floodFill(x, y - 1, oldColor, newColor);     // Top
		floodFill(x + 1, y + 1, oldColor, newColor); // Bottom-right
	}
}

// Mouse click handler to initiate the flood fill
void onMouseClick(int button, int state, int x, int y) {
	// Define the colors
	Color newColor = {1.0f, 0.0f, 1.0f}; // Red
```

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

        // Start flood fill at the given point (adjust for OpenGL coordinate system)
        floodFill(x, 480 - y, oldColor, newColor);
}

// Display function to draw the square and trigger the flood fill
void display(void) {
                Point p1 = {100, 100},   // bottom-right
    p2 = {300, 100},   // bottom-left (increased x by 200)
    p3 = {300, 300},   // top-right (increased x and y by 200)
    p4 = {100, 300};   // top-left (increased y by 200)


        glClear(GL_COLOR_BUFFER_BIT);
        glBegin(GL_POINTS);
                draw_dda(p1, p2);
                draw_dda(p2, p3);
                draw_dda(p3, p4);
                draw_dda(p4, p1);
        glEnd();
        glFlush();
}

// Main function to set up GLUT and OpenGL context
int main(int argc, char** argv) {
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(640, 480);
        glutInitWindowPosition(200, 200);
        glutCreateWindow("floodfill");

        // Initialize OpenGL
        init();

        // Register callback functions
        glutDisplayFunc(display);
        glutMouseFunc(onMouseClick);

        // Enter the main GLUT loop
        glutMainLoop();
        return 0;
}
```

Output

```
svpm@svpm-HP-EliteDesk-800-G2-SFF:~$ g++ floodfill.cpp -lGL -lGLU -lglut
svpm@svpm-HP-EliteDesk-800-G2-SFF:~$ ./a.out
```


floodfill