Practical no.04

Title:- Implement the following polygon filling methods : i) Boundary fill

Name:-Ujawala Sinha
Roll no:-S554

Boundary fill

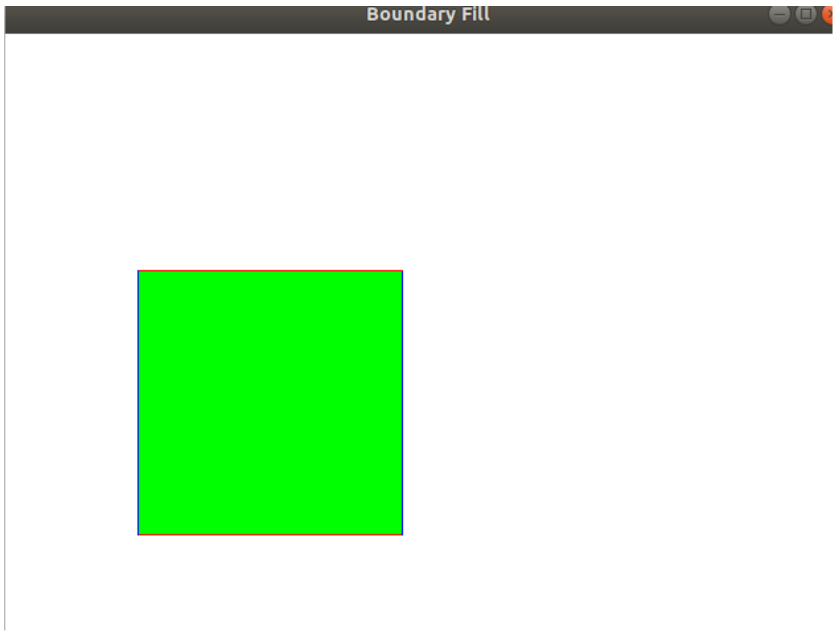
Code:

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

struct Point {
   GLint x, y;
};

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

void draw_dda(Point p1, Point p2, Color color) {
   GLfloat dx = p2.x - p1.x;
   GLfloat dy = p2.y - p1.y;
   GLfloat steps = (abs(dx) > abs(dy)) ? abs(dx) : abs(dy);
   GLfloat xInc = dx / steps;
   GLfloat yInc = dy / steps;
   GLfloat x = p1.x, y = p1.y;
   glColor3f(color.r, color.g, color.b);
   glBegin(GL_POINTS);
   for (int i = 0; i <= steps; i++) {
      glVertex2i(x, y);
      x += xInc;
      y += yInc;
   }
   glEnd();
   glFlush();
}

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

Color getPixelColor(GLint x, GLint y) { Color color;
```

```
    GLfloat pixel[3];
    glReadPixels(x, y, 1, 1, GL_RGB, GL_FLOAT, pixel);
    color.r = pixel[0];
    color.g = pixel[1];
    color.b = pixel[2];
    return color;
}

void setPixelColor(GLint x, GLint y, Color color) {
    glColor3f(color.r, color.g, color.b);
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
    glFlush();
}

void boundaryFill8(GLint x, GLint y, Color fillColor, Color boundaryColor1, Color
boundaryColor2) {
    Color color = getPixelColor(x, y);


    if ((color.r != boundaryColor1.r || color.g != boundaryColor1.g || color.b !=
boundaryColor1.b) &&
        (color.r != boundaryColor2.r || color.g != boundaryColor2.g || color.b !=
boundaryColor2.b) &&
        (color.r != fillColor.r || color.g != fillColor.g || color.b != fillColor.b)) {

        setPixelColor(x, y, fillColor);


        boundaryFill8(x, y + 1, fillColor, boundaryColor1, boundaryColor2);
        boundaryFill8(x, y - 1, fillColor, boundaryColor1, boundaryColor2);
        boundaryFill8(x + 1, y, fillColor, boundaryColor1, boundaryColor2);
        boundaryFill8(x - 1, y, fillColor, boundaryColor1, boundaryColor2);
        boundaryFill8(x + 1, y + 1, fillColor, boundaryColor1, boundaryColor2);
        boundaryFill8(x - 1, y - 1, fillColor, boundaryColor1, boundaryColor2);
        boundaryFill8(x + 1, y - 1, fillColor, boundaryColor1, boundaryColor2);
        boundaryFill8(x - 1, y + 1, fillColor, boundaryColor1, boundaryColor2);
    }
}

void onMouseClick(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        Color fillColor = {0.0f, 1.0f, 0.0f}; // Green
        Color boundaryColor1 = {1.0f, 0.0f, 0.0f}; // Red
        Color boundaryColor2 = {0.0f, 0.0f, 1.0f}; // Blue
        boundaryFill8(x, 480 - y, fillColor, boundaryColor1, boundaryColor2);
    }
}

void display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    Point p1 = {100, 100}, p2 = {300, 100}, p3 = {300, 300}, p4 = {100, 300};
    Color red = {1.0f, 0.0f, 0.0f}; // Red
    Color blue = {0.0f, 0.0f, 1.0f}; // Blue

    draw_dda(p1, p2, red);
    draw_dda(p2, p3, blue);
    draw_dda(p3, p4, red);
    draw_dda(p4, p1, blue);

    glFlush();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(200, 200);
    glutCreateWindow("Boundary Fill");
    init();
    glutDisplayFunc(display);
    glutMouseFunc(onMouseClick);
    glutMainLoop();
    return 0;
}
```

Output

```
svpm@svpm-HP-EliteDesk-800-G2-SFF:~$ g++ bound2.cpp -lGL -lGLU -lglut
svpm@svpm-HP-EliteDesk-800-G2-SFF:~$ ./a.out
```


Boundary Fill