#include <GL/glut.h>

#include <iostream>

#include <unistd.h> // For sleep function

using namespace std;

int width = 800, height = 600;

float floodFillColor[3] = {1.0, 0.0, 0.0}; // Red Color

float boundaryFillColor[3] = {0.0, 0.0, 1.0}; // Blue Color

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

float bgColor[3] = {1.0, 1.0, 1.0}; // White background

float titleColor[3] = {0.0, 0.0, 0.0}; // Blue for title

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

glBegin(GL\_POINTS);

glColor3fv(color);

glVertex2i(x, y);

glEnd();

glFlush();

usleep(1000); // Delay for visualization

}

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

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

}

void floodFill(int x, int y, float oldColor[3], float newColor[3]) {

float color[3];

getPixelColor(x, y, color);

if ((color[0] == oldColor[0] && color[1] == oldColor[1] && color[2] == oldColor[2]) &&

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

setPixel(x, y, newColor);

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

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

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

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

}

}

void boundaryFill(int x, int y, float boundaryColor[3], float newColor[3]) {

float color[3];

getPixelColor(x, y, color);

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

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

setPixel(x, y, newColor);

boundaryFill(x + 1, y, boundaryColor, newColor);

boundaryFill(x - 1, y, boundaryColor, newColor);

boundaryFill(x, y + 1, boundaryColor, newColor);

boundaryFill(x, y - 1, boundaryColor, newColor);

}

}

void drawTitle() {

// Draw "Flood Fill" text outline (left side)

glColor3fv(borderColor);

glBegin(GL\_LINE\_LOOP);

glVertex2i(50, 500);

glVertex2i(250, 500);

glVertex2i(250, 550);

glVertex2i(50, 550);

glEnd();

// Draw text inside using for loop instead of for-each

glColor3fv(titleColor);

glRasterPos2i(70, 520);

string text1 = "Flood Fill (Left Click)";

for (int i = 0; i < text1.length(); i++) {

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, text1[i]);

}

// Draw "Boundary Fill" text outline (right side)

glColor3fv(borderColor);

glBegin(GL\_LINE\_LOOP);

glVertex2i(350, 500);

glVertex2i(600, 500);

glVertex2i(600, 550);

glVertex2i(350, 550);

glEnd();

// Draw text inside using for loop instead of for-each

glColor3fv(titleColor);

glRasterPos2i(370, 520);

string text2 = "Boundary Fill (Right Click)";

for (int i = 0; i < text2.length(); i++) {

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, text2[i]);

}

glFlush();

}

void drawShape() {

// Draw a square

glColor3fv(borderColor);

glBegin(GL\_LINE\_LOOP);

glVertex2i(200, 200);

glVertex2i(300, 200);

glVertex2i(300, 300);

glVertex2i(200, 300);

glEnd();

// Draw a triangle

glBegin(GL\_LINE\_LOOP);

glVertex2i(400, 200);

glVertex2i(500, 200);

glVertex2i(450, 300);

glEnd();

glFlush();

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

drawTitle();

drawShape();

}

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

y = height - y; // Flip y coordinate to match OpenGL's coordinate system

// Check if click is within the title boxes

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

if (x >= 50 && x <= 250 && y >= 500 && y <= 550) {

// Flood fill the "Flood Fill" title

floodFill(100, 525, bgColor, floodFillColor);

} else {

// Flood fill the shape

float oldColor[3];

getPixelColor(x, y, oldColor);

if (oldColor[0] == bgColor[0] && oldColor[1] == bgColor[1] && oldColor[2] == bgColor[2]) {

floodFill(x, y, oldColor, floodFillColor);

}

}

}

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

if (x >= 350 && x <= 550 && y >= 500 && y <= 550) {

// Boundary fill the "Boundary Fill" title

boundaryFill(400, 525, borderColor, boundaryFillColor);

} else {

// Boundary fill the shape

boundaryFill(x, y, borderColor, boundaryFillColor);

}

}

}

void init() {

glClearColor(bgColor[0], bgColor[1], bgColor[2], 1.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0, width, 0, height);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(width, height);

glutCreateWindow("Flood Fill and Boundary Fill Demo");

glutDisplayFunc(display);

glutMouseFunc(mouse);

init();

glutMainLoop();

return 0;

}