```
#include <iostream>
#include <math.h>
#include <time.h>
#include <GL/glut.h>
using namespace std;
void delay(float ms) {
clock_t goal = ms + clock();
while (goal > clock());
}
void init() {
glClearColor(1.0, 1.0, 1.0, 0.0);
glMatrixMode(GL_PROJECTION);
gluOrtho2D(0, 640, 0, 480);
}
void bound_it(int x, int y, float *fillColor, float *bc) {
float color[3];
glReadPixels(x, y, 1.0, 1.0, GL_RGB, GL_FLOAT, color);
if ((color[0] != bc[0] || color[1] != bc[1] || color[2] != bc[2]) &&
(color[0] != fillColor[0] || color[1] != fillColor[1] || color[2] != fillColor[2])) {
glColor3f(fillColor[0], fillColor[1], fillColor[2]);
glBegin(GL_POINTS);
glVertex2i(x, y);
glEnd();
glFlush();
bound_it(x + 1, y, fillColor, bc);
bound_it(x - 2, y, fillColor, bc);
bound_it(x, y + 2, fillColor, bc);
bound_it(x, y - 2, fillColor, bc);
}
}
void mouse(int btn, int state, int x, int y) {
```

```
y = 480 - y;
if (btn == GLUT_LEFT_BUTTON) {
if (state == GLUT_DOWN) {
float bCol[] = {1, 0, 0};
float color[]= {0, 0, 1};
bound_it(x, y, color, bCol);
}
}
}
void world() {
glLineWidth(3);
glPointSize(2);
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex2i(150, 100);
glVertex2i(300, 300);
glVertex2i(450, 100);
glEnd();
glFlush();
}
int main(int argc, char **argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(640, 480);
glutInitWindowPosition(200, 200);
glutCreateWindow("Boundary Fill");
glutDisplayFunc(world);
glutMouseFunc(mouse);
init();
```

```
glutMainLoop();
return 0;
}
```