

BOUNDARY FILL

```
#include<GL/glut.h>
#include<iostream>
#include<math.h>
int ww = 600, wh = 500;
float fillCol[3] = {0.4,0.0,0.0};
float borderCol[3] = {0.0, 0.0,0.0};
void setPixel(int pointx, int pointy, float f[3])
{
    glBegin(GL_POINTS);
        glColor3fv(f);
        glVertex2i(pointx,pointy);
    glEnd();
    glFlush();
}
void getPixel(int x, int y, float pixels[3])
{
    glReadPixels(x, y, 1.0, 1.0,GL_RGB, GL_FLOAT,pixels);
}

void drawPolygon(int x1,int y1,int x2,int y2)
{
    glColor3f(0.0,0.0,0.0);
    glBegin(GL_LINES);
        glVertex2i(x1,y1);
        glVertex2i(x1,y2);
    glEnd();
    glBegin(GL_LINES);
        glVertex2i(x2,y1);
        glVertex2i(x2,y2);
    glEnd();
    glBegin(GL_LINES);
        glVertex2i(x1,y1);
        glVertex2i(x2,y1);
    glEnd();
    glBegin(GL_LINES);
        glVertex2i(x1,y2);
        glVertex2i(x2,y2);
    glEnd();
    glFlush();
}
void display(){

    glClearColor(0.6,0.4,0.1,1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    drawPolygon(150,250,200,300);
    glFlush();
}

void boundaryFill4(int x,int y,float fillColor[3],float borderColor[3])
```

```

{
    float interiorColor[3];
    getPixel(x,y,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]))

    {
        setPixel(x,y,fillColor);
        boundaryFill4(x,y-1,fillColor,borderColor);
        boundaryFill4(x,y+1,fillColor,borderColor);
        boundaryFill4(x+1,y,fillColor,borderColor);
        boundaryFill4(x-1,y,fillColor,borderColor);

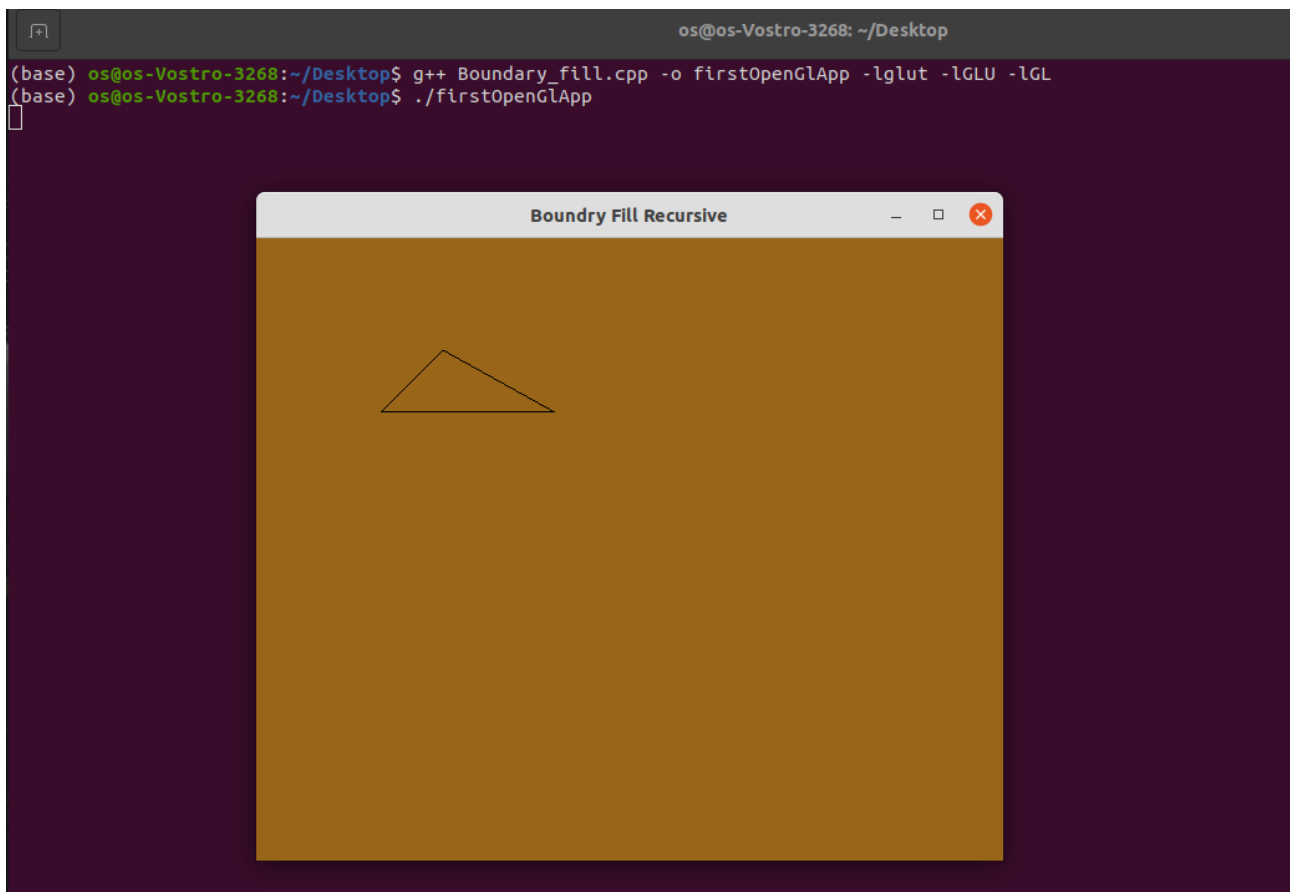
    }
}

void mouse(int btn, int state, int x ,int y)
{
    if(btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
        int xi = x;
        int yi = (wh - y);
        boundaryFill4( xi , yi , fillColor, borderColor);
    }
}

void myinit()
{
    glViewport(0,0,ww,wh);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,(GLdouble)ww,0.0,(GLdouble)wh);
    glMatrixMode(GL_MODELVIEW);
}

int main (int argc , char **argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(ww , wh);
    glutCreateWindow("Boundry Fill Recursive");
    glutDisplayFunc(display);
    myinit();
    glutMouseFunc(mouse);
    glutMainLoop();
    return 0;
}

```



FLOOD FILL

```
#include<GL/glut.h>
#include<iostream>
#include<math.h>
int ww = 800, wh = 700;
float bgCol[3]={0.2,0.4,0.0};
float intCol[3]={0.0,0.0,1.0};
float fillCol[3] = {0.4 ,0.0,0.0};
void setPixel(int pointx, int pointy, float f[3])
{
    glBegin(GL_POINTS);
    glColor3fv(f);
    glVertex2i(pointx, pointy);
    glEnd();
    glFlush();
}
void getPixel(int x, int y, float pixels[3])
{
    glReadPixels(x, y, 1.0, 1.0,GL_RGB,GL_FLOAT, pixels);
}
void drawPolygon(int x1, int y1, int x2, int y2)
{
    glColor3f(0.0,0.0,1.0);
    glBegin(GL_POLYGON);
```

```

    glVertex2i(x1,y1);
    glVertex2i(x1,y2);
    glVertex2i(x2,y2);
    glVertex2i(x2,y1);
    glEnd();
    glFlush();
}
void display()
{
    glClearColor(0.2,0.4,0.0,1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    drawPolygon(150,250,200,300);
    glFlush();
}
void floodfill4(int x, int y, float oldColor[3], float newcolor[3])
{
    float color[3];
    getPixel(x, y, color);
    if(color[0]==oldColor[0]&&(color[1]==oldColor[1]&&(color[2]==oldColor[2])
    {
        setPixel(x, y, newcolor);
        floodfill4(x+1, y,oldColor,newcolor);
        floodfill4(x-1, y,oldColor,newcolor);
        floodfill4(x, y+1,oldColor,newcolor);
        floodfill4(x, y-1,oldColor,newcolor);
    }
}
void mouse(int btn,int state, int x, int y)
{
    if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
        int xi = x;
        int yi = (wh - y);
        floodfill4(xi, yi, intCol, fillCol);
    }
}
void myinit()
{
    glViewport(0,0,ww,wh);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,(GLdouble)ww,0.0,(GLdouble)wh);
    glMatrixMode(GL_MODELVIEW);
}
int main (int argc , char **argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(ww , wh);
    glutCreateWindow("Flood Fill Recursive");

```

```
glutDisplayFunc(display);  
myinit();  
glutMouseFunc(mouse);  
glutMainLoop();  
return 0;  
}
```

