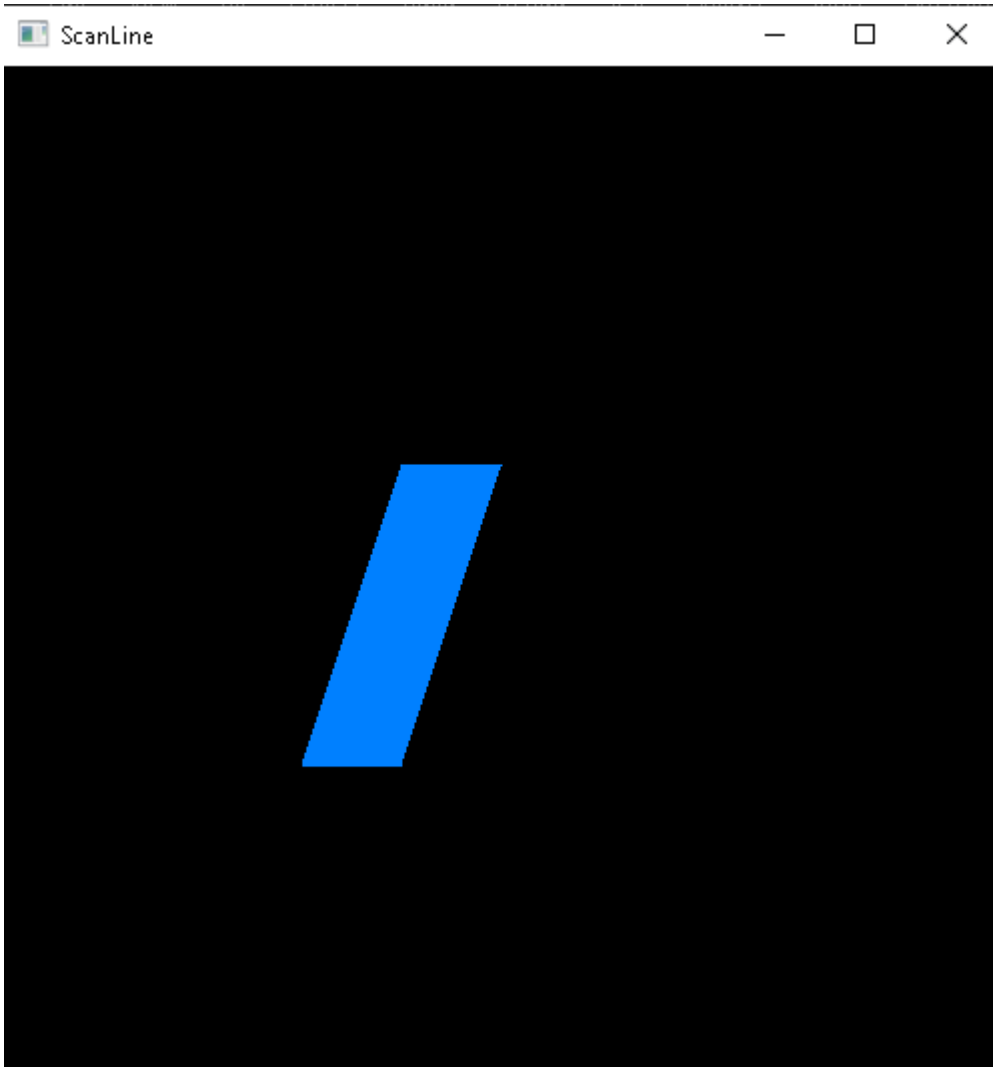
ASSIGNMENT -5
COMPUTER GRAPHICS

- Shubham Tiwari
101916126
3CS12

1. Scan line filling Algorithm

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

float x1, x2, x3, x4, y1, y2, y3, y4;

void draw_pixel(int x, int y)
{
    glColor3f(0.0, .5, 1.0);
    glPointSize(1.0);
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
}

void edgedetect(float x1, float y1, float x2, float y2, int* le, int* re)
{
    float temp, x, mx;
    int i;

    if (y1 > y2)
    {
        temp = x1, x1 = x2, x2 = temp;
        temp = y1, y1 = y2, y2 = temp;
    }

    if (y1 == y2)
        mx = x2 - x1;
    else
        mx = (x2 - x1) / (y2 - y1);

    x = x1;

    for (i = int(y1); i <= (int)y2; i++)
    {
        if (x < (float)le[i]) le[i] = (int)x;
        if (x > (float)re[i]) re[i] = (int)x;
        x += mx;
    }
}

void scanfill(float x1, float y1, float x2, float y2, float x3, float y3,
float x4, float y4)
{
    int le[500], re[500], i, j;

    for (i = 0; i < 500; i++)
        le[i] = 500, re[i] = 0;

    edgedetect(x1, y1, x2, y2, le, re);
    edgedetect(x2, y2, x3, y3, le, re);
    edgedetect(x3, y3, x4, y4, le, re);
```

```
    edgedetect(x4, y4, x1, y1, le, re);

    for (j = 0; j < 500; j++)
    {
        if (le[j] <= re[j])
            for (i = le[j]; i < re[j]; i++)
                draw_pixel(i, j);
    }
}


void display()
{
    x1 = 150.0; y1 = 150.0; x2 = 200.0; y2 = 150.0; x3 = 250.0;
    y3 = 300.0; x4 = 200.0; y4 = 300.0;
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0, 0.0, 0.0);
    glBegin(GL_LINE_LOOP);
    glVertex2f(x1, y1);
    glVertex2f(x2, y2);
    glVertex2f(x3, y3);
    glVertex2f(x4, y4);
    glEnd();

    scanfill(x1, y1, x2, y2, x3, y3, x4, y4);

    glFlush();
}


void init()
{
    glClearColor(0.0, 0.0, 0.0, 0.0); // back ground color
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 499.0, 0.0, 499.0);
}

void main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);

    glutCreateWindow("ScanLine");
    glutDisplayFunc(display);

    init();
    glutMainLoop();
}
```

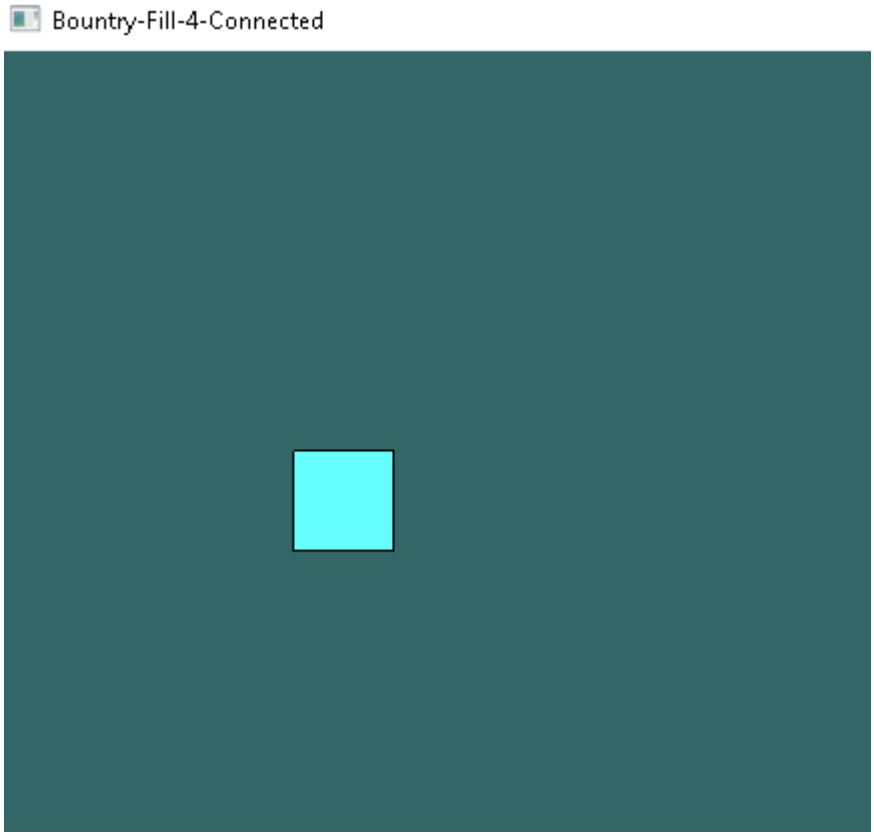


2.1 Boundary fill 4 connected

```
#include <GL/glut.h>
int maxWidth = 600, maxHeight = 500;
float fillCol[3] = { 0.4,1.0,1.0 };
float borderCol[3] = { 1.0,0.5,0.0 };
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(0.2, 0.4, 0.4, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    drawPolygon(150, 250, 200, 300);
    //drawPolygon(150, 250, 350, 400);
    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 + 1, y, fillColor, borderColor);
        boundaryFill4(x - 1, y, fillColor, borderColor);
        boundaryFill4(x, y + 1, fillColor, borderColor);
        boundaryFill4(x, y - 1, 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 = (maxHeight - y);
        boundaryFill4(xi, yi, fillCol, borderCol);
    }
}
void myinit()
{
    glViewport(0, 0, maxWidth, maxHeight);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, (GLdouble)maxWidth, 0.0,
        (GLdouble)maxHeight); glMatrixMode(GL_MODELVIEW);
}
int main(int argc, char** argv)
```

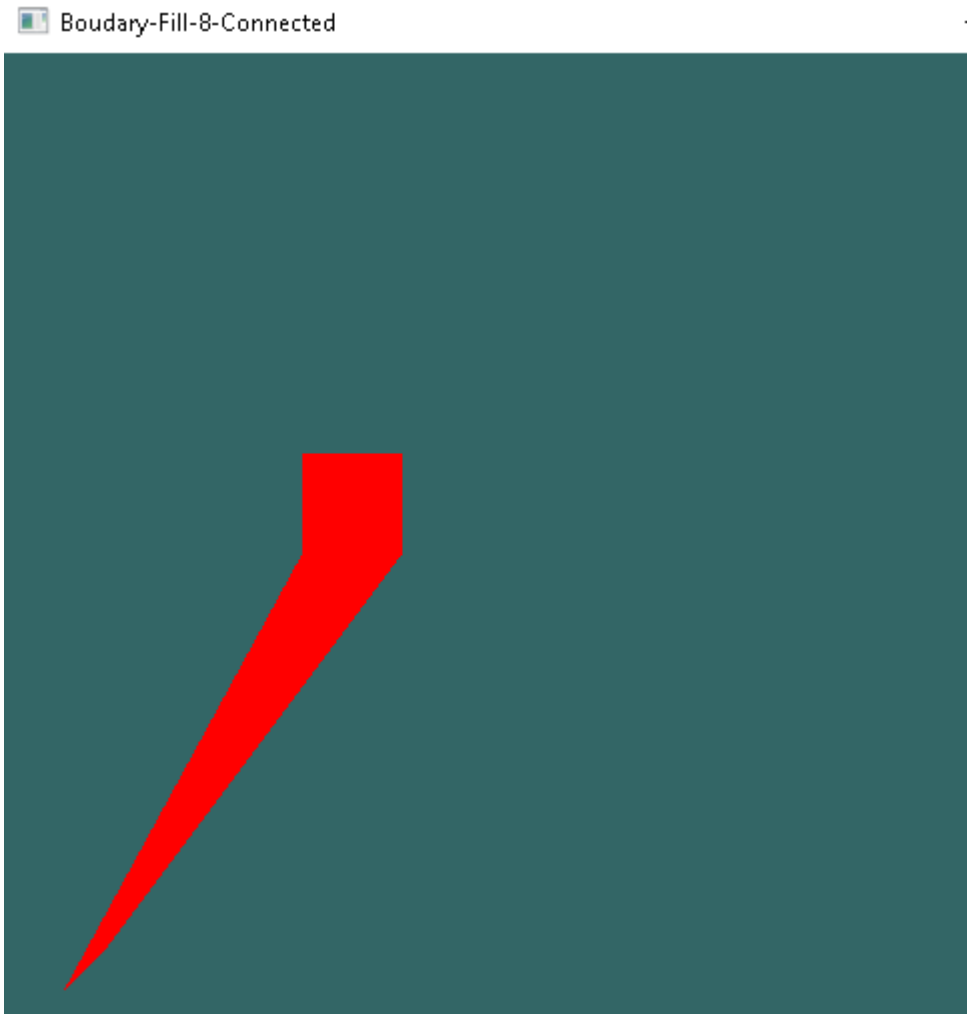
```
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE |
        GLUT_RGB); glutInitWindowSize(maxWidth, maxHeight);
    glutCreateWindow("Bountry-Fill-Recursive");
    glutDisplayFunc(display); myinit();
    glutMouseFunc(mouse);
    glutMainLoop();
    return 0;
}
```



2.1 Boundary fill 8 connected

```
#include <GL/glut.h>
int ww = 600, wh = 500;
float bgCol[3] = { 0.4, 0.4,1.0 };
float intCol[3] = { 0.0,0.5,1.0 };
float fillCol[3] = { 0.0,0.0,1.0 };
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(1.0, 0.0, 0.0);
    glBegin(GL_POLYGON);
    glVertex2i(x1, y1);
    glVertex2i(x1, y2);
    glVertex2i(x2, y2);
    glVertex2i(x2, y1);
    glVertex2i(50, 50);
    glVertex2i(30, 30);
    glEnd();
```

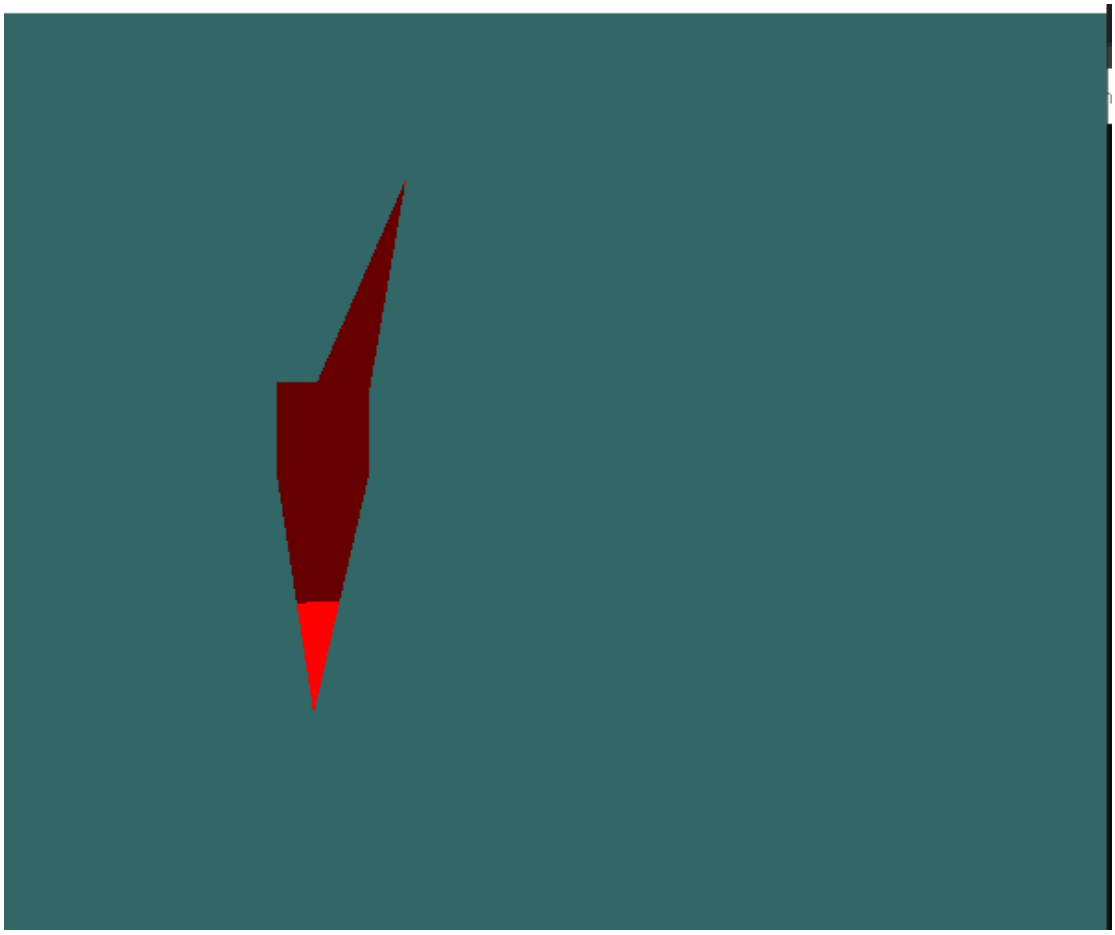
```
    glFlush();
}
void display()
{
    glClearColor(0.2, 0.4, 0.4, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    drawPolygon(150, 250, 200, 300);
    glFlush();
}
void boundaryFill8(int x, int y, float fillColor[3], float borderColor[3])
{
    float interiorColor[3];
    getPixel(x, y, interiorColor);
    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);
        boundaryFill8(x + 1, y, fillColor, borderColor);
        boundaryFill8(x, y + 1, fillColor, borderColor);
        boundaryFill8(x - 1, y, fillColor, borderColor);
        boundaryFill8(x, y - 1, fillColor, borderColor);
        boundaryFill8(x - 1, y - 1, fillColor, borderColor);
        boundaryFill8(x - 1, y + 1, fillColor, borderColor);
        boundaryFill8(x + 1, y - 1, fillColor, borderColor);
        boundaryFill8(x + 1, y + 1, 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);
        boundaryFill8(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("Boudary-Fill-8-Connected");
    glutDisplayFunc(display); myinit();
    glutMouseFunc(mouse);
    glutMainLoop();
    return 0;
}
```



2.2 Boundary fill 4 connected

```
#include <GL/glut.h>
int ww = 600, wh = 500;
float bgCol[3] = { 0.2, 0.4,0.0 };
float intCol[3] = { 1.0,0.0,0.0 };
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(1.0, 0.0, 0.0);
    glBegin(GL_POLYGON);
    glVertex2i(x1, y1);
```

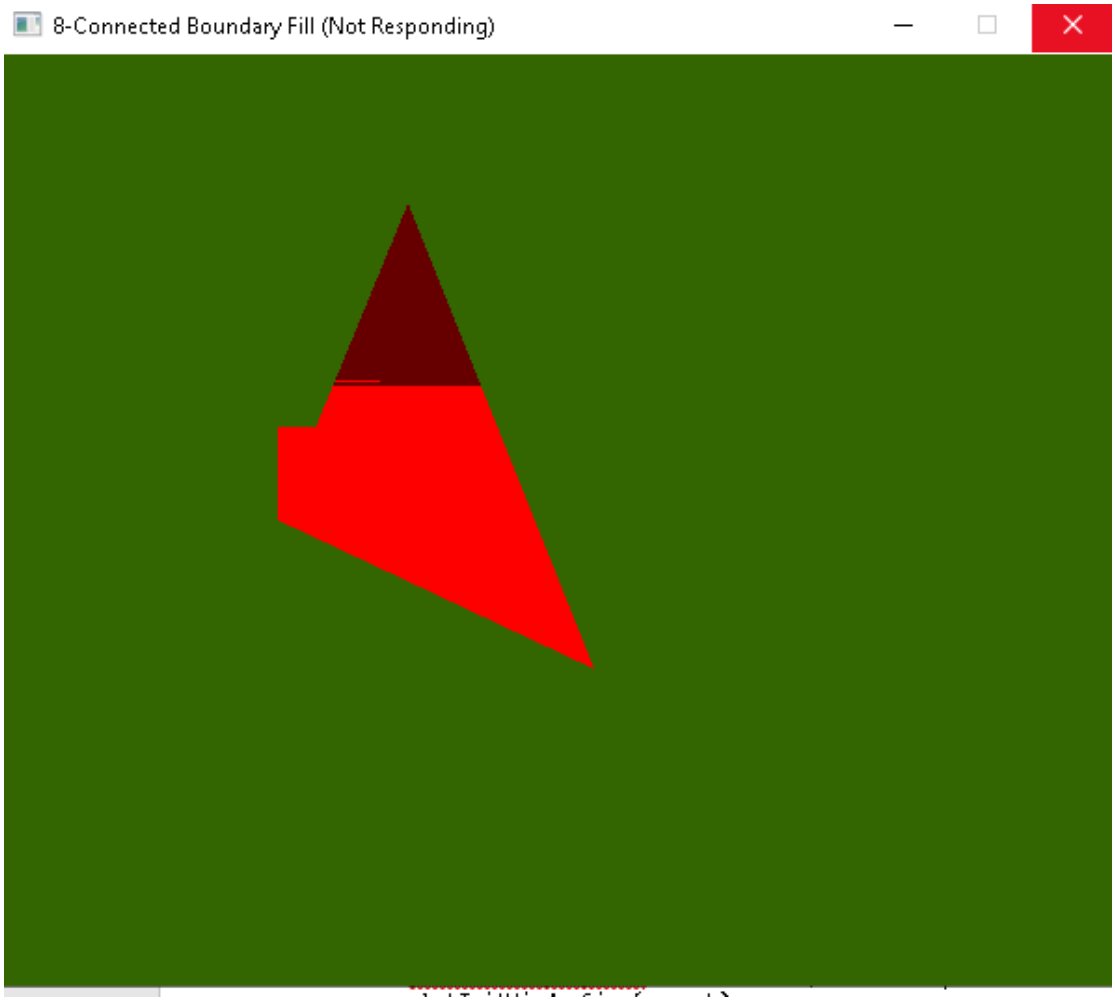
```
    glVertex2i(x1, y2);
    glVertex2i(x2, y2);
    glVertex2i(x2, y1);
    glVertex2i(170, 120);
    glVertex2i(220, 410);
    glEnd();
    glFlush();
}
void display()
{
    glClearColor(0.2, 0.4, 0.4, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    drawPolygon(150, 250, 200, 300);
    glFlush();
}
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("Boundary Fill 4 Connected");
    glutDisplayFunc(display); myinit();
    glutMouseFunc(mouse);
    glutMainLoop();
    return 0;
}
```



2.2 Boundary fill 8 connected

```
#include <GL/glut.h>
int ww = 600, wh = 500;
float bgCol[3] = { 0.2, 0.4,0.0 };
float intCol[3] = { 1.0,0.0,0.0 };
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(1.0, 0.0, 0.0);
    glBegin(GL_POLYGON);
    glVertex2i(x1, y1);
    glVertex2i(x1, y2);
    glVertex2i(x2, y2);
    glVertex2i(x2, y1);
    glVertex2i(220, 420);
    glVertex2i(320, 170);
```

```
    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 floodfill(int x, int y, float old[3], float newcol[3])
{
    float color[3];
    getPixel(x, y, color);
    if (color[0] == old[0] && (color[1]) == old[1] && (color[2]) ==
old[2])
    {
        setPixel(x, y, newcol);
        floodfill(x + 1, y, old, newcol);
        floodfill(x - 1, y, old, newcol);
        floodfill(x, y + 1, old, newcol);
        floodfill(x, y - 1, old, newcol);
        floodfill(x + 1, y + 1, old, newcol);
        floodfill(x - 1, y + 1, old, newcol);
        floodfill(x + 1, y - 1, old, newcol);
        floodfill(x - 1, y - 1, old, newcol);
    }
}
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);
        floodfill(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("8-Connected Boundary Fill");
    glutDisplayFunc(display); myinit();
    glutMouseFunc(mouse);
    glutMainLoop();
    return 0;
}
```


8-Connected Boundary Fill (Not Responding)