# LAB-4

**Implement the following polygon filling methods :**

**i) Flood fill / Seed fill ii) Boundary fill**

**using mouse click, keyboard interface and menu driven programming.**

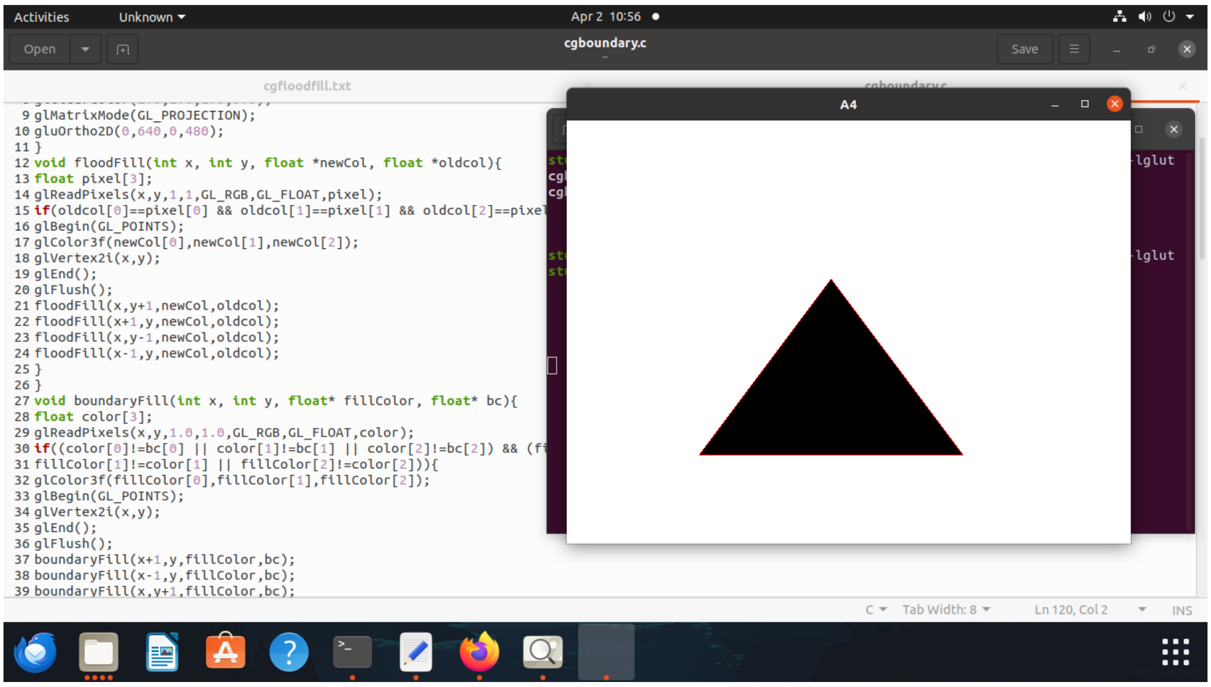
**Source Code :**

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

using namespace std;

float R=0,G=0,B=0;
int Algo;

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


void floodFill(int x, int y, float *newCol, float *oldcol){
   float pixel[3];
   glReadPixels(x,y,1,1,GL_RGB,GL_FLOAT,pixel);

   if(oldcol[0]==pixel[0] &&oldcol[1]==pixel[1] &&oldcol[2]==pixel[2]){

      glBegin(GL_POINTS);
         glColor3f(newCol[0],newCol[1],newCol[2]);
         glVertex2i(x,y);
      glEnd();
      glFlush();

      floodFill(x,y+1,newCol,oldcol);
      floodFill(x+1,y,newCol,oldcol);
      floodFill(x,y-1,newCol,oldcol);
      floodFill(x-1,y,newCol,oldcol);
   }
```

```
}
void boundaryFill(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]) && (fillColor[0]!=color[0] ||
fillColor[1]!=color[1] || fillColor[2]!=color[2])){

            glColor3f(fillColor[0],fillColor[1],fillColor[2]);
            glBegin(GL_POINTS);
                glVertex2i(x,y);
            glEnd();
            glFlush();
            boundaryFill(x+1,y,fillColor,bc);
            boundaryFill(x-1,y,fillColor,bc);
            boundaryFill(x,y+1,fillColor,bc);
            boundaryFill(x,y-1,fillColor,bc);


        }

     return;
}

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

    y = 480-y;
      if(btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN){

          float bcol[] = {1,0,0};
          float oldcol[] = {1,1,1};
          float newCol[] = {R,G,B};

          if(Algo==1){
              boundaryFill(x,y,newCol,bcol);
          }
          if(Algo==2){
              floodFill(x,y,newCol,oldcol);
          }
      }
}

void B_Draw(){

    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1,0,0);
    glBegin(GL_LINE_LOOP);
        glVertex2i(150,100);
        glVertex2i(300,300);
        glVertex2i(450,100);
    glEnd();
    glFlush();

}

void F_Draw(){

    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINES);
        glColor3f(1,0,0);glVertex2i(150,100);glVertex2i(300,300);
    glEnd();
    glBegin(GL_LINE_LOOP);
        glColor3f(0,0,1);glVertex2i(300,300);glVertex2i(450,100);
    glEnd();
    glBegin(GL_LINE_LOOP);
        glColor3f(0,0,0);glVertex2i(450,100);glVertex2i(150,100);
    glEnd();
    glFlush();

}

void goMenu(int value){

    switch(value){

        case 1:
            R = 0, G = 1, B=0;
            break;
        case 2:
            R = 1, G = 1, B=0;
            break;
        case 3:
            R = 1, G = 0, B=1;
            break;

    }
    glutPostRedisplay();
}

int main(int argc, char** argv){
    cout<<"\n \t Select the Algorithm ";
    cout<<"\n \t 1. Boundary Fill Algorithm ";
    cout<<"\n \t 2. Flood Fill Algorithm \n \t";
    cin>>Algo;

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(640,480);
    glutInitWindowPosition(200,200);
    glutCreateWindow("A4");
    init();
    glutCreateMenu(goMenu);

     glutAddMenuEntry("Color 1 Green",1);
     glutAddMenuEntry("Color 2 Yellow",2);
     glutAddMenuEntry("Color 3 Pink",3);
     glutAttachMenu(GLUT_RIGHT_BUTTON);

    if(Algo==1){
        glutDisplayFunc(B_Draw);
    }
    if(Algo==2){
        glutDisplayFunc(F_Draw);
    }
    glutMouseFunc(mouse);
    glutMainLoop();
    return 0;
}
```

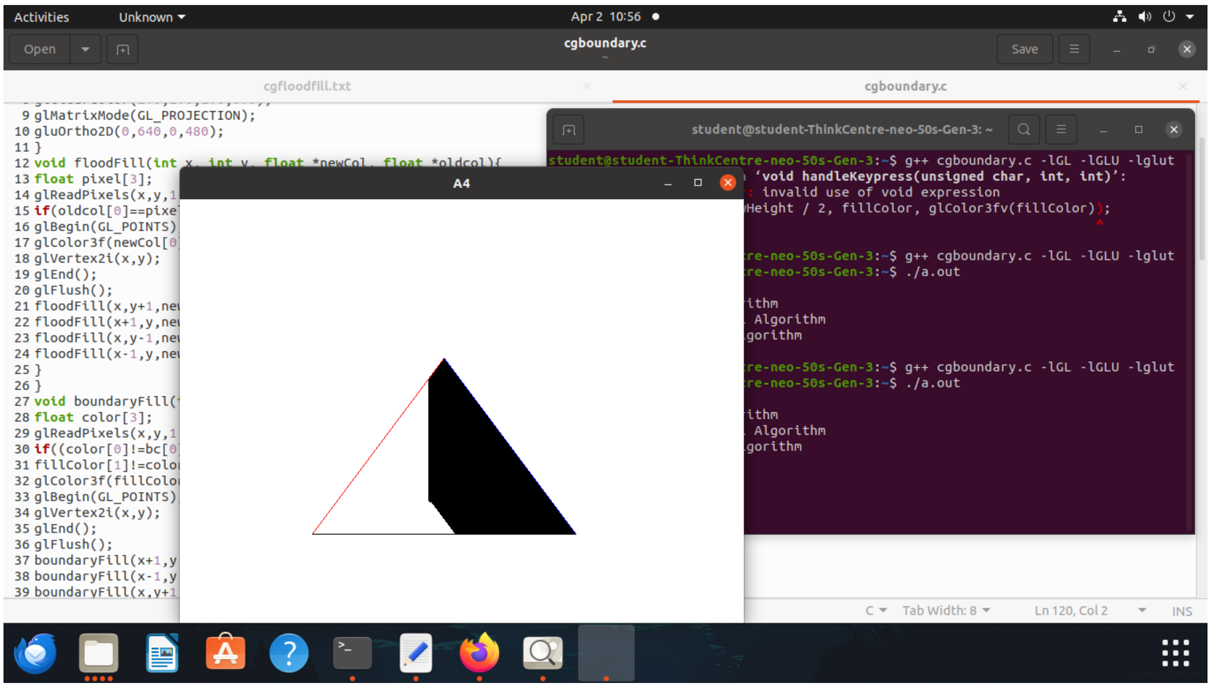
## Output :

## FLOOD FILL -



```
glMatrixMode(GL_PROJECTION);
gluOrtho2D(0,640,0,480);
}
void floodFill(int x, int y, float *newCol, float *oldcol){
float pixel[3];
glReadPixels(x,y,1,1,GL_RGB,GL_FLOAT,pixel);
if(oldcol[0]==pixel[0] && oldcol[1]==pixel[1] && oldcol[2]==pixel
glBegin(GL_POINTS);
glColor3f(newCol[0],newCol[1],newCol[2]);
glVertex2i(x,y);
glEnd();
glFlush();
floodFill(x,y+1,newCol,oldcol);
floodFill(x+1,y,newCol,oldcol);
floodFill(x,y-1,newCol,oldcol);
floodFill(x-1,y,newCol,oldcol);
}
}
void boundaryFill(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]) && (fi
fillColor[1]!=color[1] || fillColor[2]!=color[2])){
glColor3f(fillColor[0],fillColor[1],fillColor[2]);
glBegin(GL_POINTS);
glVertex2i(x,y);
glEnd();
glFlush();
boundaryFill(x+1,y,fillColor,bc);
boundaryFill(x-1,y,fillColor,bc);
boundaryFill(x,y+1,fillColor,bc);
```

## BOUNDARY FILL -



```
glMatrixMode(GL_PROJECTION);
gluOrtho2D(0,640,0,480);
}
void floodFill(int x, int y, float *newCol, float *oldcol){
float pixel[3];
glReadPixels(x,y,1
if(oldcol[0]==pixe
glBegin(GL_POINTS)
glColor3f(newCol[0
glVertex2i(x,y);
glEnd();
glFlush();
floodFill(x,y+1,ne
floodFill(x+1,y,ne
floodFill(x,y-1,ne
floodFill(x-1,y,ne
}
}
void boundaryFill(
float color[3];
glReadPixels(x,y,1
if((color[0]!=bc[0
fillColor[1]!=colo
glColor3f(fillColo
glBegin(GL_POINTS)
glVertex2i(x,y);
glEnd();
glFlush();
boundaryFill(x+1,y
boundaryFill(x-1,y
boundaryFill(x,y+1
```

```
student@student-ThinkCentre-neo-50s-Gen-3:~$ g++ cgboundary.c -lGL -lGLU -lglut
 'void handleKeypress(unsigned char, int, int)':
: invalid use of void expression
Height / 2, fillColor, glColor3fv(fillColor));
                                            ^

re-neo-50s-Gen-3:~$ g++ cgboundary.c -lGL -lGLU -lglut
re-neo-50s-Gen-3:~$ ./a.out

ithm
 Algorithm
gorithm

re-neo-50s-Gen-3:~$ g++ cgboundary.c -lGL -lGLU -lglut
re-neo-50s-Gen-3:~$ ./a.out

ithm
 Algorithm
gorithm
```