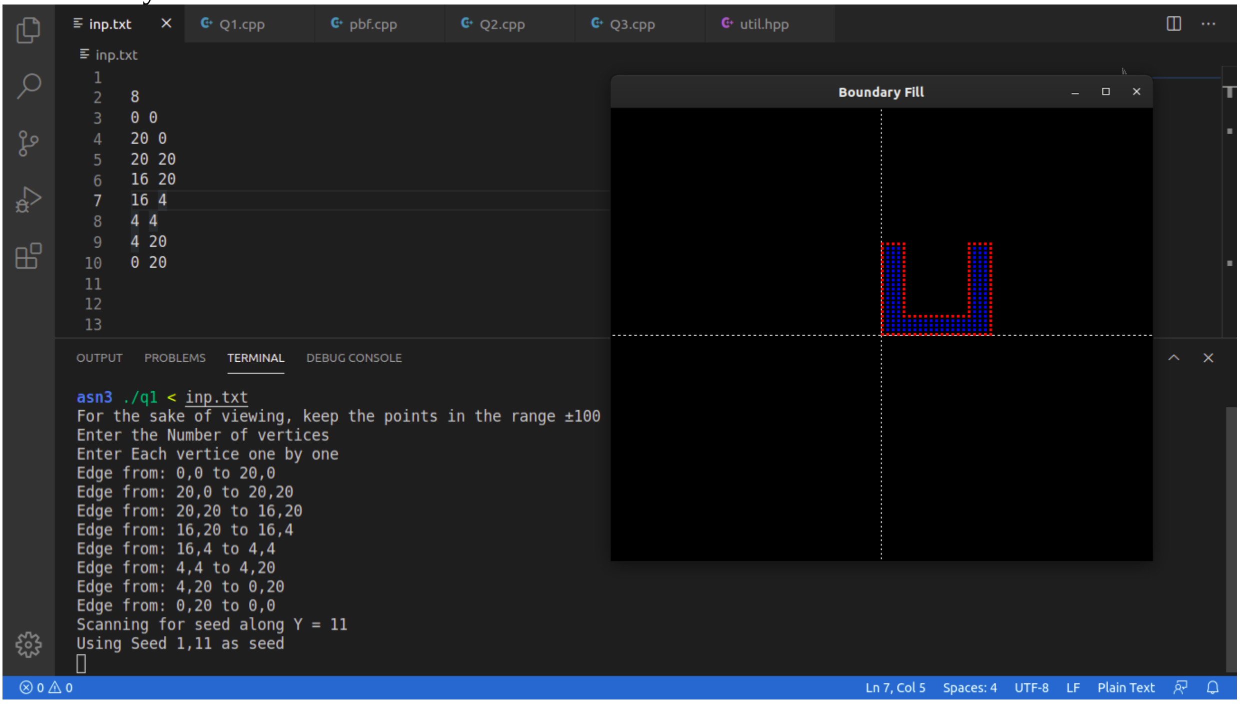
CS352 Assignment 3
Filling Algorithms

Ayush Agrawal
180001011
08-05-2021

Q1. Boundary Fill

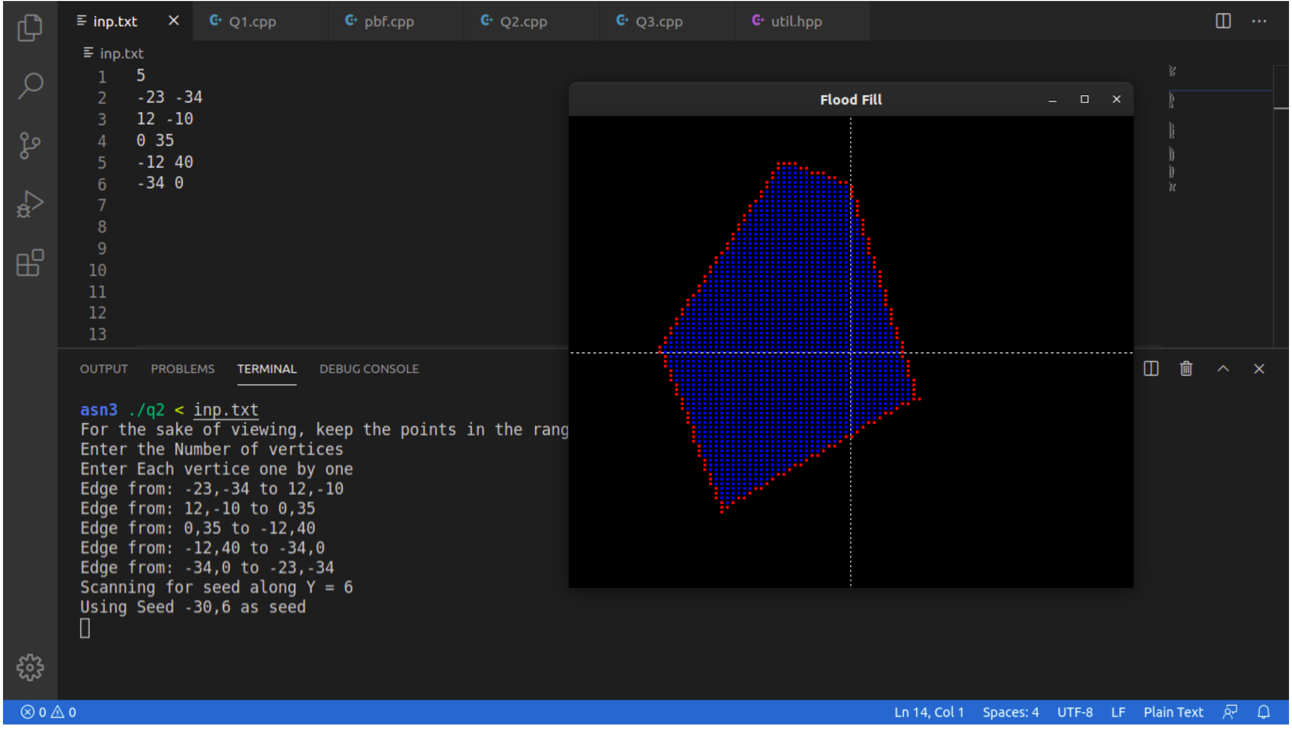


```

8
0 0
20 0
20 20
16 20
16 4
4 4
4 20
0 20
```

```
asn3 ./q1 < inp.txt
For the sake of viewing, keep the points in the range ±100
Enter the Number of vertices
Enter Each vertice one by one
Edge from: 0,0 to 20,0
Edge from: 20,0 to 20,20
Edge from: 20,20 to 16,20
Edge from: 16,20 to 16,4
Edge from: 16,4 to 4,4
Edge from: 4,4 to 4,20
Edge from: 4,20 to 0,20
Edge from: 0,20 to 0,0
Scanning for seed along Y = 11
Using Seed 1,11 as seed
```

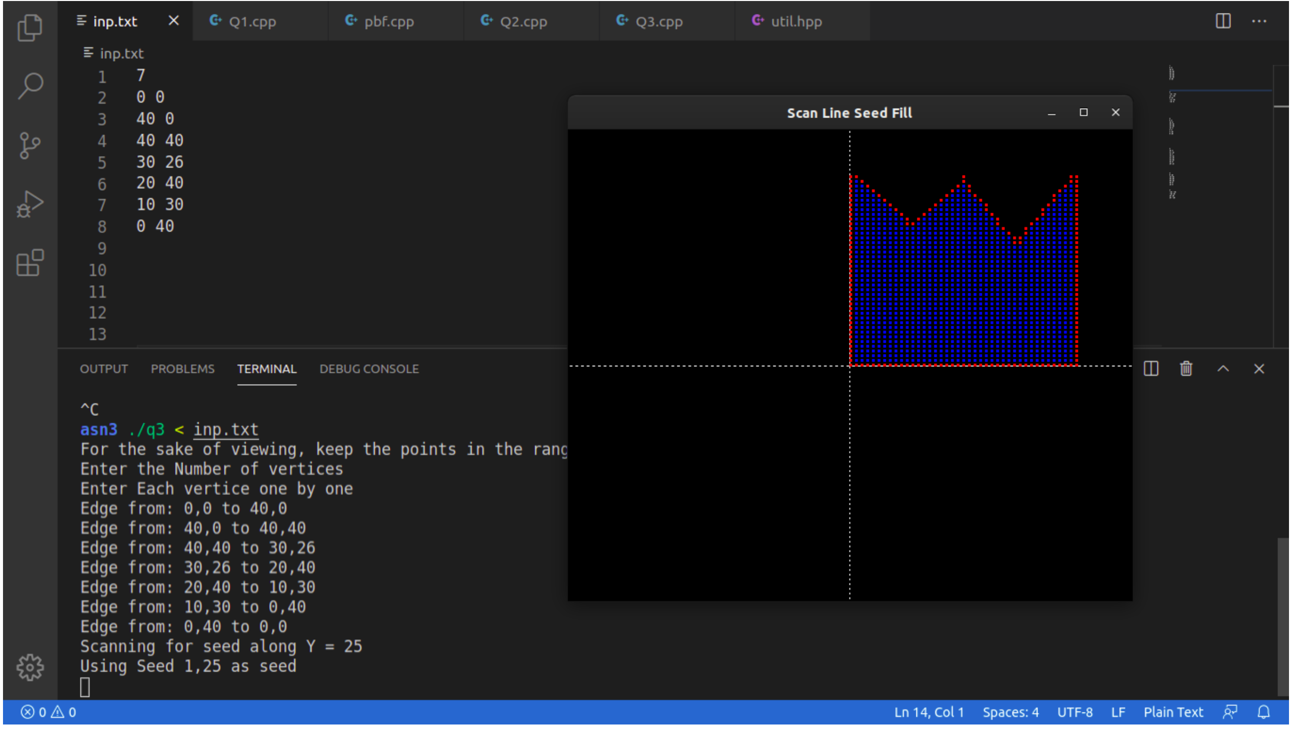
Q2. Flood Fill



```
5
-23 -34
12 -10
0 35
-12 40
-34 0
```

```
asn3 ./q2 < inp.txt
For the sake of viewing, keep the points in the rang
Enter the Number of vertices
Enter Each vertice one by one
Edge from: -23,-34 to 12,-10
Edge from: 12,-10 to 0,35
Edge from: 0,35 to -12,40
Edge from: -12,40 to -34,0
Edge from: -34,0 to -23,-34
Scanning for seed along Y = 6
Using Seed -30,6 as seed
```

Q3. Scanline Seed Fill: -



```
7
0 0
40 0
40 40
30 26
20 40
10 30
0 40
```

```
^C
asn3 ./q3 < inp.txt
For the sake of viewing, keep the points in the rang
Enter the Number of vertices
Enter Each vertice one by one
Edge from: 0,0 to 40,0
Edge from: 40,0 to 40,40
Edge from: 40,40 to 30,26
Edge from: 30,26 to 20,40
Edge from: 20,40 to 10,30
Edge from: 10,30 to 0,40
Edge from: 0,40 to 0,0
Scanning for seed along Y = 25
Using Seed 1,25 as seed
```

----------- Codes -----------

Common Header file (used in all 3 questions.)

contains boundary making, polygon initialization (input and drawing), glDisplayFunction, and constants.

Each question specific file just uses this header and has a colouring function of its own.

File ‘util.hpp’

```
#include <iostream>

#include <GL/glut.h>

#include <vector>

#include <unistd.h>

// #include <utility>




using namespace std;
```

```
typedef pair<int,int> pii;

const int WIDTH = 600;
const int HEIGHT = 500;

const int WC = 250;
const int HC = 250;

const float PIXEL_SIZE = 3;

const int BG = 0;
const int BOUNDARY = 1;
const int INSIDE = 2;

int xc = 0, yc = 0;

const int numColors = 5;

double colors[numColors][3] = {
    {0, 0, 0},
    {1, 0, 0},
    {0, 0, 1},
    {0, 1, 0},
    {0.5, 0.5, 0.5},
};
```

```
}

int screen[WIDTH + 1][HEIGHT + 1];

void printScreen(){
    for(int i=0;i<HEIGHT;i++){
        for(int j=0;j<WIDTH;j++){
            cout<<screen[j][i]<<" ";
        }
        cout<<"\n";
    }
}

void fetchSeed(int ys){
    // we'll scan for the seed along y = ys;
    for(int i=1;i<WIDTH;i++){
        if((screen[i-1][ys] == BOUNDARY) && ((screen[i][ys] != BOUNDARY))){
            xc = i;
            yc = ys;
            break;
        }
    }
}

void setBoundary(int x1, int y1, int x2, int y2){
    screen[x1 + WC][y1 + HC] = BOUNDARY;
```

```
        int dx = x2-x1;
        int dy = y2-y1;

bool mInv = 0;

if(abs(dy) > abs(dx)){
    mInv = 1;
    swap(x1, y1);
    swap(x2, y2);
    swap(dx, dy);
}

int stepX = (dx > 0);
int stepY = (dy > 0);

if(dx < 0) {
    stepX = -1;
    dx = -dx;
}
if(dy < 0) {
    stepY = -1;
    dy = -dy;
}

        int x = x1;
        int y = y1;

        int p = 2*dy-dx;

        while( x != x2 ){
                if(p >= 0){
                        y += stepY;
                        p -= 2*dx;
                }

      if(mInv){
         screen[y + WC][x + HC] = BOUNDARY;
      } else {
         screen[x + WC][y + HC] = BOUNDARY;
      }

      p += 2*dy;
                x += stepX;
        }
}

void initPolygon(){
   cout<<"For the sake of viewing, keep the points in the range ±100\n";

   cout<<"Enter the Number of vertices\n";
   int n;cin>>n;
```

```
if(n < 3) {
    cout<<"Atleast 3 vertices\n";
}
vector<pair<int,int>> points = vector<pair<int,int>>(n);



cout<<"Enter Each vertice one by one\n";


for(int i=0;i<n;i++){
    cin>>points[i].first>>points[i].second;


    xc += points[i].first;
    yc += points[i].second;
}


xc /= n; yc /= n;


for(int i=0;i<n;i++){
    int x1 = points[i].first;
    int y1 = points[i].second;


    int x2 = points[(i+1)%n].first;
    int y2 = points[(i+1)%n].second;


    cout<<"Edge from: "<<x1<<","<<y1<<" to "<<x2<<","<<y2<<"\n";
    setBoundary(x1,y1,x2,y2);
    }

    cout<<"Scanning for seed along Y = "<<yc<<"\n";

    fetchSeed(yc + HC);


    cout<<"Using Seed "<<xc-WC<<","<<yc-HC<<" as seed\n";

}


void drawObject(){

    glClear(GL_COLOR_BUFFER_BIT);


    glBegin(GL_LINES);

        glVertex2d(-50, 0);

        glVertex2d(50, 0);

        glVertex2d(0, -50);

        glVertex2d(0, 50);

    glEnd();


    glBegin(GL_POINTS);

    for(int i=0;i<HEIGHT;i++){

        for(int j=0;j<WIDTH;j++){

            glColor3dv(colors[screen[j][i]]);

            glVertex2i(j - WC, i - HC);

        }

    }

    glEnd();

    glFlush();
```

```
}



void myInit (void){

    // Reset background color with black (since all three argument is 0.0)

    glClear(GL_COLOR_BUFFER_BIT);

    glClearColor(0.0, 0.0, 0.0, 1.0);



    // Set width of point to one unit

    glMatrixMode(GL_PROJECTION);

    glLoadIdentity();



    glPointSize(PIXEL_SIZE);

    // Set window size in X- and Y- direction

    gluOrtho2D(-50, 50, -50, 50);

}
```

Q1. Boundary Fill Algorithm:

```
#include "util.hpp"

void Boundary(int x, int y){
    if( (screen[x][y] != BOUNDARY) && (screen[x][y] != INSIDE)){
        screen[x][y] = INSIDE;

        if(x+1 < WIDTH) Boundary(x+1, y);
        if(y+1 < HEIGHT) Boundary(x, y+1);

        if(x > 0) Boundary(x-1, y);
        if(y > 0) Boundary(x, y-1);
    }
}
```

```
int main(int argc, char** argv){
    initPolygon();
    Boundary(xc, yc);

    // GLute init and create window
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(WIDTH,HEIGHT);
    glutInitWindowPosition(600,100);
    glutCreateWindow("Boundary Fill");
    myInit();

    // Register display callback
    glutDisplayFunc(drawObject);

    glutMainLoop();
}
```

Q2. Flood Fill:

```
#include "util.hpp"

void FloodFill(int x, int y){
    if( screen[x][y] == BG ){
        screen[x][y] = INSIDE;

        if(x+1 < WIDTH) FloodFill(x+1, y);
        if(y+1 < HEIGHT) FloodFill(x, y+1);

        if(x > 0) FloodFill(x-1, y);
        if(y > 0) FloodFill(x, y-1);
    }
}


int main(int argc, char** argv){
    initPolygon();
    FloodFill(xc, yc);

    // GLute init and create window
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(WIDTH,HEIGHT);
    glutInitWindowPosition(600,100);
    glutCreateWindow("Flood Fill");
    myInit();

    // Register display callback
    glutDisplayFunc(drawObject);
    glutMainLoop();
}
```

Q3. Scan line Seed fill -

```
#include "util.hpp"
#include <stack>

void pushUnfilledRight(int xl, int xr, int y, stack<pii> &seed){
    for(bool span=0;xr>=xl;xr--){
        if((screen[xr][y] != BOUNDARY) && (screen[xr][y] != INSIDE)){
            if(!span){
                seed.push({xr,y});
                span = 1;
            }
        } else {
            span = 0;
        }
    }
}

void scanLineSeedFill(int x, int y){
    stack<pii> seeds;
    seeds.push({x,y});

    while(!seeds.empty()){
        pii p = seeds.top();
        seeds.pop();

        x = p.first;
        y = p.second;

        // cout<<"Seed: ("<<x<<","<<y<<")\n";
        if(screen[x][y] == INSIDE){
            // Already painted by some other seed.
            continue;
        }
        screen[x][y] = INSIDE;

        int xr,xl;

        // filling left
        for(xl=x-1;xl>=0;xl--){
            if((screen[xl][y] == BOUNDARY)) break;

            screen[xl][y] = INSIDE;
        } xl++;

        // filling left
        for(xr=x+1;xr<WIDTH;xr++){
            if((screen[xr][y] == BOUNDARY)) break;

            screen[xr][y] = INSIDE;
        } xr--;

        // cout<<"xL: "<<xl - WC<<", xR: "<<xr - WC<<", y: "<<y<<"\n";

        if(y+1 < HEIGHT) pushUnfilledRight(xl, xr, y+1, seeds);
        if(y > 0) pushUnfilledRight(xl, xr, y-1, seeds);
    }
}

int main(int argc, char** argv){
    initPolygon();
    scanLineSeedFill(xc, yc);

    // GLute init and create window
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(WIDTH,HEIGHT);
    glutInitWindowPosition(600,100);
    glutCreateWindow("Scan Line Seed Fill");
    myInit();

    // Register display callback
    glutDisplayFunc(drawObject);

    glutMainLoop();
}
```