Boundary Fill and Flood Fill Implementation

Trisha Shah

SE CMPN A 20102A0004

**Theory:**

Boundary Fill Algorithm:

Boundary Fill Algorithm starts at a pixel inside the polygon to be filled and paints the interior proceeding outwards towards the boundary. This algorithm works only if the color with which the region has to be filled and the color of the boundary of the region are different. If the boundary is of one single color, this approach proceeds outwards pixel by pixel until it hits the boundary of the region.

Flood Fill Algorithm:

Flood fill is an algorithm mainly used to determine a bounded area connected to a given node in a multi-dimensional array. It is a close resemblance to the bucket tool in paint programs.

In this method, a point or seed which is inside region is selected. This point is called a seed point. Then four connected approaches or eight connected approaches is used to fill with specified color.

The most approached implementation of the algorithm is a stack-based recursive function.

The flood fill algorithm has many characters similar to boundary fill. But this method is more suitable for filling multiple colors boundary. When boundary is of many colors and interior is to be filled with one color we use this algorithm.

**Algorithm:**

**Boundary Fill:**

1. There are two defined colours: colour of boundary (color\_boundary) and colour that needs

to be filled (colour fill)

2. Get colour (say color1) of the current pixel

3. If color1 is equal to color\_boundary or colour fill, nothing needs to be done as correct colour

is already assigned.

4. If color1 is not equal to the two values:

4.1. Add colour to the current pixel.

4.2. Do the same process for 4 adjacent pixel points: (x, y-1), (x+1, y), (x, y+1), (x-1, y) 4.3.

If 8-connected fill is being done, do the same process for 4 diagonal pixel points

additionally: (x+1, y-1), (x+1, y+1), (x-1, y+1), (x-1, y-1).

5. Do the process for every pixel point.

**Flood Fill:**

1. Take the position of the starting point.

2. Decide whether you want to go in 4 directions (N, S, W, E) or 8 directions (N, S, W, E, NW,

NE, SW, SE).

3. Choose a replacement colour and a target colour.

4. Travel in those directions.

5. If the tile you land on is a target, replace it with the chosen colour.

6. Repeat 4 and 5 until you’ve been everywhere within the boundaries.

**Implementation:**

**Boundary Fill:**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<math.h>

#include<dos.h>

void fill\_right(x,y)int x , y ;

{

if((getpixel(x,y) != WHITE)&&(getpixel(x,y) != RED))

{

putpixel(x,y,RED);

fill\_right(++x,y);

x = x - 1 ;

fill\_right(x,y-1);

fill\_right(x,y+1);

}

}

void fill\_left(x,y)int x , y ;

{

if((getpixel(x,y) != WHITE)&&(getpixel(x,y) != RED))

{

putpixel(x,y,RED);

fill\_left(--x,y);

x = x + 1 ;

fill\_left(x,y-1);

fill\_left(x,y+1);

}

}

void main()

{int x , y ,a[10][10];

int gd, gm ,n,i;

detectgraph(&gd,&gm);

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

printf("\n\n\tEnter the no. of edges of polygon : ");

scanf("%d",&n);

printf("\n\n\tEnter the cordinates of polygon :\n\n\n ");

for(i=0;i<n;i++)

{printf("\tX%d Y%d : ",i,i);

scanf("%d %d",&a[i][0],&a[i][1]);

}

a[n][0]=a[0][0];

a[n][1]=a[0][1];

printf("\n\n\tEnter the seed pt. : ");

scanf("%d%d",&x,&y);

cleardevice();

setcolor(WHITE);

for(i=0;i<n;i++)

{ line(a[i][0],a[i][1],a[i+1][0],a[i+1][1]);

}

fill\_right(x,y);

fill\_left(x-1,y);

getch();

}

Flood Fill:

#include<stdio.h>

#include<graphics.h>

void flood\_fill(int x,int y,int new\_col,int old\_col)

{

if(getpixel(x,y)==old\_col)

{

putpixel(x,y,new\_col);

flood\_fill(x+1,y,new\_col,old\_col);

flood\_fill(x-1,y,new\_col,old\_col);

flood\_fill(x,y+1,new\_col,old\_col);

flood\_fill(x,y-1,new\_col,old\_col);

}

}

void main()

{

int gdriver,gmode;

detectgraph(&gdriver,&gmode);

initgraph(&gdriver,&gmode, "C:\\TURBOC3\\BGI");

circle(200,150,35);

circle(100,150,35);

arc(150,150,45,135,35);

arc(150,150,225,315,35);

arc(150,115,0,180,50);

arc(150,185,180,360,50);

flood\_fill(200,150,BLUE,getpixel(200,150));

flood\_fill(150,150,GREEN,getpixel(150,150));

flood\_fill(80,150,BLUE,getpixel(80,150));

flood\_fill(150,100,MAGENTA,getpixel(150,100));

flood\_fill(150,200,MAGENTA,getpixel(150,200));

getch();

closegraph();

}



