CHAPTER 4

FILLING ALGORITHMS

CONTENTS

4.0 OBJECTIVES:

After reading this chapter you will be able to:

- Discuss about region filling.
- Understand Boundary fill algorithm
- Understand Flood fill algorithm
- Know the techniques of character display

4.1 INTRODUCTION:

Filling is one of the fundamental parts of computer graphics. There are several cases where filling plays an important role. For example, the interior of a certain object like circles, rectangles are to be filled often. The interior of a character needs to be filled when outlined fonts are used. Fonts are one of the most important graphic elements in computer world. How fonts can be generated is one of the major issues of computer science. In this chapter we shall discuss different aspects of filling algorithms and some techniques that are used to generate fonts. The chapter ends with some of the issues of the font generation techniques.

4.2 REGION FILLING:

Region filling is the process of coloring of a well defined image area or region by a given color. There are two types of region in terms of pixels. Boundary-defined region and interior defined region.

Boundary-Defined Region: If a region is outlined by the pixels, i.e. only the out line of the region is defined then it is called boundary-defined region.

Interior-Defined Region: A region whose total internal area is filled with pixel colours is called an interior defined region.

Depending upon the type of the region, there are two types of filling algorithms boundary fill algorithm and flood fill algorithm. Boundary fill algorithm is used in the case of boundary defined regions and flood fill is used in the case of interior-defined region.

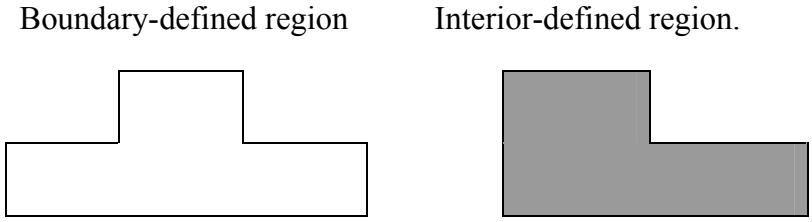

Fig-4.1 Boundary Defined Region and Interior Defined Region

If the boundary is specified in a single colour, the filling process starts from an interior part and proceeds outwards pixel by pixel unless on the boundary colour is

encountered. In case of an area defined by multiple colors boundary, the approach will be different. In this case then we can fill the area by replacing the specified interior color by the filling color instead of searching the specific boundary colour. This area is called interior specific region and the algorithm for filling this region in called flood fill algorithm.

4.3 4-CONNECTED AND 8-CONNECTED:

There are two ways in which pixels are considered to be connected to one another, 4-connected and 8-connected. In 4-connected method a pixel may have upto 4 neighbor.

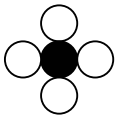

Fig-4.2 Four Connected Region

These are up, down, left and right. If the current pixel is (x, y) the neighbors are (x, y +1), (x+1, y), (x-1, y), (x, y–1).

In 8-connected method, the pixel may have up to 8 neighbors.

Fig-4.3: Eight Connected Region

These are up, down, left, right and four corner points, i.e., of the current point is (x, y) the neighbors are (x, y-1),(x, y+1), (x+1, y), (x-1, y), (x-1, y-1), (x-1, y+1), (x+1, y+1) and (x+1, y-1).

There are problems with both 4-connected and 8-connected methods. Consider the following region:

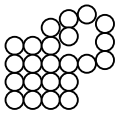

Fig-4.4 Problem-1 with 4 and 8 connected regions

The region will be partially filled up with interior fill color. That will be correctly filled up with 8-connected method. But that will be a problem for a figure like the following.

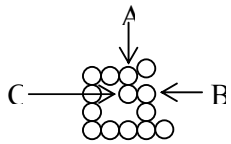

Fig-4.5: Problem-2 with 4 and 8 Connected Regions

In this case A and B are the interior part of C. So the region will be filled up correctly, in case of 4-connected method but in case of 8-connected method it will come out of the region and will fill the outer region also.

4.4 BOUNDARY FILL ALGORITHM:

In case of boundary fill the recursive algorithm takes an initial interior point (x, y) for the procedure to start. The procedure tests the neighborhood points. If a neighborhood is a boundary pixel (From the pixel color it can be tested) then we don't color that pixel otherwise the pixels are colored with the fill colour and any one of the neighborhood pixel is the chosen to continue the same process again and again. The procedure stops when all pixels up to boundary are tested. The algorithm is given below.

4.4.1 THE ALGORITHM:

```
PROCEDURE boundaryFill (int  x,int  y,int  fillcolor, int  boundarycolor)
  {
    int currentcolour = getpixel(x, y)
    If (current colour = boundary colour and current color = fill colour) Then
            putpixel ( x, y fill colour)
            boundaryFill ( x+1, y, fillcolor, boundarycolor)
            boundaryFill ( x-1, y,  fillcolor, boundarycolor)
            boundaryFill ( x, y+1, fillcolor, boundarycolor)
            boundaryFill ( x, y-1,  fillcolor, boundarycolor)
   End If
}
```

4.4.2 THE BOUNDARY-FILL PROGRAM:

```
#include<graphics.h>
#include<conio.h>
/* The Boundary-Fill Function */
void boundaryFill(int x,int y,int fillcol,int boundarycol)
  {
   int currcol;
   currcol=getpixel(x,y);
   if(currcol!=boundarycol && currcol!=fillcol)
     {
          putpixel(x,y,fillcol);
          boundaryFill(x+1,y,fillcol,boundarycol);
          boundaryFill(x-1,y,fillcol,boundarycol);
          boundaryFill(x,y+1,fillcol,boundarycol);
          boundaryFill(x,y-1,fillcol,boundarycol);
     }
  }
/* The main program */
int main()
  {
   int gd=DETECT,gm;
   initgraph(&gd,&gm,"F:\\tc\\bgi");   /* Assuming the graphics  */
   setcolor(GREEN);                    /* files are at F:\tc\bgi location */
   circle(250,200,40);
   getch();
   boundaryFill(260,210,YELLOW,GREEN);
   getch();
   closegraph();
   restorecrtmode();
   return(0);
  }
/* End of the program*/
```

4.4.3 DISADVANTAGES:

The boundary fill algorithm may not fill the region correctly of some interior pixels had already been pointed in the fill color. The algorithm, stated above, fills the region in horizontal spans. If any point is encountered with the fill colour, that recursive branch stops there, and a part of the region may not be filled. To avoid this, one may change the color of any interior point if it is set to fill color before applying this process. Another drawback of this algorithm is its extensive recursive function calls and therefore stacking of neighboring paints.

4.5 FLOOD FILL ALGORITHM:

When the boundary of a region is painted with several colors, we cannot apply boundary fill algorithm. In this case we shall apply flood fill algorithm. In case of flood fill algorithm, the specified interior colour is replaced by the fill colour.

To start with the algorithm, we take a starting interior point. Then we check if it has the old interior color or not and if yes then we replace the old color by fill color and consider all its neighborhood points for the same procedure.

The algorithm is given below.

4.5.1 THE ALGORITHM:

```
PROCEDURE  floodFill (int  x, int  y, int  oldcol, int  fillcol)
  {
     If (getpixel( x, y) = oldcol) Then
         putpixel( x,y, fillcol)
         floodFill ( x+1, y, oldcol, fillcol)
         floodFill ( x-1, y, oldcol, fillcol)
         floodFill ( x, y+1, oldcol, fillcol)
         floodFill ( x, y-1, oldcol. fillcol)
     End If
}
```

4.5.2 THE FLOOD-FILL PROGRAM:

```
#include<graphics.h>
#include<conio.h>
/* The Flood-Fill Function */
void floodFill(int x,int y,int oldcol,int fillcol)
  {
    int currcol;
    currcol=getpixel(x,y);
    if(currcol==oldcol)
      {
           putpixel(x,y,fillcol);
           floodFill(x+1,y,oldcol,fillcol);
           floodFill(x-1,y,oldcol,fillcol);
           floodFill(x,y+1,oldcol,fillcol);
           floodFill(x,y-1,oldcol,fillcol);
      }
  }
/* The main program */
int main()
  {
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"F:\\tc\\bgi");   /* Assuming the graphics  */
    setcolor(GREEN);                    /* files are at F:\tc\bgi location */
    rectangle(250,200,300,250);
    getch();
    floodFill(260,210,BLACK,YELLOW);
    getch();
    closegraph();
    restorecrtmode();
    return(0);
  }
```

In this algorithm, the extensive recursive call is a disadvantage.

4.6 CHARACTER DISPLAY:

Letters, numbers and special character display is one of the fundamental parts of computer graphics. The design style of a character set is called typeface or font. There are hundreds of different fonts available in any general purpose graphics software or word processing software. There size may be different like 10 point, 12 point etc. and these types may also be different like Bold, Italic, Bold italic etc.

There are mainly two types of fonts. One is serif and the other is sans serif. In serif, small lines at the end of the character are there and for that large block of texts are more readable written in serif font. The sans serif fonts on the other hand have no end lines so they are less readable but individual sans serif characters are quickly recognizable. For this reason, sans serifs are good for figure labeling.

Internal storage structure wise fonts can be divided into two categories, bitmap fonts and out lined fonts.

4.6.1 BITMAP FONTS:

In bitmapped type, a rectangular grid pattern is used. In this grid the type face or font is set by making the particular pixels on. As this is a bi-level pixel grid pattern, the on (or 1) pixels will cause the font to be displayed.

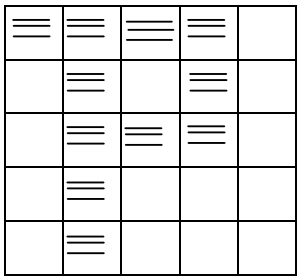

Fig-4.6 Bitmap font P

The representation of bitmap font is very simple and as it is stored in grid pattern. The font can be directly mapped into the image space without any conversion. Though it is simple for representation, it has some disadvantages also. It requires more memory to store each font type and it requires different memory grid for different types, sizes and styles i.e., for 10 points, one bold, one normal,

one italic, one bold italic style is needed so there is a requirement of four memory grids. Same thing is needed for 12 points, 14 points etc. Though it is possible to generate one type from other (such as italic from normal) but that does not produce a good result.

Another major point is the size of the bitmap image depends on the screen resolution. If the screen resolution is 72 pixels/inch, a 12 pixel high will produce a longer character than a pattern displayed on a screen with 96 pixel/ inch.

4.6.2 OUT LINED FONT:

Out lined font (also known as vector font) is the other type of font. In this case, the geometrical primitives such as lines, arcs, curves are used to draw the out line of each character. After generating the outline, the interior of character is then filled by the fill colour by using any filling algorithm (generally scan line algorithm). On the advantage part, it takes less memory for storing the character. Not only that, here font size and type can be generated from the basic definition. For example size can be changed by sealing; italic font can be obtained through shearing transformation. Not only that, rotation of a character is also possible by using rotation transformation. All these we can do as per our desire which is not possible in case of bitmap font.

The only disadvantage of this type of font is that it takes longer time to produce a character because in this method, the scan conversion of performed and then filling of the interior is also performed.

4.7 ALIASING EFFECTS:

We know that the lines, geometric shapes, colour boundaries are continuous where as a raster device is a discrete one. For this reason, the picture is distorted. This distortion is due to the under sampling. There are various types of distortions. These distortions are collectively called aliasing.

Unequal Brightness: A slanted line appears to be less bright than a horizontal or vertical line. This is because the slanted pixels are far apart than the horizontal or vertical pixels.

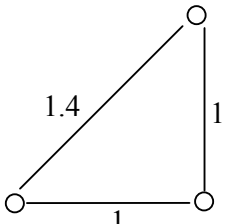

Fig-4.7: Cause of Aliasing Effects

So, though all pixels are in some intensity level, the slanted lines look dimmer.

Mapping of Periodic Figures: In this case, we may also have under sampling as shown in the figure. This problem can be solved. In case of periodic information, we need the sampling frequency at least twice the highest frequency of the object. This is called Nyquist frequency.

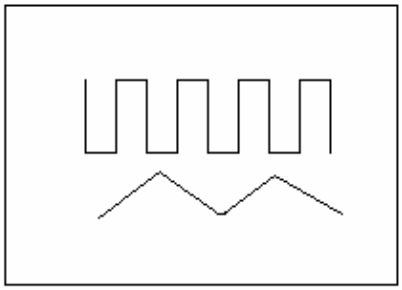

Fig-4.8: Effect of Under Sampling

Antialiasing: Aliasing effects are tolerable or negligible in most of the static pictures but it may be significant in case of moving pictures or animations. A rotating object with small surface may loose features or even may disappear during rotation.

Fundamentally there are two methods of antialiasing. The first is to increase the screen resolution and the second one is to treat the pixel as a finite area rather than a point.

The first method is straight forward but there is a problem of the finiteness of the screen resolution. We know that the screen resolution has a limit and we cannot go beyond that. So, even the highest resolution is considered, aliasing may take place because to present correct objects we need arbitrary small sampling intervals.

4.8 KEY WORDS:

- 4-Connected
- 8-Connected
- Antialiasing
- Bitmap font
- Boundary fill
- Flood fill
- Outlined font

4.9 SAMPLE QUESTIONS:

4.9.1 What is filling?

4.9.2 What do you mean by 4-connected and 8-Connected

4.9.3 What do you mean by boundary defined region and interior defined region?

4.9.4 Describe the boundary-fill algorithm. What are the disadvantages of boundary-fill algorithm?

4.9.5 Describe the flood-fill algorithm also discuss its disadvantages.

4.9.6 Write down the function code for boundary-fill algorithm.

4.9.7 Write down the code for flood-fill algorithm.

4.9.8 Briefly discuss bitmap fonts and outlined fonts.

4.9.9 What is Aliasing? How can it be removed?

4.9.10 What is Nyquist frequency?