name Pranav Narayan Pisal
roll no 63 se 2

Aim: To implement Area Filling Algorithm: Boundary Fill, Flood Fill.

Objective:
Polygon is an ordered list of vertices as shown in the following figure. For filling polygons with particular colors, we need to determine the pixels falling on the border of the polygon and those which fall inside the polygon. Objective is to demonstrate the procedure for filling polygons using different techniques.

Theory:
1) Boundary Fill algorithm –
Start at a point inside a region and paint the interior outward toward the boundary. If the boundary is specified in a single color, the fill algorithm processed outward pixel by pixel until the boundary color is encountered. A boundary -fill procedure accepts as input the coordinate of the interior point (x, y), a fill color, and a boundary color.

Procedure:

```
boundary_fill (x, y, f_color, b_color)
{
if (getpixel (x, y) != b_colour && getpixel (x, y) != f_colour)
 {
 putpixel (x, y, f_colour)
 boundary_fill (x + 1, y, f_colour, b_colour);
 boundary_fill (x, y + 1, f_colour, b_colour);
 boundary_fill (x - 1, y, f_colour, b_colour);
 boundary_fill (x, y - 1, f_colour, b_colour);
 }
}
```

Program:

1.program 2(boundary filll)

```
#include<conio.h>
#include<stdio.h>
#include<graphics.h>
void bf (int x,int y,int fcolor,int bcolor);
void main()
{ int gd=DETECT,gm;
initgraph(&gd,&gm,"");
rectangle(50,50,100,100);
bf( 70,70,10,18);
getch();
closegraph();
}
void bf(int x,int y,int fcolor,int bcolor)
{
 int ccolor=getpixel(x,y);
if(ccolor!=fcolor && ccolor!=bcolor)
{putpixel(x,y,fcolor);
bf(x+1,y,fcolor,bcolor);
bf(x-1,y,fcolor,bcolor);
bf(x,y+1,fcolor,bcolor);
bf(x,y-1,fcolor,bcolor);
}}
```

Output

2) Flood Fill algorithm –

Sometimes we want to fill an area that is not defined within a single color boundary. We paint such areas by replacing a specified interior color instead of searching for a boundary color value. This approach is called a flood-fill algorithm.

1. We start from a specified interior pixel (x, y) and reassign all pixel values that are currently set to a given interior color with the desired fill color.
2. If the area has more than one interior color, we can first reassign pixel values so that all interior pixels have the same color.
3. Using either 4-connected or 8-connected approach, we then step through pixel positions until all interior pixels have been repainted.

Procedure -

```
flood_fill (x, y, old_color, new_color)
{

if (getpixel (x, y) = old_colour)
 {
 putpixel (x, y, new_colour);
 flood_fill (x + 1, y, old_colour, new_colour);
 flood_fill (x - 1, y, old_colour, new_colour);
 flood_fill (x, y + 1, old_colour, new_colour);
 flood_fill (x, y - 1, old_colour, new_colour);
 flood_fill (x + 1, y + 1, old_colour, new_colour);
 flood_fill (x - 1, y - 1, old_colour, new_colour);
 flood_fill (x + 1, y - 1, old_colour, new_colour);
 flood_fill (x - 1, y + 1, old_colour, new_colour);
 }
}
```

Program:

2nd program (flood fill)

```
#include<conio.h>
#include<stdio.h>
#include<graphics.h>
void ff (int x,int y,int ocolor,int ncolor);
void main()
{ int gd=DETECT,gm;
initgraph(&gd,&gm,"");
rectangle(50,50,100,100);
ff( 70,70,0,15);
getch();
closegraph();
}
void ff(int x,int y,int ocolor,int ncolor)
{
if(getpixel(x,y)==ocolor)
{putpixel(x,y,ncolor);
delay(5);
ff(x+1,y,ocolor,ncolor);
ff(x-1,y,ocolor,ncolor);
ff(x,y+1,ocolor,ncolor);
ff(x,y-1,ocolor,ncolor);
}}
```

Output

conclusion

Flood fill is an algorithm that can be used to fill a region of pixels with a different color in a 2D array, such as an image. It can be useful for various applications, such as coloring maps, editing images, or creating games. However, it also has some limitations and challenges, such as:

It can be very slow if the region to be filled is large or complex, since it requires visiting many pixels recursively or iteratively.
It may fail or produce unexpected results if the region is not well-defined or has holes or gaps in the boundary.
It may cause stack overflow if the recursion depth is too high or the stack size is too small.
It may require more knowledge about the surrounding pixels, such as the old color and the new color, to avoid overwriting or skipping some pixels.
Therefore, flood fill is an important and interesting algorithm, but it also needs to be implemented carefully and efficiently. There are different variations and optimizations of flood fill, such as scanline fill, boundary fill, or fast flood fill1234, that can improve its performance and accuracy.