Area Filling Algorithm: Boundary Fill, Flood Fill

Algorithm:

Input: Specify the seed point (x, y) inside the area to be filled.

Define the fill color or pattern.

Define the boundary color (the color that marks the boundary of the area to be filled).

Boundary Color Check:

Check the color of the pixel at the seed point (x, y).

If it is equal to the boundary color, exit (nothing to fill).

Filling Process:

Change the color of the pixel at the seed point to the fill color.

Recursively or iteratively perform the following for neighboring pixels:

Check the color of the pixel to the north (x, y - 1).

Check the color of the pixel to the south (x, y + 1).

Check the color of the pixel to the west (x - 1, y).

Check the color of the pixel to the east (x + 1, y).

If the color of the neighboring pixel matches neither the fill color nor the boundary color, change its color to the fill color and apply the fill algorithm recursively or iteratively to that pixel.

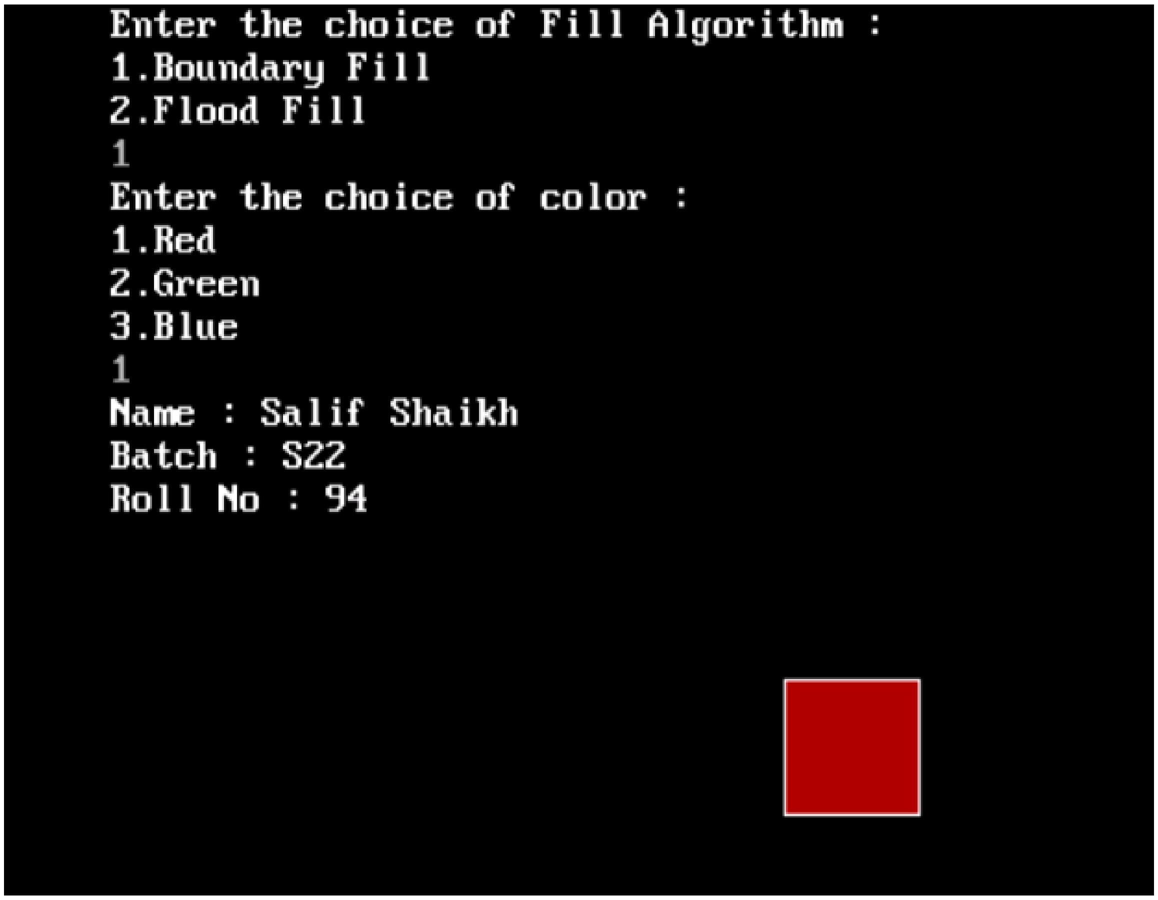
Code:

```
#include<stdio.h>

#include<graphics.h>

void boundaryFill4(int x, int y, int fillColor, int bColor){

 if(getpixel(x,y)!=bColor && getpixel(x,y)!=fillColor){

delay(1);

putpixel(x, y, fillColor);

boundaryFill4(x+1, y, fillColor, bColor);

boundaryFill4(x, y+1, fillColor, bColor);

boundaryFill4(x-1, y, fillColor, bColor);

boundaryFill4(x, y-1, fillColor, bColor);

 }

}

void floodFill4(int x, int y, int newColor, int oldColor){

 if(getpixel(x,y) == oldColor){

delay(1);

putpixel(x, y, newColor);

floodFill4(x+1, y, newColor, oldColor);

floodFill4(x, y+1, newColor, oldColor);

floodFill4(x-1, y, newColor, oldColor);

floodFill4(x, y-1, newColor, oldColor);

 }

}
```

```
int main()


{

 int choice,cochoice;

 int gdriver=DETECT, gmode, xc, yc, radius;

 initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");

 printf("Enter the choice of Fill Algorithm : \n");

 printf("1.Boundary Fill \n2.Flood Fill \n");

 scanf("%d",&choice);

 printf("Enter the choice of color : \n");

 printf("1.Red \n2.Green \n3.Blue\n");

 scanf("%d",&cochoice);

 printf("Name : Salif Shaikh \nBatch : S22 \nRoll No : 94\n");

 switch(choice)

 {

case 1 :

 switch(cochoice)

 {

case 1 :

 rectangle(250, 250, 300, 300);

 boundaryFill4(275, 275, 4, 15);

 break;

case 2 :
 rectangle(250, 250, 300, 300);

 boundaryFill4(275, 275, 2, 15);

 break;

case 3 :


 rectangle(250, 250, 300, 300);

 boundaryFill4(275, 275, 7, 15);

 break;

 }

 break;

case 2 :

 switch(cochoice)

 {

case 1 :

printf("\nFlood Fill Algorithm");

 rectangle(250, 250, 300, 300);

floodFill4(275, 275, 4, 0);

break;

case 2 :

printf("\nFlood Fill Algorithm");

 rectangle(250, 250, 300, 300);

floodFill4(275, 275, 2, 0);

break;
```

```
case 3 :
printf("\nFlood Fill Algorithm");
 rectangle(250, 250, 300, 300);
floodFill4(275, 275, 7, 0);
break;
 }
 break;

 }
 delay(500);
 closegraph();
 return 0;
}
```

Output:



```
Enter the choice of Fill Algorithm :
1.Boundary Fill
2.Flood Fill
1
Enter the choice of color :
1.Red
2.Green
3.Blue
1
Name : Salif Shaikh
Batch : S22
Roll No : 94
```

Conclusion: In conclusion, the Boundary Fill Algorithm, also known as the Flood Fill Algorithm, is a fundamental technique in computer graphics for filling closed regions or polygons with a specified color or pattern. It offers simplicity and effectiveness in many applications, such as coloring, masking, and area filling.