**Practical No. 6**

***Title:-*** Implement Flood fill algorithm for Polygon filling.

***Course outcome*** :- Apply the algorithms to draw lines, circle and polygons.

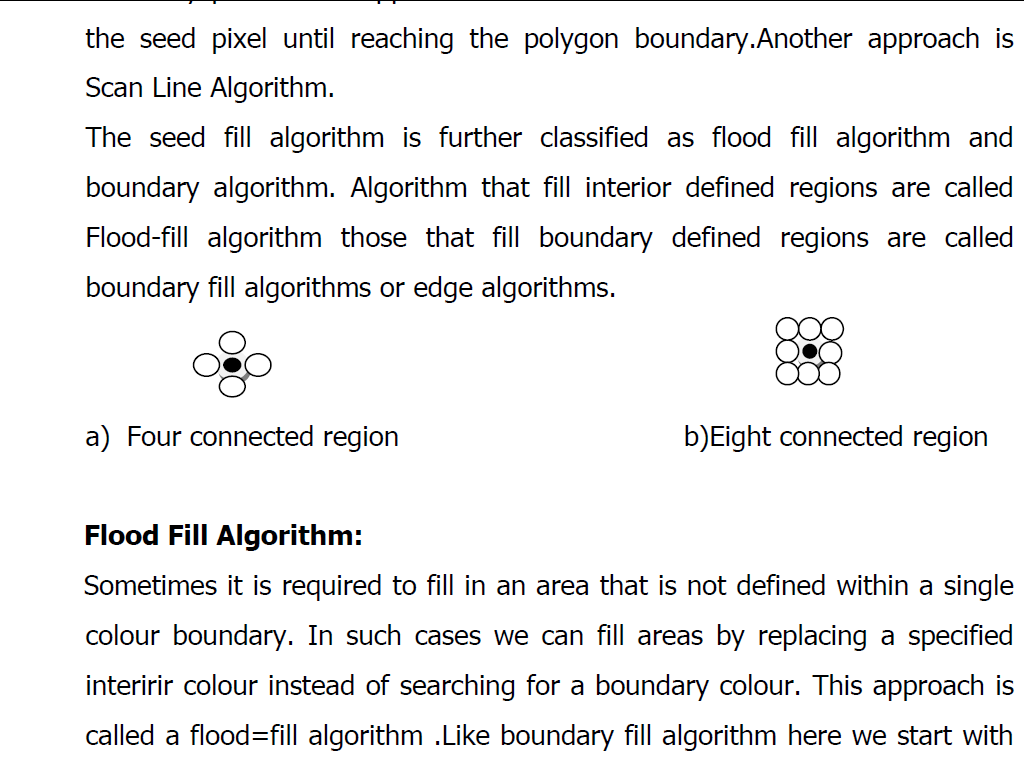
***Resources Required (Hardware & Softwares):-***

1. A Desktop PC/ Laptop
2. Ansi C/ Turbo C/ (Any distribution) installed

***Theory:-***

Filling the polygon means highlighting all the pixels which lie inside the polygon with any colour other than background colour. There are two basic approaches used to fill the polygon. One way to fill a polygon is to start from a given seed point known to be inside the polygon and highlight outward from this point i.e. neighboring pixels until we encounter the boundary pixels. This approach called seed fill because colour flows from the seed pixel until reaching the polygon boundary.Another approach is Scan Line Algorithm.

The seed fill algorithm is further classified as flood fill algorithm and boundary algorithm. Algorithm that fill interior defined regions are called Flood-fill algorithm those that fill boundary defined regions are called boundary fill algorithms or edge algorithms.

******

a) Four connected region b)Eight connected region

***Flood Fill Algorithm***

Sometimes it is required to fill in an area that is not defined within a single colour boundary. In such cases we can fill areas by replacing a specified interred colour instead of searching for a boundary colour. This approach is called a flood=fill algorithm .Like boundary fill algorithm here we start with some seed and examine the neighbouring pixels. However here pixels are checked for a specified interior colour instead of boundary colour and they are replaced by new colour. Using either a 4-connected or 8-connected approach, we can step through pixel positions until all interior point have been filled.

***Algorithm:-***

1. **4 -connected Flood Fill Algorithm:-**

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

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

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

}

}

1. **8-connected Flood Fill Algorithm:-**

flood\_fill(x,y,old\_colour,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 for Flood Fill Algorithm**

#include<stdio.h>

#include<dos.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

#include<stdlib.h>

void flood\_fill(int ,int ,int ,int );

void main()

{

int gm=DETECT,gd;

clrscr();

detectgraph(&gd,&gm);

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

rectangle(50,50,100,100);

flood\_fill(51,51,0,15);

getch();

closegraph();

}

void flood\_fill(int x,int y,int oc,int nc)

{

if(getpixel(x,y)==oc)

{

putpixel(x,y,nc);

flood\_fill(x+1,y,oc,nc);

flood\_fill(x-1,y,oc,nc);

flood\_fill(x,y+1,oc,nc);

flood\_fill(x,y-1,oc,nc);

flood\_fill(x+1,y+1,oc,nc);

flood\_fill(x-1,y-1,oc,nc);

flood\_fill(x+1,y-1,oc,nc);

flood\_fill(x-1,y+1,oc,nc);

}

}

**Output:- ( Paste your own Output )**

***Conclusion:-***

Thus, we have implemented **Flood Fill Algorithm** for Polygon Filling.