**Ashesh Kumar**

**501254**

**I.T. – 5**

**Q. Implement Seed Fill algorithms.**

1. **Boundary fill**
2. **Flood fill**

**Boundary\_fill.c**

#include<stdio.h>

#include<graphics.h>

/\* Defining the structure to store edges

-----------------------------------------\*/

struct edge

{

int x1;

int y1;

int x2;

int y2;

int flag;

};

void main()

{

int gd = DETECT, gm, n, i, j, k;

struct edge ed[10],temped;

float dx,dy,m[10],x\_int[10],inter\_x[10];

int x[10],y[10],ymax = 0, ymin = 480, yy,temp;

initgraph (&gd, &gm, "C:\\Turboc3\\BGI");

/\* Read the number of vertices of the polygon

------------------------------------------------ \*/

printf("Enter the number vertices of the graph: ");

scanf("%d", &n);

/\* Read the vertices of the polygon and also find Ymax and Ymin

--------------------------------------------------------------- \*/

printf("Enter the vertices: \n");

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

{

printf("x[%d] : ", i);

scanf("%d", &x[i]);

printf("y[%d] : ", i);

scanf("%d", &y[i]);

if(y[i] > ymax)

ymax = y[i];

if(y[i] < ymin)

ymin = y[i];

ed[i].x1 = x[i];

ed[i].y1 = y[i];

}

/\* Store the edge information

-------------------------------\*/

for(i=0;i<n-1;i++)

{

ed[i].x2 = ed[i+1].x1;

ed[i].y2 = ed[i+1].y1;

ed[i].flag=0;

}

ed[i].x2 = ed[0].x1;

ed[i].y2 = ed[0].y1;

ed[i].flag=0;

/\* Check for y1>y2, if not interchange y1 and y2

with corresponding x1 and x2 ---------------\*/

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

{

if(ed[i].y1 < ed[i].y2)

{

temp = ed[i].x1;

ed[i].x1=ed[i].x2;

ed[i].x2=temp;

temp = ed[i].y1;

ed[i].y1=ed[i].y2;

ed[i].y2=temp;

}

}

/\* Draw the polygon

---------------------- \*/

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

{

line(ed[i].x1,ed[i].y1,ed[i].x2,ed[i].y2);

}

/\* sorting of edges in the order of y1,y2,x1

--------------------------------------------- \*/

for(i=0;i<n-1;i++)

{

for(j=0;j<n-1;j++)

{

if(ed[j].y1<ed[j+1].y1)

{

temped = ed[j];

ed[j]=ed[j+1];

ed[j+1] = temped;

}

if(ed[j].y1==ed[j+1].y1)

{

if(ed[j].y2<ed[j+1].y2)

{

temped = ed[j];

ed[j]=ed[j+1];

ed[j+1] = temped;

}

if(ed[j].y2==ed[j+1].y2)

{

if(ed[j].x1<ed[j+1].x1)

{

temped = ed[j];

ed[j]=ed[j+1];

ed[j+1] = temped;

}

}

}

}

}

/\* calculating 1/slope of each edge and storing top x

co-ordinate of the edge --------------------------- \*/

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

{

dx = ed[i].x2 - ed[i].x1;

dy = ed[i].y2 - ed[i].y1;

if(dy==0)

{

m[i]=0;

}

else

{

m[i] = dx/dy;

}

inter\_x[i]= ed[i].x1;

}

yy=ymax;

while(yy>ymin)

{

/\* Marking active edges

-------------------------- \*/

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

{

if(yy > ed[i].y2 && yy <= ed[i].y1)

{

ed[i].flag = 1;

}

else

{

ed[i].flag = 0;

}

}

/\* Finding the x intersections

-------------------------------- \*/

j=0;

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

{

if(ed[i].flag==1)

{

if(yy==ed[i].y1)

{

x\_int[j]=ed[i].x1;

j++;

if(ed[i-1].y1==yy && ed[i-1].y1<yy)

{

x\_int[j]=ed[i].x1;

j++;

}

if(ed[i+1].y1==yy && ed[i+1].y1<yy)

{

x\_int[j]=ed[i].x1;

j++;

}

}

else

{

x\_int[j] = inter\_x[i]+(-m[i]);

inter\_x[i]=x\_int[j];

j++;

}

}

}

/\* Sorting the x intersections

--------------------------------\*/

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

{

for(k=0;k<j-1;k++)

{

if(x\_int[k]>x\_int[k+1])

{

temp =x\_int[k];

x\_int[k] = x\_int[k+1];

x\_int[k+1] = temp;

}

}

}

/\* Extracting pairs of x values to draw lines

---------------------------------------------- \*/

for(i=0;i<j;i+=2)

{

line(x\_int[i],yy,x\_int[i+1],yy);

}

yy--;

delay(1000);

}

sleep(200);

getch();

}

**flood\_fill.c**

#include<stdio.h>

#include<graphics.h>

void flood(int, int, int, int);

void main()

{

int gd,gm;

/\* Initialise graphics mode

---------------------------------- \*/

detectgraph(&gd,&gm);

initgraph(&gd,&gm,"C:\\Turboc3\\BGI");

rectangle(50,50,100,100);

flood(55,55,4,15);

getch();

closegraph();

}

void flood(seed\_x,seed\_y,foreground\_col,background\_col)

{

int i;

if(getpixel(seed\_x,seed\_y)!= background\_col && getpixel(seed\_x,seed\_y)!= foreground\_col)

{

putpixel(seed\_x,seed\_y,foreground\_col);

flood(seed\_x+1,seed\_y,foreground\_col,background\_col);

flood(seed\_x-1,seed\_y,foreground\_col,background\_col);

flood(seed\_x,seed\_y+1,foreground\_col,background\_col);

flood(seed\_x,seed\_y-1,foreground\_col,background\_col);

flood(seed\_x+1,seed\_y+1,foreground\_col,background\_col);

flood(seed\_x-1,seed\_y-1,foreground\_col,background\_col);

flood(seed\_x+1,seed\_y-1,foreground\_col,background\_col);

flood(seed\_x-1,seed\_y+1,foreground\_col,background\_col);

}

}