Experiment No : 8

Aim : To draw and colour animated boat using Boundaryfill and Floodfill Algorithm

Theory :
Boundaryfill
If the boundary of some region is specified in a single color, we can fill the interior of this region, pixel by pixel, until the boundary color is encountered. This method, called the boundary-fill algorithm, is employed in interactive painting packages, where interior points are easily selected. Using a graphics tablet or other interactive device, an artist or designer can sketch a figure outline, select a fill color from a color menu, specify the area boundary color, and pick an interior point. The figure interior is then painted in the fill color. Both inner and outer boundaries can be set up to define an area for boundary fill.

Floodfill
Sometimes we want to fill in (or recolor) an area that is not defined within a single color boundary. We can paint such areas by replacing a specified interior color instead of searching for a particular boundary color. This fill procedure is called a flood-fill algorithm. We start from a specified interior point (x, y) and reassign all pixel values that are currently set to a given interior color with the desired fill color. If the area that we want to paint has more than one interior color, we can first reassign pixel values so that all interior points have the same color. Using either a 4-connected or 8-connected approach, we then step through pixel positions until all interior points have been repainted

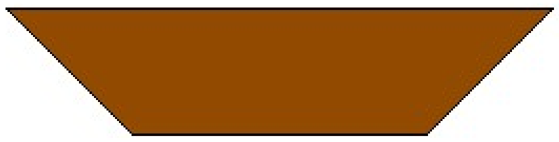
Algorithm
Boundaryfill
1. Add a pixel of the interior region on stack
2. If the stack is empty exit
3. Pop a pixel from stack
4. If color of pixel is not borderColor and not fillColor go to step 3
5. color the pixel to fillColor
6. push the 4 neighbours of the pixel on the stack.
7. Go to step 3

Floodfill
1. Add a pixel of the interior region on stack
2. If the stack is empty exit
3. Pop a pixel from stack
4. If color of pixel is interiorColor go to step 3
5. color the pixel to fillColor
6. push the 4 neighbours of the pixel on the stack.
7. Go to step 3

Code & Output :

```
#include <stdlib.h>
#include<graphics.h>
#include <math.h>
#include<iostream>
using namespace std;
void floodFill(int xi,int yi, int interiorColor, int fillColor){
if(getpixel(xi,yi)==interiorColor){
putpixel(xi,yi,fillColor);
floodFill(xi+1,yi,interiorColor,fillColor);
floodFill(xi-1,yi,interiorColor,fillColor);
floodFill(xi,yi+1,interiorColor,fillColor);
floodFill(xi,yi-1,interiorColor,fillColor);
}
}
void boundaryFill(int xi, int yi, int borderColor, int fillColor){
int color=getpixel(xi,yi);
if(color!=borderColor && color!=fillColor){
putpixel(xi,yi,fillColor);
boundaryFill(xi+1,yi,borderColor,fillColor);
boundaryFill(xi-1,yi,borderColor,fillColor);
boundaryFill(xi,yi+1,borderColor,fillColor);
boundaryFill(xi,yi-1,borderColor,fillColor);
}
}
int main(){
int gd=DETECT,gm;
initgraph(&gd,&gm,NULL);
setcolor(BLACK);
int x=100,y=100;
// Boat params
int slope=60,length=200,total=slope+length;
for(int i=0;i<5;i++){
setbkcolor(WHITE);
int boat[]={x,y, x+slope,y+slope, x+length,y+slope, x+total,y, x,y
};
drawpoly(5,boat);
floodFill(x+3,y+1,WHITE,BROWN);
//boundaryFill(x+3,y+1,BLACK,BROWN);
x+=15;
delay(500);
}
closegraph();
}
```

Output :



Conclusion : Program to color a boat using boundary and floodfill algorithms was successfully written and executed

Deepraj Bhosale
181105016