Experiment- 5

Aim

To implement the Flood Fill Algorithm and color a closed figure.

Description/Theory

Flood fill, also called seed fill, is an algorithm that determines the area connected to a given node in a multi-dimensional array. It is used in the "bucket" fill tool of paint programs to fill connected, similarly-colored areas with a different color, and in games such as Go and Minesweeper for determining which pieces are cleared.
The flood-fill algorithm takes three parameters: a start node, a target color, and a replacement color. The algorithm looks for all nodes in the array that are connected to the start node by a path of the target color and changes them to the replacement color. There are many ways in which the flood-fill algorithm can be structured, but they all make use of a queue or stack data structure, explicitly or implicitly.
Instead of relying on the boundary of the object, it relies on the fill color. In other words, it replaces the interior color of the object with the fill color. When no more pixels of the original interior color exist, the algorithm is completed.

Algorithm

Depending on whether we consider nodes touching at the corners connected or not, we have two variations: eight-way and four-way respectively.

Stack-based Recursive Implementation (Four-Connected)

One implicitly stack-based (recursive) flood-fill implementation (for a two-dimensional array) goes as follows:

Flood-fill (node, target-color, replacement-color):
1. If *target-color* is equal to *replacement-color*, return.
2. If the color of *node* is not equal to *target-color*, return.
3. Set the color of *node* to *replacement-color*.
4. Perform **Flood-fill** (one step to the south of *node*, *target-color*, *replacement-color*).
   Perform **Flood-fill** (one step to the north of *node*, *target-color*, *replacement-color*).
   Perform **Flood-fill** (one step to the west of *node*, *target-color*, *replacement-color*).
   Perform **Flood-fill** (one step to the east of *node*, *target-color*, *replacement-color*).
5. Return.

Code

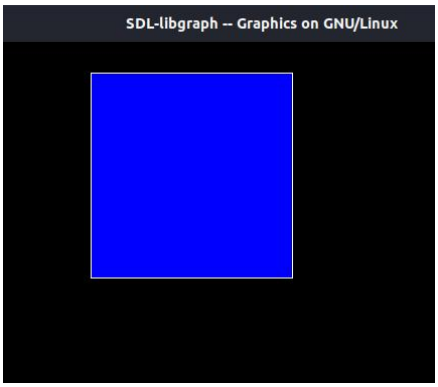
```
#include <graphics.h>
#include <stdio.h>

void flood(int x, int y, int new_col, int old_col)
{
   if (getpixel(x, y) == old_col) {
      putpixel(x, y, new_col);
      flood(x - 1, y, new_col, old_col);
      flood(x, y + 1, new_col, old_col);
      flood(x, y - 1, new_col, old_col);
      flood(x + 1, y, new_col, old_col);
   }
}
```

```
int main()
{
   int gd, gm = DETECT;
   initgraph(&gd, &gm, "");
   int top, left, bottom, right;
   top = left = 50;
   bottom = right = 300;
   rectangle(left, top, right, bottom);
   int x = left + 1;
   int y = top + 1;
   int newcolor = 12;
   int oldcolor = 0;
   flood(x, y, newcolor, oldcolor);
   getch();

   return 0;
}
```

Result



Discussion

Flood-Fill is an algorithm that determines the area connected to a given node in a multi-dimensional array. Flood fill colors an entire area in an enclosed figure through interconnected pixels using a single color. So, Flood Fill is one in which all connected pixels of a selected color get replaced by a fill color.

Finding and learning

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 interior colour instead of searching for a boundary colour. This approach is called a flood-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.