# Computer Graphics

Karthik S
LIT2021012
5th Semester - Information Technology
Mail: lit2021012@iiitl.ac.in

## Table of Contents

*> All the code to this lab can be found in my github repository :* [*https://github.com/KarthikS373/graphics*](https://github.com/KarthikS373/graphics)

## Lab 03

Implementation of Flood fill algorithms:

1. 4 Connected
2. 8 Connected

### AIM

The aim of this lab is to implement the flood fill algorithms - 4 connected and 8 connected. This involves understanding the theory behind these algorithms, coding them and evaluating their performance. The assignment provides a Python GUI application that allows users to fill using these algorithms interactively

### Submission

#### How to Run

- Install poetry from [here] if not already installed
- Clone the project and install the dependencies using poetry install
- Run the project using `poetry run python path_to_main.py`

#### How to Use

- Launch the application
- Select the drawing algorithm by clicking the respective buttons
- Input the center coordinates and radius of the circle in the provided entry fields
- Click the "Submit" button to execute the selected algorithm and draw the circle
- View the calculation logs displayed on the GUI

## 4 Connected

### Algorithm

```
Inputs:
       - (x, y): Starting coordinates
       - canvas: The canvas to perform the flood-fill on
       - min_x, min_y, max_x, max_y: Optional bounding box for the fill
       - color: Desired color for the fill

Max bounds initialization:
       max_x = max_x if max_x else canvas.winfo_width()
       max_y = max_y if max_y else canvas.winfo_height()

       If the starting point is already occupied:
           Return logs

       Initialize a stack to store coordinates to fill:
       coords_to_fill = [(x, y)]

       Initialize a set to keep track of filled coordinates:
       filled_coords = set()

       While there are coordinates to fill in the stack:
           Pop a coordinate (x, y) from the stack

           If (x, y) is already filled or outside the bounding box:
               Continue to the next iteration

           setPixel(x, y, x+1, y+1)
```

```
    For each direction (dx, dy) in [(0, 1), (1, 0), (0, -1), (-1, 0)] (4 connected):
        Compute the new coordinates (new_x, new_y) by adding (dx, dy) to (x, y)

        If the new pixel is already occupied:
            Continue to the next direction

        Add (new_x, new_y) to the coordinates to fill stack

    Add (x, y) to the filled coordinates set
```

## Code

```
def flood_fill_4_connected(x, y, canvas, min_x=0, min_y=0, max_x=None, max_y=None, color=”red”):
    max_x = max_x if max_x else canvas.winfo_width()
    max_y = max_y if max_y else canvas.winfo_height()

    if canvas.find_overlapping(x, y, x, y):
        return

    coords_to_fill = [(x, y)]
    filled_coords = set()

    while coords_to_fill:
        x, y = coords_to_fill.pop()

    if (x, y) in filled_coords or x < min_x or x >= max_x or y < min_y or y >= max_y:
        continue
```

```
canvas.create_rectangle(x, y, x+1, y+1, outline=color, fill=color)

directions = [(0, 1), (1, 0), (0, -1), (-1, 0)]


for dx, dy in directions:

    new_x, new_y = x + dx, y + dy

if canvas.find_overlapping(new_x, new_y, new_x, new_y):

    continue

    coords_to_fill.append((new_x, new_y))


filled_coords.add((x, y))
```

## 8 Connected

### Algorithm

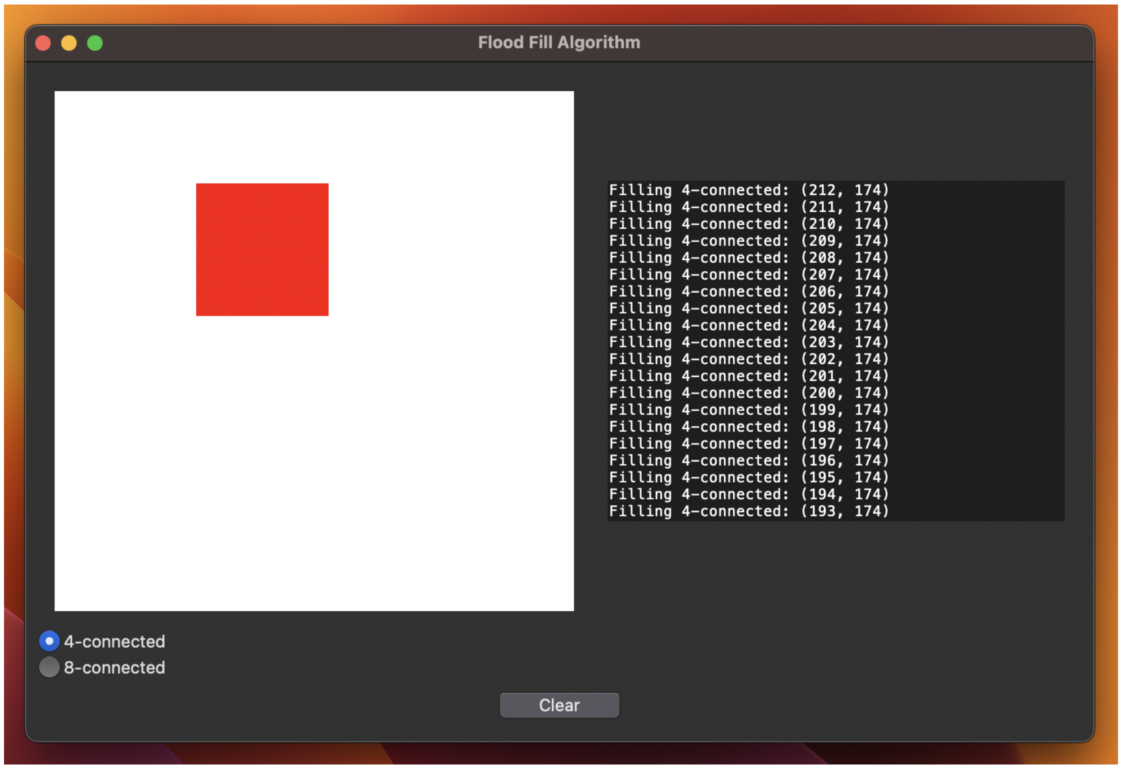
```
Inputs:
      - (x, y): Starting coordinates
      - canvas: The canvas to perform the flood-fill on
      - min_x, min_y, max_x, max_y: Optional bounding box for the fill
      - color: Desired color for the fill

Max bounds initialization:
      max_x = max_x if max_x else canvas.winfo_width()
      max_y = max_y if max_y else canvas.winfo_height()

      If the starting point is already occupied:
          Return logs

      Initialize a stack to store coordinates to fill:
      coords_to_fill = [(x, y)]

      Initialize a set to keep track of filled coordinates:
      filled_coords = set()

      While there are coordinates to fill in the stack:
          Pop a coordinate (x, y) from the stack

          If (x, y) is already filled or outside the bounding box:
              Continue to the next iteration

          setPixel(x, y, x+1, y+1)
```

```
        For each direction (dx, dy) in [(0, 1), (1, 0), (0, -1), (-1, 0), (1, 1), (1, -1), (-1, -1), (-1, 1)] (8 connected):
            Compute the new coordinates (new_x, new_y) by adding (dx, dy) to (x, y)

            If the new pixel is already occupied:
                Continue to the next direction

            Add (new_x, new_y) to the coordinates to fill stack

        Add (x, y) to the filled coordinates set
```

## Code

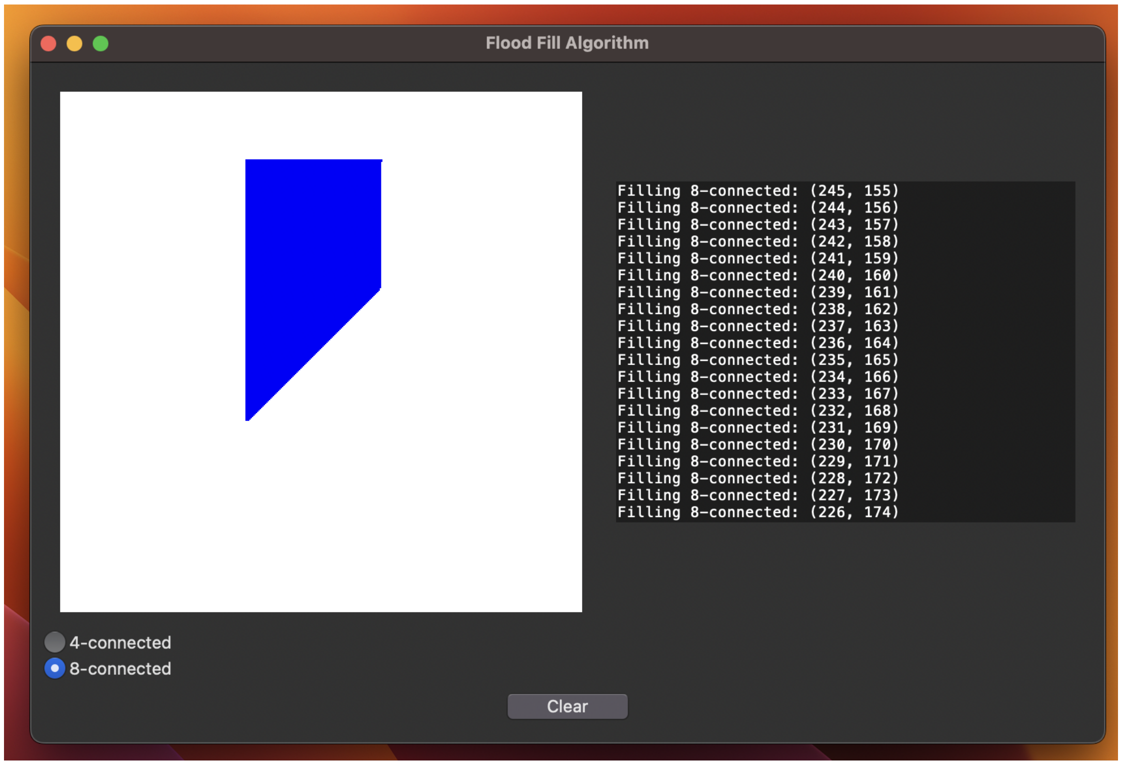
```python
def flood_fill_8_connected(x, y, canvas, min_x=0, min_y=0, max_x=None, max_y=None):
    color = "blue"
    max_x = max_x if max_x else canvas.winfo_width()
    max_y = max_y if max_y else canvas.winfo_height()

    if canvas.find_overlapping(x, y, x, y):
        return logs

    coords_to_fill = [(x, y)]
    filled_coords = set()

    while coords_to_fill:
        x, y = coords_to_fill.pop()

        if (x, y) in filled_coords or x < min_x or x >= max_x or y < min_y or y >= max_y:
            continue
        canvas.create_rectangle(x, y, x+1, y+1, outline=color, fill=color)


        directions = [(0, 1), (1, 0), (0, -1), (-1, 0), (1, 1), (1, -1), (-1, -1), (-1, 1)]


        for dx, dy in directions:

            new_x, new_y = x + dx, y + dy

            if canvas.find_overlapping(new_x, new_y, new_x, new_y):

            continue

            coords_to_fill.append((new_x, new_y))


       filled_coords.add((x, y))

       logs.append(f"Filling 8-connected: ({x}, {y})\n")

    return logs
```

## ScreenShots



```
Filling 4-connected: (212, 174)
Filling 4-connected: (211, 174)
Filling 4-connected: (210, 174)
Filling 4-connected: (209, 174)
Filling 4-connected: (208, 174)
Filling 4-connected: (207, 174)
Filling 4-connected: (206, 174)
Filling 4-connected: (205, 174)
Filling 4-connected: (204, 174)
Filling 4-connected: (203, 174)
Filling 4-connected: (202, 174)
Filling 4-connected: (201, 174)
Filling 4-connected: (200, 174)
Filling 4-connected: (199, 174)
Filling 4-connected: (198, 174)
Filling 4-connected: (197, 174)
Filling 4-connected: (196, 174)
Filling 4-connected: (195, 174)
Filling 4-connected: (194, 174)
Filling 4-connected: (193, 174)
```

[✓ 4-connected]
8-connected



```
Filling 8-connected: (245, 155)
Filling 8-connected: (244, 156)
Filling 8-connected: (243, 157)
Filling 8-connected: (242, 158)
Filling 8-connected: (241, 159)
Filling 8-connected: (240, 160)
Filling 8-connected: (239, 161)
Filling 8-connected: (238, 162)
Filling 8-connected: (237, 163)
Filling 8-connected: (236, 164)
Filling 8-connected: (235, 165)
Filling 8-connected: (234, 166)
Filling 8-connected: (233, 167)
Filling 8-connected: (232, 168)
Filling 8-connected: (231, 169)
Filling 8-connected: (230, 170)
Filling 8-connected: (229, 171)
Filling 8-connected: (228, 172)
Filling 8-connected: (227, 173)
Filling 8-connected: (226, 174)
```

4-connected
[✓ 8-connected]