BANGLADESH UNIVERSITY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Lab Report

Course Title: Computer Graphics Lab

Course Code: CSE 342

Submitted By:

NAME	ID	INTAKE/SECTION
Abdullah Al Mahmud Joy	21225103506	49/7

Submitted To:

MD. SAKHAWAT HOSSAIN

Lecturer

Dept. of CSE

Bangladesh University of Science & Technology

May 6, 2025

1. **Problem Statement:**
   • Draws a circle using Bresenham's Circle Drawing Algorithm cantered in the window.
   • Uses the Flood Fill algorithm to fill the circle and an area outside the circle with different colors.
   • Draws a rectangular boundary around the area.

2. **Overview:**
   The code performs the following actions:
   - Initializes the graphics window.
   - Draws a rectangle centered in the screen as a bounding area.
   - Draws a circle centered using Bresenham's circle drawing algorithm.
   - Fills:
   - The circle's interior with red.
   - The area outside the circle (but inside the rectangle) with green.

3. **Algorithm:**
   a. Bresenham's Circle Drawing Algorithm:
   A midpoint algorithm is used to draw a circle using integer arithmetic. It avoids floating-point calculations and is highly efficient for raster devices.
   void bresenham(int x, int r, int c, int d)
    x is the starting x-coordinate (0).
    r is the radius.
    c and d are the circle's center coordinates.
    Symmetry is used to plot 8 points per iteration.

   b. Flood Fill Algorithm (4-directional):
   A recursive region-filling algorithm. It fills a region with a given color, starting from a point, and spreads to all connected pixels of the same original color.
   void FloodFill(int x, int y, int original, int fill_clr)
- It checks the current pixel color.
- If the color matches original, it sets it to fill_clr.
- Then recursively calls the function in four directions (left, right, up, down).

4. **Code:**

```
#include <graphics.h>
#include <math.h>
#include <stdio.h>
#include <bits/stdc++.h>
#include <conio.h>
#include <windows.h>
using namespace std;
void FloodFill(int x, int y, int original,int
fill_clr)
{
    int clr = getpixel(x,y);
    if(clr==original)
    {
        putpixel(x,y,fill_clr);

        FloodFill(x+1,y,original,fill_clr);
```

```
        else
        {
            p=p+4*x+6;
            x=x+1;
        }
    }
}
int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");
    int r=50;
    float th=0;
    int A = getmaxx();
    int B = getmaxy();
    int c,d,x,y,original,fill_clr;
```

```
      FloodFill(x-1,y,original,fill_clr);
      FloodFill(x,y+1,original,fill_clr);
      FloodFill(x,y-1,original,fill_clr);
   }
}
void bresenham(int x,int r,int c,int d)
{
   int y,p;
   p=3-2*r;
   y=r;
   while( x<=y)
   {
      putpixel(x+c,-y+d,RED);
      putpixel(x+c,d+y,RED);
      putpixel(-x+c,d+y,RED);
      putpixel(-x+c,d-y,RED);
      putpixel(y+c,d+x,RED);
      putpixel(y+c,d-x,RED);
      putpixel(-y+c,d+x,RED);
      putpixel(-y+c,d-x,RED);

      if(p>=0)
      {
         p=p+4*(x-y)+10;
         x=x+1;
         y=y-1;
      }
```

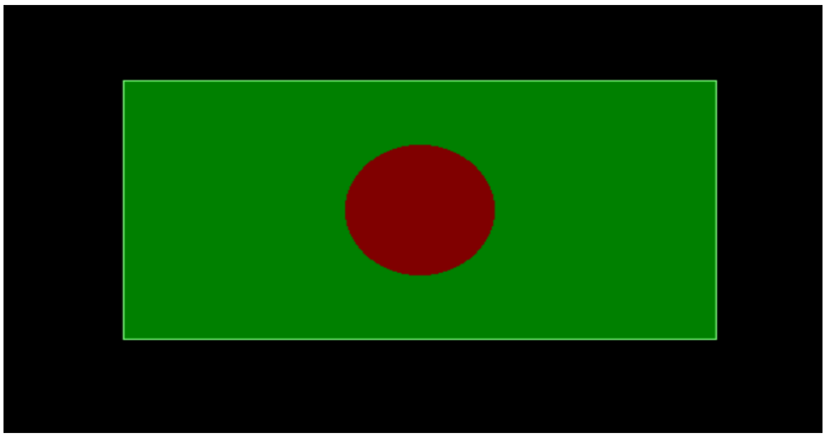
```
    setcolor(10);

    line(-200+A/2,-100+B/2,200+A/2,-100+B/2);
    line(200+A/2,-100+B/2,200+A/2,100+B/2);
    line(200+A/2,100+B/2,-200+A/2,100+B/2);
    line(-200+A/2,-100+B/2,-200+A/2,100+B/2);

    bresenham(0,50,A/2,B/2);
    FloodFill(A/2, B/2,BLACK,RED);
    FloodFill(A/2+51,B/2+51,BLACK,GREEN);

    getch();
    closegraph();
    return 0;
}
```

5. Output:



6. Discussion:

- Simple demonstration of **midpoint circle drawing** without relying on high-level libraries.
- Recursive **flood fill** shows region filling.
- Clean separation between drawing and filling logic.
- Uses symmetry in circle drawing to minimize computation.