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

April 29, 2025

1. Problem Statement

Drawing a circle using four classic computer graphics methods:

- Direct Equation Method
- Trigonometric (Parametric) Method
- Bresenham's Circle Algorithm
- Midpoint Circle Algorithm

Each method generates a circle pixel-by-pixel on a 2D Cartesian plane, relying on mathematical formulas and discrete plotting

2. Overview

Circle Drawing is a foundational task in computer graphics. Computers don't think in real "perfect curves"; they only know how to light up square pixels. So we use math-based approximations to simulate a circle on a pixel grid.

- **Direct Equation** — y = $\sqrt{r^2 - x^2}$ (basic but slow and inaccurate sometimes).
- **Trigonometric** — (x,y)=(rcosθ,rsinθ)) parametric form (better, but floating-point heavy).
- **Bresenham's Circle Algorithm** — Highly optimized, uses only **integers** (no float, no sqrt, no sin/cos).
- **Midpoint Circle Algorithm** — A cleaner, more "geometrically justified" version of Bresenham.

3. Algorithm
a) Direct Equation Method

1. For a circle centered at (xc, yc) with radius r, basic formula is: $x^2 + y^2 = r^2$
2. For each x from 0 to $\frac{r}{\sqrt{2}}$:
    - Calculate y= $\sqrt{r^2 - x^2}$
    - Plot (x, y) and its 8 symmetric points (circle is symmetric).

Problem: Needs `sqrt` function every time — expensive

b) Trigonometric Method

1. The parametric equations of a circle:
   x = rcosθ y = rsinθ
2. Start angle θ=0, increment it by small steps (like 0.01 radians) until 2π
3. For each step:

- Calculate x and y.
- Plot point (x, y).

Problem: Needs `sin` and `cos` — floating point operations are heavy, slow on old machines.

c) Bresenham's Circle Algorithm

1. Uses a decision variable `d` to choose the next pixel without needing floats.
2. Initialize:
   - x = 0, y = r
   - Decision parameter d=3−2r
3. While x <= y:
   - Plot 8 points (symmetric points).
   - If d < 0:
     - Next pixel is (x+1, y)
     - Update d: d = d + 4x + 6
   - Else:
     - Next pixel is (x+1, y-1)
     - Update d: d = d + 4(x−y)+ 10

Advantage: No floating point, no sqrt — just add, subtract, and shift. Much faster.

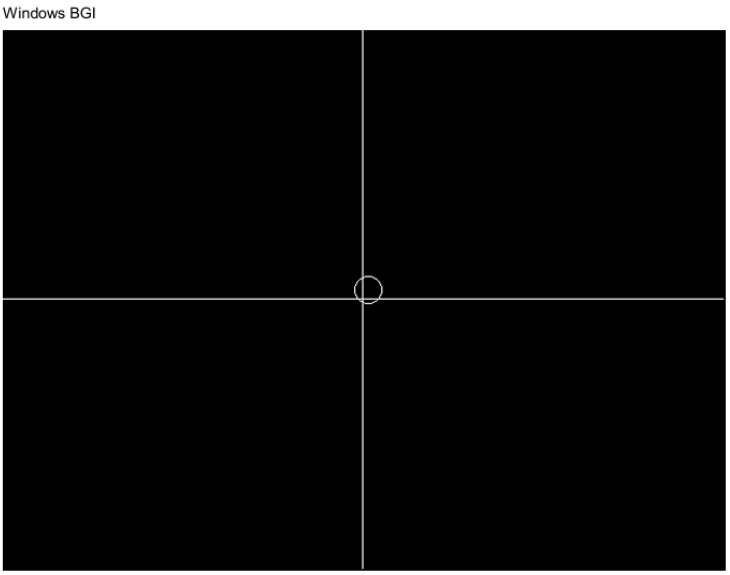
d) Midpoint Circle Algorithm

1. Almost the same as Bresenham, but usesthe midpoint concept:
   - Check whether midpoint between two candidate pixels is inside or outside the circle.
2. Initialize:
   - x = 0, y = r
   - Decision parameter p = 1−r
3. For each step:
   - If p<0: choose East pixel, update p = p + 2x + 3
   - Else: choose South-East pixel, update p = p + 2(x−y) + 5

Advantage: Even simpler logic behind the updates.

4. Output

```
0. direct Equation
1. trigonometic
2. Besenham
3. Mid Point
enter a number from this: 3
Enter center (x y): 5 8
Enter radius: 12
(0, 12)
(1, 12)
(2, 12)
(3, 12)
(4, 11)
(5, 11)
(6, 10)
(7, 10)
(8, 9)
```



5. Discussion

- **Direct Equation** is **naive**. It works, but is computationally expensive because of `sqrt`. Unacceptable in performance-demanding applications.

- **The Trigonometric Method** is **better**, because it naturally draws the circle smoothly as a parametric curve. But it still uses heavy float-point calculations (`sin`, `cos`), which older machines hate.

- **Bresenham's Algorithm** is a **game-changer**. No floating point, no square root. Only integer arithmetic. Super fast. It became the industry standard for real-time graphics.

- **The Midpoint Circle Algorithm** is a **theoretical refinement** of Bresenham, with very similar performance but an even cleaner decision-making process based on evaluating midpoints.