Lab Report

Course title : Computer Graphics Lab
Course code : CSE 422
Experiment Name: Implementation of Mid Point Circle Drawing Algorithm

Submitted By

Name: Aong Sing Marma
ID : 221-15-6050
Sec : 61_H1

Submitted To
Mr. Md Aman Ullah
Lecturer Department
of CSE Daffodil International University

Submission Date: *14.12.2025*

Title of the Report

Implementation of Mid Point Circle Drawing Algorithm

1. Objective(s)

The main objectives of this experiment are:

- To study the concept of circle drawing in raster graphics.
- To understand the mathematical fundamentals behind the Mid Point Circle Drawing Algorithm.
- To implement the Mid Point Circle Drawing Algorithm step by step.
- To observe how decision parameters are used to select pixel positions.

2. Introduction

Computer graphics is a field of computer science that deals with the creation, manipulation, and displaying images on computer screens. One of the most important tasks in computer graphics is drawing basic geometric shapes such as lines, circles, and ellipses on a raster display. Since raster displays are made up of many small dots, continuous mathematical equations cannot be directly applied.

A circle is defined mathematically by the equation:

$$x^2 + y^2 = r^2$$

However, evaluating this equation for every pixel is computationally expensive. To overcome this limitation, efficient algorithms such as the Mid Point Circle Drawing Algorithm are used. This algorithm determines the next pixel position based on a decision parameter and uses only integer arithmetic, making it fast and suitable for real-time graphics applications.

3. Required Tools

- **Hardware Requirements:**
    - Computer or Laptop.
    - Keyboard and Mouse.
- **Software Requirements:**
    - Programming Language: C .
    - Compiler or Interpreter : GCC / Turbo C.
    - Graphics Library: graphics.h for C.
    - Operating System: Windows.

4. Algorithm Implementation

The Mid Point Circle Drawing Algorithm draws a circle by calculating points only in one octant and then reflecting those points in the other seven octants using symmetry.

Step-by-Step Algorithm:

1. Read the radius r of the circle.
2. Set the center coordinates (xc, yc) of the circle.
3. Initialize the starting point:
    - $x = 0$
    - $y = r$
4. Initialize the decision parameter:
    - $p = 1 - r$
5. Plot the initial points using 8-way symmetry.
6. Repeat the following steps while $x < y$:
    - Increment x by 1.
    - If $p < 0$:
        - The next point is chosen as $(x + 1, y)$
        - Update decision parameter:

        - $p = p + 2x + 1$
    - Else:
        - The next point is chosen as $(x + 1, y - 1)$
        - Decrement y by 1
        - Update decision parameter:
            - $p = p + 2x + 1 - 2y$
    - Plot all eight symmetrical points for the calculated (x, y).
7. Stop when $x \geq y$.

5. Sample Input(s)

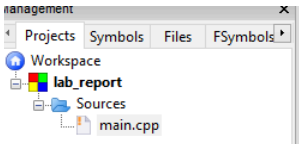
- Radius of the circle (r): User input (range: 1 to 100)
- Center of the circle: $(0, 0)$

6. Sample Output(s)

- A proper circle is drawn using the user-specified radius.
- Pixel points plotted symmetrically in all eight octants.
- Smooth and accurate representation of the circle on the screen.

7. Screenshots

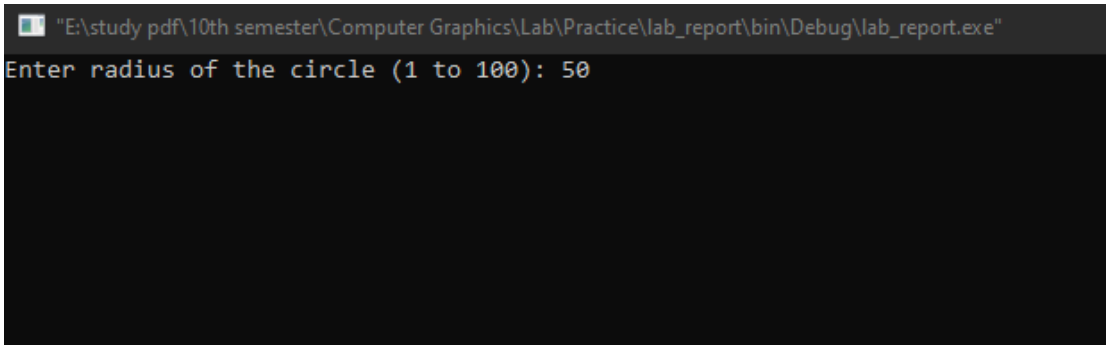
- Screenshot of program code:



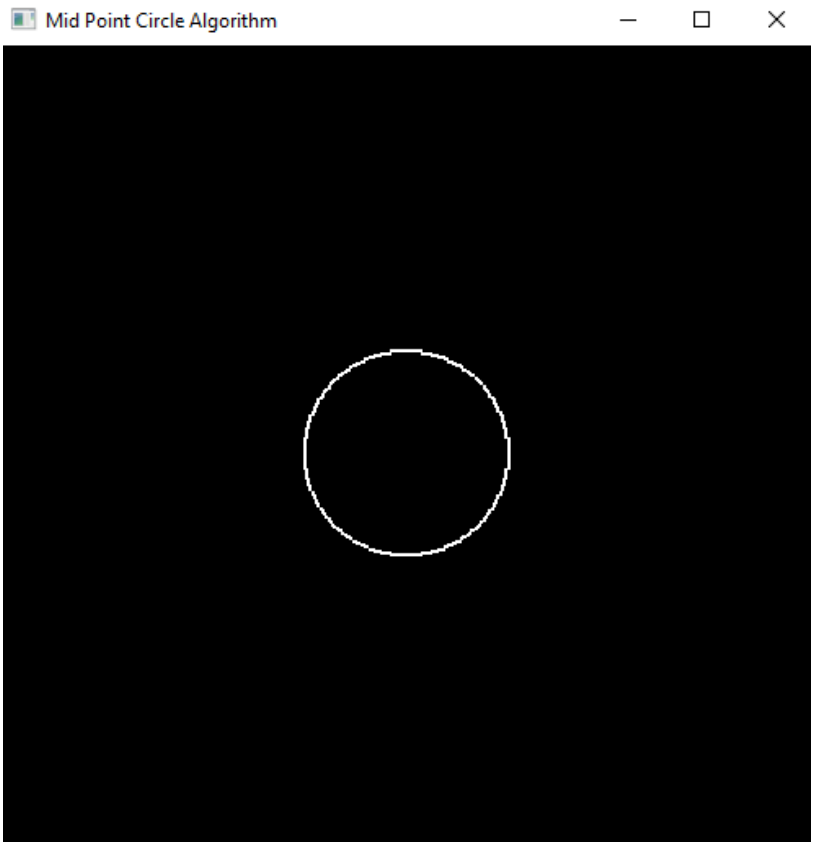

```
#include <GL/glut.h>
#include <stdio.h>
int xc = 0, yc = 0, r;
void plotPoints(int x, int y)
{
    glBegin(GL_POINTS);
        glVertex2i(xc + x, yc + y);
        glVertex2i(xc - x, yc + y);
        glVertex2i(xc + x, yc - y);
        glVertex2i(xc - x, yc - y);
        glVertex2i(xc + y, yc + x);
        glVertex2i(xc - y, yc + x);
        glVertex2i(xc + y, yc - x);
        glVertex2i(xc - y, yc - x);
    glEnd();
}
void drawCircle()
{
    int x = 0;
    int y = r;
    int p = 1 - r;
    glClear(GL_COLOR_BUFFER_BIT);
    while (x <= y)
    {
        plotPoints(x, y);
        x++;
        if (p < 0)
            p = p + 2 * x + 1;
        else
        {
            y--;
            p = p + 2 * x + 1 - 2 * y;
        }
    }
    glFlush();
}
```

```
void init()
{
    glClearColor(0, 0, 0, 1);
    glColor3f(1, 1, 1);
    glPointSize(2.0);
    gluOrtho2D(-200, 200, -200, 200);
}
int main(int argc, char** argv)
{
    printf("Enter radius of the circle (1 to 100): ");
    scanf("%d", &r);
    if (r < 1 || r > 100)
    {
        printf("Invalid radius! Please enter a value between 1 and 100.\n");
        return 0;
    }
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Mid Point Circle Algorithm");
    init();
    glutDisplayFunc(drawCircle);
    glutMainLoop();
    return 0;
}
```

- Screenshot of program execution



- Screenshot of the displayed circle



8. Output Analysis

The output obtained from the Mid Point Circle Algorithm is a smooth and symmetrical circle. The algorithm efficiently determines the nearest pixel positions without using

floating-point calculations. By exploiting the symmetry of a circle, only one-eighth of the circle is computed, which significantly reduces computational overhead.

The plotted points closely approximate the ideal circle equation, making the algorithm accurate for raster displays.

9. Discussion

- The decision point is a value used to choose the next pixel.
- It decides whether the next pixel lies inside or outside the circle.
- The initial decision point is calculated as $\mathbf{p = 1 - r}$.
- If $\mathbf{p < 0}$, the next pixel chosen is $\mathbf{(x + 1, y)}$.
- If $\mathbf{p \geq 0}$, the next pixel chosen is $\mathbf{(x + 1, y - 1)}$.
- The decision point is updated after each step.
- This process continues until the circle is completed.

10. Conclusion

From this experiment, it can be concluded that the Mid Point Circle Algorithm is an efficient and reliable method for drawing circles in raster graphics. It reduces computational cost by using integer arithmetic and symmetry properties. The algorithm produces accurate results and is suitable for real-time graphics applications. This experiment helped in gaining a deeper understanding of decision-based algorithms in computer graphics.

11. References

1. GeeksforGeeks, *"Mid-Point Circle Drawing Algorithm"*.
2. TutorialsPoint, *"Midpoint Circle Algorithm"*.
3. Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F., *Computer Graphics: Principles and Practice.*