
# Daffodil International University

## Department of Computer Science and Engineering

### Final Lab Report

**Topic:** Implementation of Midpoint Circle Drawing Algorithm Using OpenGL (GLUT)

**Course Code:** CSE422
**Course Title:** Computer Graphics Lab

**Submitted To:**

Md. Aman Ullah
Lecturer
Department of CSE
Daffodil International University

**Submitted By:**

**Adnan Rahman Sayeem**
**ID:** 221-15-5505
**Section:** 61_H1
Department of CSE
Daffodil International University

**Date of Submission: December 14, 2025**

# 1 Title of the Report

Implementation of Midpoint Circle Drawing Algorithm Using OpenGL (GLUT)

# 2 Objective(s)

The objectives of this lab experiment are:

1. To understand the working principle of the Midpoint Circle Drawing Algorithm
2. To implement the algorithm using C++ and OpenGL (GLUT)
3. To generate a visual representation of a circle on a raster display
4. To analyze the efficiency of the algorithm using integer-based calculations

# 3 Introduction

The Midpoint Circle Drawing Algorithm is a fundamental algorithm in computer graphics used to draw a circle on a pixel-based display. Unlike direct mathematical equations, this algorithm uses a decision parameter to determine the next pixel position, which makes it computationally efficient. The algorithm takes advantage of the symmetry of a circle by calculating points in one octant and reflecting them across the other seven octants. In this experiment, the algorithm is implemented using C++ and OpenGL (GLUT) to visually demonstrate its working principle.

# 4 Required Tools

The following tools are required to perform this experiment:

- Programming Language: C++
- Graphics Library: OpenGL with GLUT
- IDE: Code::Blocks
- Operating System: Windows

# 5 Algorithm Implementation

```
#include <windows.h>
#include <GL/glut.h>
#include <stdio.h>

int r;
int x = 0, y;
int pk;

void init()
{
    // Yellow background (RGB 1.0, 1.0, 0.0)
    glClearColor(1.0, 1.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-200.0, 200.0, -200.0, 200.0, -1.0, 1.0); //// Orthographic
    representation; multiply the current matrix by an orthographic
   matrix 2D= left right,bottom,top equivalent near=-1,far=1
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    // Black circle color
    glColor3f(0.0, 0.0, 0.0);

    glBegin(GL_POINTS);

    x = 0;
    y = r;
    pk = 1 - r;

    while (x <= y)
    {
        glVertex2i( x,  y);
        glVertex2i(-x,  y);
        glVertex2i( x, -y);
        glVertex2i(-x, -y);
        glVertex2i( y,  x);
        glVertex2i(-y,  x);
        glVertex2i( y, -x);
        glVertex2i(-y, -x);

        x++;

        if (pk < 0)
        {
            pk = pk + 2*x + 1;
        }
        else
        {
            y--;
            pk = pk + 2*(x - y) + 1;
        }
    }

```

```
55      glEnd();
56      glFlush();
57  }
58
59  int main(int argc, char** argv)
60  {
61      printf("Enter radius: ");
62      scanf("%d", &r);
63
64      glutInit(&argc, argv);
65      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
66      glutInitWindowSize(500, 500);
67      glutInitWindowPosition(100, 100);
68      glutCreateWindow("Midpoint Circle Algorithm");
69
70      init();
71      glutDisplayFunc(display);
72      glutMainLoop();
73
74      return 0;
75  }
```

Listing 1: Midpoint Circle Drawing Algorithm

# 6 Sample Input

- **Radius of the circle was inputed as 170.**

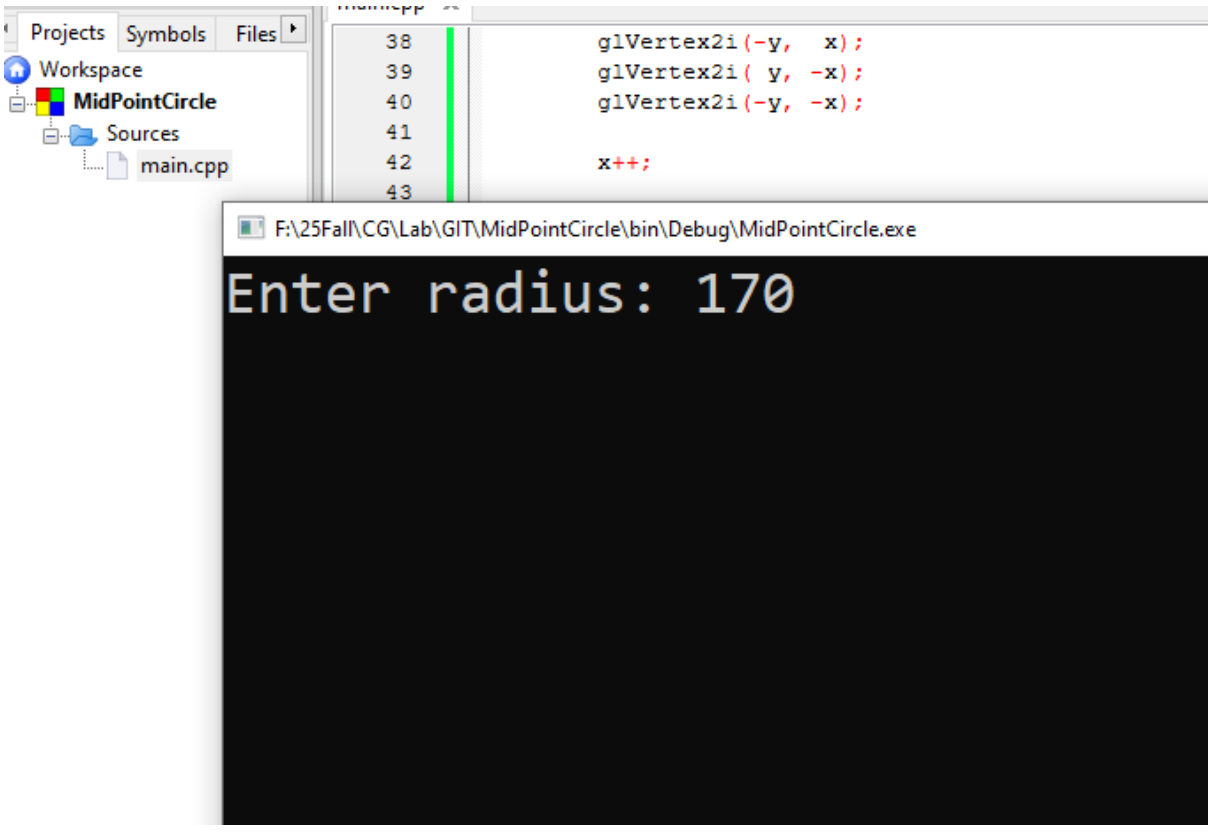


Figure 1: Input radius of 170.

# 7 Sample Output

- A circular shape drawn using pixel points.

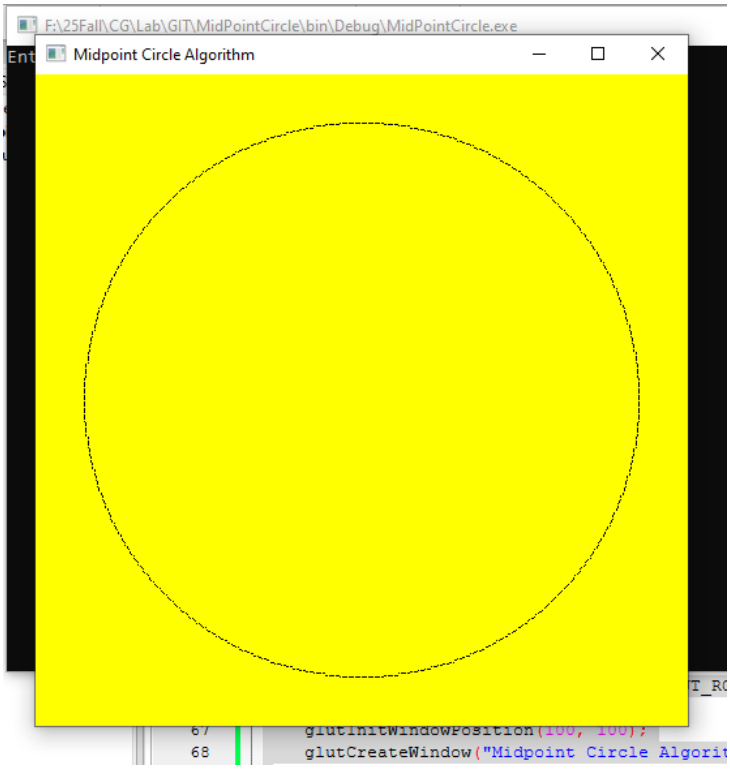


Figure 2: Output circle generated using the Midpoint Circle Algorithm in CodeBlocks.

The circle appears symmetric and smooth due to 8-way symmetry. sam

# 8 Output Analysis

The generated output successfully displays a circle of the given radius using discrete pixel points. The algorithm efficiently calculates each point using integer arithmetic, which improves performance. The symmetry of the circle is clearly visible, as points are plotted uniformly in all eight octants. The output verifies the correctness of the Midpoint Circle Drawing Algorithm.

# 9 Conclusion

In this lab experiment, the Midpoint Circle Drawing Algorithm was successfully implemented using C++ and OpenGL (GLUT). The algorithm efficiently generated a circular shape using integer-based calculations and symmetry. The experiment helped in understanding the practical application of computer graphics algorithms in real-time rendering environments.

# 10 References


- Hearn, D., & Baker, M. P. (2011). *Computer Graphics with OpenGL (4th Edition).* Pearson Education.
- Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F. (2013). *Computer Graphics: Principles and Practice (3rd Edition).* Addison-Wesley.
- OpenGL Architecture Review Board. (1999). *OpenGL Programming Guide: The Official Guide to Learning OpenGL (Red Book, 3rd Edition).* Addison-Wesley.
- Lecture Slide – Slide 5: Midpoint Circle Algorithm (MID – Computer Graphics Lab, Department of CSE, [Daffodil International University], 2025).