Lab Report

Course Code: CSE 422

Course Title: Computer Graphics Lab

Report No: 07

Title: Implementation of Mid-Point Circle Drawing algorithm and Implement it and draw half moon using it.

Submitted To:

Syada Tasmia Alvi

Senior Lecturer

Department of Computer Science and Engineering (C.S.E)

Submitted by:

Name: Mahmudul Hasan Piash

ID: 221-15-5606

Section: 61_D1

Department of Computer Science and Engineering (C.S.E)

Date of submission: 14th December, 2025

Title

Implementation of Mid-Point Circle Drawing algorithm and Implement it and draw half moon using it.

Introduction

In this project, an OpenGL-based graphical program has been developed to draw a Moon shape. The concept is implemented using two overlapping circles, where one circle is drawn in warm white and another circle matching the night sky background is drawn slightly shifted to create the crescent shape. The circle is generated using trigonometric calculations inside OpenGL's GL_TRIANGLE_FAN primitive, which is similar in spirit to the Mid-Point Circle Drawing approach for approximating circular shapes. GLUT is used for window management, rendering, and handling display events.

Contents

The following functions and shapes have been used in this project:

1. init()
   - Sets the background color of the window to simulate a night sky.
   - glClearColor(0.05f, 0.05f, 0.2f, 1) is used to make the background dark blue.
2. draw()
   - This function performs all rendering tasks.
   - The first circle is drawn in warm white (1.0f, 1.0f, 0.9f), representing the main moon body.
   - A second circle matching the background color is drawn slightly offset, creating a shadow effect to form the crescent moon shape.
   - glutSwapBuffers() is used for double buffering.
3. reshape()
   - Adjust the viewport and projection when the window is resized.
   - Uses gluOrtho2D() to define a fixed 2D coordinate system from -300 to 300 on both axes.
4. circle()
   - Draws a circle using GL_TRIANGLE_FAN with 100 segments.
   - Takes radius (rx, ry) and center (cx, cy) as parameters
   - Uses trigonometric functions cos() and sin() to calculate each vertex of the circle.

Code

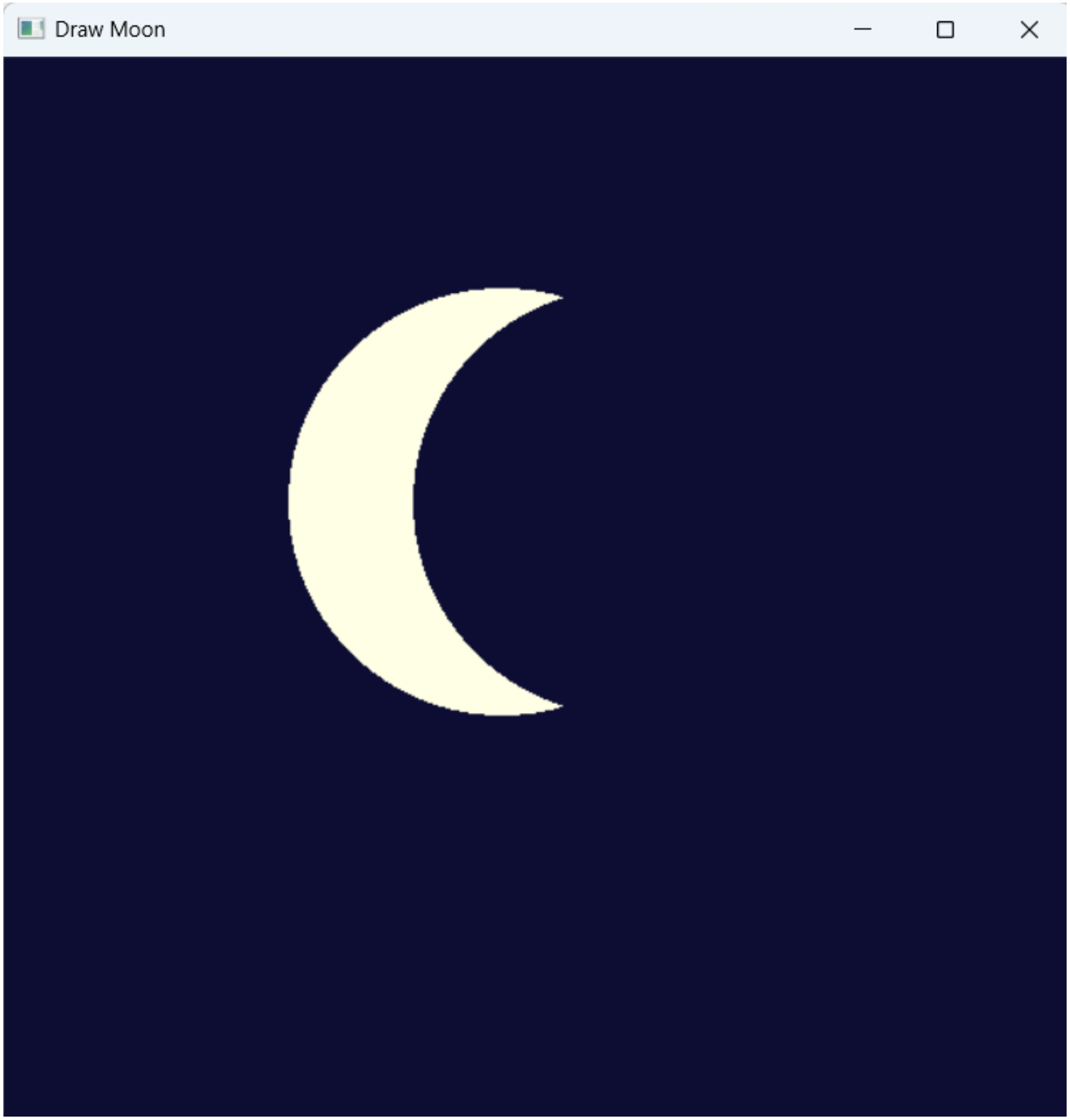
```
#include <GL/glut.h>

#include <cmath>

void init();
```

```
void draw();
void reshape(int w, int h);
void circle(GLfloat rx, GLfloat ry, GLfloat cx, GLfloat cy);
int main(int argc, char **argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
   glutInitWindowPosition(300, 100);
   glutInitWindowSize(600, 600);
   glutCreateWindow("Draw Moon");

   glutDisplayFunc(draw);
   glutReshapeFunc(reshape);
   init();

   glutMainLoop();
}
void init()
{
   glClearColor(0.05f, 0.05f, 0.2f, 1);
}
void draw()
{
   glClear(GL_COLOR_BUFFER_BIT);
   glLoadIdentity();

   // Draw white circle (main moon body)
```

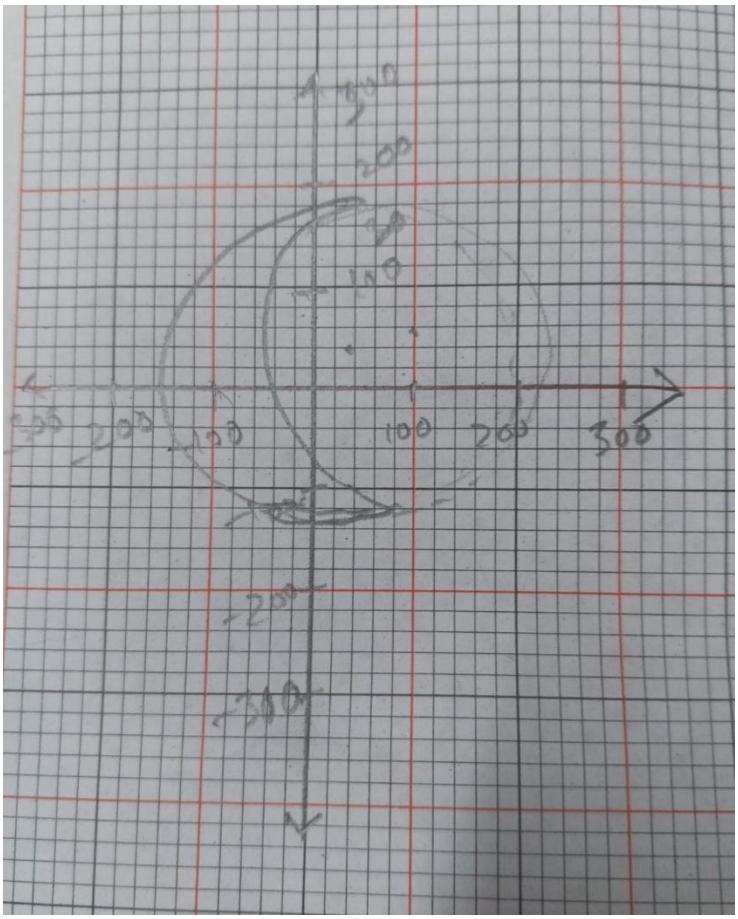
```
    glColor3f(1.0f, 1.0f, 0.9f);
    circle(120, 120, -20, 50);

    // Draw dark circle (shadow on moon to create crescent)
    glColor3f(0.05f, 0.05f, 0.2f);
    circle(120, 120, 50, 50);

    glutSwapBuffers();
}
void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-300, 300, -300, 300);
    glMatrixMode(GL_MODELVIEW);
}
void circle(GLfloat rx, GLfloat ry, GLfloat cx, GLfloat cy)
{
    glBegin(GL_TRIANGLE_FAN);
    glVertex2f(cx, cy);
    for (int i = 0; i <= 100; i++)
    {
        float angle = 2 * M_PI * i / 100;
        float x = rx * cosf(angle);
        float y = ry * sinf(angle);
        glVertex2f((x + cx), (y + cy));
```

```
    }
    glEnd();
}
```

Output



Graph



Discussion

The main idea of this project is to create a crescent moon by overlapping two circles. First, a full warm white circle is drawn to represent the moon body. Then a circle matching the dark night sky background is drawn slightly offset, creating a shadow effect that hides part of the white circle and producing the realistic crescent moon shape.

Although the traditional Mid-Point Circle Drawing algorithm plots pixels step-by-step, in OpenGL we use trigonometric calculations and primitives to approximate the circle smoothly. The background color simulates a night sky, enhancing the visual appearance of the moon. This project demonstrates how simple 2D shapes can be combined to form more complex shapes. Furthermore, GLUT is used to manage window creation, display callbacks, and reshape events, making the program interactive and scalable.