Lab Report 5

Name: **Arnob Das Shacha**

ID: **221-15-5277**

Section: 61_D1

Course Code: CSE 422

Experiment Name: Implementation of Mid-Point Circle Drawing algorithm. (Half Moon).

Submitted To:

Sayeda Tasnim Alvi

Senior Lecturer

Dept of CSE

Daffodil International University

Title

Implementation of Mid-Point Circle Drawing algorithm.

Introduction

In this project, an OpenGL-based graphical program has been developed to draw a **Half Moon** shape. The concept is implemented using two overlapping circles, where one circle is drawn in white and another black circle 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

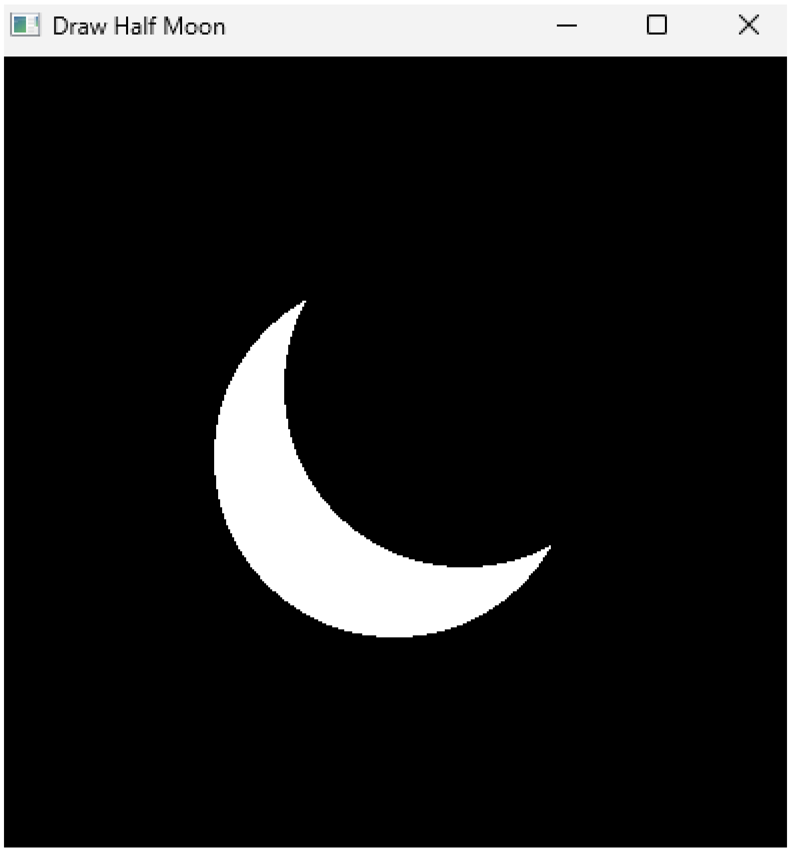
The following functions and shapes have been used in this project:

1. init()
    - Sets the background color of the window.
    - glClearColor(0, 0, 0, 1) is used to make the background black.
2. draw()
    - This function performs all rendering tasks.
    - The first circle is drawn in white, representing a full moon.
    - A second black circle is drawn slightly to the right, overlapping the first circle to create the Half Moon effect.
    - glutSwapBuffers() are 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, ensuring proper scaling.
4. circle()
    - Draws a circle using GL_TRIANGLE_FAN with 100 segments.
    - Take radius (rx, ry) and center (cx, cy) as parameters.
    - Uses trigonometric functions cos() and sin() to calculate each vertex of the circle.

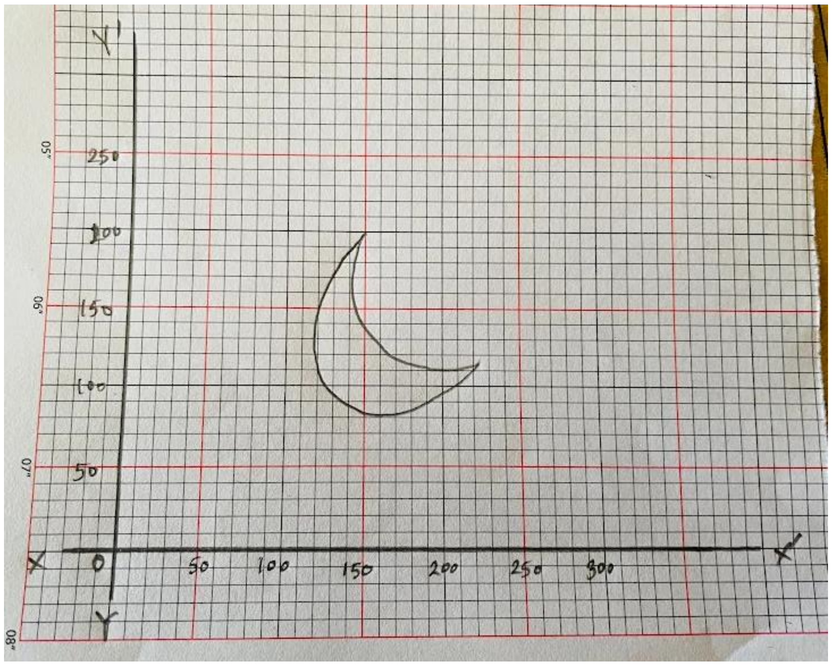
Code

```
#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(400, 400);

   glutCreateWindow("Draw Half Moon");

   // registration of callbacks

   glutDisplayFunc(draw);

   glutReshapeFunc(reshape);

   init();

   // enter the glut loop

   glutMainLoop();

}


void init()

{
   glClearColor(0, 0, 0, 1); // make bg color to black
}
void draw()
{
   glClear(GL_COLOR_BUFFER_BIT); // clear frame color
   glLoadIdentity();
   // left circle
   glColor3f(1.0f, 1.0f, 1.0f);
   circle(180, 180, 0, 0);
   // right circle
   glColor3f(0.0f, 0.0f, 0.0f);
   circle(180, 180, 70, 70);
   glutSwapBuffers();
}
void reshape(int w, int h)
{
   glViewport(0, 0, (GLsizei)w, (GLsizei)h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(-400, 400, -400, 400);
   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 Half Moon by overlapping two circles. First, a full white circle is drawn to represent the moon. Then a black circle is drawn slightly offset to the right, hiding part of the white circle and producing the Half 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. 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.

Conclusion

This project successfully demonstrates how basic OpenGL primitives can be combined to create visually meaningful objects such as a Half Moon. By overlapping two circles, the project illustrates fundamental graphics concepts like layering, shape approximation, and coordinate-based rendering. Overall, the program provides a clear learning example for understanding circle drawing, color manipulation, and the OpenGL rendering pipeline.