NOTRE DAME UNIVERSITY BANGLADESH

Computer Graphics Lab Report-02

Course Code: CSE-4204
Course Title: Computer Graphics Lab
Lab Task Topic: Draw a Flag

Submitted by:

Name: Istiak Alam
ID: 0692230005101005
Batch: CSE-20

Submission Date: January 27, 2026

Submitted to:

Humayara Binte Rashid
Lecturer, Dept. of CSE
Notre Dame University Bangladesh

Table of Contents

1 Objective

The objective of this lab is to understand and apply fundamental Computer Graphics concepts using OpenGL and GLUT. The experiment focuses on drawing a national flag by combining basic geometric primitives such as quadrilaterals, triangles, and circles within a 2D orthographic projection. Through this task, the use of coordinate systems, color models, and custom drawing functions is practiced to build a complete graphical object.

2 Tools and Environment

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

3 Graph Implementation

Here is the Graph file Link

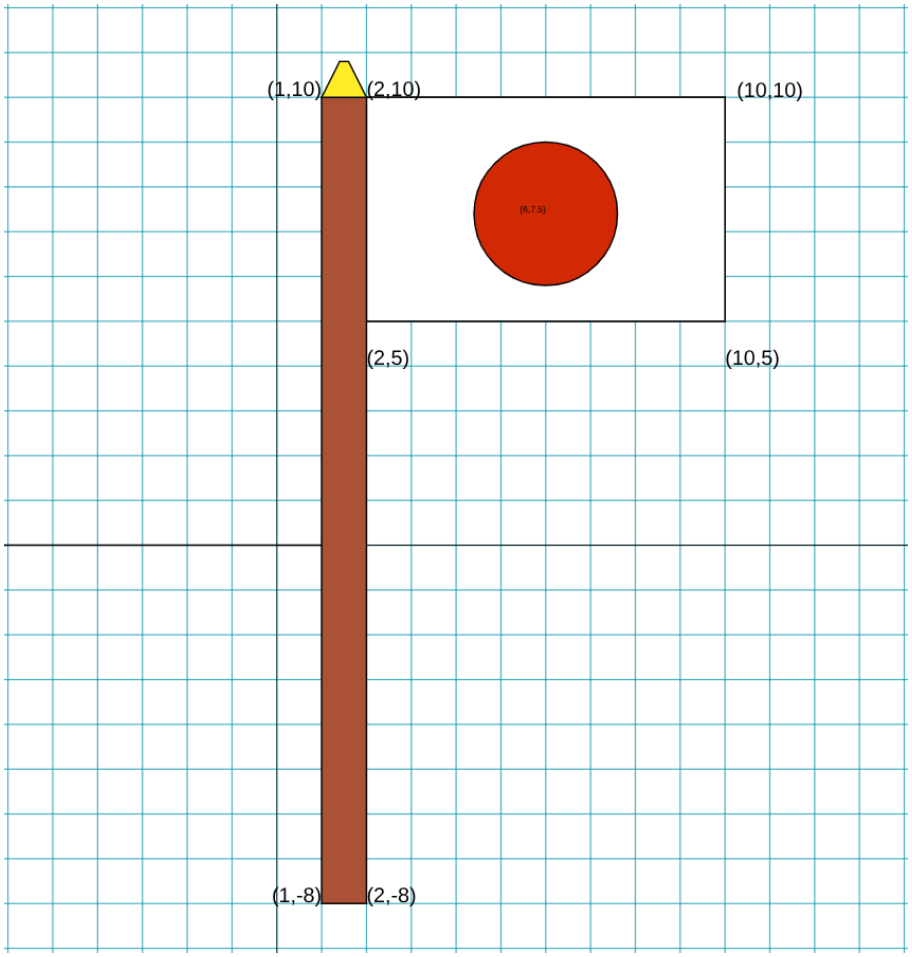

Figure 1: Graph View

4 Source Code

The program starts by initializing the OpenGL environment, where the background color is set and a 2D orthographic projection is defined using the `glOrtho()` function. A custom function named `DrawCircle()` is implemented to draw the circular portion of the flag using the triangle fan technique and trigonometric calculations.

The `display()` function handles the rendering process. It draws the rectangular portion of the flag using quadrilateral primitives and places a filled circle at the center to represent the emblem. Additional geometric shapes are used to create the flag stand and decorative elements. Different colors are applied using the `glColor3f()` function to achieve a realistic appearance.

The `main()` function initializes the GLUT framework, creates the display window, registers the display callback function, and starts the main rendering loop to continuously display the flag.

```
/*
 * Lab Task-02
 * Creating a Flag
 * Written by Istiak Alam
 */

#include <GL/glut.h>  // Include GLUT header here
#include <stdlib.h>
#include <math.h>


//float p=-2.3;
void DrawCircle(float cx, float cy, float rx,float ry, int
   num_segments)
{
        glBegin(GL_TRIANGLE_FAN);
        for(int ii = 0; ii < num_segments; ii++)
        {
float theta = 2.0f * 3.1415926f * float(ii) / float(num_segments);
   //current angle

        float x = rx * cosf(theta);  //calculate the x component
        float y = ry * sinf(theta);  //calculate the y component

        glVertex2f(x + cx, y + cy);  //output vertex

        }
        glEnd();
}
void init(void)
{
        glClearColor (0.0, 0.0, 0.0, 0.0);
         glOrtho(-20.0, 20.0, -20.0, 20.0, -1.0, 1.0);
}
```

```
void display(void)
{

    glClear(GL_COLOR_BUFFER_BIT);
        glPushMatrix();

    glColor3f(1.0f, 1.0f, 1.0f);  // Set color to white
    glBegin(GL_QUADS);  // Draw a square
        glVertex3d(2.0, 5.0, 0.0);
        glVertex3d(10.0, 5.0, 0.0);
        glVertex3d(10.0, 10.0, 0.0);
        glVertex3f(2.0, 10.0, 0.0);
    glEnd();

        glColor3f(0.7, 0.2, 0.3);
    DrawCircle(6,7.5f,1.5f,1.5f,100);
    glPopMatrix();

    glColor3f(0.3f, 0.0f, 0.0f);
    glBegin(GL_QUADS);  // Draw a square
        glVertex3d(1.0, -8.0, 0.0);
        glVertex3d(2.0, -8.0, 0.0);
        glVertex3d(2.0, 10.0, 0.0);
        glVertex3f(1.0, 10.0, 0.0);
    glEnd();

    glColor3f(0.9f, 1.0f, 0.3f);
        glBegin(GL_TRIANGLES);
        glVertex3d(1,10,0);
        glVertex3d(2,10,0);
        glVertex3d(1.5f,10.5f,0);
    glEnd();

        glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
        glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize (800, 800);
        glutInitWindowPosition (500, 100);
        glutCreateWindow ("Circle");
        init();
    glutDisplayFunc(display);
        glutMainLoop();
        return 0;
}
```

5 Output

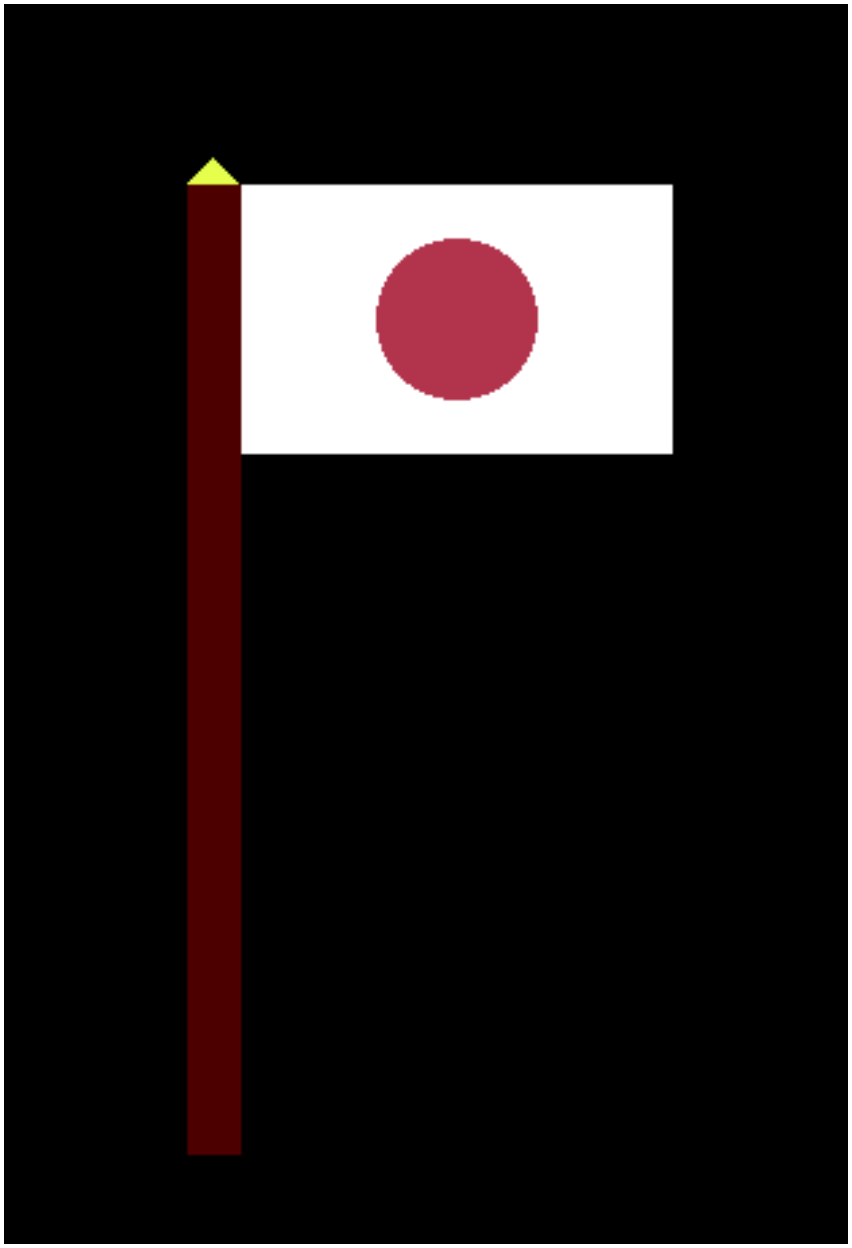

Figure 2: Output of Flag Using OpenGL

6 Discussion

This experiment illustrates how basic OpenGL drawing primitives can be combined to create a meaningful graphical object. The use of an orthographic projection simplifies positioning and scaling in a 2D space. Implementing a custom circle-drawing function enhances understanding of how curves are approximated using line segments in computer graphics.

The experiment also demonstrates effective use of color composition and coordinate transformation. Overall, it provides practical insight into structured graphical programming and reinforces theoretical concepts taught in computer graphics lectures.

7 Conclusion

In this lab experiment, a 2D flag was successfully drawn using OpenGL and GLUT. The task improved understanding of OpenGL initialization, coordinate systems, geometric primitives, and custom shape generation. This experiment serves as a strong foundation for advanced graphics topics such as transformations, animations, and interactive graphics applications.