

# **Lab Report 3**

Name: **Arnob Das Shacha**

ID: **221-15-5277**

Section: **61\_D1**

Course Code: **CSE 422**

Experiment Name: Drawing different objects using circle.

Submitted To:

**Sayed Tasnim Alvi**

Senior Lecturer

Dept of CSE

Daffodil International University

## Title

Drawing different objects using circle

## Introduction

Computer Graphics allows us to represent real-world objects visually using mathematical techniques. OpenGL is one of the most powerful graphics libraries used to create 2D and 3D objects. In this program, different objects are drawn using circles and polygons. Circles are generated using trigonometric functions while polygons are drawn using sets of vertices. This assignment demonstrates how multiple shapes (circles, triangles, square-like polygons) can be combined together to form a complete graphical scene in OpenGL using GLUT. The main objective of the program is to understand how geometric shapes are created, colored, and displayed using basic OpenGL primitives.

## Contents

**init():** This function initializes OpenGL settings.

It sets the background color and defines the 2D viewing coordinate system using gluOrtho2D().

**drawCircle():** This function draws a filled circle using trigonometric functions.

It loops from  $0^\circ$  to  $360^\circ$  and plots the x,y points using cosine and sine.

**drawCircleRed():** Same logic as drawCircle(), but draws a red colored circle.

**drawCircleBlue():** Draws a filled blue colored circle using the same circular plotting technique.

**drawCircleGreen():** Draws a filled green circle using sine-cosine based points.

**drawTriangle():** Draws a quadrilateral shape (rectangle-like) using a polygon.

Used to represent a vertical block/structure on the left side.

**drawSideTriangle():** Draws another rectangular shape similar to the previous one, but positioned differently.

**drawSideSquare():** Draws a square-like polygon on the right side of the window.

**display():** This is the main rendering function.

It clears the screen and calls all other shape-drawing functions to display the final scene.

Finally, glFlush() forces the drawing to appear on screen.

## Code

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

void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0); // Set background to blue
    gluOrtho2D(0, 500, 0, 500);      // Set coordinate system
}

// Function to draw a circle
void drawCircle(int h, int k, int rx, int ry)
{
    glColor3f(1.0, 1.0, 1.0); // Yellow color
    glBegin(GL_POLYGON);
    for (int i = 0; i <= 360; i++)
    {
        glVertex2f(h + rx * cos(3.14159 * i / 180), k + ry * sin(3.14159 * i / 180));
    }
    glEnd();
}

void drawCircleRed(int h, int k, int rx, int ry)
{
    glColor3f(1.0, 0.0, 0.0); // Yellow color
    glBegin(GL_POLYGON);
    for (int i = 0; i <= 360; i++)
    
```

```

{
    glVertex2f(h + rx * cos(3.14159 * i / 180), k + ry * sin(3.14159 * i / 180));
}
glEnd();
}

void drawCircleBlue(int h, int k, int rx, int ry)
{
    glColor3f(0.0, 0.0, 1.0); // Yellow color
    glBegin(GL_POLYGON);
    for (int i = 0; i <= 360; i++)
    {
        glVertex2f(h + rx * cos(3.14159 * i / 180), k + ry * sin(3.14159 * i / 180));
    }
    glEnd();
}

void drawCircleGreen(int h, int k, int rx, int ry)
{
    glColor3f(0.0, 1.0, 0.0); // Yellow color
    glBegin(GL_POLYGON);
    for (int i = 0; i <= 360; i++)
    {
        glVertex2f(h + rx * cos(3.14159 * i / 180), k + ry * sin(3.14159 * i / 180));
    }
    glEnd();
}

// Function to draw a triangle

void drawTriangle(void)

```

```
{  
    glColor3f(1.0, 1.0, 0.0);  
    glBegin(GL_POLYGON);  
    glVertex2f(50, 100);  
    glVertex2f(80, 100);  
    glVertex2f(80, 0);  
    glVertex2f(50, 0);  
    glEnd();  
}  
  
}
```

```
void drawSideTriangle(void)  
{  
    glColor3f(1.0, 1.0, 0.0);  
    glBegin(GL_POLYGON);  
    glVertex2f(200, 100);  
    glVertex2f(230, 100);  
    glVertex2f(230, 0);  
    glVertex2f(200, 0);  
    glEnd();  
}
```

```
void drawSideSquare(void)  
{  
    glColor3f(0.0, 1.0, 0.5);  
    glBegin(GL_POLYGON);  
    glVertex2f(180, 250);  
    glVertex2f(270, 250);  
    glVertex2f(270, 100);  
}
```

```
glVertex2f(180, 100);
glEnd();
}

// Function to display both the circle and triangle
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT); // Clear screen (sets background to blue)

    drawCircleBlue(70, 135, 50, 80); // Draw the circle at (250, 350)
    drawTriangle();
    drawSideTriangle();
    drawSideSquare();

    drawCircleRed(222, 222, 12, 12);
    drawCircleBlue(222, 170, 12, 12);
    drawCircle(222, 120, 12, 12);

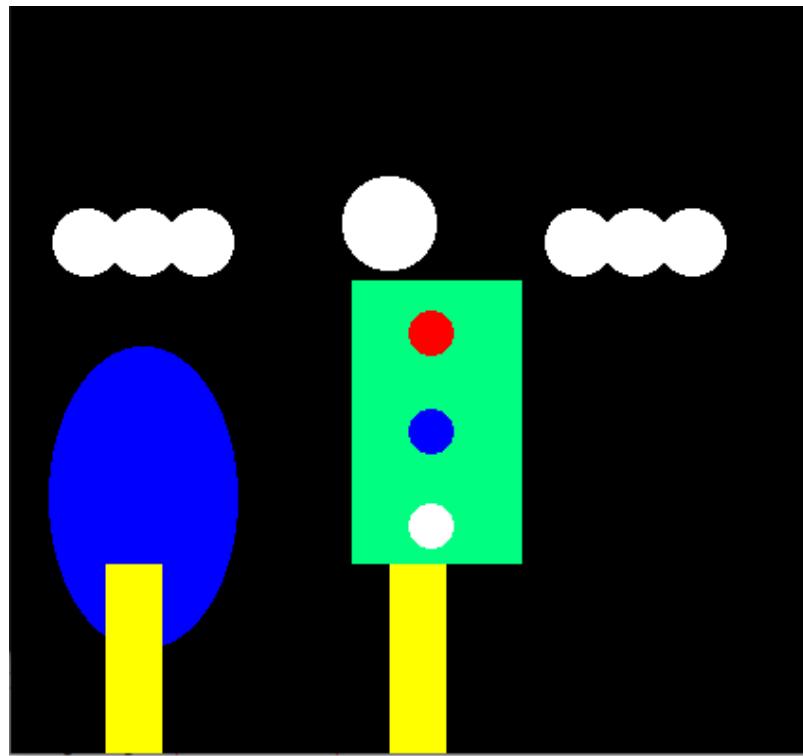
    drawCircle(40, 270, 18, 18);
    drawCircle(70, 270, 18, 18);
    drawCircle(100, 270, 18, 18);

    drawCircle(300, 270, 18, 18);
    drawCircle(330, 270, 18, 18);
    drawCircle(360, 270, 18, 18);

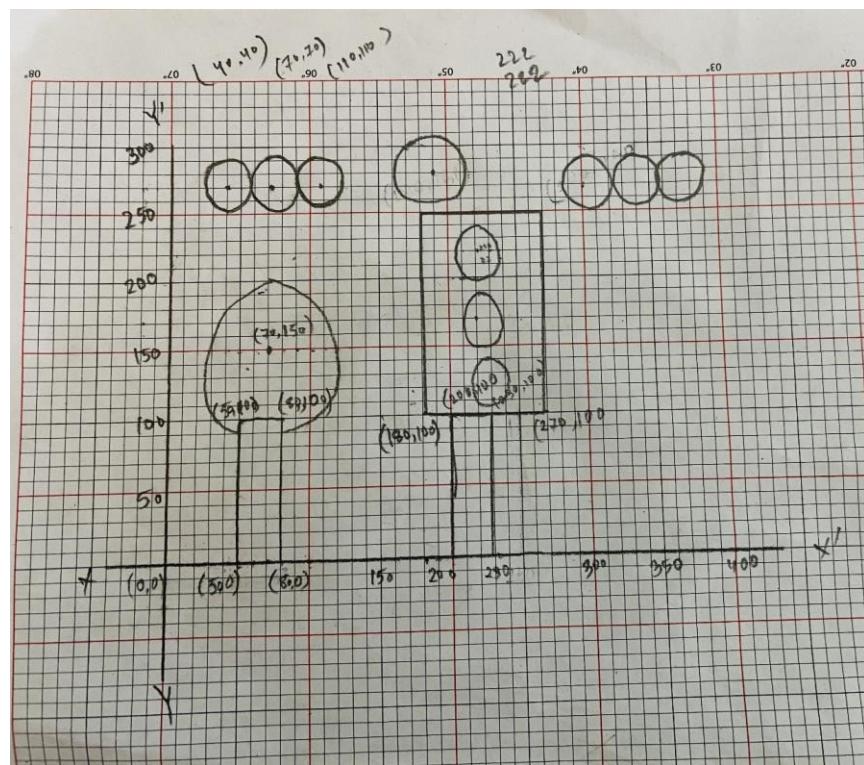
    drawCircle(200, 280, 25, 25);
```

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

## Output



## Graph



## Discussion

In this project, different objects are displayed using circle-based drawing techniques. The `drawCircle()` function draws a filled ellipse by repeatedly plotting points between  $0^\circ$  and  $360^\circ$ , using cosine and sine functions for X and Y coordinates. Separate color functions (red, blue, green, white) are used to visually differentiate shapes.

The scene contains several graphical objects:

- A large blue circle acting as the main background object.
- Multiple small circles arranged in different positions.
- Two rectangular blocks representing vertical shapes.
- A green square drawn on the right side.
- A mixture of circles forming decorative patterns.