Title

Drawing different object using Circle.

Introduction

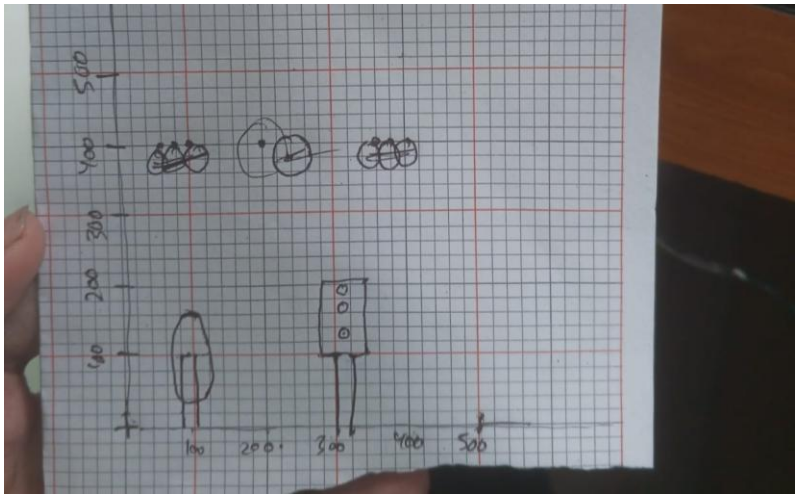
In this lab task, I used OpenGL with GLUT to draw a Scenery with a tree, sun, cloud and traffic light shape using basic geometric primitives. The program demonstrates how to create 2D graphics by combining Circles and polygons with different colors. Through this task, we learned how to initialize an OpenGL window, set background and object colors, and use functions like glBegin(), glVertex3f(), and glFlush() to display objects on the screen. This experiment helps in understanding the fundamentals of computer graphics and how shapes are formed using coordinates in OpenGL.

Contents

In this lab task :

1. Functions used
   - ‘glClear()’ – clears the screen.
   - ‘glColor3f()’ – sets color.
   - ‘glBegin()’/ ‘glEnd()’ – start and end shape drawing.
   - ‘glVertex3f()’ – defines shape corners.
   - ‘glFlush()’ – displays the drawing.
   - ‘drawCircle()’ – draws circle.
2. Shapes used:
   - Circle – Sun, clouds, tree leafs, cloud.
   -
   - Polygons (rectangles)– tree body, traffic light stand & body.

Graph



Code

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

void init(void)
{
   glClearColor(0.0, 0.0, 0.0, 0.0); // Black background as per original values
   gluOrtho2D(0, 500, 0, 500);       // Set coordinate system
}

void drawCircle(int h, int k, int rx, int ry)
{
   glColor3f(1.0, 1.0, 0.0);  // Yellow color (1.0, 1.0, 1.0 is White)
   glBegin(GL_POLYGON);
   for (int i = 0; i <= 360; i++) {
      // Corrected the line by replacing non-breaking spaces
      // The angle needs to be in radians for sin/cos
      float angle = 3.14159 * i / 180.0;
      glVertex2f(h + rx * cos(angle), k + ry * sin(angle));
   }
   glEnd();

}
void drawCircle1(int h, int k, int rx, int ry)
{
   glColor3f(1.0, 0.0, 0.0);  // Yellow color (1.0, 1.0, 1.0 is White)
```

```
    glBegin(GL_POLYGON);
    for (int i = 0; i <= 360; i++) {
        // Corrected the line by replacing non-breaking spaces
        // The angle needs to be in radians for sin/cos
        float angle = 3.14159 * i / 180.0;
        glVertex2f(h + rx * cos(angle), k + ry * sin(angle));
    }
    glEnd();

}
void drawCircle3(int h, int k, int rx, int ry)
{
    glColor3f(0.0, 1.0, 0.0);  // Yellow color (1.0, 1.0, 1.0 is White)
    glBegin(GL_POLYGON);
    for (int i = 0; i <= 360; i++) {
        // Corrected the line by replacing non-breaking spaces
        // The angle needs to be in radians for sin/cos
        float angle = 3.14159 * i / 180.0;
        glVertex2f(h + rx * cos(angle), k + ry * sin(angle));
    }
    glEnd();

}
void drawCirclew(int h, int k, int rx, int ry)
{
    glColor3f(1.0, 1.0, 1.0);  // Yellow color (1.0, 1.0, 1.0 is White)
    glBegin(GL_POLYGON);
```

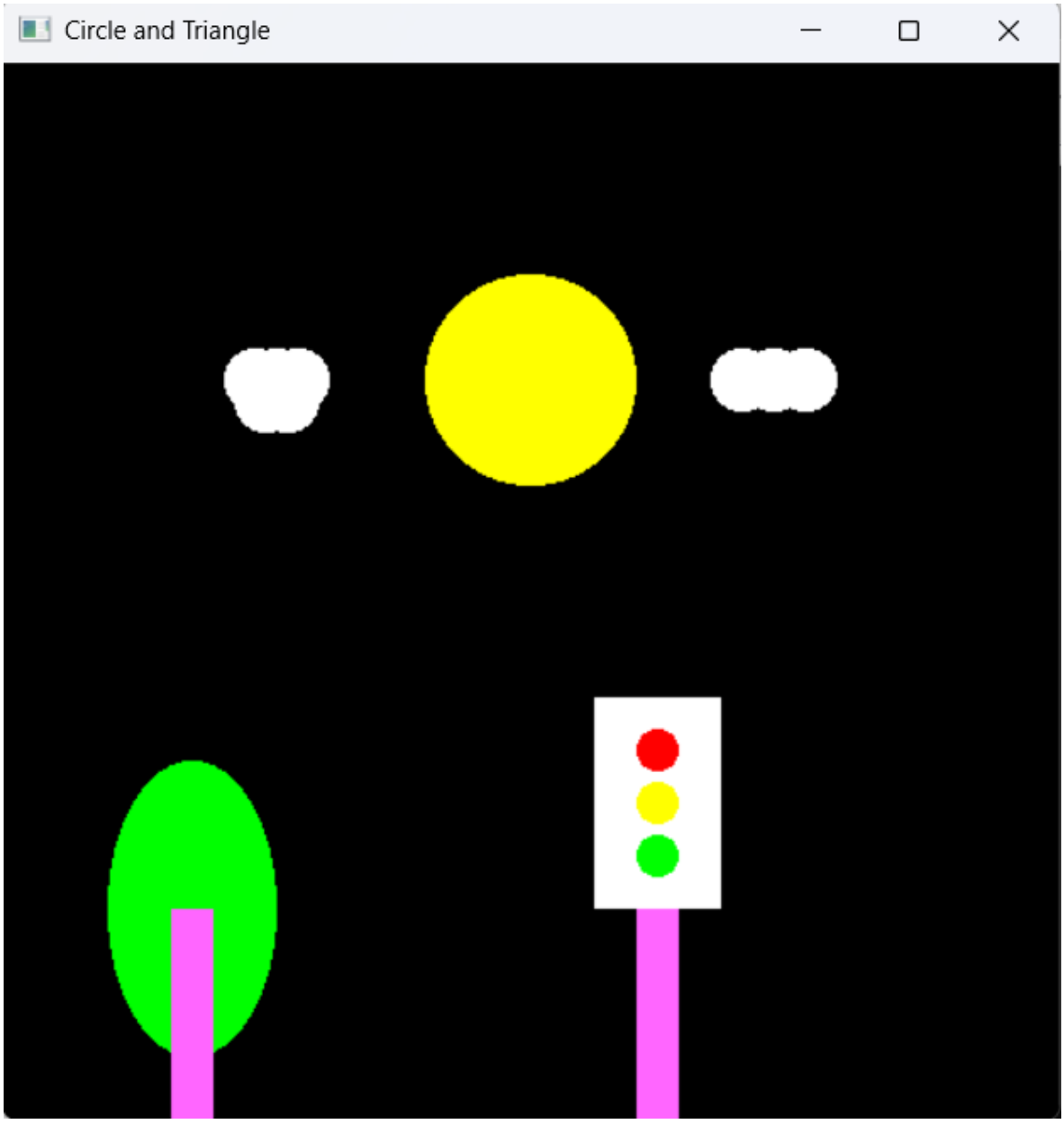
```
    for (int i = 0; i <= 360; i++) {
        // Corrected the line by replacing non-breaking spaces
        // The angle needs to be in radians for sin/cos
        float angle = 3.14159 * i / 180.0;
        glVertex2f(h + rx * cos(angle), k + ry * sin(angle));
    }
    glEnd();

}

// Function to draw a triangle
void drawTriangle(void)
{
    glColor3f(1.0, 1.0, 0.0);  // Yellow color
    glBegin(GL_TRIANGLES);
        glVertex2i(150, 150);  // Bottom-left
        glVertex2i(350, 150);  // Bottom-right
        glVertex2i(250, 300);  // Top peak
    glEnd();
}

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

```
drawCircle(250, 350, 50, 50); // Draw the circle at (250, 350)
//drawTriangle();               // Draw the triangle at predefined coordinates
//r cloud
drawCirclew(350, 350, 15, 15);
drawCirclew(365, 350, 15, 15);
drawCirclew(380, 350, 15, 15);
//l cloud

drawCirclew(120, 350, 15, 15);
drawCirclew(130, 350, 15, 15);
drawCirclew(140, 350, 15, 15);
drawCirclew(125, 340, 15, 15);
drawCirclew(135, 340, 15, 15);

// tree
drawCircle3(90, 100, 40,70);
glColor3f(1.0, 0.4, 1.7);
    glBegin(GL_POLYGON);
    glVertex3f (80, 0, 0);
    glVertex3f (80, 100, 0);
    glVertex3f (100, 100, 0);
    glVertex3f (100, 0, 0);
    glEnd();

//trafic light

    //lightbox
```

```
        glColor3f(1.0, 1.4, 1.7);
            glBegin(GL_POLYGON);
            glVertex3f (280, 100, 0);
            glVertex3f (280, 200, 0);
            glVertex3f (340, 200, 0);
            glVertex3f (340, 100, 0);
            glEnd();
        //stand
        glColor3f(1.0, 0.4, 1.7);
        glBegin(GL_POLYGON);
        glVertex3f (300, 0, 0);
        glVertex3f (300, 100, 0);
        glVertex3f (320, 100, 0);
        glVertex3f (320, 0, 0);
        glEnd();
        drawCircle3(310, 125, 10,10);
        drawCircle(310, 150, 10,10);
        drawCircle1(310, 175, 10,10);



    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();

    return 0; // Added return 0 for good practice
}
```

Output



Discussion

In this lab task, I successfully created a robot using basic OpenGL functions. The Scenery was drawn by combining simple shapes such as circles and rectacgles for the traffic lights, cloud, sun, tree leaves and polygons for the tree body, traffic light body. Each part was given a different color using glColor3f() to make the figure more visually clear. The program used glBegin() and glEnd() to define shapes, while glVertex3f() specified their coordinates. Overall, this lab demonstrated how OpenGL can be used to design graphical objects using coordinating geometry & color control.