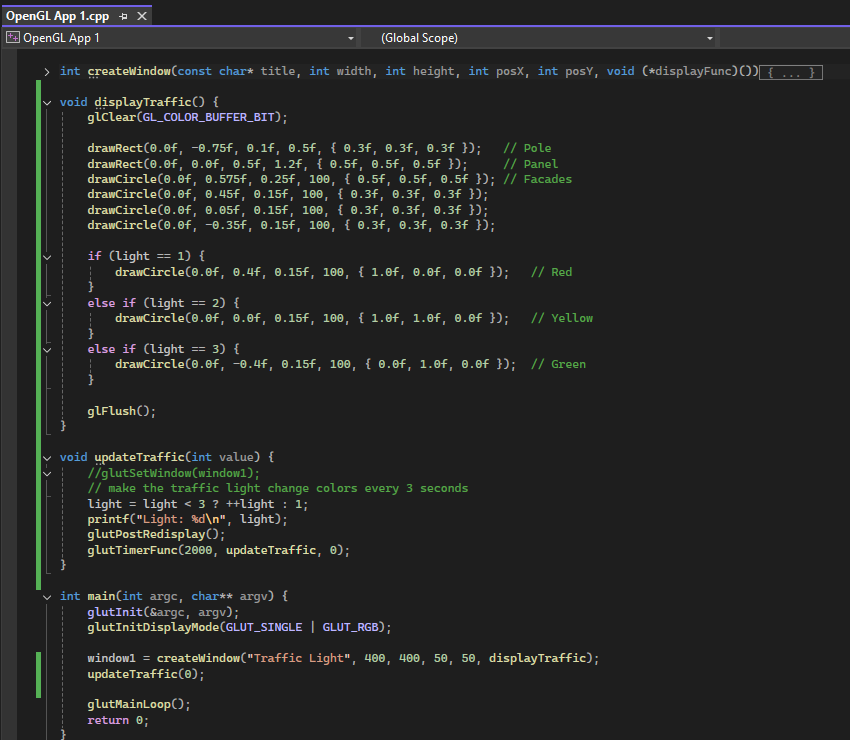
**Homework 3 Report**

D11315807

Ardiawan Bagus Harisa

Department of CSIE

In this traffic light homework, I draw a traffic light including its pole and façade. The light color will change after every two seconds. Instead of blocking the color as the animation mechanism, I draw the red, green, and yellow color based on a global variable.

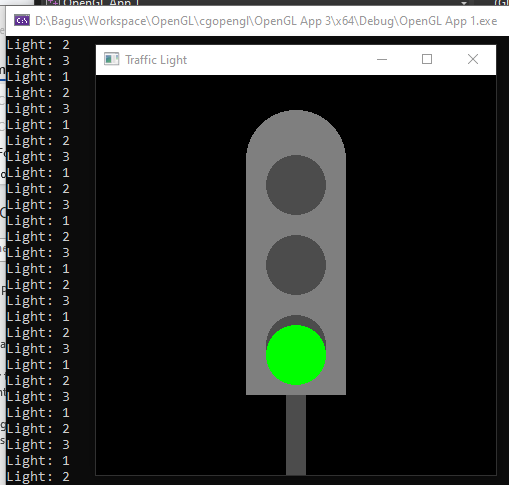
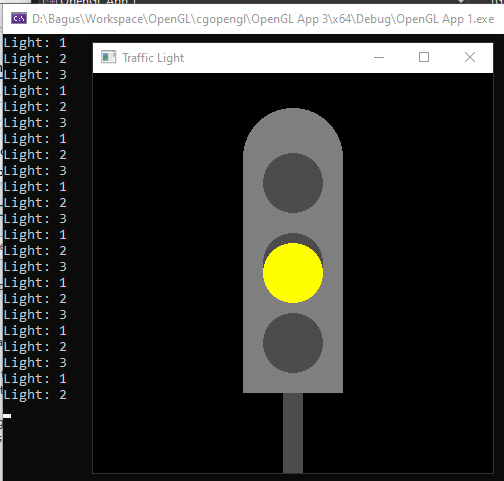
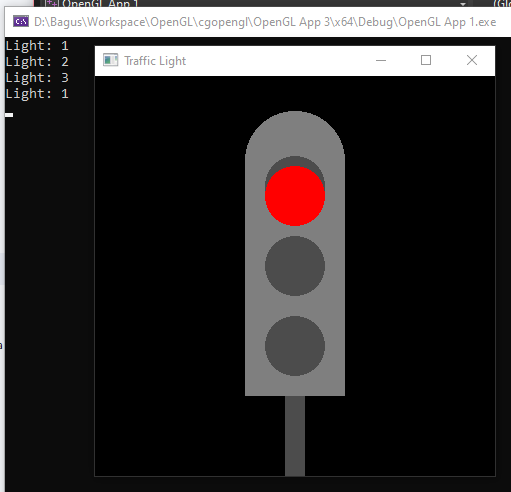
****

As you can see, I create a window that will call the displayTraffic() function to draw the traffic light animation, in which will be updated using updateTraffic() function.

First, I draw the pole and panel using drawRect() function that I have created on the previous homework. The parameters I use on the functions are: center of *x, center of y, width, height, and color*. I also draw the main façade for the main panel and three façades for each light with center of *x, center of y, radius, number of segments, and color* as the parameters.

Second, according to the value of global variable: *light*, I draw the correct color accordingly.

Finally, I update the value in light simply by add 1 value for every 2000ms or 2 seconds.

****

Notes:

* I push my code here: <https://github.com/ardiawanbagusharisa/cgopengl>

The complete code:

#include <GL/glew.h>

#include <GL/freeglut.h>

#include <cmath>

#include <stdio.h>

#include "OpenGL App 1.h"

const float PI = 3.14159265359f;

struct Color {

float r, g, b;

};

int window1, window2;

int light = 0;

void drawRect(float cx, float cy, float sizeX, float sizeY, Color color) {

glColor3f(color.r, color.g, color.b);

glBegin(GL\_QUADS); // Start drawing a square

glVertex2f(cx - sizeX / 2, cy - sizeY / 2); // Change the method to be more parametric

glVertex2f(cx + sizeX / 2, cy - sizeY / 2);

glVertex2f(cx + sizeX / 2, cy + sizeY / 2);

glVertex2f(cx - sizeX / 2, cy + sizeY / 2);

glEnd();

}

void drawCircle(float cx, float cy, float r, int segments, Color color) {

glColor3f(color.r, color.g, color.b);

glBegin(GL\_TRIANGLE\_FAN);

//glVertex2f(0.0f, 0.0f);

for (int i = 0; i <= segments; i++) {

float theta = 2.0f \* PI \* float(i) / float(segments); // Compute the radian angle

float x = r \* cosf(theta); // Set coordinates of points on the perimeter of the circle using polar to cartesian

float y = r \* sinf(theta);

glVertex2f(x + cx, y + cy);

}

glEnd();

}

int createWindow(const char\* title, int width, int height, int posX, int posY, void (\*displayFunc)()) {

glutInitWindowSize(width, height);

glutInitWindowPosition(posX, posY);

int windowID = glutCreateWindow(title);

glewInit();

glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Black background

glutDisplayFunc(displayFunc); // Register the display function

return windowID;

}

void displayTraffic() {

glClear(GL\_COLOR\_BUFFER\_BIT);

drawRect(0.0f, -0.75f, 0.1f, 0.5f, { 0.3f, 0.3f, 0.3f }); // Pole

drawRect(0.0f, 0.0f, 0.5f, 1.2f, { 0.5f, 0.5f, 0.5f }); // Panel

drawCircle(0.0f, 0.575f, 0.25f, 100, { 0.5f, 0.5f, 0.5f }); // Facades

drawCircle(0.0f, 0.45f, 0.15f, 100, { 0.3f, 0.3f, 0.3f });

drawCircle(0.0f, 0.05f, 0.15f, 100, { 0.3f, 0.3f, 0.3f });

drawCircle(0.0f, -0.35f, 0.15f, 100, { 0.3f, 0.3f, 0.3f });

if (light == 1) {

drawCircle(0.0f, 0.4f, 0.15f, 100, { 1.0f, 0.0f, 0.0f }); // Red

}

else if (light == 2) {

drawCircle(0.0f, 0.0f, 0.15f, 100, { 1.0f, 1.0f, 0.0f }); // Yellow

}

else if (light == 3) {

drawCircle(0.0f, -0.4f, 0.15f, 100, { 0.0f, 1.0f, 0.0f }); // Green

}

glFlush();

}

void updateTraffic(int value) {

//glutSetWindow(window1);

// make the traffic light change colors every 3 seconds

light = light < 3 ? ++light : 1;

printf("Light: %d\n", light);

glutPostRedisplay();

glutTimerFunc(2000, updateTraffic, 0);

}

int main(int argc, char\*\* argv) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

window1 = createWindow("Traffic Light", 400, 400, 50, 50, displayTraffic);

updateTraffic(0);

glutMainLoop();

return 0;

}