Homework 1 Report

D11315807

Ardiawan Bagus Harisa

Department of CSIE

In this homework project, I do not read the sample code from TA on Moodle, therefore I apologize that my implementation may be varied from the sample.

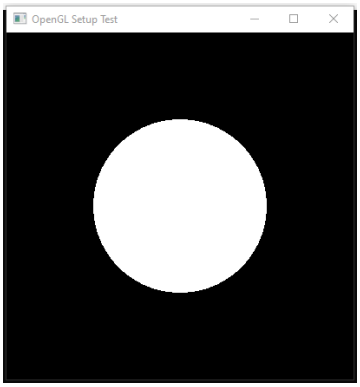
1. Draw a Crescent Moon

First, I modify the sample code from the first meeting and create a drawCircle() functions. I set the color of the vertex to be white (RGB 1,1,1). Then using a loop, I calculate the x, y coordinates on the perimeter of the circle, by converting the polar coordinate into cartesian. Through that loop, we draw the vertex from x, y = 0,0 to x, y calculated from the loop, relative to the center of circle.

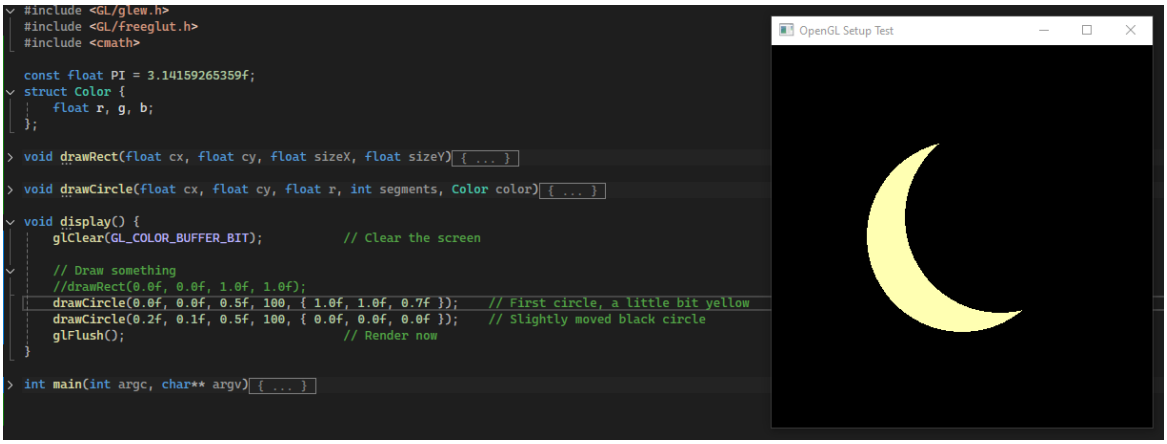
This is the drawCircle() function:

```
void drawCircle(float cx, float cy, float r, int segments) {
    glColor3f(1.0f, 1.0f, 1.0f);    // Set color to white
    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();
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT);           // Clear the screen

    // Draw something
    //drawRect(0.0f, 0.0f, 1.0f, 1.0f);
    drawCircle(0.0f, 0.0f, 0.5f, 100);
    glFlush();                              // Render now
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(400, 400);
    glutCreateWindow("OpenGL Setup Test");
    glewInit();
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);       // Black background
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

As you may see, I just modify the setup sample code, and modified the display() function. There, I just call the drawCircle() inside the display() where it will also be called in main(). Then, this is the resulting circle:



So, the next step is just to draw another circle with black color above the first one. But then, I just realized that the color variable in original OpenGL is not provided. So, I just create the Color struct, therefore I can call drawCircle() in a more convenient way. I also changed the moon's color. This is the final result:



2. Draw Smiley Face

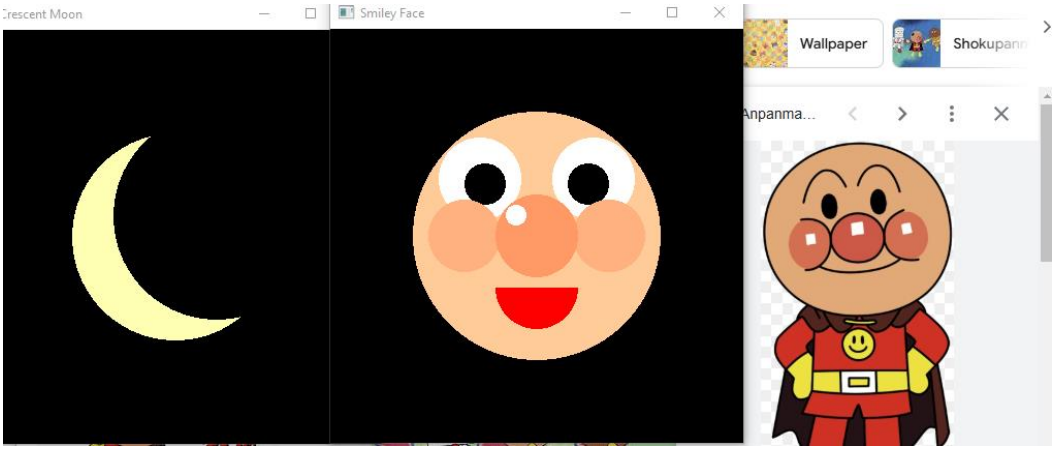
For the second project, again, I just modify the first project because I want to work in the same project. Instead, I just create multiple windows in a project. First, I create an integer variable to hold the window's ID. Then, create the function to initialize the window, where I can later pass the parameters like size, position, and what drawing function I need to call. Finally, I just call the window function to the main function. With this, I don't need to create two projects.

```
int window1, window2;

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

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

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 displayMoon() { ... }

void displaySmiley() { ... }

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

    window1 = createWindow("Crescent Moon", 400, 400, 50, 50, displayMoon);
    window1 = createWindow("Smiley Face", 400, 400, 450, 50, displaySmiley);
    glutMainLoop();
    return 0;
}
```

With this displaySmiley() function, I aim to draw Anpanman.

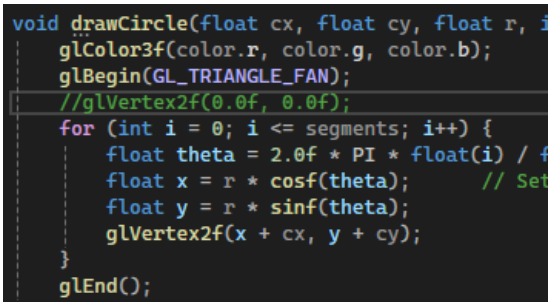
```
void displaySmiley() {
    glClear(GL_COLOR_BUFFER_BIT);           // Clear the screen
    drawCircle(0.0f, 0.0f, 0.6f, 100, { 1.0f, 0.8f, 0.6f });        // Main face
    drawCircle(-0.275f, 0.275f, 0.2f, 100, { 1.0f, 1.0f, 1.0f });   // Left eye
    drawCircle(-0.25f, 0.25f, 0.1f, 100, { 0.0f, 0.0f, 0.0f });
    drawCircle(0.275f, 0.275f, 0.2f, 100, { 1.0f, 1.0f, 1.0f });    // Right eye
    drawCircle(0.25f, 0.25f, 0.1f, 100, { 0.0f, 0.0f, 0.0f });
    drawCircle(0.0f, -0.25f, 0.2f, 100, { 1.0f, 0.0f, 0.0f });      // Mouth
    drawRect(0.0f, -0.15f, 0.4f, 0.2f, { 1.0f, 0.8f, 0.6f });
    drawCircle(-0.35f, 0.0f, 0.175f, 100, { 1.0f, 0.7f, 0.5f });    // Cheeks
    drawCircle(0.35f, 0.0f, 0.175f, 100, { 1.0f, 0.7f, 0.5f });
    drawCircle(0.0f, 0.0f, 0.2f, 100, { 1.0f, 0.6f, 0.4f });        // Nose
    drawCircle(-0.1f, 0.1f, 0.05f, 100, { 1.0f, 1.0f, 1.0f });      // Highlight
    glFlush();
}
```

The results and the reference:


Crescent Moon
Smiley Face
Wallpaper
Anpanma...

Notes:

- I push my code here: https://github.com/ardiawanbagusharisa/cgopengl
- I just realized that I write unnecessary code, after I finish my code, and then re-evaluate using Sohan's.



```
void drawCircle(float cx, float cy, float r, i
    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) / f
        float x = r * cosf(theta);      // Set
        float y = r * sinf(theta);
        glVertex2f(x + cx, y + cy);
    }
    glEnd();
```

The complete code:

```
#include <GL/glew.h>
#include <GL/freeglut.h>
#include <cmath>

const float PI = 3.14159265359f;
struct Color {
     float r, g, b;
};

int window1, window2;

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 displayMoon() {
     glClear(GL_COLOR_BUFFER_BIT);           // Clear the screen

     drawCircle(0.0f,  0.0f,  0.5f,  100,  {  1.0f,  1.0f,  0.7f  });
     // First circle, a little bit yellow
     drawCircle(0.2f,  0.1f,  0.5f,  100,  {  0.0f,  0.0f,  0.0f  });
     // Slightly moved black circle

     glFlush();                                          //  Render
now
}

void displaySmiley() {
     glClear(GL_COLOR_BUFFER_BIT);           // Clear the screen
     drawCircle(0.0f, 0.0f, 0.6f, 100, { 1.0f, 0.8f, 0.6f });
     // Main face
     drawCircle(-0.275f, 0.275f, 0.2f, 100, { 1.0f, 1.0f, 1.0f });
     // Left eye
     drawCircle(-0.25f, 0.25f, 0.1f, 100, { 0.0f, 0.0f, 0.0f });
     drawCircle(0.275f, 0.275f, 0.2f, 100, { 1.0f, 1.0f, 1.0f });
     // Right eye
     drawCircle(0.25f, 0.25f, 0.1f, 100, { 0.0f, 0.0f, 0.0f });
     drawCircle(0.0f, -0.25f, 0.2f, 100, { 1.0f, 0.0f, 0.0f });
     // Mouth
     drawRect(0.0f, -0.15f, 0.4f, 0.2f, { 1.0f, 0.8f, 0.6f });
     drawCircle(-0.35f, 0.0f, 0.175f, 100, { 1.0f, 0.7f, 0.5f });
     // Cheeks
     drawCircle(0.35f, 0.0f, 0.175f, 100, { 1.0f, 0.7f, 0.5f });
     drawCircle(0.0f, 0.0f, 0.2f, 100, { 1.0f, 0.6f, 0.4f });
     // Nose
     drawCircle(-0.1f, 0.1f, 0.05f, 100, { 1.0f, 1.0f, 1.0f });
     // Highlight
     glFlush();

}

int main(int argc, char** argv) {
     glutInit(&argc, argv);
     glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
```

```
     window1 = createWindow("Crescent Moon", 400, 400, 50, 50,
displayMoon);
     window1 = createWindow("Smiley Face", 400, 400, 450, 50,
displaySmiley);
     glutMainLoop();
     return 0;
}
```