2020 CG HW1 Report

一、 環境

完全遵照 HW0 所介紹的環境進行搭建。
OpenGL
Visual Studio 編寫 C++
32 bit GLFW
32 bit freeglut

二、 作業內容

1. 繪製正立方體，貼上材質並使其自動旋轉
2. 手動繪製兩個球體（不可使用 gluSphere()），貼上兩個不同材質並使其自動旋轉
3. 加入光照
4. 設定鍵盤按鍵，使能夠切換三個物體
5. 設定鍵盤或滑鼠，使相機視角能夠改變

三、 實作

1. 建立畫面視窗。

   使用 GLFW 套件，可以輕鬆建立視窗。並在此也設定需要鍵盤、滑鼠、滾輪等功能。

```
GLFWwindow* window;
glfwSetErrorCallback(error_callback);
if (!glfwInit())
    exit(EXIT_FAILURE);
window = glfwCreateWindow(SCR_WIDTH, SCR_HEIGHT, "CG_HW1_TA", NULL, NULL);

if (!window) {
    glfwTerminate();
    exit(EXIT_FAILURE);
}
glfwMakeContextCurrent(window);
glfwSetKeyCallback(window, key_callback);
glfwSetFramebufferSizeCallback(window, framebuffer_size_callback);
glfwSetMouseButtonCallback(window, mouse_button_callback);
glfwSetScrollCallback(window, scroll_callback);
```

2. 載入材質
   (1.) 開啟新的材質暫存器
   (2.) 開啟材質暫存器的綁定模式
   (3.) 做基礎設置
   (4.) 將圖檔載入至 char pointer
   (5.) 將圖檔設置在開啟的材質暫存器上
   (6.) 取消綁定

```
glGenTextures (1, &texture1);
glBindTexture (GL_TEXTURE_2D, texture1);
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
data = stbi_load ("../resources/container.jpg", &w, &h, &nrChannels, 0);
glTexImage2D (GL_TEXTURE_2D, 0, GL_RGB, w, h, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
glBindTexture (GL_TEXTURE_2D, 0);
```

3. 設定光源
   (1.) 開啟光照
   (2.) 設定某光源之 ambient、diffusion、specular 參數
   (3.) 設定某光源之位置
   (4.) 開啟該光源

```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);

GLfloat AmbientLight[] = {0.5, 0.5, 0.5, 1.0};
GLfloat DiffuseLight[] = {1.0, 1.0, 1.0, 1.0};
GLfloat LightPosition[] = {10, 10, 10, 1.0};

glLightfv(GL_LIGHT0, GL_AMBIENT, AmbientLight);
glLightfv(GL_LIGHT0, GL_DIFFUSE, DiffuseLight);
glLightfv(GL_LIGHT0, GL_POSITION, LightPosition);
```

4. 進入 loop

```
while (!glfwWindowShouldClose(window))
```

5. 設定視窗大小

```
int width, height;
glfwGetFramebufferSize(window, &width, &height);
```

6. 設定投影模式

   在此使用的是 perspective 投影，需要設置 y 方向（上下）的視野範圍、橫向比例、投影範圍近平面與相機的距離、遠平面與相機的距離

   （基本上只有設定投影模式時才使用 Projection Matrix，其餘時間都會切換回 ModelView Matrix）

```
//Projection Matrix
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(45.0f, width / (GLfloat)height, 0.1, 1000);
```

7. 設定相機

   (1.) 設定相機位置、相機目標位置、相機正上方之向量

   (2.) 設定投影到相機視窗的位置

   (3.) 開啟深度測試並設定深度值低的在前

(4.) 設定 清除緩存後 的值

```
gluLookAt(xCameraPosition, yCameraPosition, zCameraPosition, xCameraTarget, yCameraTarget, -1.0f, 0.0f, 1.0f, 0.0f)
glViewport(0, 0, width, height);
 glEnable(GL_DEPTH_TEST);
 glDepthFunc(GL_LESS);
glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
glClearDepth(1.0f);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

8. 繪製圖形
   繪製圖形請見第四大點

9. 離開迴圈
   關閉視窗與程式
10. 設定鍵盤按鍵與滑鼠
    原理相當簡單，用 if 判斷式來做控制。只需要上網尋找需要的按鍵定義、輸入與輸出，即可快速應用。
    在我的設計中有：
    (1.) 主鍵盤區域的數字 123 可以切換圖形
    (2.) WASD 控制畫面上左下右
    (3.) 滑鼠滾輪控制前後
    (4.) 滑鼠左鍵停止旋轉
    (5.) 滑鼠右鍵將旋轉方向反過來

四、 繪製圖形

1. 正立方體
   (1.) 實際上是繪製了六個四邊形，各有四個頂點，因此為 24 個頂點所組成
   (2.) 啟動材質並選定要使用的材質暫存器
   (3.) 選擇繪製模式並可開始繪製

```
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texture1);
glBegin(GL_QUADS);
```

   (4.) 設定接下來的節點的法向量
   (5.) 設定材質的座標（圖片座標）
   (6.) 繪製節點
   (7.) 使用 QUADS 模式時，在繪製完四個節點後會自動形成填滿的四邊形

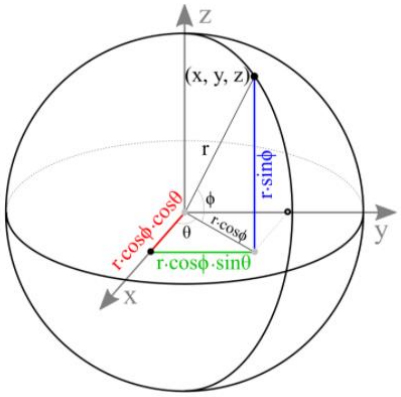
```
glNormal3f(0.0f, 0.0f, -1.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(-0.5f, 0.5f, -0.5f);
```

   (8.) 需要重複上述步驟直到 24 個點（六個面）全部繪製完成
   (9.) 結束繪製並取消綁定材質

```
glEnd();
glDisable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, 0);
```

2. 球形
   (1.) 題目要求不能使用函式庫中的函式，因此需要使用大量四邊形完成球體的繪製
   (2.) 以下為示意圖，圖中左右方向應為 x 軸，上下方向為 y 軸， 前後則為 z 軸。（圖片來源與程式參考）

x 方向為 r * cos(φ) * sin(θ)
y 方向為 r * sin(φ)
z 方向為 r * cos(φ) * cos(θ)



A point on a sphere using sector and stack angles

而其中 φ 與 θ 由經度緯度數量決定，作業要求 360 條經度與 180 條緯度。

   (3.) 於主迴圈外建立了 create_sphere_vertices 函式，完成所有球體節點座標的尋找，並存入 vector 內。需要繪製時只需要到 vector 尋找座標點即可。
   (4.) 繪製球體時與繪製正立方體一樣，需要繪製大量四邊形，組合形成球狀，並在表面貼上材質。（球體材質的黏貼只需要將每個節點，按比例對應到圖片上的座標即可）
3. 繪製圖形的重點提醒
   (1.) 在繪製圖形時，需要給每一個節點設定其法向量，光照才會產生效果。
   (2.) 設定法向量時，不需要有同一個節點該對應哪一個相鄰面的法向量的困擾。以立方體為例，每個節點看似都有三個面相鄰，但實際上六個面中，每個面都使用了 4 個節點，因此總共有 24 個不同節點，只不過最後圖形呈現時，有不少節點都在同一個座標上。因此在繪製該平面的節點時，設定法向量，不需要擔心未來同一個座標節點的法向量會被改變（實際上並非同一個節點）。
4. 旋轉
   (1.) 旋轉只需要使用現存函式即可

```
glRotatef(angle, 1.0, 1.0, 0.0);
```

   (2.) 可以使用時間函式設定旋轉速度

```
float delta_time = glfwGetTime() - time;
```

五、 難點（疑問）與解法

此次最大的問題在於，原本並不了解任何函式，要從 0 開始讀懂架構程式相當不容易。就連立方體是由六個面組成的概念都花了好一陣子才了解，更不用說 OpenGL 特別的「開啟設定」的概念（例如材質暫存器的開啟）。

1. MatrixMode 的使用

   不了解 MatrixMode 的作用，因此不曉得程式順序該是什麼樣子。

   ⇨ 搜尋了相當多資料才了解到 Projection 才是特別的 Matrix 模式，其餘時間都使用 ModelView 即可。

2. 材質的黏貼

   認為材質暫存器開啟後就要貼完一整個四邊形，不知道貼完一個四邊形後下一個四邊形該怎麼辦，也不知道材質圖片座標的意義。

   ⇨ 在想辦法貼球體時，無意間看到圖片本身也有座標，才了解到其實 OpenGL 只是需要我在畫節點前，告訴它會對應到哪個材質的哪個座標點，它就會幫我貼上去。以節點為主而非平面為主。

3. 光線的設定

一開始設定好光源後，發現旋轉中的物體，暗面保持昏暗，亮面仍保持亮面，而不是如想像中的光源固定，轉到面相光源面的方向漸漸變亮。

⇨ 雖然不確定當時做出錯誤情況的主因為何，但我認為是因為 MatrixMode 並沒有放在正確的位置。也嘗試過 Push 與 Pop Matrix，但後來發現那個並不是主因。

六、 結語

由於是首次接觸 OpenGL，因此對於任何函式都相當陌生，必須上網一條一條搜尋，才能漸漸理解其行為模式，前置作業就花了兩到三天。OpenGL 讓我覺得最特別的部分就在於，它許多設定（Settings）都是永久開啟。正常來說設定應該都是偏向單次使用的按鈕，需要設定 A 就使用 A 函式，要設定 B 就使用 B 函式。但 OpenGL 可以開啟此材質暫存器，之後若是有貼材質的動作都是使用該材質，除非將材質取消綁定或是換成綁定別的材質，才會出現改變，有點類似開了一扇門，若是沒有主動關上，它就永遠保持開啟狀態。也因為時間因素，我無法做出更多額外項目，因此放棄加分部分。