計算機圖學 作業一報告

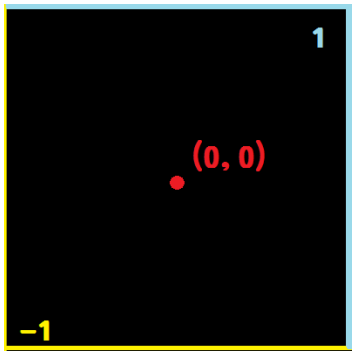
103062234 張克齊

一、 The method of operating program:

執行程式前，要先確定在 CG_HW1 資料夾底下有 ColorModels 這個資料夾，裡面用來存放要讀取的.obj 檔；接著，要修改程式中所開的 filename 陣列和 file_size，若 filename 中寫了 12 個路徑的名稱，則要將 file_size 改成 12，如下圖所示，這樣就能順利執行程式。

```
char filename[][50] = { "ColorModels/al7KC.obj", "ColorModels/blitzcrank_incognito.obj",
                        "ColorModels/texturedknot11KC.obj", "ColorModels/elephant16KC.obj",
                        "ColorModels/frog2KC.obj",  "ColorModels/dragon10KC.obj",
                        "ColorModels/igea17KC.obj",  "ColorModels/Dino20KC.obj",
                        "ColorModels/lion12KC.obj",  "ColorModels/lucy25KC.obj",
                        "ColorModels/armadillo12KC.obj",  "ColorModels/ziggs.obj" };
int file_size = 12;
```

二、 The method of normalization:



<- normalize to [-1, 1] 示意圖

首先，我們要從 traverseColorModel()這個 function 中進行修改，必須如同範例的三角形一樣，創建兩個 GLfloat 的 array 分別儲存 vertices 座標和 colors 座標(x, y, z)，而因為一張圖中是由很多三角形所構成，可以利用 OBJ->numtriangles 取得這張圖由多少個三角形組成，又因為每個三角形有三個點和三個座標，代表存一個三角形需要 9 個數值，因此至少要(OBJ->numtriangles)*9 大小的 array 才能存完所有的點，如下圖。

```
vertices = new float[(OBJ->numtriangles) * 9];
colors = new float[(OBJ->numtriangles) * 9];
```

接著是 normalization 的部分，要分成 translation 和 scaling 兩個步驟來操作。第一個是 translation 的部分，我們要先找出讀進來 OBJ 的中心，方法是找出 x, y 及 z 軸上對於這個 OBJ 最大和最小的 vertex，也就是這個 OBJ 的邊界，並利用 code 中所給的 maxVal 和 minVal 來儲存，而在讀每一個新的 vertex 時要進行 max function 做比較更新值；當我們取到三個軸的邊界後，就能利用 (maxVal+minVal)/2 找出 OBJ 的中心座標 (Midpoint)，在利用一個新的 for 迴圈對所有的點 (vertices array) 進行平移，將 OBJ 移到視窗中心 (0, 0) 的位置，完成 translation 的動作。再來我們要處理 scaling，要注意的是不能直接將整張圖縮放至 [-1, 1]，因為會改變 OBJ 的比例；因此，我們也要利用剛剛得到的邊界 maxVal, minVal 來實現；利用 max function 找出 x, y, z 軸的最長邊界，並將這個邊界控制在 [-1, 1] 間，其他軸則是造著這個比例 (下圖的變數 scale) 來進行 scaling，這樣就能完成 normalization 的動作。下圖為找到 x, y, z 軸 maxVal 和 minVal 後的操作。

```
float mid_x = (maxVal[0] + minVal[0]) / 2;
float mid_y = (maxVal[1] + minVal[1]) / 2;
float mid_z = (maxVal[2] + minVal[2]) / 2;
printf("\nMidpoints: ");
printf("[%f, %f, %f]\n", mid_x, mid_y, mid_z);

for (i = 0; i < now_v; i ++)
{
    vertices[i++] -= mid_x;
    vertices[i++] -= mid_y;
    vertices[i] -= mid_z;
}
```

Translation

```
float max_edge = max(max(maxVal[0] - minVal[0], maxVal[1] - minVal[1]), maxVal[2] - minVal[2]);
float scale = 2 / max_edge;
for (i = 0; i < now_v; i++)
{
    vertices[i] *= scale;
}
```

Scaling

三、 Other implementation:

1. Shader:

在處理只顯示 R、G、B channel 時，可以利用一個 uniform variable 連結 main 與 shader，達到切換 channel 的效果。首先，我們可以先對 shader.frag 進行修改，創建一個 uniform variable 來控制我們想要的模式，並透過取出 vv3color(OBJ 的 color)的[0](R channel)、[1](G channel)、[2](B channel)彆設其他為 0 來達到單色的效果，如下圖。

```
varying vec3 vv3color;
uniform int RGB = 0;
void main() {
    if(RGB == 0)    gl_FragColor = vec4(vv3color, 1.0);
    if(RGB == 1)    gl_FragColor = vec4(vv3color[0], 0, 0, 1);
    if(RGB == 2)    gl_FragColor = vec4(0, vv3color[1], 0, 1);
    if(RGB == 3)    gl_FragColor = vec4(0, 0, vv3color[2], 1);
}
```

接著在 main 的部分，因為在 setShaders()中已經將兩個 shader 彼此 bind，所以我們可以直接建一個 GLint 來取出我們的 uniform variable，所用的函式為 glGetUniformLocation()，如下圖所示，第二個參數雙引號內的 RGB 就是我們在 shader 中創建的變數名稱，這樣就能取出該變數的位置。

```
iLocPosition = glGetAttribLocation (p, "av4position");
iLocColor    = glGetAttribLocation (p, "av3color");
myLoc = glGetUniformLocation(p, "RGB");
```

最後，在進行操作的部分，我們必須先確定 myLoc 不等於-1，才能確定有連接到 shader 的變數；確認後我們就可以利用 glUniform1i 來設定 shader 中 uniform variable 的值，1i 的意思為一維的 int，若想創建不同類型會有不同對應的 glUniform 函式。下圖為按鍵 c 的操作，順序為 normal, R, G, B channel。

```
case 'c': // ** R, G, B channel
    if (myLoc != -1)
    {
        color ++;
        if(color == 1)  glUniform1i(myLoc, 1);
        else if(color == 2) glUniform1i(myLoc, 2);
        else if(color == 3) glUniform1i(myLoc, 3);
        else {
            glUniform1i(myLoc, 0);
            color = 0;
        }
    }
    printf("%d", (int)myLoc);
    break;
```

2. Wireframe/Solid mode:

要切換填滿和三角形模式，可以利用 PDF 中提示的 glPolygonMode() 來實現；只要設定其中的第二個參數，GL_FILL 代表會著色填滿，GL_LINE 代表指連點但不填滿顏色，就能達到三角形的效果，下圖為按鍵 w 的操作。

```
case 'w': // ** mode
    if(judge == 0)  glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    else glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    judge++;
    if (judge > 1) judge = 0;
    break;
```

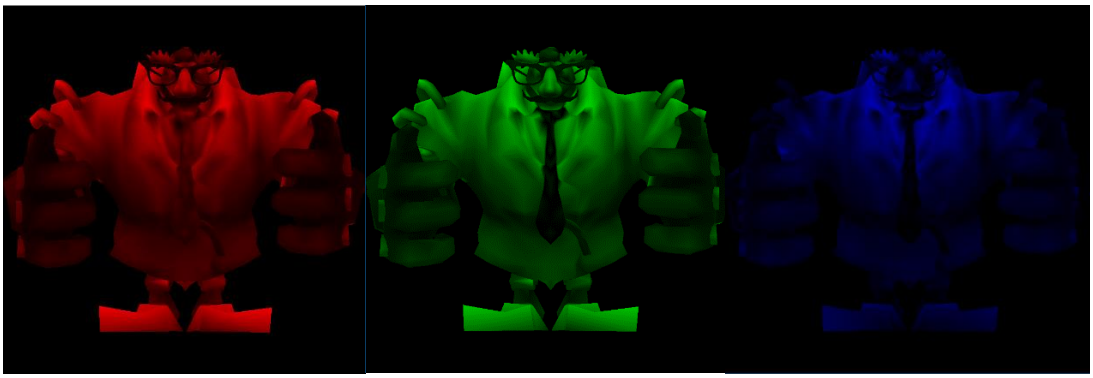
四、 Problem and efforts:

因為助教在一開始 code 架構和 PDF 中都給了很多的提示，所以在 normalization 的部分是很順利進行的，只有因為忘記要 new float array 而稍微卡住。而在改 shader 的部分就比較多挑戰，雖然一開始就有著要從 main 傳一個變數進 shader，但試了一些方法都沒有成功；直到研究 OpenGL 官方 tutorial 中有關 uniform variable 的部分，才成功做出題目的要求。

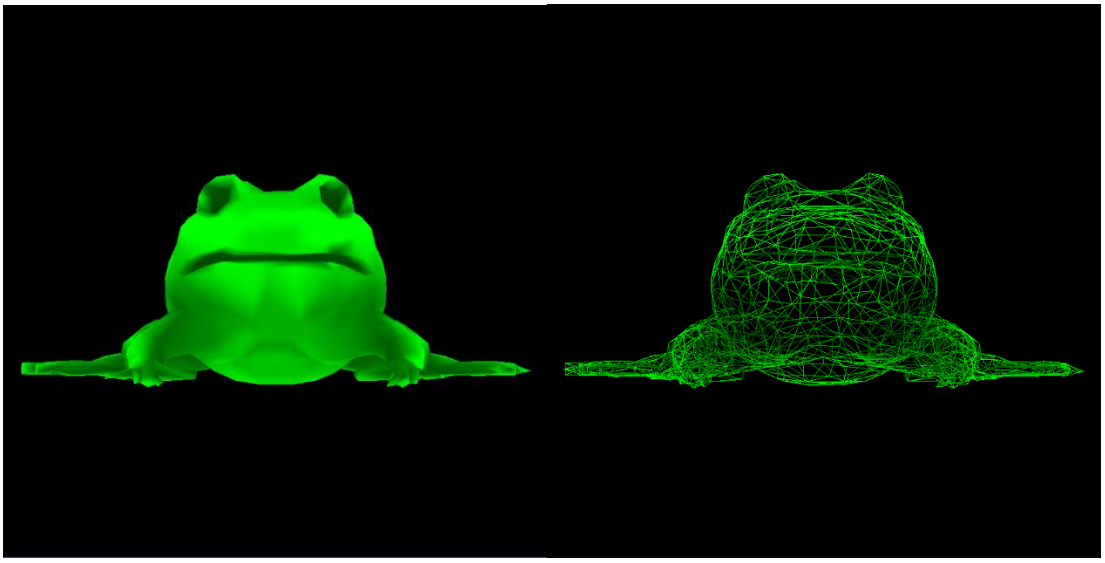
在這次的作業中，前面的部分讓我瞭解了整份 code 的架構，後面則是對 shader 有非常深刻的認識，是一個很棒的學習！

五、 Results:

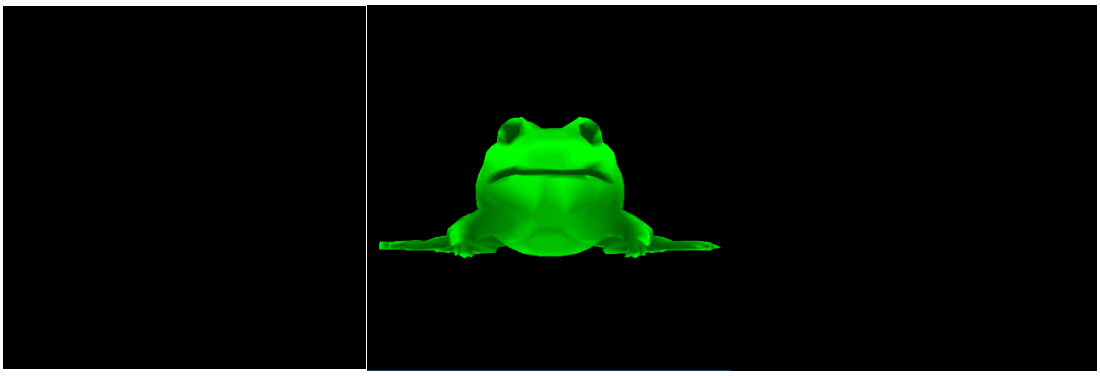
Solid mode　　Wireframe mode



R channel G channel B channel



Solid mode Wireframe mode



R channel G channel B channel