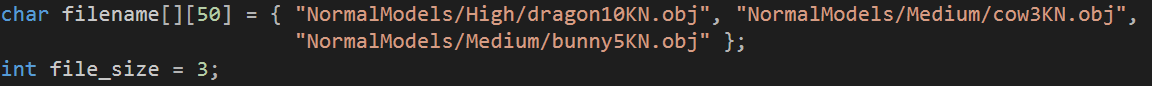
計算機圖學 作業三報告

103062234 張克齊

一、 The method of operating program:

執行程式前，要先確定在CG\_HW3底下有NormalModels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3個路徑的名稱，則要將file\_size改成3，如下圖所示，這樣就能順利執行程式。一開始為perspective mode，按o可切換至orthogonal mode，按p可切回去；而操作如同要求所述，也可按h鍵有詳細的按鍵說明。



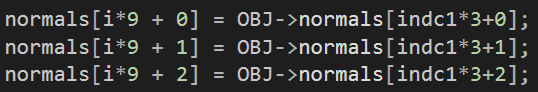
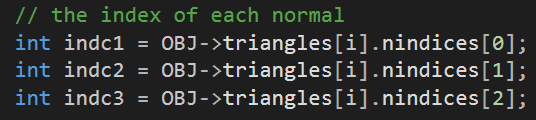
二、 Implementation and problems:

這次的重點為Lighting的實作，我的想法是在main.cpp中處理完所有變數的設定與改變，並把所有要傳的參數透過uniform variable傳入shader；Shader主要處理的是怎麼利用傳進來的參數算出結果，也就是公式的展現。因此，以下分為兩大部分來講解：

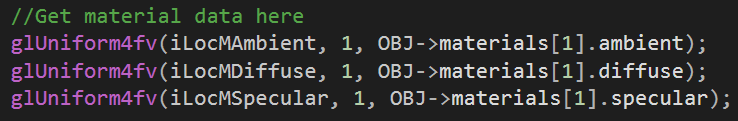
1. 主程式:

(1) OBJ檔的讀取：

這部分的讀取跟前兩次有所不同，因為這次是讀他的Normal vector而非Color，因此在讀取的時候可以利用助教提示的方法取得Normal的index，如下圖所示。

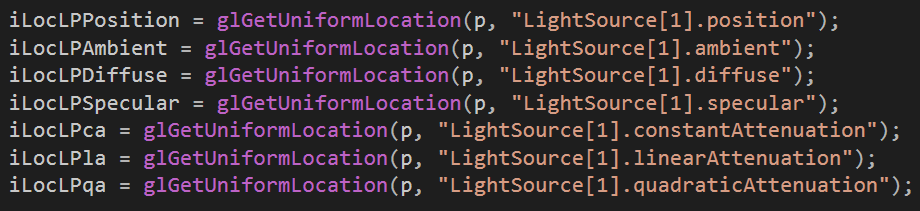


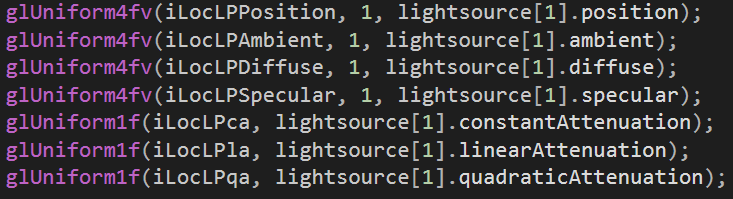
而讀OBJ的過程中也可以獲得material的參數ambient, diffuse, specular，提示中說可以利用group的概念，但我在實作時發現因為讀進來的都是單色系，代表group中其實只有一項有值，因此我利用下圖的方法只取出第一項的參數傳入Shader中，這樣我們就得到所有OBJ檔提供的參數並都傳入Shader做後續的處理。



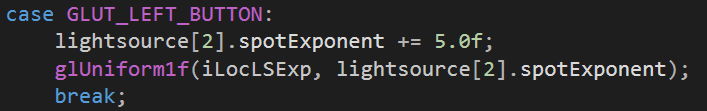
(2) Lightsource的設定與傳遞:

Lightsource的設定中，首先我先在setLightingSource()中設定起始值和3種光源的參數，在這邊我開lightsource[4]的struct陣列，0代表Directional Light，1代表Point Light，2代表Spot Light，3代表初始狀態；接著就透過設定不同的Loc取得shader中uniform的變數並利用glUniformXX()寫入，XX代表不同的型態，將所有一開始設定好的參數全數傳入shader中，圖示擷取部分的抓位置和寫入。



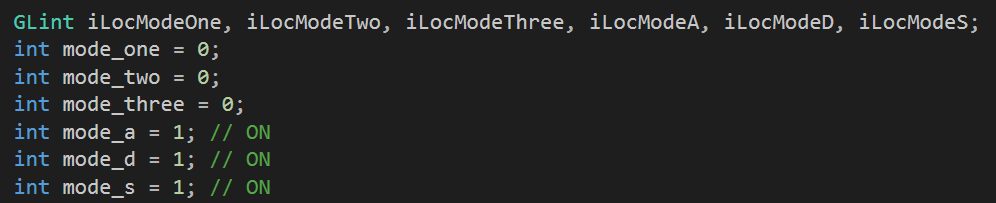
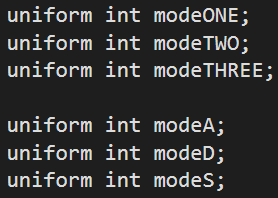


功能實現中，會改到lightsource中的參數，我的方法是直接更改原本存在主程式變數的值，改變後利用glUniformXX()寫入，也就是setLightingSource()我只有在一開始跑一遍，其他都是在更改後立即寫入shader。圖中為增加spot light EXP的操作。



(3) mode控制:

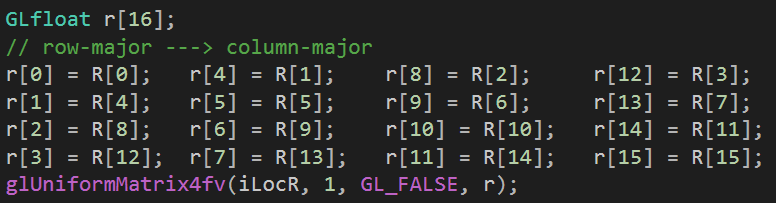
因為這次的作業中會有疊加的效果，因此我的方法是有六種mode參數來控制各項的開關，1代表開，0代表關；分別為三種不同光及三種組成Light的元素，並將6種mode傳入shader，在shader中控制開關時給的值，下圖為6種Mode的傳遞，one, two, three代表三種光源，a是ambient，d是diffuse，s是specular。

(4) Rotate:

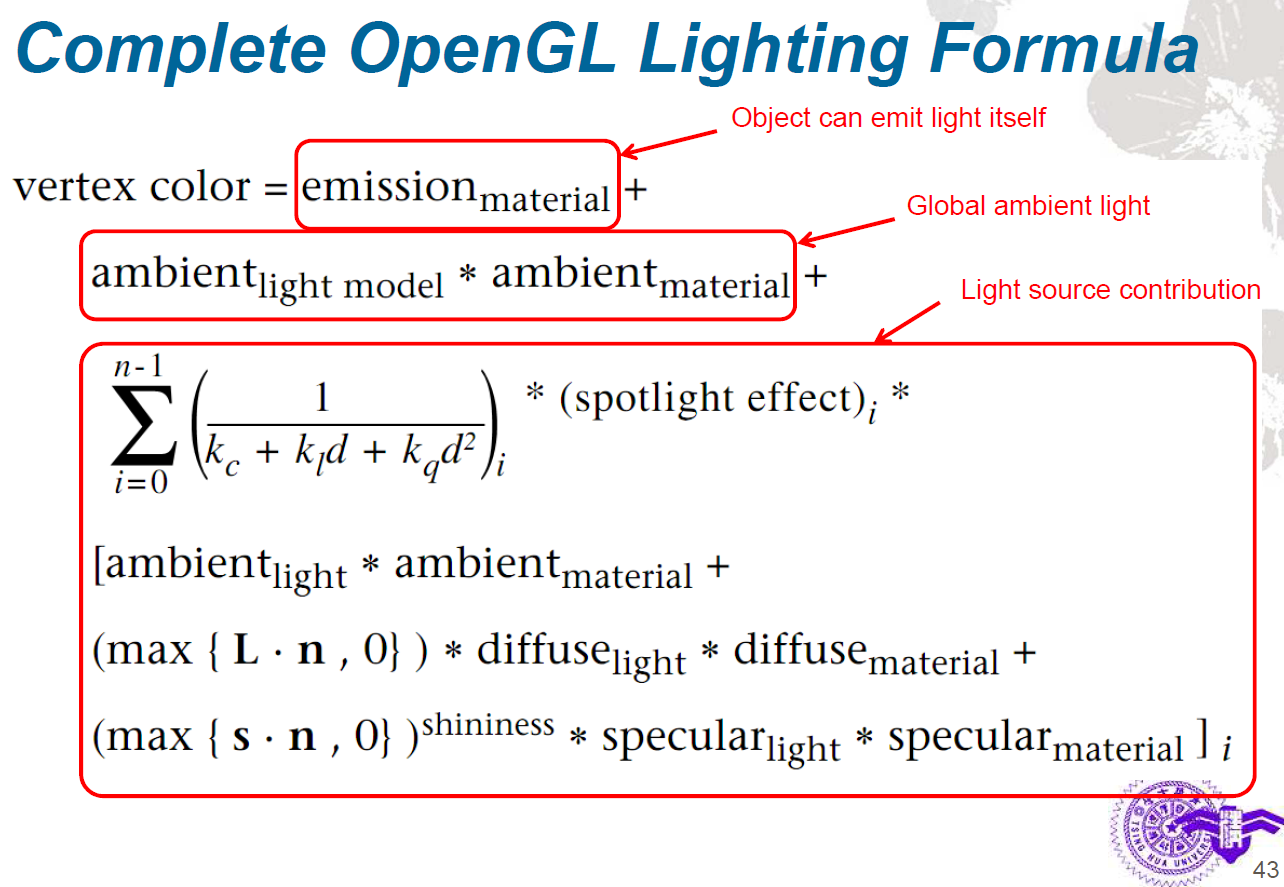
在旋轉的部分Matrix跟上次作業一樣，而因為我只有做y軸的旋轉因此只有一個R，但是只做這樣會有一個問題，就是光源會跟著你的物體旋轉而非固定。因此，從助教得提示中，我們必須傳入R矩陣的invert跟一開始讀入的normal vector相乘才會抵消R的作用，可以讓光源固定在我們設的地方。突圍R的invert的傳遞和normal vector的改變，其中M為y軸旋轉矩陣，傳入方法跟傳mvp一樣要轉成column major。





2. Shader:

大方向是要實作出下圖slide中講的加總公式，而我分別討論三項作法。



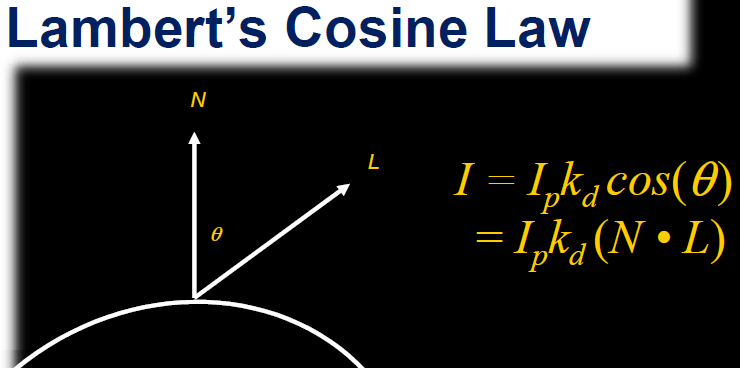
(1) Ambient:

Ambient的部分比較單純，指的是環境光，代表及時在黑暗的環境下還是會有點微量的光，因此在實作時是固定的，不會因為不同的lightsource而改變，圖為我設定的ambient，給的是灰色定值(0.7, 0.7, 0.7, 1)。



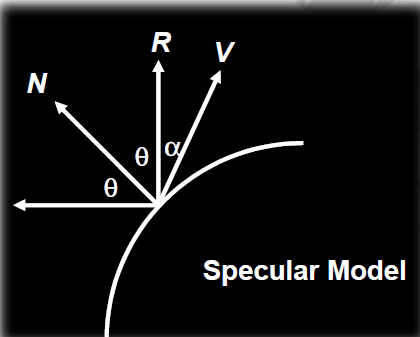
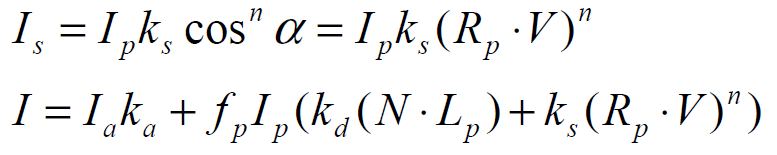
(2) Diffuse:

這個代表著漫反射的效果，也是主要顯示顏色的方法，跟方向有很大的影響，算法為講義寫的那樣，能夠反射出物體主要的顏色。其中，可利用dot()算內積和利用pow()算次方，並記得將指定的向量做normalize()，像是normal vector和光源到點的向量。



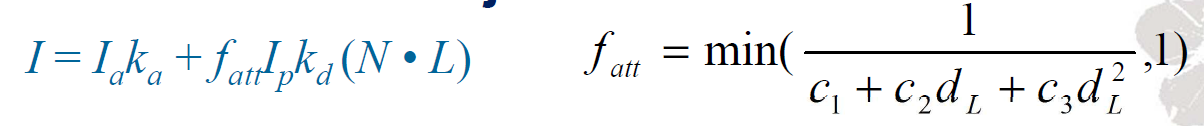
(3) Specular:

為光打在平滑物體上會產生如同鏡面的光澤，是不同的亮點，顏色為光的顏色(白)而非物體的顏色。作法也是參照講義即可完成，這邊要注意的是有關V的向量，他是人眼看的方向向量，因此要將做perspective時設定的眼睛位置傳入shader中(命名為eye)，並與物體的位置相減取normalize就能得到V的向量。

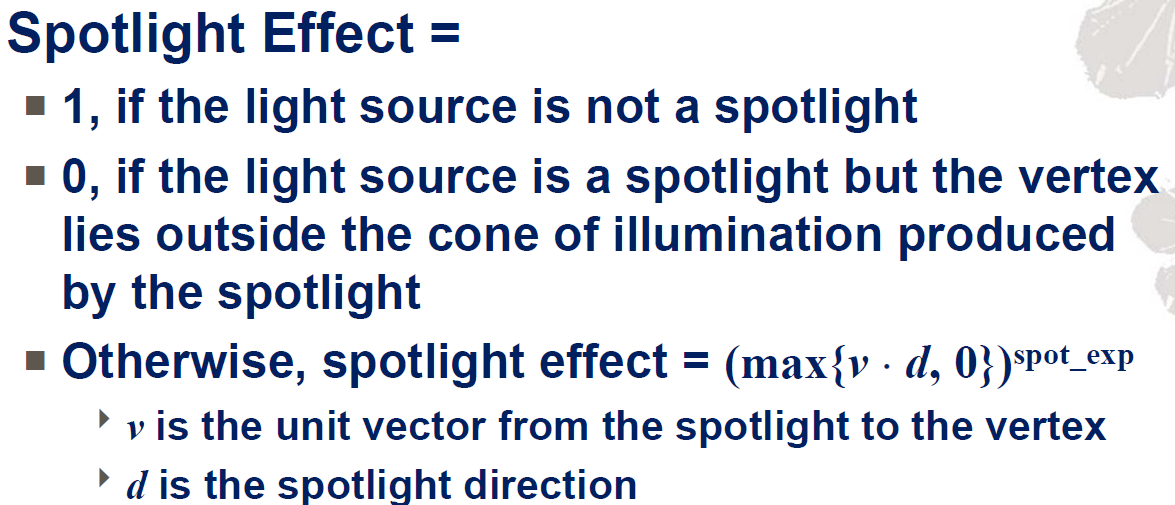
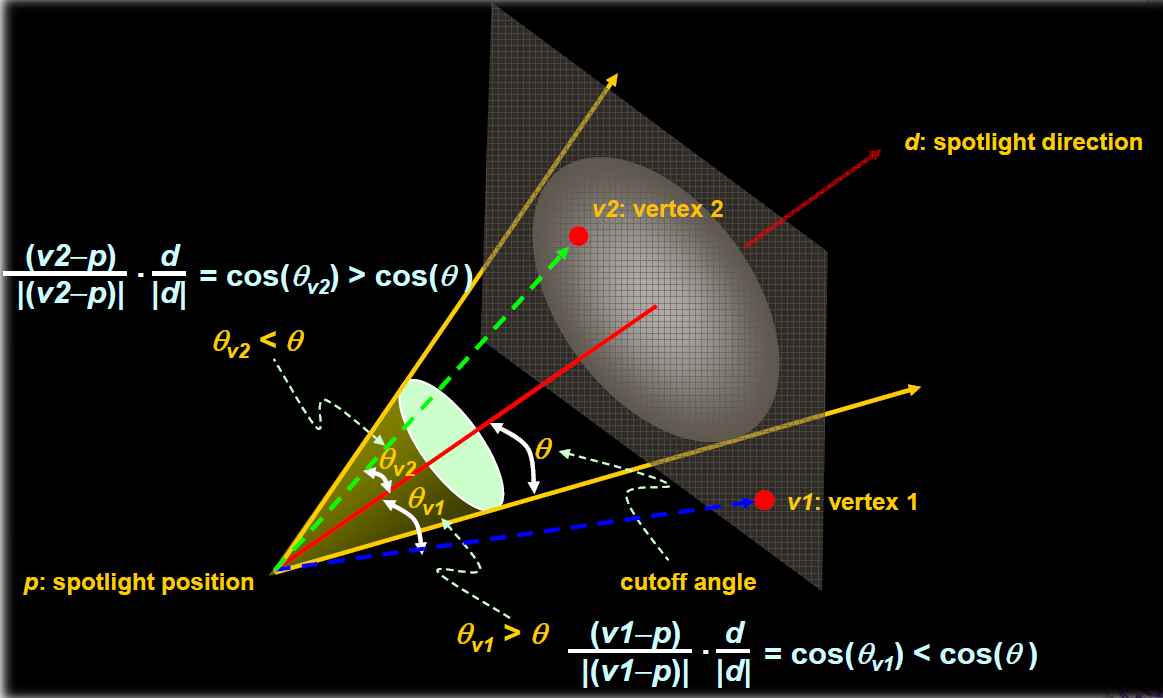


(4) Point & Spot Light adjusting:

做點光源時，要在Diffuse和Specular上乘上fatt的參數，才會得到是從一個點看出去的效果而非平行光，作法如同講義所述，其中c1對應的是傳入的constantAttenuation，c2對應的是linearAttenuation，c3對應的是quadraticAttenuation，dL則是點光源到該點的長度，可以利用length()計算。



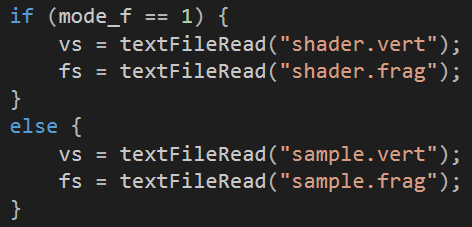
而Spotlight的部分，除了要乘上fatt外，還要去算他的spotlight effect，算法是要先判斷是否超出物體邊界，超出的話值為0，若沒超出的話可以利用講義公式計算出spotlight effect效果並跟fatt一樣乘在diffuse和specular後的結果，就可以達到spotlight的效果。



3. Bonus:

(1) Per pixel mode **(‘f’)**: (15%)

原本的概念是每傳進一個vertex就對它進行光照的運算，而要把它改成pixel才進行運算，也就是在fragment shader才進行處理。我的作法是開兩組(vert, frag)shader來控制不同的mode，切換的方法是在setShaders()中利用if else判斷要讀取哪組shader，並在每次按下f鍵在set一遍，這樣可以讓我們傳入兩組shader的變數都是一樣的。在shader中，原本寫在vertex shader的運算全部移到fragment shader來做，而要在兩個shader間傳遞參數可以利用varying來宣告，這樣就能達到光澤更加圓滑的效果，圖為控制讀哪個shader。

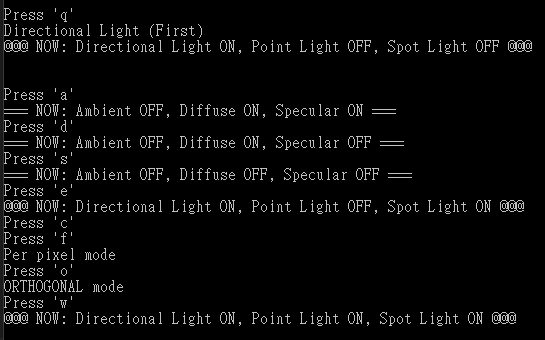


(2) Change spot light to another kind **(‘c’)**: (5%)

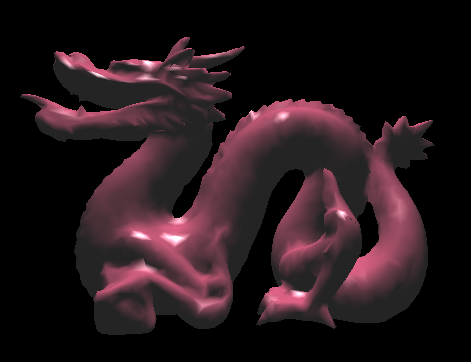
這個部份其實比較簡單，就是要再多一個變數(我取名mode\_c)傳入shader來控制現在要是哪種光源，並且切換的每種光源都可以透過滑鼠的hover來改變光的position，這樣就能順利切換。

三、 Other effort I have done:

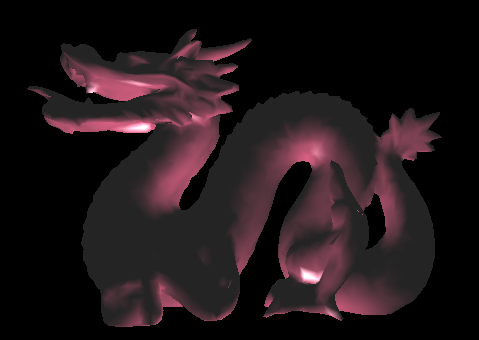
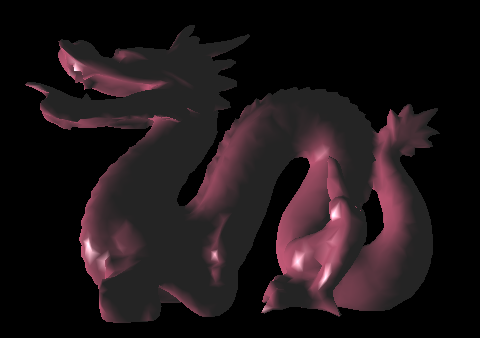
在這次的作業中，因為有很多不同的組合和控制，所以我有適當的Print出現在的狀態和改變，按不同的按鍵也會有不同的敘述，可以讓我在Debug中更加清晰易懂，也可以讓其他人知道我現在做的控制，圖為一些印出的敘述。



四、 Results:

**Directional Light(1) Directional Light(2) Directional Light(3)**

**Point light at (0,-1,-0.3) Point light at (-2.7,-1,0.3)**

** **

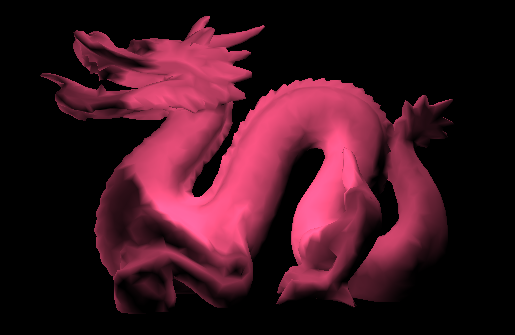
**Spot light Spot light(low spotCosCutff)**

** **

**Spot light(low EXP) Spot light(high EXP)**

****

**Blending for three kinds of light**

**  **

**Ambient Diffuse Specular**