計算機圖學HW1

1. Specify your programming environment.

我使用的編輯環境為visual studio 2019，template的內建建置目標為v142(專屬於vs2019)，其餘部分皆依照assignment之template初始設置。

2. Explain how you implement the above requirements.

1. Render a **3D cube** with texture and let it rotate over time.

Render方塊的過程我使用glbegin(GL\_QUADS)，依照每一個面的順序分別將四個點以glvertex3f的方式寫入。針對每一個點再分別加入glTexCoord2f(貼上texture)和glnormal3f設定該點的法向量(提供資訊進行光源的渲染)。進行這些操作必須在modelview matrix中進行，因此在操作之前需要加上glMatrixMode(GL\_MODELVIEW)。

旋轉的部分則根據instruction給予的參數，每個cycle逐步增加旋轉角度值(到360時歸零)，並使用glRotatef(angle,旋轉軸參數)進行旋轉。

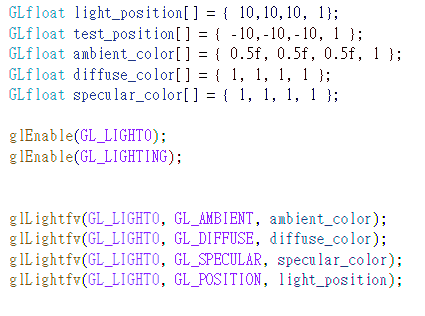
1. Render two **spheres** with two textures and let them rotate over time.

設定點與旋轉的方式與Render cube的方式相同，決定點的座標的方式上，我採用助教提供的經緯度的方法。將經度分成360等分，緯度分成90等分，在每一個經緯度點 Pi,j 上根據該點經度u、緯度v、半徑r，算出要繪製的座標點x = cos(u) \* sin(v) \* r，y = cos(v) \* r，z = sin(u) \* sin(v) \* r。並且由於所設置的球體中心在原點，因此計算vector normal同計算座標點公式，但不乘上半徑r即為法向量。

在面的繪製上，對每個經緯度座標點Pi,j，我們取用四個點分別為Pi,j、Pi+1,j、Pi,j+1、Pi+1,j+1組合成一個平面，經緯度分割數量越多越接近球型。

在進行Texture mapping時，我們將經緯度(0,0)視為圖片座標的原點，並將經度remap到(0,1)的範圍視為圖片座標軸S，將緯度remap到(0,1)的範圍視為圖片座標軸T，根據每個點的經緯度u,v設置對應的glTexCoord2f(s,t)。

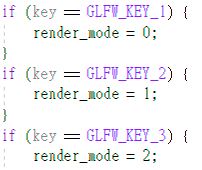
1. Add lighting component to the scene.



Enable lighting的部分則較為單純，首先enable(GL\_LIGHTING)，將系統預設光源關閉，改為全使用自己設定的光源。接著enable(GL\_LIGHT0)將編號為0的燈打開(open\_gl環境下最少有8個光源可以使用)，下面四行則是使用glLightfv指令去設置GL\_LIGHT0這個光源裡面的參數，從上至下分別是ambient light color, diffuse light color, specular light color以及光源的絕對位置。

1. Some keyboard callback functions for **switching** three objects (2 spheres and 1 cube)

在switching object這部分，我設置了一個全域變數為render\_mode，用以判斷現在應render哪一個物件，並在key\_callback function中進行更改，如下:

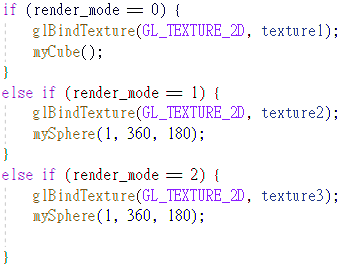


若按下'1'，則mode = 0，要render 3D cube。

若按下'2'，則mode = 1，要render 地球圖案的sphere。

若按下'3'，則mode = 2，要render 月球圖案的sphere。

在Main Loop裡面判斷如下:



在繪製點之前需bind對應的texture。

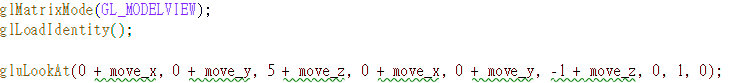
1. Use keyboard or mouse event to change **camera view**.

為了改變camera view，首先我做了幾項設置。

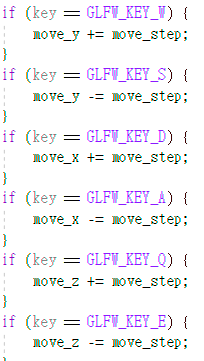
第一，我在使用gluLookAt function先將glMatrixMode設置為Model view，原因是我們通常會盡量使用modelview matrix對camera的位置參數進行修改，否則在投影時會有不必要的麻煩，第三部分會詳談。

第二，我設置了全域變數move\_x, move\_y, move\_z，用以表示相對於起始點時camera的位移量。

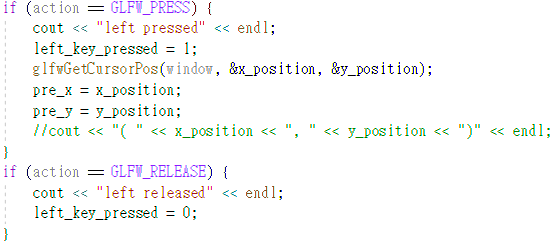
結合上述兩點結果如下:



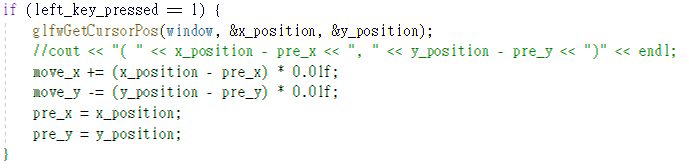
並且在按下W(上)、S(下)、A(左)、D(右)、Q(後)、E(前)時會改變move\_x, move\_y, move\_z以達成camera位移的效果。如下:



在Mouse event的實作中，我同樣使用move\_x, move\_y, move\_z作為位移量的表示。並額外增加了left\_key\_pressed的flag用以表示左鍵是否被按著(當左下按下去的時候改為1，放開時改為0)。為了紀錄滑鼠拖曳的方向以及量值，我在left\_key\_pressed = 1時以x\_position以及y\_position紀錄當下的滑鼠座標點，並以prev\_x和prev\_y表示上一個cycle的滑鼠座標點，以兩者之間的向量作為要更改的move\_x和move\_y值。合併如下:

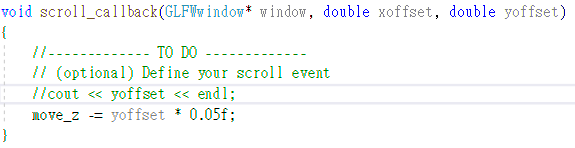


此為在mouse\_button\_callback中判斷是否有按下按鍵的判斷式。



此為根據前一個cycle與當下cycle的滑鼠座標的差異值進行move\_x, move\_y更動的部分。

滑鼠右鍵的call back function同左鍵，僅改動if內的條件，並且在按下右鍵時將move\_x, move\_y, move\_z歸零，以達成位置初始化的效果。

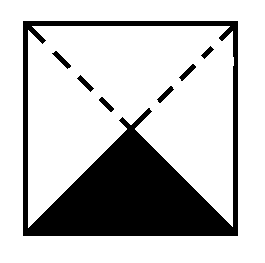
滾輪的效果中我使用scroll\_callback function進行修改。function中的input → yoffset為滾輪滾動的量值，向前為正，向後為負，滾動一圈就是1，因此我以這個yoffset的量值直接修改move\_z的值，達成深度的改變，如下:

3. Describe the problems you met and how you solved them.

1. 我第一個遇到的問題是在sphere中的texture mapping的部分。

起初我並不了解texture mapping的原理，所以我將製作成圓形的每一個四邊形都設置了texture coordinate(0,0)(0,1)(1,0)(1,1)，這會讓每一個正方形都出現一個不完整的地球圖案(不完整的原因在第二段)，而後我學到了texture mapping只是在該點設置要使用texture的哪個座標的顏色，因此才使用了第二大部分提到的方法。

在我完成了sphere所有點的繪製並貼上貼圖後，我發現我的球看起來"破圖"很嚴重。因此我把整個拉大來看之後發現每個方塊都只出現1/4個三角形，大致如下



在我上網搜尋資料之後發現，在GL\_QUADS中我們必須以順時針或是逆時針的順序進行點的設置(glTexCoord2f也是一樣)，否則就會像上圖一樣只出現其中一部分的三角形的情況。

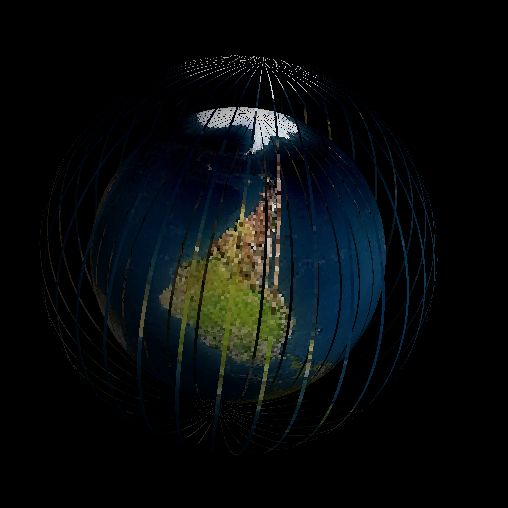
修改繪製點的順序後即修好。

1. 第二個遇到的問題則是在lighting與camera

當我大致設定好第一版後，我發現一個嚴重的問題，是我的light會隨著物體的自轉以同心圓的方式跟著轉。在我搜尋資料之後發現是因為我將camera和繪製的物件都放在projection matrix下進行運算，因此在rotate時會連同camera和光源將整個座標系統進行旋轉。

因此我將camera和渲染物件的區塊換成modelview matrix進行運算，成功解決。

4. Illustrate extra features of your design. (optional)



此為成果截圖。

我設置了一個簡單的animation，以P鍵作為動畫開關的按紐。動畫限於sphere，當滿足經度u%10 == 0時，我將所有的點Pu,v之半徑r以sin函數的方式進行縮放，使那些點的半徑r在0.5~1.5之間以sin波的方式運行，呈現一種竹燈籠狀的樣貌。