# CG\_HW1 作業報告

## 程式碼介紹

* 1. **讀取obj並更新資料**
* 先將v、vt、vn的資訊分別加入到三個vector中，以便之後用於建立VertexPTN。
* 更新std::vector<VertexPTN> vertices;，但有些PTN組會是重複的，對於Cube.obj來說，會有6\*2\*3=36 組PTN，但我們不應該要存36組，因為其中有些是重複的，在儲存PTN時候，應該要注意是否有重複，並為其建立一個int findVertexPTNIndex(VertexPTN VertexPTN) const;去找出在vertices中的index，若沒有找到則加入vector中，並return -1。
* 為了解決多邊形需要分解成多個三角形的問題，建立一個std::vector<unsigned int> polyIndices;，讀取每一行f就會先將VertexPTN的index存入，若先分解的話，這findVertexPTNIndex要做很多次，會讓效能降低。
* 針對polyIndices處理好分解三角形的問題，再存入vertexIndices中。

※ 原本我是直接使用vertices遞迴尋找有無儲存過的點，但我發現這樣兔子會跑超級無敵久，所以後來改用hashMap，使用(p, t, n)做hash計算以後，對應到該點在vertices的index，因為hash好像不能使用float型態(順便多新增一個structure VertexPTNIndexKey來當作hashMap映照儲存使用)

※ 在紀錄PTN時候，紀錄的是實際在vector index的值，而非文本PTN值

* 1. **標準化所有頂點**
* 有了vertices中的所有頂點，我們可以找出*minVertex*和*maxVertex*，並計算出Bounding box以後，將最長邊縮放為1，讓所有vertex.position都進行縮放。
* 中心點的計算則是*(minVertex+maxVertex)/2*以後，進行縮放得到新的中心點。
* 最後，將所有點減去新的中心點，會將整個模型移動到中心位置。
  1. **建立buffer和render**
* 建立&vboId和&iboId buffer
* Render中，比較重要的是下面這裡stride對應到sizeof(VertexPTN)，offset則對應到(void \*)offsetof(VertexPTN, position))，因為有定義structure VertexPTN，所以stride要以元素間隔為主，offset則是指第n個元素需要的位移量，但VertexPTN.position不用位移。glVertexAttribPointer(0, 3, GL\_FLOAT, GL\_FALSE, sizeof(VertexPTN), (void \*)offsetof(VertexPTN, position)); // pos offset is 0
  1. **資源釋放**
* 在ReleaseResources()這一塊，我是將前面建立的&vboId和&iboId buffer刪除，並把mesh物件也刪除。(不是很確定是不是這樣寫)
  1. **UI介面**
* 用右鍵彈跳出menu的方式，讓使用者可以切換models。
* 介面上方增加熱鍵提示。

## 結果(截圖)

