CG_HW1 作業報告

1. 程式碼介紹

1.1 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

1.2 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。

1.3 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
```

1.4 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)

1.5 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。

2. 結果(截圖)