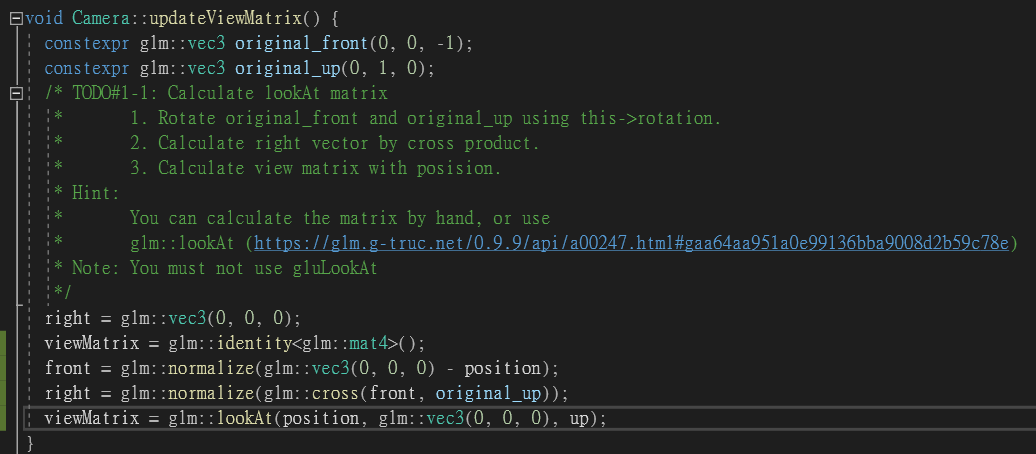
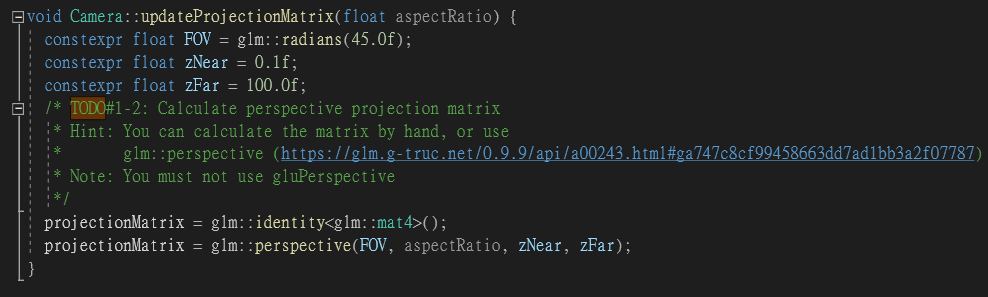
**camera.cpp**



front設定為camera的前方並normalize降低camera移動速度

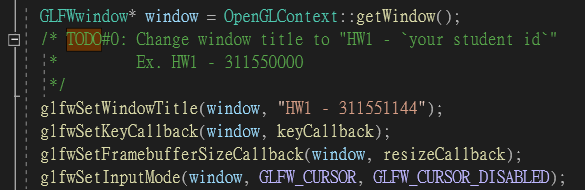
right為front和up的cross product

viewMatrix用lookat(camera位置, 看向的位置, up)算出

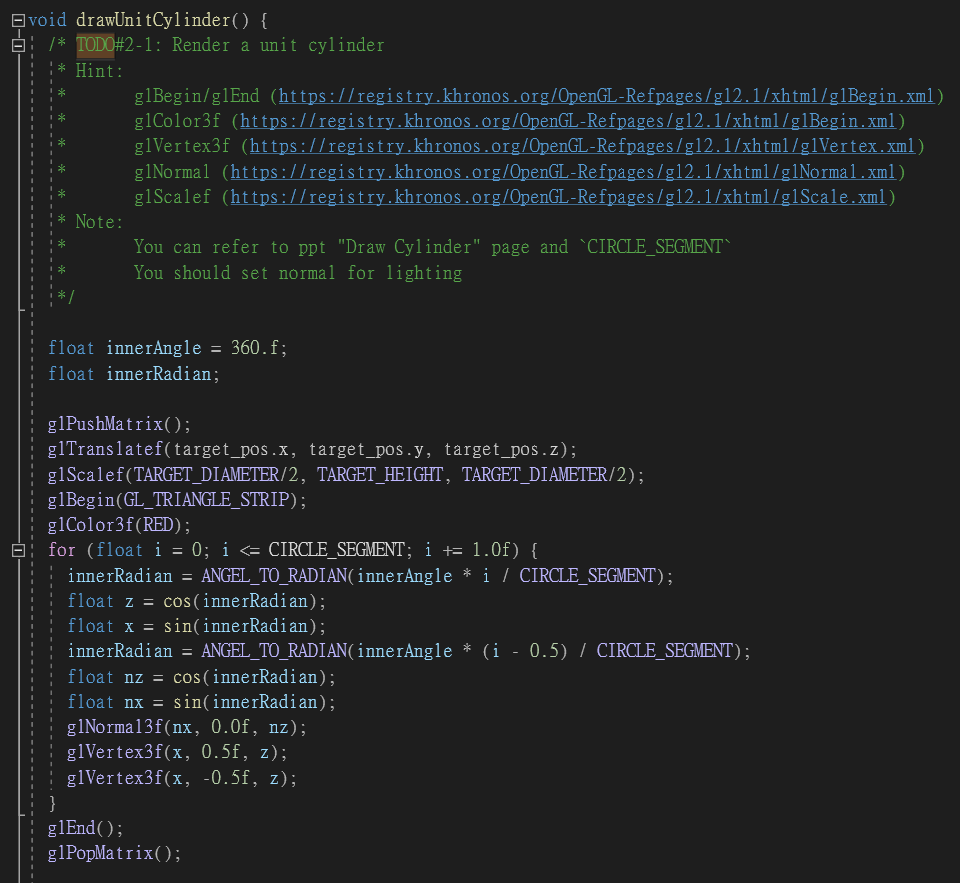


projecttionMatrix用perspective算出(照hint網址給的參數輸入就好)

**main.cpp**

****

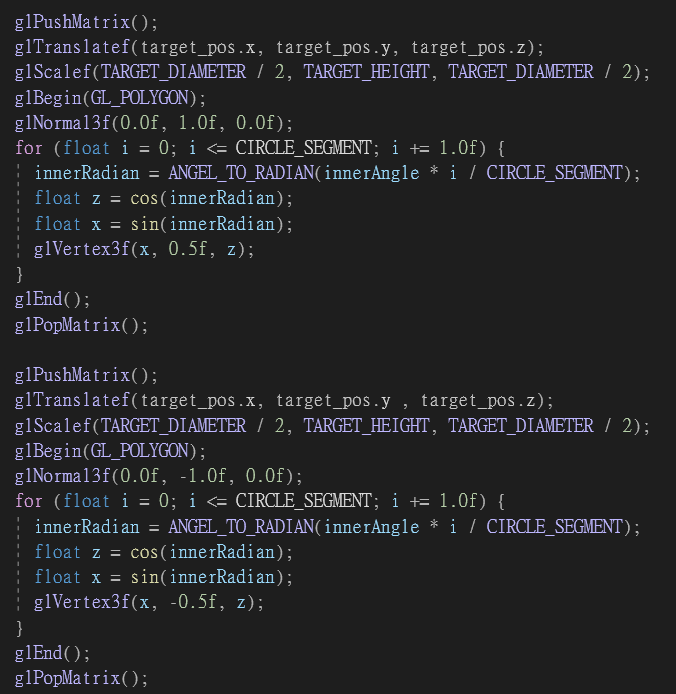
改學號



innerAngle/ innerRadian為多邊形中心對頂點的角度(正方形即0、90、180、270、360度)用來計算各頂點的位置

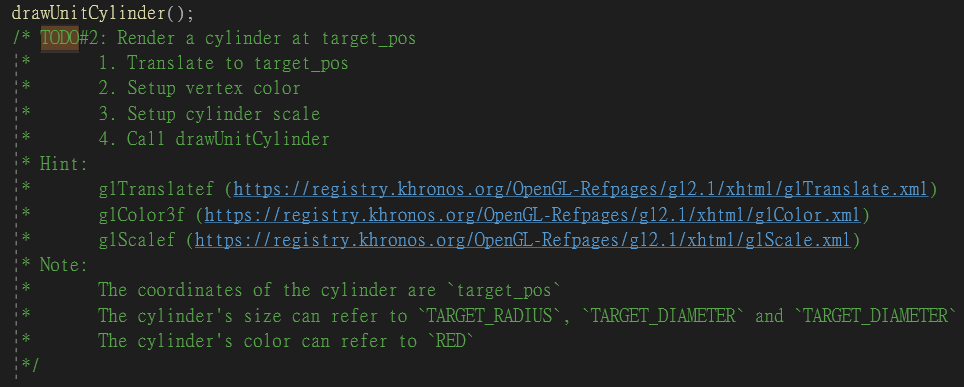
畫target外環順序

1. Loop畫不同頂點的三角形
2. 縮放物件
3. 轉移物件

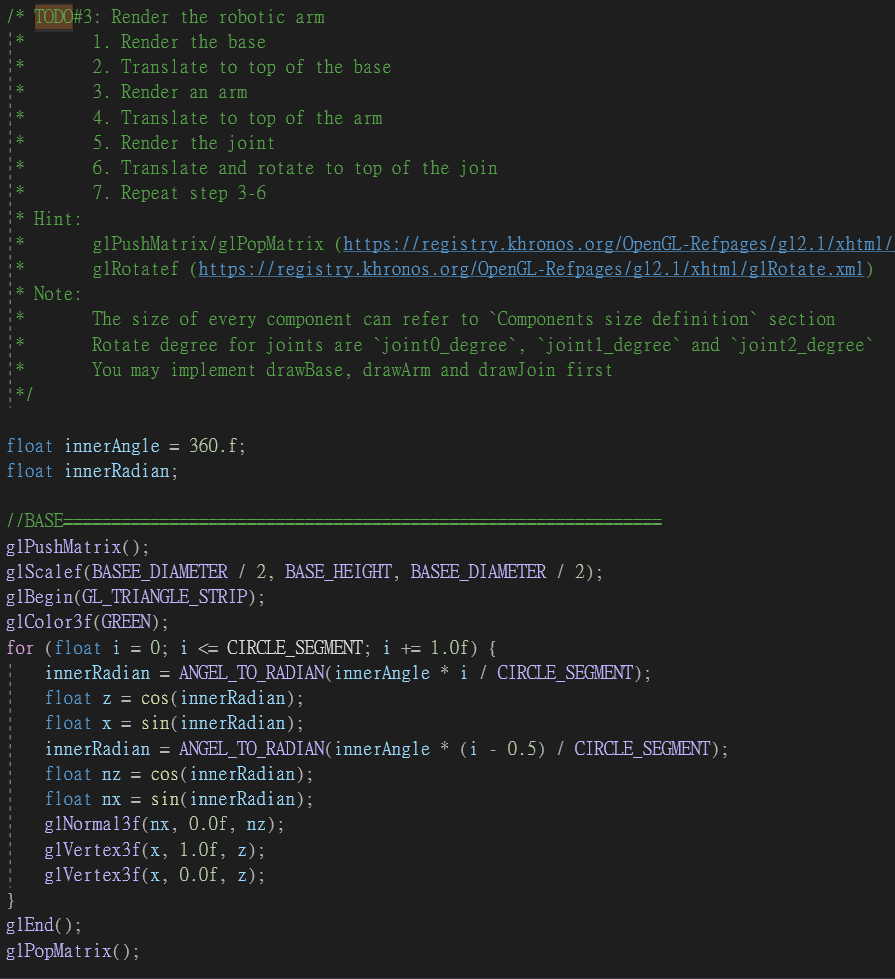


畫target上/下蓋順序

1. Loop畫多邊形
2. 縮放物件
3. 轉移物件

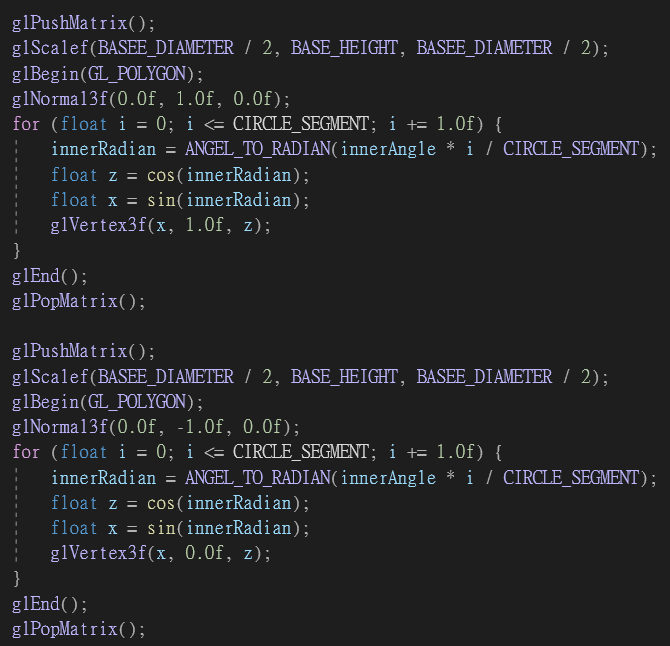


呼叫drawUnitCylinder



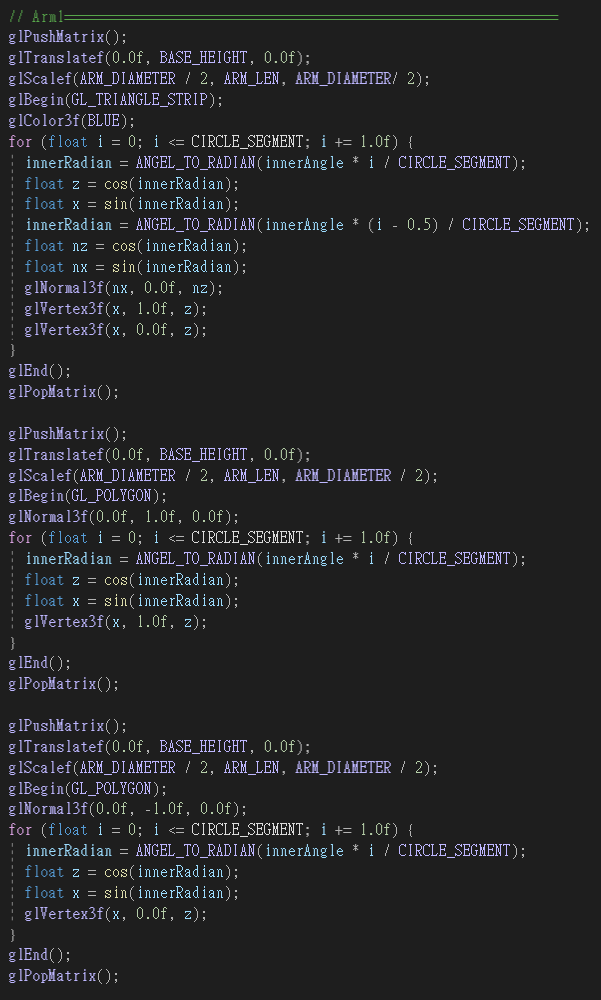
畫BASE外環順序

1. Loop畫不同頂點的三角形
2. 縮放物件



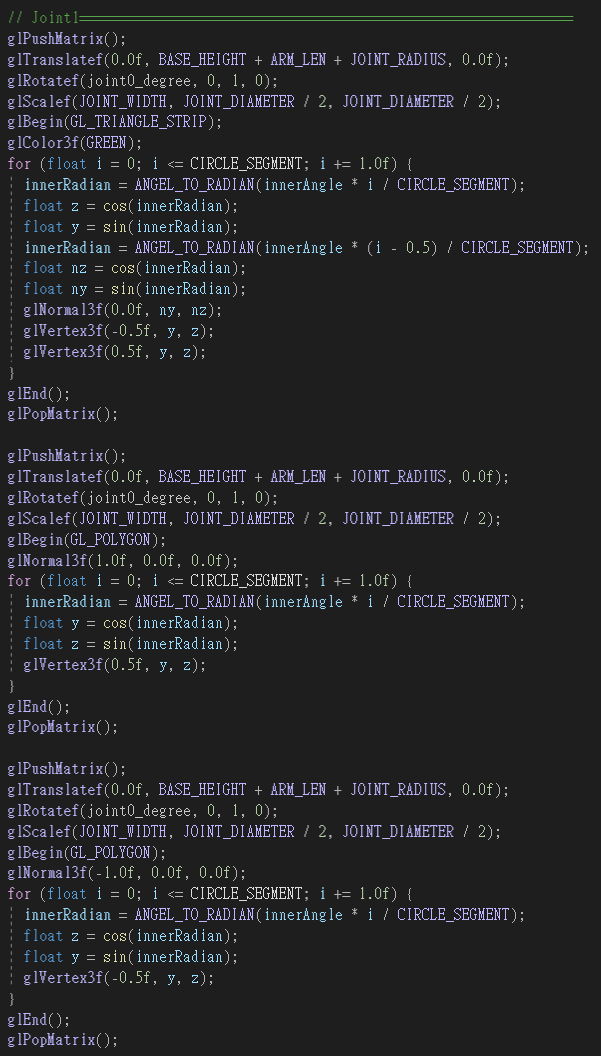
畫BASE上/下蓋順序

1. Loop畫多邊形
2. 縮放物件



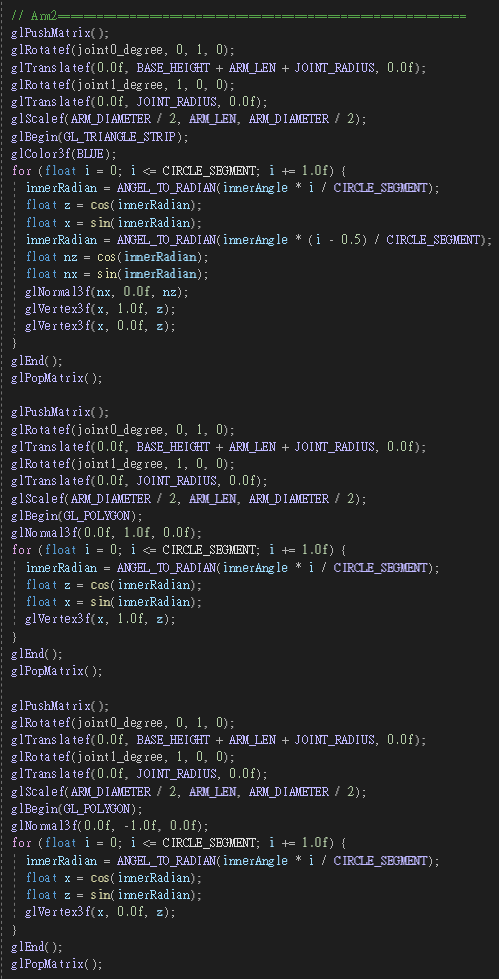
畫ARM1外環與上/下蓋順序

1. Loop畫不同頂點的三角形/多邊形
2. 縮放物件
3. 轉移物件到BASE上



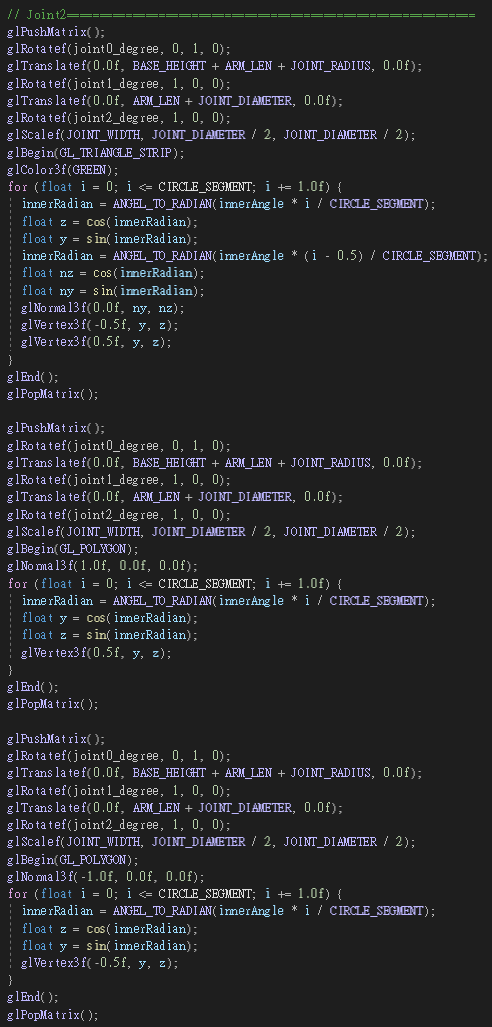
畫JOINT1外環與上/下蓋順序

1. Loop畫不同頂點的三角形/多邊形
2. 縮放物件
3. 旋轉物件(Y軸)
4. 轉移物件到ARM1上



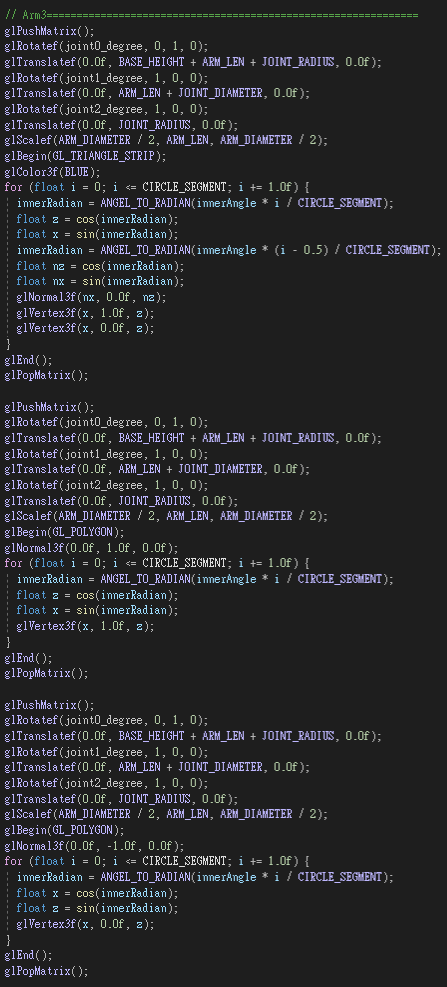
畫ARM2外環與上/下蓋順序

1. Loop畫不同頂點的三角形/多邊形
2. 縮放物件
3. 轉移物件(joint半徑)
4. 旋轉物件(X軸)
5. 轉移物件到JOINT1上
6. 旋轉物件(Y軸)



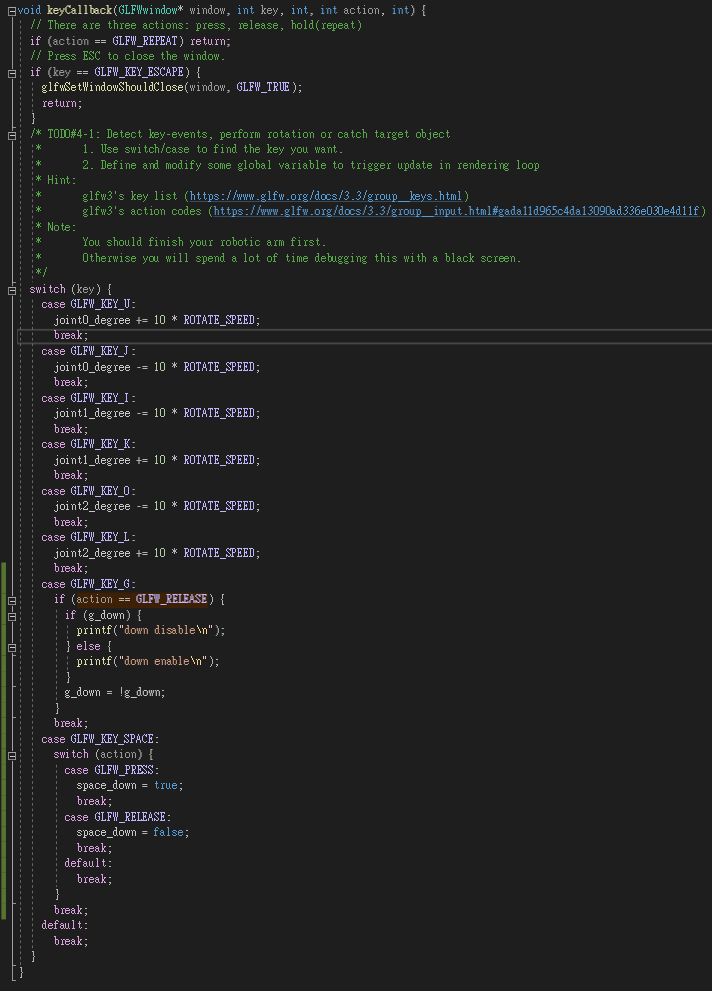
畫JOINT2外環與上/下蓋順序

1. Loop畫不同頂點的三角形/多邊形
2. 縮放物件
3. 旋轉物件(X軸)
4. 轉移物件到ARM2上
5. 旋轉物件(X軸)
6. 轉移物件到JOINT1上
7. 旋轉物件(Y軸)



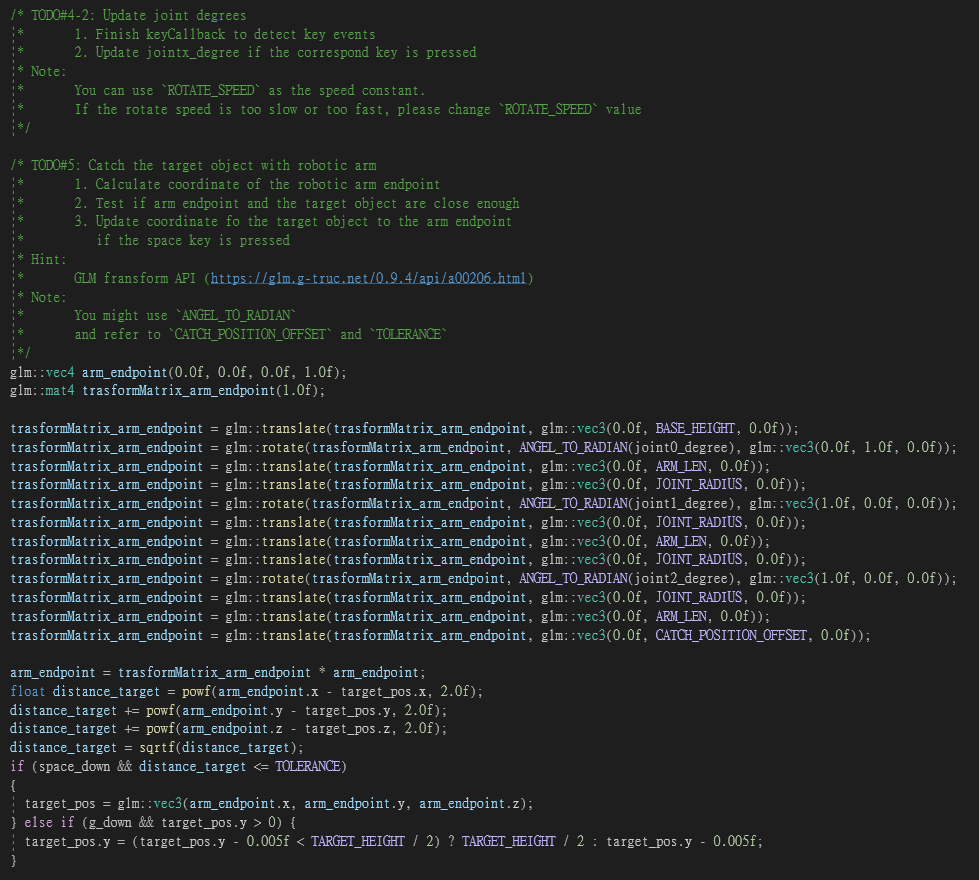
畫ARM3外環與上/下蓋順序

1. Loop畫不同頂點的三角形/多邊形
2. 縮放物件
3. 轉移物件(joint半徑)
4. 旋轉物件(X軸)
5. 轉移物件到ARM2上
6. 旋轉物件(X軸)
7. 轉移物件到JOINT1上
8. 旋轉物件(Y軸)



判別key為什麼按鍵然後改變joint\_degree的值，空白鍵的畫就判定action是按下還是鬆開，然後再改變space\_down的bool值

g\_down是target是否墜落的bool值



從原點根據joint\_degree0-3計算出手臂端點的位置

如果space\_down等於true(按著空白鍵時)且端點位置與target的距離小於TOLERANCE時target位置設定為端點位置

BONUS: Target墜落(按一下G開啟/關閉，預設關閉)

**遭遇問題:**

* Camera.cpp中rotation不知道怎麼用(HINT1-1. Rotate original\_front and original\_up using this->rotation.)
* 繪圖順序(轉移、旋轉、縮放)

多嘗試幾次就可以發現要從最遠做到最近