**Introduction to Computer Animation HW2 Report**

109350008 張詠哲

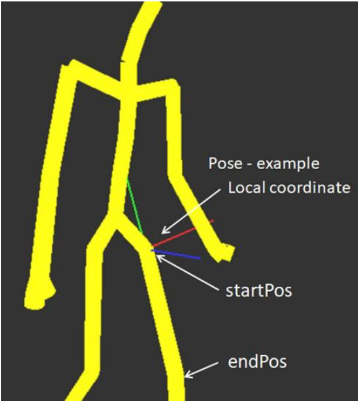
**• Introduction/Motivation**

這次我們要實作的是模擬人體骨架(很多關節再加連接的骨頭)在跑步和投籃(丟球)來展示Forward Kinematics和Time Warping。藉由現有的Function和Eigen提供的各種計算來實現旋轉矩陣和Quaternion的轉換和運算。我們會從骨骼的根部開始計算骨骼之間的起始位置、結束位置和旋轉，模擬整個人體形狀，以及其在運動中的姿勢。

**• Fundamentals**

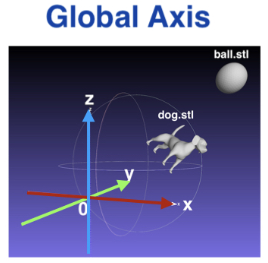
* **Local Coordinate:**

每個關節都有一個各自的坐標和自由度(角度)，是相對於上一個骨頭的末端點或某個關節定義的座標。示意圖:



* **Global Coordinate:**

Global Coordinate就是指我們平常所常見的3D空間，用來描述整個場景的位置和姿態。示意圖:



* **Time Warping:**

是指對動畫時間軸上的時間進行壓縮或擴展。通過Time warping，可以改變動畫中物體運動的速度和節奏，從而達到一些特殊的效果，例如慢動作、加速運動、時光倒流等等。

* **Forward Kinematics:**

是把物體的local coordinate轉到global coordinate，意思就是說給定每個關節的角度後，可以算出各個關節的位置。但是上網也查到說，Forward Kinematics 有其局限性，像是無法很好地處理物體之間的碰撞和交互。這些問題需要使用更複雜的技術，例如 Inverse Kinematics解決。

**• Implementation**

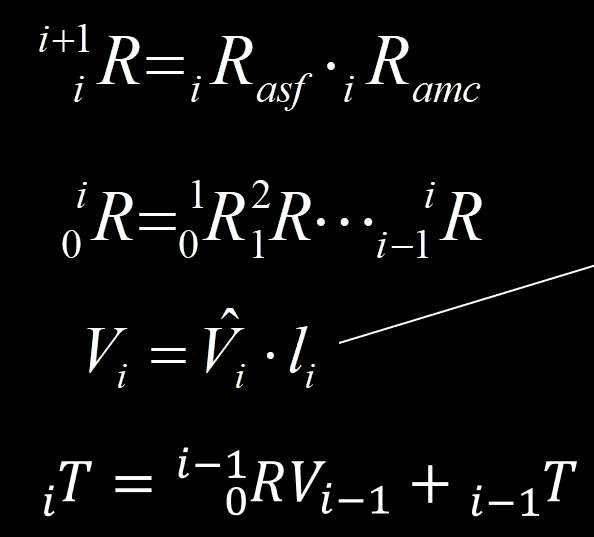
* **ForwardSolver**

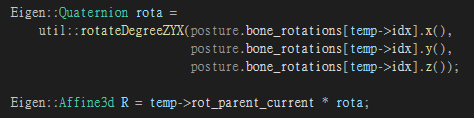
在這個部份我採用的是DFS去歷遍每個骨頭，接著去計算他們在global coordinate的起始位置、結束位置、旋轉。以下是一些code的解釋以及完整程式碼。

1. 將起始位置接到parent的結束位置

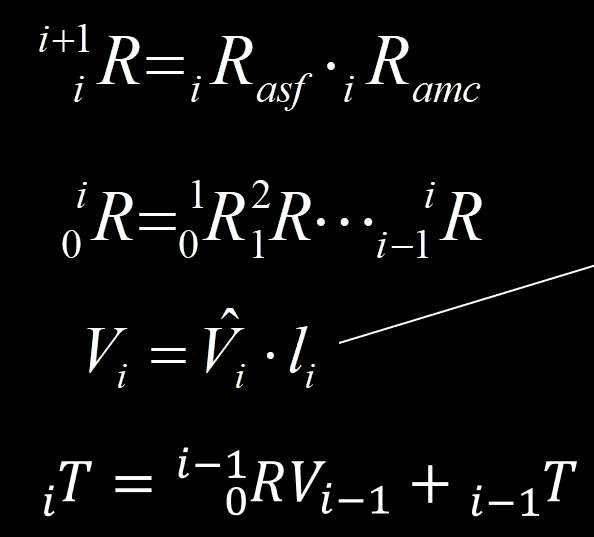


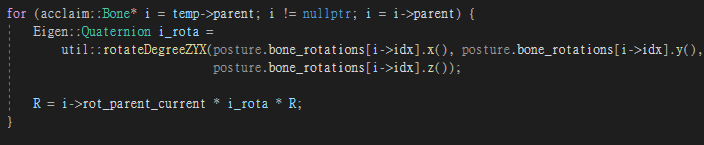
2. 將 asf 文件中的child對其parent的rotation與 amc 文件中的rotation相乘以獲得global coordinate的child相對其parent的rotation。



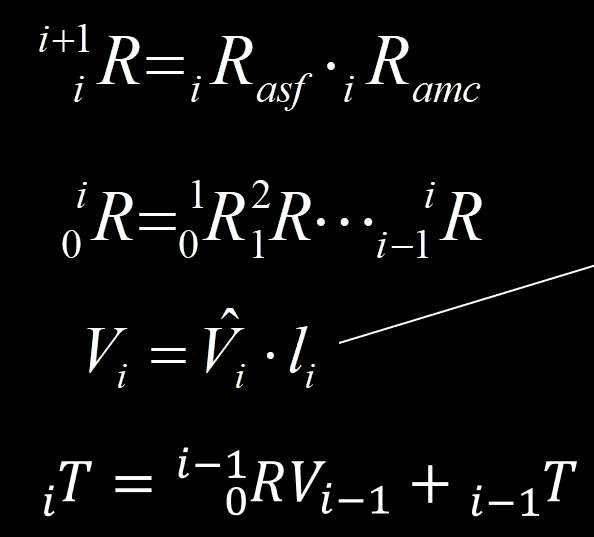


3. 計算local coordinate to global coordinate的quaternion



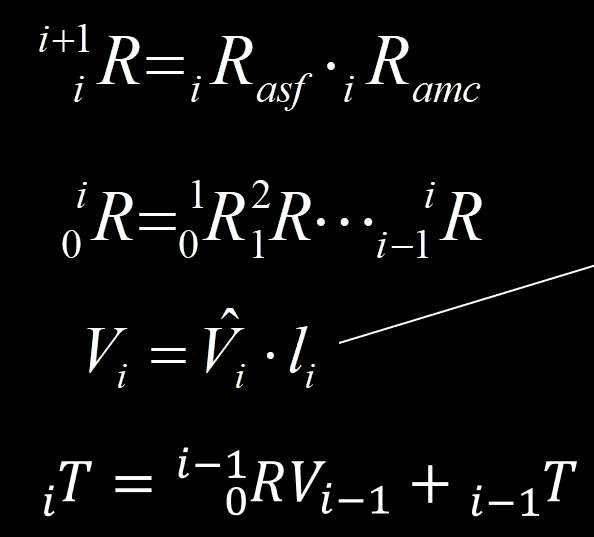


4. 由於 asf 文件中的方向是單位向量，我們需要乘以長度才能得到正確的方向



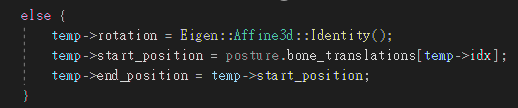


5. 把剛剛計算的rotation乘上dir並加上起始位置以得到結束位置。

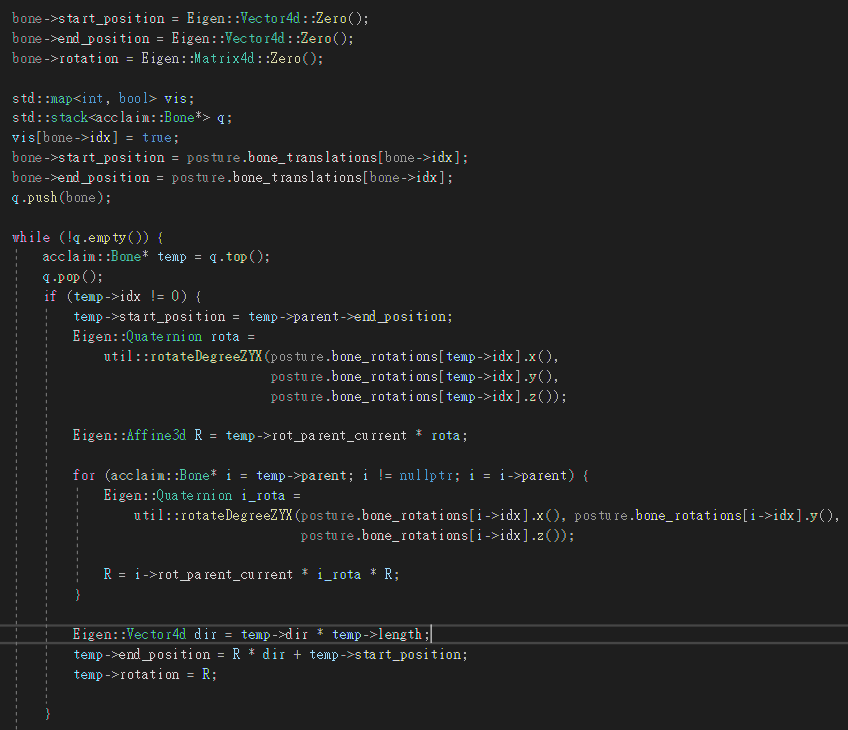


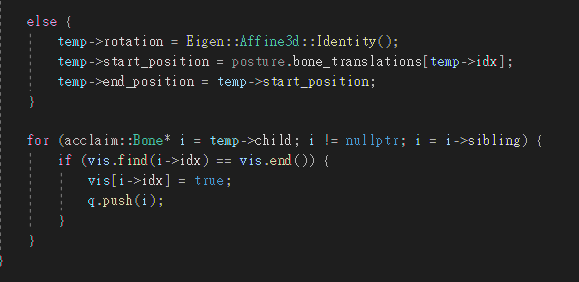


6. 如果是原點的話，因為root是沒有長度的，所以root的start position和end position會相同



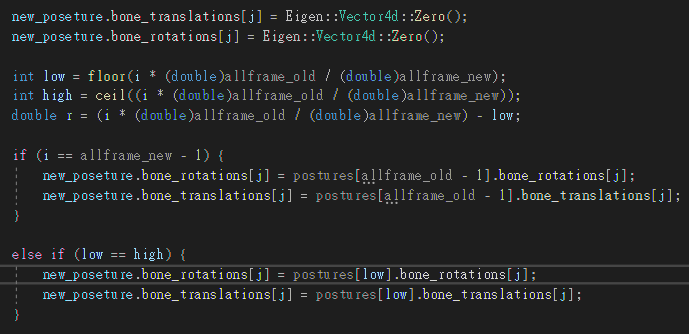
7. 全部的程式碼:

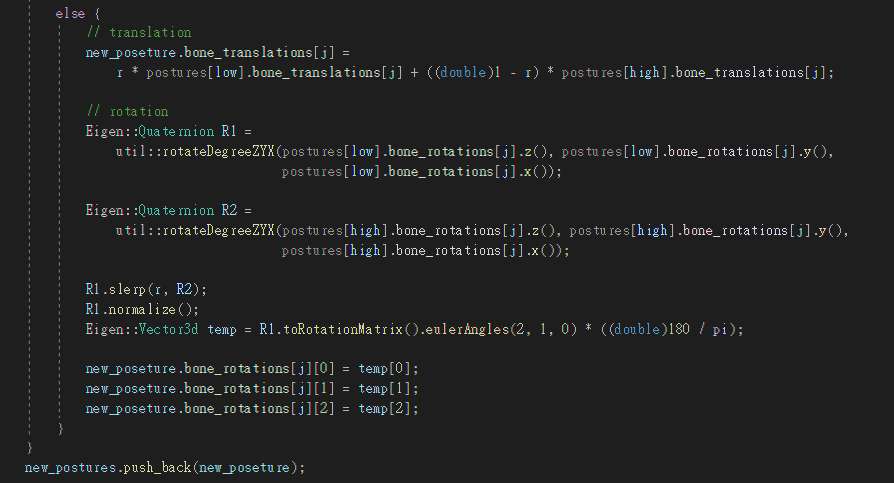




* **Time Warping:**實作這部分中的low和high是用來計算將new frame的每一幀i換算到old frames時會是在哪兩個frame中間。而r是，表示要內插在兩個frame(low & high)中的地方。而以下是判斷式的解釋:
  + 當i == allframe\_new - 1時，表示到轉換後的最後一幀，此時直接將該幀設置為原始動作的最後一幀。
  + 當 low == high 時，表示r為0，不需做內插。直接將該幀設置為原始動作中對應的幀。
  + 當 low != high 時，表示轉換後的當前幀落在原始動作的兩個相鄰幀之間，因此需要做插值。計算兩幀之間的平移變化使用線性插值，而球面線性插值(slerp)則是為了計算兩幀之間的旋轉變化。

以下為完整的程式碼:





**• Result & Discussion**

* **Result:**

ForwardSolver中我們利用amc file 和 asf file中的資訊去計算每個關節的旋轉角度，以及其在世界坐標系下的位置。

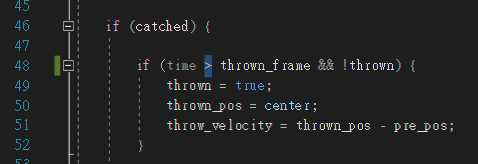
TimeWarper中我們計算了對時間軸上的時間進行壓縮或擴展。並對平移變化做線性插值，對旋轉做球面線性插值(slerp)。

* **Problems Encountered:**

1. 起初剛開始的時候要搞懂那些旋轉式怎麼操作的花了一些功夫，常常一進到執行畫面看到骨架歪七扭八，就直接跳出來不跑了。還好經過多方嘗試以及理解後，最後大致上有實作出來。歪七扭八大概像這樣:



2. 再來是在做timewarper部分時，剛開始寫的時候很常跳出未載入符號檔，雖然最後不太確定是為什麼會這樣，但似乎是因為有些地方計算錯誤才會這樣。還有就是不知道為甚麼在高幀數的時候(大約>550)，球就會丟不出去。還好看到討論區有人提出了解法: 將ball.cpp中的第 48行的>=改成>就可以了。有可能是因為差距太小以至於判定丟出去的速率為0。



**• Bonus**

我還是先去材質包那裡看看有沒有甚麼酷酷的材質包可以選，發現好像只能換天空的skin，因此我就順便換了一下。大概長這樣:





**• Conclusion**

此次要實作的部分有2個ForwardSolver以及TimeWarper來模擬人體骨架在跑步和投籃(丟球)。我們從骨骼的根部開始計算骨骼之間的起始位置、結束位置和旋轉，模擬整個人體形狀，以及其在運動中的姿勢。

在ForwardSolver中我們利用amc file 和 asf file中的資訊去計算每個關節的旋轉角度，以及其在世界坐標系下的位置。TimeWarper中我們計算了對時間軸上的時間進行壓縮或擴展。並對平移變化做線性插值，對旋轉做球面線性插值(slerp)。

也對於找到了為什麼在高幀數下球丟不出去的大致原因(有可能是因為差距太小以至於判定丟出去的速率為0)，也找到了解法: 將ball.cpp中的第 48行的>=改成>。

因為有修改ball.cpp(丟球)以及main.cpp(加分)檔的關係，因此也會額外在繳交作業的地方附上這兩個檔案。