Introduction To Computer Animation

Assignment2 Report_A022029 邱政凱

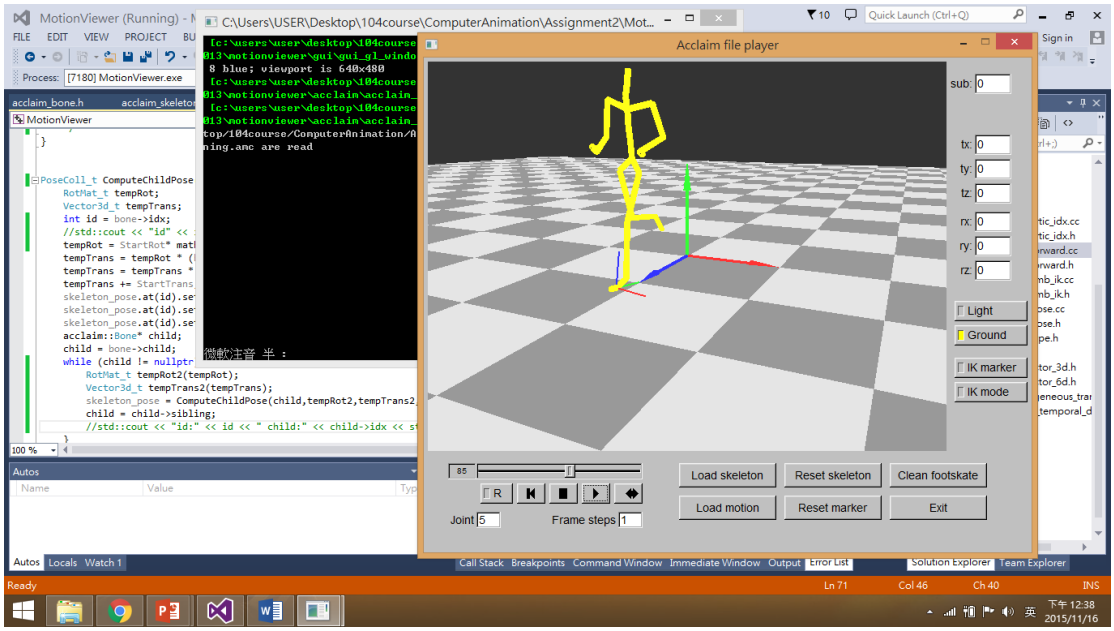
這次的作業是要我們實作 Forward Kinematics，是一種藉由給予每個關節(Joint)的位移和旋轉，用重複遞迴的矩陣運算去把最後整個角色的每個骨架在世界中的動作給算出來的運動學，是一種用於 rendering 的方式，相對於 Inverse Kinematics，藉由給予 End Effector 的參數去算出它帶動的關節的旋轉和位移以偵測環境對角色動作的影響，Forward Kinematics 算是相對而言比較直覺和好運算處理的。

要描述一個角色的動畫，我們首先需要角色的骨架(skeleton)，這部分在這次的作業中用的是 Acclaim 的 ASF 骨架格式。在 ASF 檔中，會宣告每個骨頭(bone)的 index、長度、方向、neutral pose 的角度以及 Degree of Freedom。而要讓一個處於 neutral pose 的 bone 去表現一段特定的動畫，我們在這次作業中使用的是 Acclaim 的 AMC 檔，在 AMC 檔中的資訊很簡單，AMC 檔是每個 FRAME 會有一個的檔案格式，除了根節點(root)包含了關節的位移跟旋轉外，其他所有的關節都只有旋轉的資訊，描述每個關節在每個 Frame 中的變化情形。

實作上，助教提供的 Project 檔裡面其實已經把讀檔、初始化、渲染等步驟都已經完成了，我們需要做的部分主要就只有從 ASF 和 AMC 檔中的已知資訊去把角色整個骨架的位置和旋轉算出來而已。

我們要完成的只有 ComputeSkeletonPose()這一個函數而已，這個函數藉由傳入 frame number 讓我們知道要從哪一個 AMC 檔的資訊來做 POSE 的計算。我的做法很簡單，先用`(*motion_).JointSpatialPos(frame_idx)`把我們需要的 `motion` 資訊取出來放在一個 `Vector6dColl_t joint` 裡面，接著先取出 `root_bone` 並計算它的位移和旋轉。計算旋轉的方式是從 bone 裡面取出 `rot_parent_current` 矩陣，乘上 `math::ComputeRotMatXyz(math::ToRadian(joint.at(idx).AngularVector()))`，也就是把原本的旋轉乘上 `motion` 指定的旋轉就 ok 了。位移則是因為是根節點的關係，我們需要從 `motion` 檔中抽出 `LinearVector` 加到它原本的 `root_pos` 上面，而骨頭的 `distal end` 的算法則是把原本骨頭的 `dir` 乘上新算出來的旋轉矩陣再乘上骨頭長度就可以得出來。算出 `root` 的位移跟旋轉之後，我用一個遞迴的函式 `ComputeChildPose`，藉由遞迴的方式把父節點的 `distal end` 位置和旋轉矩陣傳給屬於那個父節點的所有子節點，子節點再把父節點的旋轉矩陣乘上自己原本的旋轉矩陣`(rot_parent_current)`然後再乘上 `motion` 檔指定的旋轉角度(作法根 `root` 一樣)，只不過位置的部分因為非根節點的 `motion` 裡面的 `LinearVector` 都固定為 `0` 所以不須理會，只需要把父節點的 `distal end` 直接當作 `proximal end`，然後把新算出來的旋轉矩陣乘上 `bone` 的 `dir` 然後在乘上 `length`，就可以算出每個子節點的起始和終點位置。算完之後再用遞

迴的方式把當前骨頭的參數傳給所有它的子節點做計算。



這是最後的結果。

在實作的過程中其實遇到了不少麻煩，因為光是 Trace code 就不知道花上多少時間了，到後來其實有點把各個 class 的功能搞混，在計算骨架 pose 時要嘛就是沒有看清楚各個不同函式對於旋轉的定義是 Radian 還是 degree 導致單位一直混淆，就是忘記要把 rot_parent_current 先計算 transpose 再拿來用，整個實作過程其實一直在這些小細節上卡了很久。
(Radian 跟 Degree 的單位搞混的話動畫會變成以下這樣 XDD)