Appear in Proceeding of 2007 Computer Graphics Workshop (CG2007), Taiwan, 2007

人體低姿勢運動的程序式模擬

Simulating Human Low-Posture Motions with Procedural Animation

梁長宏

國立政治大學資訊科學系
台北市指南路二段 64 號

s9204@cs.nccu.edu.tw

李蔡彥

國立政治大學資訊科學系
台北市指南路二段 64 號

li@cs.nccu.edu.tw

摘要

目前製作動畫的主流方法除了由動畫師手工製作之外，也常採用動作捕捉（Motion Capture）。這兩種方法都相當耗費人力與時間，因此我們提出了一個高階的動畫控制系統，幫助我們更快速地產生動畫。因爲人體運動的種類很多，難以窮舉，所以在此我們就以低姿勢運動作爲代表。我們利用程序式動畫（Procedural Animation）來模擬人體低姿勢運動。此方法的重點是針對每種運動定義可調整的參數、關鍵格（Keyframe）以及內插方式（Interpolation），其精神與傳統的關鍵格動畫類似，不同點在於程序式動畫在關鍵格和內插方式中加入了「參數化」的概念。藉由參數化的關鍵格與內插方式，可使得原本低階、耗費人力的傳統關鍵格動畫，提升成高階的動畫控制系統。

Categories and Subject Descriptors

H.5 [**Information Interfaes and Presentation**]: H.5.1 Multimedia Information Systems – *Animations, Artificial, augmented, and virtual realities*. I.3 [**Computer Graphics**]: I.3.7 Three-Dimensional Graphics and Realism – *Animation, Virtual reality.*

關鍵字

程序式動畫、人體低姿勢運動、電腦動畫模擬、運動規劃

1. 前言

目前動畫最常用的製作方式有兩種，一種是關鍵格法（Keyframing），也就是由動畫師以滑鼠或其他輸入設備調整出一張張的關鍵格，再由電腦計算關鍵格之間的內插格以產生動畫。製作成果的好壞完全仰賴動畫師的專業程度，缺乏經驗的使用者不太可能使用這種方式製作出自然流暢的動畫，而且短短幾秒鐘的動畫就可能讓動畫師花上許多的時間。爲了節省時間與成本，於是有所謂的動作捕捉（Motion Capture）的技術發展出來。這種技術直接捕捉人體或物體的動作，輸入到電腦產生動畫格。這種做法不但擬真度較高，而且一旦校正設定完成，多個動作可以一次擷取，省時許多。

雖然以上兩種做法（手動和動作捕捉）都可以讓動畫做到很細膩，但是基本原理都是把輸入的關鍵格資料直接呈現在螢幕上，而這些關鍵格資料是死的，無法被彈性地調整。例如，在平地上走路和在崎嶇地形上走路是不太一樣的，雖然是同類動作，但還是得重新調整或擷取才能得到正確的動畫，非常費時費力。近年來許多動畫技術的研究都是在如何將擷取下來的動作，做進一步的變化以加值運用在其他應用環境下，但畢竟能變化的程度有限。所以像這種需要高互動性的環境，我們偏好使用程序式動畫的技術即時產生所需的動畫。程序式動畫的概念是建立動畫角色的關節式的骨架，同時以運動的基本原則和對環境的適應方法建構一個演算法的模型，在某種程度上，電腦就可以幫我們自動產生該運動的動畫。

在本論文中，我們嘗試利用程序式動畫定義出幾種常見的運動，以應用在互動性高的環境上。而爲了簡化問題以方便進行計算，我們將某些運動投影在 2D 平面上。這個平面隨著運動特性之不同，可能是側視圖或上視圖，在平面上設計出運動後，再轉換恢復原來缺少的自由度（Degrees of Freedom），就可轉回到 3D 空間上呈現了。.

2. 相關研究

雖然人類運動的模擬在電腦動畫領域中歷史悠久，但一直到現在，由於觀眾最爲熟悉的就是人類的運動，所以要能成功地模擬人類的運動，仍是一大挑戰。學術上許多自動產生人體動畫的相關研究 [1][2][3][4][7]，根據一篇人類走路動畫相關研究的概述 [6]，人體動畫的相關研究可大致分爲三類：動力學（Dynamics）、取樣法（Example-Based）、程序式動畫（Procedural Animation）。

動力學方法是建構人體骨架的動力學模型，利用控制器（Controller）控制角色以產生模擬的動畫 [3][4][7]。因爲動力學方法是建立在物理學模型上，所以產生出來的動畫符合物理學，具真實性，也很有彈性。但缺點是計算量大，所以不適合應用在即時環境上，更重要的是，這個方法最大的缺點是難以控制，使用者很難找出一組起始條件來達到心中所想的動畫。

取樣法通常參考動作捕捉的資料，在其中尋求變化，以達到彈性的目的 [9]。因爲此方法直接參考動作捕捉之資料，所以產生出來的動畫最爲逼真，這類研究都將動畫曲線視爲訊號，然後用訊號處理的方法來產生變化，這個方法的缺點是對於運動的結構或意義一無所知，所以變化程度有限，難以適應不同的環境。

程序式動畫則是在運動中找出規則，定義出一般化的關鍵格，關鍵格之間再利用適當的數學函數做內插 [1][2]。程序式動畫的計算量介於前兩種方法之間，所以效率佳，也具有彈性，適合應用在即時環境。然而缺點是擬真性無法與前兩種方法相提並論，所以爲了改善程序式動畫的擬真性，我們可以搭配實際捕捉的運動資料或物理學的知識作爲經驗法則。

近年來人體運動的相關研究大多著重在人類的行走。例如，Bruderlin 和 Calvert [1] 利用程序式動畫以及其他領域的知識，設計出一個人體跑步風格的高階控制介面，其中所取材的知識分爲經驗知識（Empirical Knowledge）、物理學知識（Physical Knowledge）和肢體座標（Limb-Coordination Knowledge），其中 Empirical Knowledge 用來定義跑步參數之間相互影響的關係；Physical Knowledge 計算骨盆的運動軌跡；Limb-Coordination Knowledge 則用來產生骨盆之外其他部位的內插。Chen 和 Li [2] 利用程序式動畫以及腳步計劃器，設計出可適應不同地形、即時自動規劃的人類走路運動。van de Panne [7] 把足跡和時間資訊作爲輸入，計算每個姿勢的舒適違反程度（Comfort Penalty）和物理性質的違反程度（Physics Penalty），然後透過最佳化（Optimization）演算法，算出身體合理的運動軌跡，腳步動作則利用程序式方法產生，其成果可產生出適應不同地形的走路、跑步或跳躍運動。

行走以外的人體運動相關研究大多使用動力學方法。例如，Faloutsosh 等人 [3] 提出先決條件模型（Pre-Conditions Model），用來管理多個控制器，另外他們也利用支援向量機（Support Vector Machine, SVM）讓電腦學習 Pre-Conditions 的定義，他們以此法成功模擬出跌倒、從躺下到站立及走路等動作。Hodgins 等人 [4] 利用控制理論和有限狀態機（State Machine）模型設計出各種模擬運動員運動的演算法，包括騎單車、跑步及鞍馬體操等。因爲程序式動畫很少用在行走以外的運動，所以我們希望能運用程序式動畫的優點，模擬各種低姿勢運動，然而走路是最常見的運動，所以我們也會將其納入範圍。

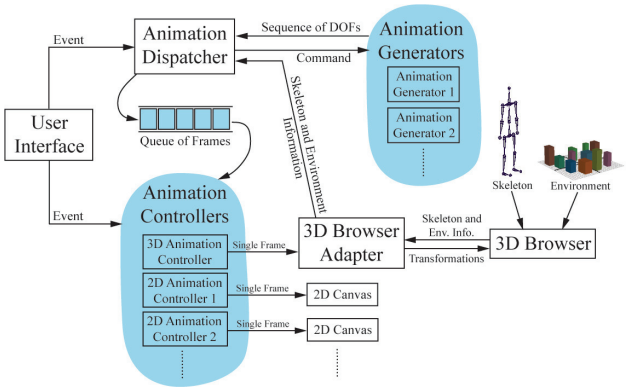


圖 1：IMHAP 的系統架構

3. 系統綜覽

我們開發了一個名爲 IMHAP（Intelligent Media Lab's Humanoid Animation Platform）的動畫平台來測試我們的程序式動畫，IMHAP 的系統架構如圖 1 所示。IMHAP 的架構主要是根據 MVC 模式設計的，在 MVC 中，系統被區分成三大模組：Model、View 以及 Controller，Model 用來封裝資料，View 與使用者互動，而 Controller 則負責前兩者之間的溝通。對照圖 1，圖中每個方塊表示一個模組，而模組間的箭頭表示資料流向，其中 Animation Dispatcher 和 Animation Generator 是整個系統的核心，它們扮演 Model 的角色。Animation Controller 負責控制動畫的播放，所以它是 MVC 裡的 Controller。剩下的 3D Browser、2D Canvas 以及 User Interface 都是與使用者互動的元件，所以它們是 MVC 裡的 View。以下段落將詳細描述各個模組的功能。

3D Browser 負責繪圖工作，它可以讀取人體模型和境資料。3D Browser Adapter 用來封裝 3D Browser，對其他模組提供一個統一的介面，這麼做的優點是可以讓系統動態地抽換不同的 3D Browser，而不用更動到系統本身。

Animation Dispatcher 和 Animation Generator 是整個系統的核心。Animation Dispatcher 可以作爲全域規劃器（Global Planner），它從使用者介面接收高階命令，間接指示它所掌管的多個 Animation Generator。而 Animation Generator 則是區域規劃器（Local Planner），負責產生個別運動。舉個例子來解釋這兩個模組之間的關係，使用者透過使用者介面輸入了一條行進路徑給 Animation Dispatcher，Animation Dispatcher 接收這道命令之後，它根據環境中的障礙物，計算出何時該用何種運動，如此便將一個高階的命令拆成數個低階的指令，之後將工作分派給適當的 Animation Generator，讓 Animation Generator 產生運動細節後，才把結果交由給 Animation Dispatcher 整理成一連串的畫格。

Animation Controller 從 Animation Dispatcher 讀取畫格，並控制動畫的播放，使用者可以透過使用者介面去控制 Animation Controller，進而控制 3D Browser 或 2D Canvas 應該顯示第幾格畫格。系統中允許多個 Animation Controller 存在，多個 Animation Controller 可以讓應用程式同時擁有多個視圖。例如，2D Animation Controller 可連接到 2D Canvas，提供像是

上視圖、側視圖、或是曲線圖等 2D 視圖。而 3D Animation Controller 則可連接到 3D Browser Adapter，提供 3D 透視圖的瀏覽。

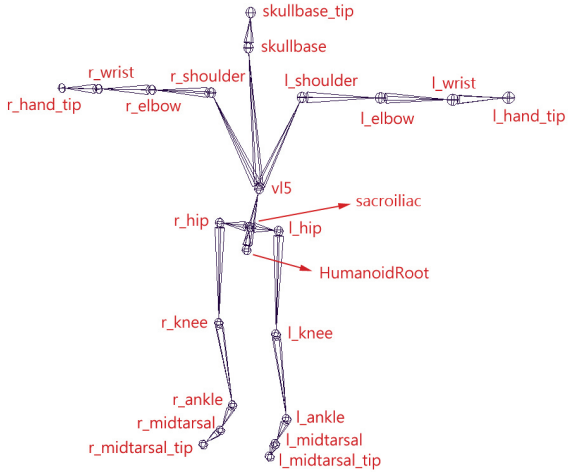


圖 2：H-Anim LOA1 所定義的人體骨架

4. 動作設計

我們所採用的人體模型爲 VRML [11] 裡 H-Anim [8] LOA 1 所定義的骨架，如圖 2 所示，此人體模型由十八個關節組成。在 3D 空間中，HumanoidRoot 有 6 個自由度（平移和旋轉各有 XYZ 三個自由度），而其他關節有 1~3 個自由度。爲了減輕負擔，我們在設計運動時，只考慮側視圖投影平面（Sagittal Plane），如圖 3 所示，所以如此一來，除了 HumanoidRoot 的自由度爲 3（兩個平移自由度加上一個旋轉自由度）之外，其他關節的自由度均減爲 1，大幅簡化問題的複雜度。

然而大部分的應用仍是在 3D 空間中，所以我們必須設計一個將動作從 2D 轉成 3D 的程序。在 2D 平面上，除了 HumanoidRoot 有 3 個自由度之外，其餘關節都只有 1 個自由度。到了 3D 空間中，因爲 HumanoidRoot 的平移座標表示法從 (x, y) 變成 (x, y, z)，而旋轉角度的表示法從 1 個實數變成 3 個實數，所以 HumanoidRoot 的自由度需增加爲 6 個。其他關節的自由度視個別情況而定，例如，肩膀（r_shoulder 和 l_shoulder）原本在 2D 平面上只需要一個實數來表示旋轉角度，但到了 3D 空間中則需要三個實數；而手肘（r_elbow 和 l_elbow）因爲實際上就只有 1 個旋轉軸，所以無論在 2D 平面上或 3D 空間中都只有 1 個自由度。

從 2D 轉換到 3D，重點就是還原這些新的自由度。因爲我們在設計動作時，把 3D 空間投影到 2D 側視圖平面上，而這個平面在 3D 空間的方程式是 $z = 0$，所以這些新的自由度就應該還原爲 0。舉例來說，HumanoidRoot 的座標表示法會從 (x, y) 變成 (x, y, 0)，而旋轉角度的表示法則從一個實數 θ 變成 (θ, 0, 0)。

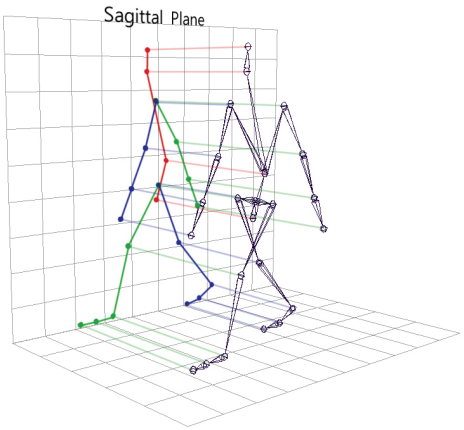


圖 3：將 3D 人體骨架投影到 2D 側視圖平面上

針對不同的運動，就必須設計不同的 Animation Generator，每種運動的設計原則大同小異，其步驟如下：

- 定義參數：每種運動都有一些可調整的參數，以走路爲例，可調整的參數可能有：步伐大小、走路快慢及抬腳高低等。
- 定義關鍵格：大部分運動可以分成好幾個階段（Phase），而階段和階段之間就是關鍵格，關鍵格的定義中會包含前步驟所設計之參數，如此一來，不同的參數組合，便能合成出不同風格的動作。
- 定義內插：關鍵格之間的動作必須用內插產生，最簡單的內插法是線性內插，複雜一點的則是用運動軌跡（Trajectory）與時間資訊（Timing）共同決定。本研究對於簡單的動作採用線性內插，較複雜的動作則採用貝茲曲線作爲軌跡和時間資訊的表達方式。

我們設計了四種運動，分別是坐下、蹲下、走路和低身行走，以下就分別討論其定義方法。

4.1 坐下

坐下這個運動相當於「從坐姿站起來」的反向，因爲「從坐姿起來」比較容易設計，所以我們在設計坐下動作時，其實是先設計「從坐姿站起」，再將其結果反向播放即成爲坐下動作。因此以下我們將以「從坐姿站起來」來替代坐下動作的設計。

從坐姿站起來的重點在於重心轉移，爲了能在運動過程中隨時追蹤重心位置，每段骨頭都被賦予一個「重量」參數，有了每段骨頭的重量，就可以算出整個骨架的重心位置。除了骨頭的重量之外，可調整的參數還有椅子高度（Chair Height）與坐椅深度（Sitting Depth）。此運動有四個關鍵格、三個階段，關鍵格之間的內插均採用線性內插，各關鍵格定義如下：

- 第一關鍵格：根據椅子高度和坐椅深度兩參數，將骨盆定於如圖 4 (a) 之位置，手掌平放在腿上，脊椎挺直。

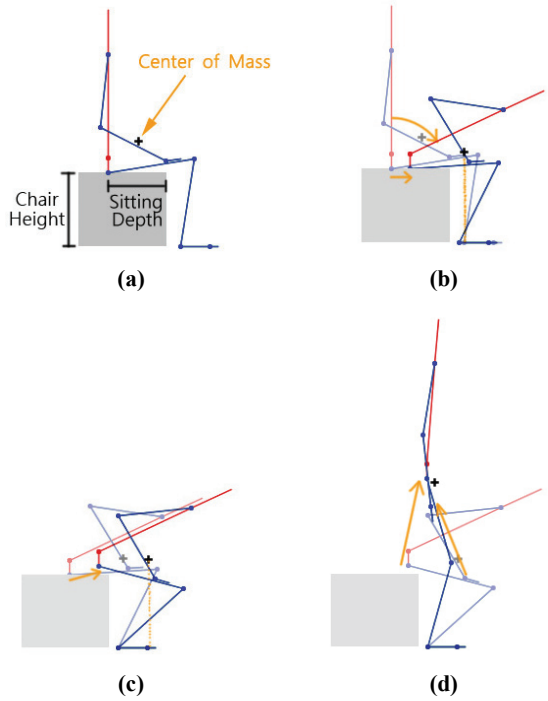


圖 4：從坐姿站起來的四個關鍵格

- 第二關鍵格：如圖 4 (b)，骨盆往前移動，脊椎向前轉，直到重心投影在地面上的位置落於腳跟。
- 第三關鍵格：如圖 4 (c)，骨盆向前上方移動，直到重心投影在地面上的位置落於腳尖。
- 第四關鍵格：如圖 4 (d)，骨盆繼續向前上方移動，雙手放下，最後成站立姿勢。

4.2 蹲下

蹲下這個運動非常單純，只有一個臀部高度（Hip Height）參數，指明蹲下時臀部與地面的高度。關鍵格有兩個，關鍵格間也是使用線性內插，關鍵格定義如圖 5。第一關鍵格爲一般的站姿，第二關鍵格爲骨盆往後下方移動一段距離，而雙手向前擺以保持平衡。

圖 5：蹲下的兩個關鍵格

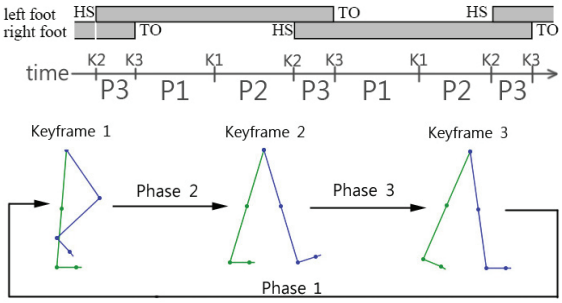


圖 6：走路循環圖

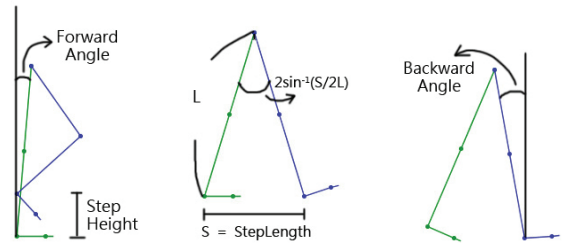


圖 7：走路的三個關鍵格

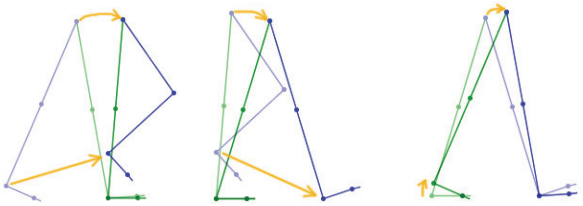
4.3 走路

走路較爲複雜，我們參考文獻 [2] 之作法，將行走分成三個階段、三個關鍵格，可調整的參數有步伐大小（Step Length）、抬腳高度（Step Height）、後傾角度（Backward Angle）及前傾角度（Forward Angle）。

文獻 [2] 中指出，人類的行走過程以 HS（Heel Strikes）與 TO（Toe Off）做分隔，其中 HS 表示腳跟踏地瞬間，TO 表示腳趾離地瞬間。我們以此爲基礎，將行走運動分成三個階段、三個關鍵格，如圖 6 所示，圖中時間軸上的 K1 表示第一關鍵格（Keyframe 1），P1 表示第一階段（Phase 1），其他以此類推。

4.3.1 走路關鍵格的定義

- 第一關鍵格：如圖 7 (a)，前腳腳跟抬起，使其高度等於參數 Step Height。後腳則伸直，以腳跟爲圓心、腳長爲半徑向前畫弧，使後腳與鉛垂線的夾角等於前傾角度。
- 第二關鍵格：爲前腳腳跟踏地瞬間，如圖 7 (b)，此時前後腳與地面形成一個等腰三角形，底邊長等於步伐大小。若我們令 S =（步伐大小）、L =（腳伸直時的長度），則可推導出兩腳之間的夾角爲 $2\sin^{-1}(S/2L)$。
- 第三關鍵格：爲後腳腳趾離地瞬間，如圖 7 (c)，根據第二關鍵格，後腳腳趾固定，前腳以腳跟爲圓心、腳長爲半徑向前畫弧，直到前腳與鉛垂線的夾角等於後傾角度。



(a) (b) (c)

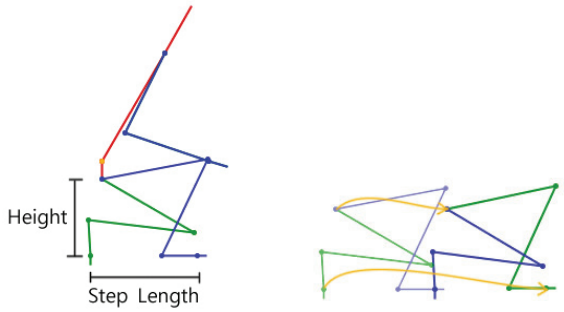
圖 8：走路三個階段的內插

4.3.2 走路關鍵格間的內插

- 第一階段（第三到第一關鍵格）：如圖 8 (a)，骨盆的運動軌跡爲圓弧，且爲等速運動。後腳腳跟的運動軌跡爲直線，並以緩入（愈來愈快）作運動。
- 第二階段（第一到第二關鍵格）：如圖 8 (b)，骨盆的運動軌跡爲圓弧，且爲等速運動。懸浮腳的運動軌跡爲直線，並以緩出（愈來愈慢）作運動。
- 第三階段（第二到第三關鍵格）：如圖 8 (c)，此階段爲兩腳著地，骨盆的運動軌跡爲圓弧，且爲等速運動。後腳則以腳趾爲圓心，順勢畫弧。

4.4 低身行走

低身行走和走路都屬於週期性的運動。若我們把對稱左右腳的動作視爲一致，可以把它分成一個階段、一個關鍵格。可調整的參數包括：步伐大小（Step Length）、臀部高度（Hip Height）及腳掌在空中時的運動軌跡。低身行走的關鍵格如圖 9 (a) 所示，根據步伐大小和臀部高度，將臀部和雙腳置於適當位置。關鍵格之間的內插如圖 9 (b) 所示，我們利用貝茲曲線表達臀部和腳掌的運動軌跡，其中臀部做等速度運動，而腳掌在離地和落地瞬間分別採用緩入和緩出做內插。



(a) (b)

圖 9：低身行走的 (a) 關鍵格 (b) 內插

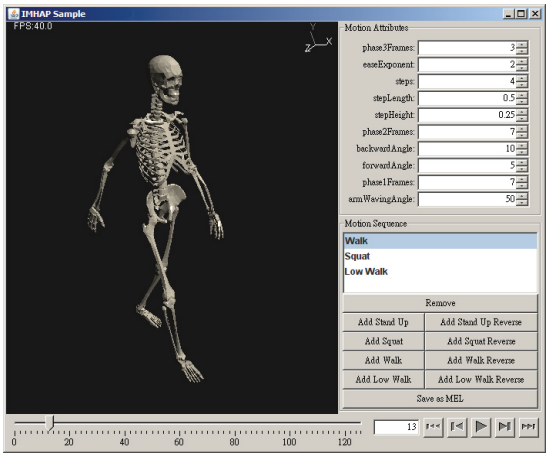


圖 10：實驗平台的使用者介面

5. 實驗結果

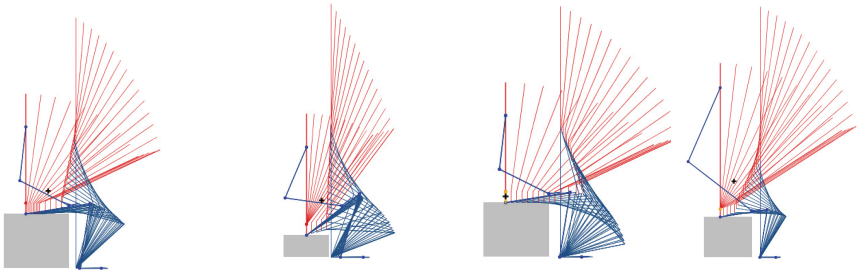
我們使用 Java 作爲實作系統的程式語言，以 IMHAP 爲基礎，搭配我們之前開發的 IMBrowser [5] 作爲 3D 瀏覽器，實作了一個應用程式，如圖 10。介面中央爲 3D 視圖，可顯示輸出的動畫結果。右側面板分成上下兩部分，上方面板用來讓使用者設定個別運動的參數，而下方面板則是用來指定運動序列。下方則是控制動畫面板，可控制動畫的播放、暫停或指定跳到某一格。

我們一共設計了四種運動，每種運動有共同的參數，也有自己特有的參數。共同的參數是骨架中各段骨頭的重量和長度，其數值來自 3D 模型檔，使用者無法自己調整。參數可以分爲重量參數、長度參數和角度參數，重量和長度的單位是我們自訂的，而角度單位爲度（˚）。以下我們就針對不同的運動，使用不同的參數來比較程序產生的結果。

坐下這個動作有兩個特有參數，分別爲椅子高度及坐椅深度。圖 11 爲四組不同的參數所產生的結果，(a) 和 (b) 都是使用預設的骨頭重量和長度，(a) 的椅子高度爲 5、坐椅深度爲 4；(b) 的椅子高度爲 2、坐椅深度爲 2；(c) 的骨盆重量被設爲極大值，其他參數則與 (a) 相同，可以看到 (c) 因爲重心集中在骨盆，所以它必須將身體更向前傾，以維持平衡並站起來；(d) 的脊椎和手臂長度增大，其他參數則與 (a) 相同。

蹲下的變化程度有限，主要參數是骨頭的長度，圖 12 爲兩個腿長不同的骨架。(a) 使用預設的骨頭長度，而 (b) 則是加長雙腿的結果。

走路有四個參數，分別爲步伐大小（Step Length）、抬腳高度（Step Height）、前傾角度（Forward Angle）及後傾角度（Backward Angle）。圖 13 爲四組不同參數的走路風格，(a) 的步伐大小爲 10、抬腳高度爲 3、前傾角度爲 5˚、後傾角度爲 10˚；(b) 的步伐大小爲 8、抬腳高度爲 1、前傾角度爲 4˚、後傾角度爲 12˚；(c) 的步伐大小爲 10、抬腳高度爲 4、前傾角度爲 10˚、後傾角度爲 15˚。



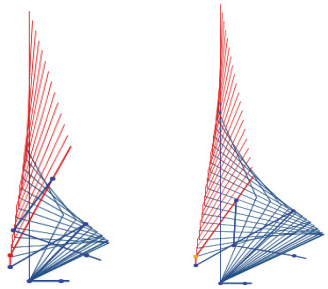
(a)
椅子高度 = 5
坐椅深度 = 4
預設骨頭重量和長度

(b)
椅子高度 = 2
坐椅深度 = 2
預設骨頭重量和長度

(c)
椅子高度 = 5
坐椅深度 = 4
骨盆重量極大

(d)
椅子高度 = 5
坐椅深度 = 4
加長脊椎和手臂

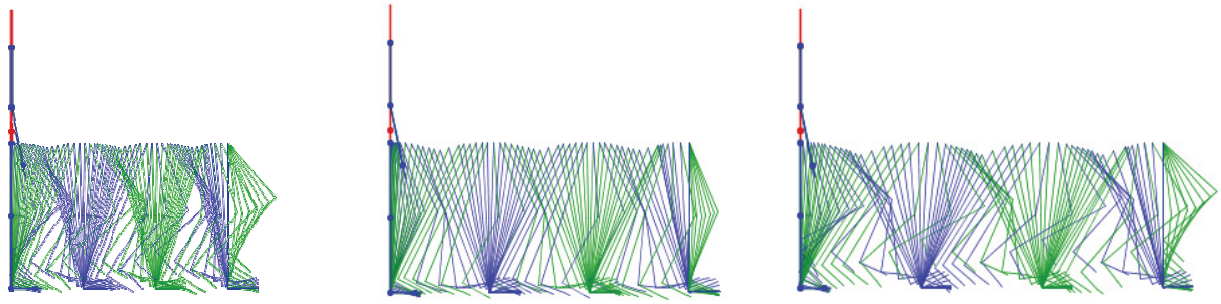
圖 11：坐下的實驗結果



(a)
預設骨頭長度

(b)
加長雙腿

圖 12：蹲下的實驗結果

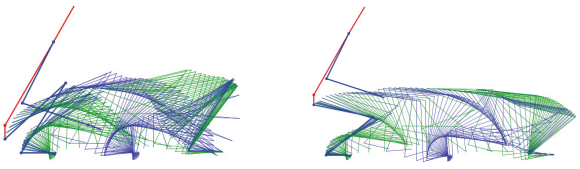


(a)
步伐大小 = 10
抬腳高度 = 3
前傾角度 = 5°
後傾角度 = 10°

(b)
步伐大小 = 8
抬腳高度 = 1
前傾角度 = 4°
後傾角度 = 12°

(c)
步伐大小 = 10
抬腳高度 = 4
前傾角度 = 10°
後傾角度 = 15°

圖 13：走路的實驗結果



(a)
步伐大小 = 6
臀部高度 = 1

(b)
步伐大小 = 10
臀部高度 = 5

圖 14：低身行走的實驗結果

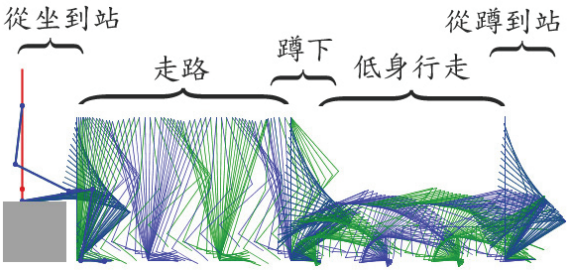


圖 15：組合多種運動

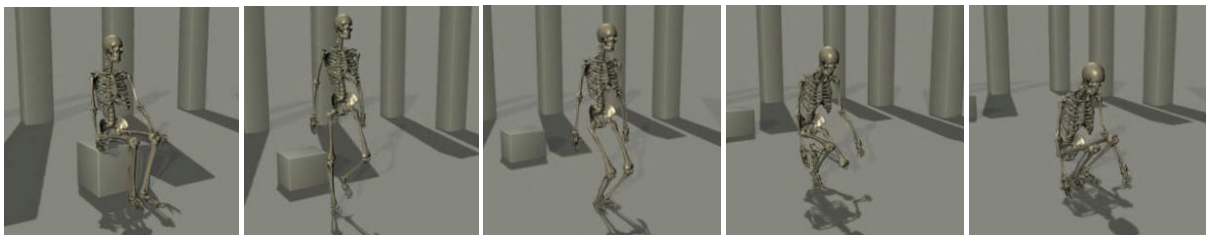


圖 16：從 2D 平面轉換到 3D 空間中的結果

圖 17：相同的動作序列套用在不同的角色上

低身行走有兩個參數：步伐大小（Step Length）和臀部高度（Hip Height）及雙腳在空中的運動軌跡。圖 14 爲兩組不同參數所產生出來的結果，(a) 的步伐大小爲 6，臀部高度爲 1；(b) 的步伐大小爲 10，臀部高度爲 5。

我們可以輕易將前述幾種運動連接起來，設計出一連串的動作。如圖 15，角色先是從椅子上站起來，走了四步後蹲下來，並低身行走四步，最後再從蹲著到站起來。我們也嘗試將這些程序式動畫輸出到商業 3D 動畫軟體 Maya 中，經過 Maya 的算圖（Render），將動畫輸出成品質較高的動畫影片，圖 16 便是成果之一。而在圖 17 中，我們將相同的動作序列套用在不同的角色，我們所設計的程序式動畫會依照骨架的不同，自動產生出適當的動畫。

6. 結論與未來研究

四種運動的程序式動畫已被定義，且每種運動各有不同的參數可供調整，以合成出不同風格的動作。此外，從輸入參數到輸出動畫期間，其延遲時間短，我們使用 CPU 2.4GHz、1GB RAM 的 PC 測試，產生一個動作所需的時間皆在 30~80 毫秒以內，故本研究可被應用在即時互動的環境上。

在未來我們可以開發更多種動作，目前在動作庫裡只有四種動作，在未來希望可以設計更多種動作，讓角色可以有更有趣的能力。目前每個動作都是分開獨立的，等到動作庫裡有足夠多的動作時，我們可以整合動作庫，並設計一個全域規劃器，讓角色可以隨著環境或使用者的指示，自動從動作庫當中選出適當的動作來執行。程序式動畫的特點是有彈性但擬真性不足，動作捕捉則是有擬真性但沒彈性，若搭配動作捕捉的資料，加強程序式動畫的擬真性，如此便可粹取兩種技術之優點，達成既有彈性，又有擬真性的動畫。另外，因爲我們目前僅利用簡單的方法來做 2D 到 3D 的轉換，所以造成模擬出來的動作在 3D 空間中有些許的不自然，我們可以改進 2D 轉 3D 的程序，以改善這個問題。

7. 誌謝

此研究在國科會 NSC 95-2815-C-004-006-E 及 NSC 96-2221-E-004-008 計劃的支助下完成，特此感謝。

8. 參考文獻

[1] A. Bruderlin and T. W. Calvert. Knowledge-Driven, Interactive Animation of Human Running. *Graphics Interface 1996*, pp. 213-221, 1996.

[2] P. F. Chen and T. Y. Li. Generating Humanoid Lower-Body Motions with Real-time Planning. *Proceedings of 2002 Computer Graphics Workshop*, Taiwan, 2002.

[3] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable Controllers for Physics-Based Character Animation. *Proceedings of ACM SIGGRAPH*, 2001.

[4] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating Human Athletics. *Proceedings of ACM SIGGRAPH*, 1995.

[5] T.Y. Li, M.Y. Liao, and P.C. Tao. IMNET: An Experimental Testbed for Extensible Multi-user Virtual Environment Systems. *ICCSA 2005*, LNCS 3480, O. Gervasi et al. (Eds.), pp. 957–966, 2005.

[6] F. Multon, L. France, M-P. Cani-Gascuel, and G. Debunne. Computer Animation of Human Walking: a Survey. *Journal of Visualization and Computer Animation*, 10:39-54, 1999.

[7] M. van de Panne, From Footprints to Animation. *Computer Graphics Forum*, 16(4): 211-223, October 1997.

[8] A. Witkin and M. Kass. Spacetime Constraints. *Computer Graphics*, 22(4):159–168, August 1988.

[9] A. Witkin and Z. Popovic. Motion Warping. *Proceedings of ACM SIGGRAPH*, *Addison Wesley*, pp.105-108, August 1995.

[10] Humanoid Animation Working Group (H-Anim): http://www.h-anim.org

[11] VRML, http://www.web3d.org/x3d/specifications/vrml/