一個有延展性的動畫劇本描述語言
A Scripting Language for Extensible Animation

廖茂詠 李蔡彥
國立政治大學資訊科學系
電話: +886-2-29393091 Ext. 63379
傳真: +886-3-22341494
E-mail: {g9005, li}@cs.nccu.edu.tw

摘要

目前 3D 虛擬環境中虛擬人物的動作，多是以動作抓取等離線方式錄製後，再以罐裝動作的方式播放，產生過程繁瑣且耗時。這篇論文提出了一套供虛擬演員使用的動畫腳本語言，稱為 eXtensible Animation Markup Language(XAML)。這套語言讓使用者可以彈性的使用不同層次的方式指定虛擬演員的動作。藉由這套語言，使用者可以重複利用之前創造的動畫元件來產生出一連串的動畫，或者是可以更改已經定義好的動畫的部分內容來產生一套新的動畫。這樣的方式可以提昇動畫元件的可重複性，並縮短動畫的開發時間。動畫腳本的結構是以 XML 為基礎的樹狀結構；藉由樹狀結構的父子關係和順序，我們定義出動畫元件間的優先權關係。一般而言，子代的優先權會大於親代，如果再同一個親代的前提下，越晚出現的元素有越大的優先權。因為各個動畫元素有優先關係，這樣就可以重新的融合成新的動畫元件，或做出個人化的修改。另外，我們以限制傳遞的方式說明此語言在時間上指定動畫的完整性。我們並實做了一個可用來組合現成動畫的製作軟體，可以用來即時預覽合成後的動畫。

關鍵字: 虛擬人物(Virtual Characters), 動畫腳本語言(Animation Script Language), 行為動畫(Behavior Animation), XML, 電腦動畫(Computer Animation)

1. 簡介

隨著電腦科技的進步，電腦運算能力已經可以輕易地負荷 3D 的複雜運算，因此 3D 動畫的應用已漸漸地也開始進入日常生活中。沒有了運算上的問題，在加上 3D 的動畫呈現上比起 2D 的圖像可以表現的更豐富，照理來說 3D 的應用應該會如火如荼的展開，但是實際上 3D 應用目前大部分還是侷限在撥放事先已經錄製完的 3D 罐裝動畫上。這個現象的主要原因之一辨識3D動畫製作的複雜程序，3D 動畫的生成是由一連串連續的影像結合而成，為了產生這一連串影像，開發者必須藉由設定許多低階的參數來完成這一項工作，沒有受過特殊訓練的一般的使用者很難去了解這些低階參數所代表的意義，也就降低了一般人開發 3D 動畫的意願。就算是專業人士，繁瑣的設定也使得開發 3D 動畫的時程大幅提昇，因此如何降低 3D 動畫所耗費的大量人工和時間變成了一項重要的議題。所以如果可以讓低階的繁瑣工作交由電腦自動化控制，開發人員只要使用高階的指令就可以產生一連串的動作，那麼就可以大幅地減少製作的時程和人力。

本論文提議以 XML 為基礎，設計一個具有延展性的動畫腳本語言。設計此語言的目的，在於提供使用者一個完整且具延展性的動畫指定方式。此語言必須具備完整指定動畫元件時間關係的特性，並且能容納各層次的輸入指令。例如，同一種動畫可以用低階、中階、或高階方式表達，或混合使用不同層次的指令。另外，此語言必須能將現有的動畫進行彈性的組合及篩選，以達到動畫元件重新利用的目的。我們將以一個實做出的設計系統，展示此語言的延展性及開放性。

本論文後續的章節裡，我們將逐一介紹此系統的設計理念。首先，我們將在第二節裡介紹動畫腳本語言的相關研究。在第三節裡，我們將說明此腳本語言設計的方式。在第四節中，我們將說明為此語言所實做出的設計及展示系統。設計成果及實驗的範例則將於第五節中呈現。最後我們會以未來延展做為本文的結論。

2. 相關研究

早期在動畫設計軟體還不是非常成熟的時候，動畫腳本語言的研究一直是個熱門的課題。但隨著電腦軟硬體的發展，動畫的產生多是以動畫設計軟體(如 3D MAX 及 MAYA 等)或運動抓取(Motion Capture)的方式製作，最後並以動畫格中個別物體座標轉換的低階方式儲存動畫結果。至於設計時所使用的概念，則多以設計軟體獨特的原始格式存在。

使用高階的描述來控制角色動畫並不是一個

```
<AnimItem DEF =”WaveWalk” cycle=”2000”>
  <AnimImport src=”Walk”>
  <AnimItem DEF=”SimpleWave” cycle=”1000”>
    <Node target=”r_shoulder”>
      <OrientationInterpolator
          key =”…” keyValue=”…” />
    </Node>
  <AnimItem>
</AnimItem>
```

圖一: XAML 動畫描述語言的範例

新點子，從 Improv [1] 開始就許多的相關研究投入這項議題。其中同樣是針對人體的動畫描述還有 Avatar Markup Language(AML) [3] 和 Scripting Technology for Embodied Persona(STEP) [3] 。AML 雖然也有肢體運動的設計，但是主要的動作是由一個 bap 檔案做描述，使用者只能改變速度、強度等有限屬性，對整體的動作更改的能力是很薄弱的。語言設計的重點在於同步上，他們對於臉部動畫和語音的表現相當重視；例如臉部表情可以用「情緒」這種抽象的字眼來對同一動畫做出不同的效果。

STEP 是提供給 Agent 使用的一套動作語言。此語言的特色在於其背後實做的語言是以邏輯推理見長的 prolog 語言，因此這個系統能將自動推理的功能與電腦動畫的結果進行整合。STEP 的設計則是跟我們相當類似，都是專注在肢體的動畫語言，STEP 也可以藉由組合定義出新動作，此外，配合 logic 語言設計出因應環境的改變決定對應的動作，這種事件觸發的動畫機制在多人環境是有相當的潛力，開發者只要修改動畫語言而不需要修改程式，即可作出動態的動畫反應。不過 STEP 的動畫的豐富性並不強，動畫指定完成就無法去更改其強度、動畫撥放時間等動畫屬性。另外，電腦動畫的描述，如必須遷就宣告式(Declarative)語言的特性，則在表達方式上將較為繁瑣不自然。

3. XAML 腳本語言的設計

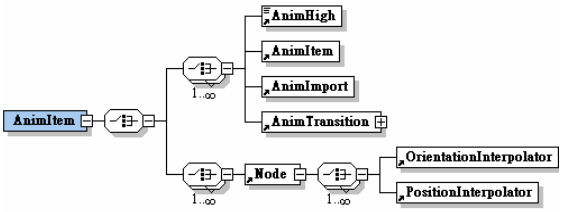
3.1. 不同動畫指定方式的比較

電腦動畫中物體運動腳本的指定方式，可以分為兩個極端：低階指定與高階控制[8]。低階指定的方式優點在於能指定所有參數的細節，但缺點在於當要指定的參數過多時，製作過程過於繁瑣耗時。一般具有動畫表示能力的資料格式，如 VRML 或 Direct3D 均屬於這類低階的表示法。這類指定方式的腳本資料量通常較大，但動畫呈像的軟體通常只需要知道如何作剛體的座標轉換，即可做出各式各樣的動畫表現。

相對而言，另一個極端是高階的動畫控制方式。動畫設計者透過高階的指令，便能指定所希望

```
<AnimItem> := (<AnimItem> | <AnimHigh> | <AnimIm-
   port> | <AnimTransition> )* | <Node>*
<AnimHigh> := text
<AnimTransition> := (<AnimItem> | <AnimImport>)*
<Node> := (<OrientationInterpolator> | <PositionInterpo-
   lator>)*
```

圖二: XAML 語言的語法



圖三： XAML 語言的結構

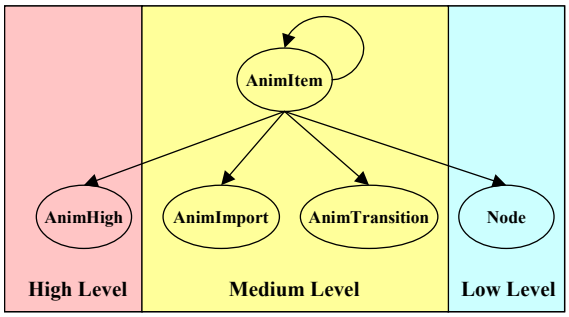
表現的動畫。優點是設計者可以用較簡便的方式表達動畫設計的理念。相對而言，此方式的缺點是動畫最終的結果可能因動畫呈像軟體的不同而有差異。另外，將高階指令轉換成低階的動畫指定有可能因計算的複雜度而無法達到即時動畫的要求。

整體而言，高階與低階的方式各有利弊，最佳的使用方式應是能依照所要製作的動畫特性、現有的播放軟體、及動畫的應用要求，選擇適當的表示方法。即使是對同一個應用，也可能因現有動畫庫或播放軟體的能力，而需在不同時機彈性地選擇不同層次的動畫指定方式。本論文所提出 XAML 動畫腳本語言的設計目標，便是希望能讓不同的的動畫指定方式互相協調運作，在想要輕鬆指定的時候能使用高階的語法，而在想要作細微動作指定的時候亦能使用低階語法。

3.2. XAML 語言的語法與結構

圖一標示了 XAML 的基本文法，在 XAML 中，任何自訂的動畫都是由<AnimItem>作為起始的標籤，新的動畫內容指定可以分為兩個部分：屬性(Attributes)和內容(Content)，屬性主要是管理邏輯上的控制，例如撥放時間、動畫內容篩選等來控制新動畫的表現；至於內容方面，在<AnimItem>中則可以放入要新產生的動畫實體，動畫實體內容可以藉由<AnimImport>作動畫元件重複利用或者是利用<Node>作低階的動畫指定。

XAML 語言的語法及的結構如圖二及圖三所示。粗略地說，我們將動畫指定方式區分為三種不同的形式：高階、中階和低階(如圖四所示)。在低階的表示法中，首先要在 Node 的 target 參數中指定要變化的物體。例如，動畫的目標如果是虛擬人物的人體動作，則 target 給定的字串可以是 H-Anim [1] 中所定義的各個關節點(Node)的名稱。另外，



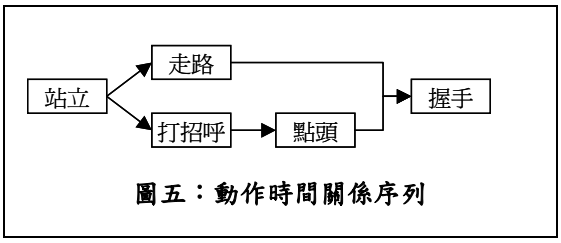
圖四: 不同層次的動畫指定方式

在低階動畫中，物體移動轉動都是參照所給定的 key frame mapping，這部分參考 X3D [3] 的動畫指定方式，將他分為<OrientationInterpolator>和<PositionIntepolator>兩種，而 key 和 key value 則是該標籤的屬性。

高階的動畫指定目前先暫定使用口語的文字，如"Walk to café."。不過目前只能接受簡單的文法結構，也就是動詞+介詞+名詞，而且動詞及受詞也必須在動畫資料庫中有紀錄的才可以被接受。

動畫的指定除了空間上移動的關係，還有時間軸的指定，在 XAML 中，我們將時間控制的參數放置於 AnimItem 的屬性，藉由調整時間屬性來控制不同的動畫效果，由 AnimItem 所包含的動畫元件會參考相同的時間設定，例如動畫的起始時間；另外一方面，子代動畫元件也會受到親代的時間限制，例如親代的動畫結束時，所有子代動畫將會被強制結束。如此一來，上層動畫所設定的時間參數就是真實的撥放時間。使用者只要修改最上層的時間參數即是定義最後呈現的風貌，不用擔心其中一個動畫元件可能會超出限制。

3.3. XAML 語言的時間控制模型



圖五：動作時間關係序列

XAML 的時間控制模型是使用限制傳遞（constraint propagation）的方式[1]。在人工智慧的領域裡，限制傳遞方式通常被應用在需要對時間作推理的系統上。這類的系統最大的相同點就是使用圖的結構表達事件時間上的關係。XAML 就語言的結構而言是樹狀結構，但是意義上是圖狀的結構，我們每次新定義一個動作都必需指定他的撥放時間跟起始時間，就如圖五所示，在站立完成後會有一個分岔點，會進行不同的動作，等這些分歧的動作都撥放完畢後，整體動畫會回歸至下一個動作，所以我們說他是圖狀的結構。

```
<AnimItem mode="seq">
      <AnimItem DEF="A" start="5"> …
      </AnimItem>
      <AnimItem DEF="B" start="50"> …
      </AnimItem>
</AnimItem>
```

圖六：以 seq 指定動畫元件循序播放的範例

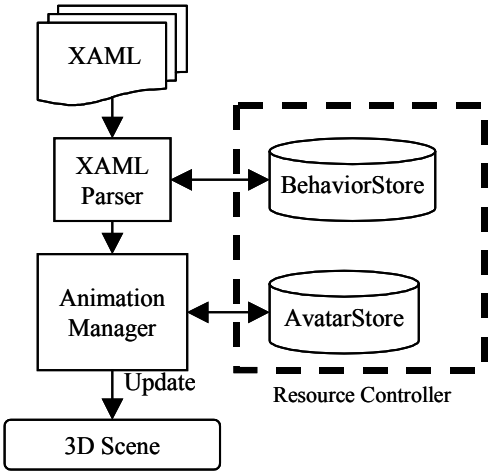
```
<AnimItem DEF="DrunkWalk">
  <AnimImport src="walk" only="LowerBody" />
  <AnimImport src="bow" only="UpperBody" />
  <AnimImport src="nod" only="skullbase" />
  <AnimImport src="five" />
</AnimItem>
```

圖七：從現有動畫項目中合成新動畫

這些動畫跟動畫之間的箭頭是表示兩者之間的時間關係，這些時間關係共有 13 種，在 XAML 中可以利用 start、end、cycle 等時間屬性來指定想要表現的時間關係，start 跟 end 是表示該動畫元件的起始時間和結束時間，預設值是以親代的起始時間點作為時間的參考點，時間是以相對的關係作為指定。另外也可以使用絕對時間指定動畫開始結束時間，但是子代的動畫依舊會受到親代的限制，只有親代的動畫開始運作，子代動畫的時間才有可能被考慮。cycle 是指定動畫撥放一次所需要的時間，藉由 cycle 的再次定義，可以將動畫作時間上的縮放，以便加速或減緩該動作。也可以調整 loop 的值並更動畫重複撥放的次數。

雖然使用區間的方式可以提昇動畫表達的能力，但是如果需要使用者去配置所有的時間關係仍是有些繁瑣。在某些情況下我們會希望給予動畫撥放的方式一些限制，以使動畫內容的製作更佳方便。在眾多的條件中，最常使用的就是動畫的循序撥放。所以我們在 AnimItem 這個標籤中，加入一個撥放模式的屬性。這個屬性可以有兩種值：平行撥放(par)和循序撥放(seq)。不同的撥放方式會讓子代的時間屬性有者不同的意義；平行撥放讓所有子動畫元件有者相同的時間參考起點。在一個 AnimItem 所管轄的範圍下，所有的動畫時間關係均使用 Allen 所提出的時間模型。

當一個 AnimItem 的撥放模式是循序撥放時，就是假設各個動畫事件彼此並不重疊，也就是說子動畫是一個撥完之後另一個才可以被撥放。所以動畫開始撥放的時間是以前一個動畫結束的時間當作起點，如圖六所示，A 動畫會在整個動畫開始 5 毫秒後開始撥放，B 動畫則是在 A 動畫結束後 50 毫秒後撥放。seq 可視為是 par 的一種特殊變畫，



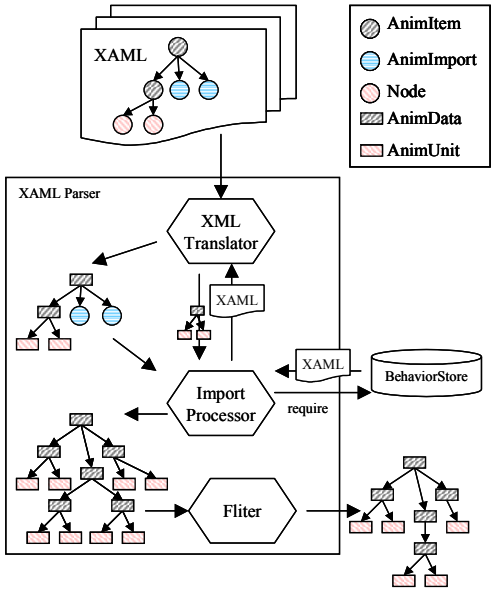
圖八：系統架構綜覽

一個用 seq 表示的內容就可以找到一個相對應的 par 的表示，seq 因為被限定不能 overlap，所以在 13 種時間關係上只剩下 4 種關係可能被發生。雖然 seq 所表達的空間一定會被 par 所表達的空間所包含，但是增加使用上的便利性，因此我們在 AnimItem 上還是加了撥放模式這一項屬性。

3.4. 動畫元件的重複利用與銜接

XAML 另一項重點就是動畫元件的重複利用，所以我們另外設計了 AnimImport 這個標籤。這個標籤可以輸入之前定義過的動畫，不過如果原本的動畫內容可能會一些不想要的部分，則需要有方法可以排除；尤其在平行撥放動畫的時候，使用者會希望藉由不同部分的局部動作組合成新的動作。所以，如圖七所示，我們加入了一個 only 的屬性，根據使用者給定的值作篩選(Filter)的動作。藉由 import 跟 filter 的結合，使用者便可以輕易地製作出新的動畫內容。

引用已經定義好的動作除了有內容修改的問題外，另一個值得關心得問題就是動作的銜接。例如我們希望虛擬演員走到特定地方後揮手，虛擬演員前一刻手應該還停留在走路的震臂動作，下一秒就要變成揮手的準備動作，兩個動作之間手臂會有跳躍的現象。所以我們設計了一個 AnimTransition 標籤，用來自動產生動畫間的連接動作。在目前的設計，AnimTransition 中動畫的時間模式都是循序(seq)的；因為當動畫有 overlap 的情況發生時，這部分的自動產生還未完成，所以我們強制限定在此標籤下僅能為 seq 的撥放方式。這個作產生過渡動作所使用的 AnimTransition 可以被設計成一個 AnimItem 的屬性。但是為了增加此銜接動作未來變化的可能，因此我們將此功能獨立出來成為一個標籤。如果在銜接動作中要加許多的規則以增加其表現方式，我們可以透過屬性設定的方式，加入到



圖九： XAML 語法剖析器的運作方式

AnimTransition 的標籤中，達到延展性的目的。

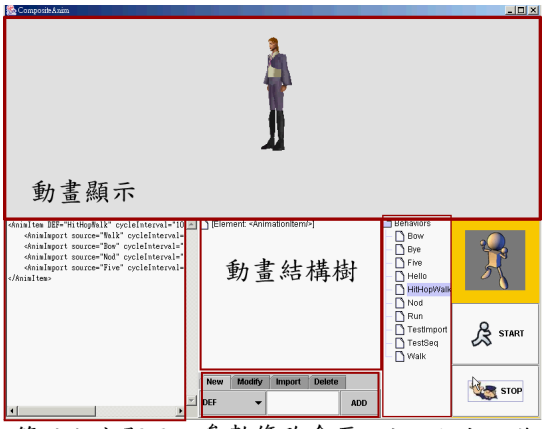
4. 設計與展示系統的實做

目前我們已為 XAML 腳本語言實作了一個設計(Authoring)與展示的系統。這個系統使用 JAVA 3D [6] 作為 3D 物件的顯像程式庫，並以 VRML 做為幾何模型的表達格式，整個運作流程如圖八所示。這個系統主要由三個部分組成：XAML 語法剖析器(Parser)、動畫管理器(Animation Manager)、及資源控制器(Resource Controller)。

4.1. XAML 語法剖析器(Parser)

在本系統中，所有以 XAML 表示的動畫腳本，在實作上我們都先轉換成由更低階的表示法：AnimUnit 和 AnimData 兩種組成。AnimUnit 這個元件單純地存放物體移動旋轉的數值資料，也就是物體幾何描述的座標轉換(Transformation)關係。這個標籤記錄了物體如何旋轉和移動，但是有關於時間的參數完全不紀錄，所有時間上的相對關係都是放置於 AnimData 中。從階層關係來看，AnimUnit 不能獨立存在，因為 AnimUnit 一定會被一個 AnimData 管理。但是 AnimData 所管轄的元件卻不一定都是 AnimUnit，因為也有可能是另外一個 AnimData。XAML 語法剖析器在這裡所扮演的角色便是將外部的 XAML 腳本轉換成動畫顯示用的內部表示法；也就是將使用者所輸入的 XML 樹狀結構轉換成內部的樹狀結構。

XMAL 語法剖析氣得運作方式分成三個階段：XML 轉換器(Translator)、語法處理器(Syntax Processor)、及動畫篩選器(Animation Filter)，如圖



圖十：應用程式介面

圖十一：酒醉走路的範例展示

九所示。首先，一個 XAML 的 XML 文件中比較單純的部分(如 AnimItem)會先被語法剖析器中的 XML 轉換器轉換成相對應的 AnimData 所構成的樹狀結構。像 AnimImport 或 AnimTransition 等需要額外資訊的部分則留待語法處理器(Syntax Processor)再處理。匯入處理器是語法處理器的一個特例。此語法處理器主要在處理以匯入(Import)方式建立的節點，方式是從外部檔案取得動作的 XAML 文件，再交由 XML 轉換器轉換成一個 subtree，然後在將此 subtree 取代原本匯入的節點。最後，動畫篩選器根據 only 的屬性或是動畫的優先順序作節點的刪除，最後所得到的樹再交由動畫管理器進行管理及播放。

4.2. 動畫管理器(Animation Manager)

在 XAML 語法剖析器處理之後，動畫腳本中的所有動畫均可以樹狀結構表示。一個樹狀結構的根節點(Root Node)即代表一個獨立的動畫。因此，動畫管理器只需記錄此樹狀結構最上層節點的 AnimData，並讓此 AnimData 自己管理他所管轄的動畫內容。在實作上，動畫管理器只是一個簡單計時器；它會定期的被觸發，然後循序的呼叫 AnimData 去撥放下一個動畫資料。此時 AnimData 會自己判斷該不該開始撥放動畫；如果已經過了開始時間，它會根據已經撥放時間和使用者所給定動畫設定計算出他此次應該撥放的進度，也就是相對一個完整動畫的百分比，然後將這個百分筆值傳給它所有直接管轄的 AnimUnit，當 AnimUnit 收到後會根據之前所給定的 key mapping 參考出新的 Key Value，然後進行低階的座標轉換，計算出相對應的畫格；另外它也會去呼叫它所管轄的 AnimData，讓它不斷的遞迴呼叫，直至這個數讓結構所有的動畫都被重新更新。

誠如前文所敘，一個動畫元件的撥放會受到親代的限制，也就是親代動畫撥完一個迴圈要開始一個新的迴圈之時，子代的所有動畫也會重新設定，必須跟隨著親代展開新的迴圈，因此當 AnimData 在進行動畫更新時，它同時會檢查該度畫是否為一新迴圈的開始，如果是，它會重新設定所有子代的 AnimData 的起始時間，起始時間的設定會根據子代不同的撥放方式而作不同的更改，藉由重新設定這些參數強制子代結束動畫，開始新的迴圈。

雖然在語法剖析器中已經有做過 Filter 的動作，但是某些部分的撥與不撥還是需要動態才能決定。Filter 只是做基本的靜態取代，降低系統的負荷。在動畫管理器中，會動態的依據親代的限制決定子代動畫撥放內容，例如撥放時間的長短。也就是說親代的屬性限制會在這時候被檢查。

4.3. 資源控制器(Resource Controller)

資源管理器包含兩類資料的管理：行為的動畫資料及虛擬人物模型資料。虛擬人物的行為動畫資料是透過一個叫做 BehaviorStore 的機制取得事先定好的動作。目前在這個機制下，每一個動作都是以一個 XML 文件的方式儲存。就長期管理的角度看是有點繁瑣，不過這個機制保留了比較大的彈性，可以讓取得動作的方式更多樣化。另外，為了之後此腳本語言在多人虛擬環境系統上使用的考量，所有的人物模型都是以動態的方式加入場景中。目前在實做上使用者可以透過一個稱為 AvatarStore 的機制來獲得各項虛擬人物的資訊，並動態加入新的人物。

5. 實驗成果

如圖十所示，我們開發了一個動畫設計軟體以檢視動畫的合成效果。這個程式在介面上約略可以分為五個區塊，分別是動畫顯示、外部動畫定義、動畫結構樹、參數修改介面、及節點內容顯示。

「動畫結構樹」區域是這個軟體最主要的地方，使用者製作的 XAML 會以簡單的階層結構顯

示在此視窗。使用者藉由點選結構樹上任一節點來指定要修改的動畫節點；至於要做哪些部分的修改，則可以用參數修改介面來修改。另外，使用者可以新增、修改、外部引用和刪除節點，藉由這些動作產生不同的動畫內容。「節點內容顯示」區則是顯示出使用者所點選的任一節點的 XML 內容；也就是說，當使用者動畫結構數或外部動畫定義點選了任何節點，這個視窗會顯示出該節點和節點以下的所有內容。這可以讓使者輕易了解單一動畫項目的內容，而不是每一次都要從一大段的動畫定義找出要修改的部分。「外部動畫定義」區則是列出一些已經定義好的動畫，方便使用者引用到自己新定義的動畫中；如此以來就相同的動作就不需要重新定義。最後，輸入的語言所造成的效果則可在「動畫顯示」區中呈現得知。

在圖七的範例中，我們利用外部動畫的引用合成出一個新的動畫。在酒醉走路的動畫裡，包含走路、彎腰、點頭、搖手四個動作，整個動畫就如圖十一所示。我們假設除了第四個動畫都可以在原貌完整播放外，其餘前三個動作雖可以在動作資料庫中找到，但每一個動作我們都只取其中一部份肢節的動作。

6. 結論與未來延展

在此論文中，我們提出了一個具有延展性的動畫腳本語言，稱為 XAML。此以 XML 為基礎所建構的語言，能完整表示所有動畫項目在時間上應有的關係，並且能彈性的協助使用者以不同層次的方式，重新使用現成的動畫元件。我們並實做了一個能剖析此語言並將動畫展現出來的軟體系統，以說明此腳本語言的延展性。我們相信透過此語言的設計，虛擬環境中動畫的指定方式，將更具有彈性。

目前此語言的功能還是著重在動作的重新組合上。只要動畫資料庫夠豐富，要再延伸出新的動畫並不困難。但是 XAML 所遇到的挑戰就是動作的豐富性。例如，同樣是揮手，目前只能作揮手的頻率、次數等簡單的設定；但是揮手可以有很多種不同的可能，像節奏的改變目前就無法使用簡單的方式來設定。但是此語言的特性就是能依據使用者的需求，加入特定的動畫節點及修飾屬性。只要腳本的設計者與顯示者能有共識，任何節點的設計都是可能的延展。

XAML 目前的定位是虛擬演員所使用的動畫劇本，多人的虛擬環境是它一個可以發揮的舞台。當他被應用在多人虛擬環境的時候，互動性的功能便變得很重要。我們可以因應使用者的輸入或不同的環境作出不同反應的動作。另外，不同範圍下動作的同步化也是一個問題，這些互動上的議題都需作進一步的探討。

最後，這一套語言的可延展性也可以讓開發人員加入動畫以外的延伸標籤。例如語音的部分，只要參照 AnimItem 的定義就可以套用 XAML 的時間模型，讓虛擬演員一邊說話一邊作動作，讓 XAML 所能展現的效果更為豐富。

7. 致謝

此研究在國科會 NSC 91-2213-E-004-005 計畫的支助下完成，特此致謝。

參考文獻

[1] J.F. Allen, “Time and time again: The many ways to represent time,” *International Journal of Intelligent Systems*, 6, 4, pp. 341-356, July 1991.

[2] J.F. Allen, “Maintaining Knowledge about Temporal Intervals,” *Communications of the ACM*, 26, 11, pp. 832-843, November 1983.

[3] Extensible 3D(X3D), http://www.web3d.org

[4] H-Anim, http://www.h-anim.org

[5] Zhisheng Huang, Anton Eliens, and Cees Visser, “STEP: A Scripting Language for Embodied Agents,” *Proceedings of the Workshop on Lifelike Animated Agents*, 2002.

[6] JAVA 3D, http://www.j3d.org

[7] S. Kshirsagar, A.Guye-Vuilleme, and K. Kamyab, “Avatar Markup Language,” *Proceedings of 8th Eurographics Workshop on Virtual Environments*, pp. 169-177., May, 2002.

[8] R. Parent, *Computer Animation: Algorithm and Techniques*, Morgan Kaufmann, 2001.

[9] K. Perlin, and A. Goldberg, “Improv: A System for Scripting Interactive Characters in Virtual Worlds,” *Proceedings of SIGGRAPH 96, ACM Press*, pp. 205-216, 1996.

[10] P. Ladkin, and R. Maddux, “Representation with Convex Time Intervals,” TR KES.U.88.2 Kestrel Institute, Palo Alto, CA, 1988.