以 OSGi 實現多人虛擬環境系統中客製化動畫之產生機制

Realizing Customizeable Animations in a Multi-user Virtual Environment using OSGi Framework

朱鈺琳
國立政治大學資訊科學系
g9505@cs.nccu.edu.tw

李蔡彥
國立政治大學資訊科學系
li@cs.nccu.edu.tw

陳正佳
國立政治大學資訊科學系
chencc@cs.nccu.edu.tw

摘要

多人虛擬環境(Multi-user Virtual Environment)系統的應用越來越多，而系統的延展性及內容的豐富性是未來 3D 內容服務能否普及的關鍵之一。IMNET 是一套具延展性的多人虛擬環境系統，可在編譯或執行期間，整合不同的動畫元件模組，增加系統的彈性[8]。使用者可以透過 XAML(eXtensible Animation Modeling Language)語言的設計，產生虛擬角色的動畫；然而，對於高階指令的實現，目前多僅能由系統提供有限的選擇，而無法由使用者自行設計。本論文的目的在 IMNET 的平台上，設計一個能由使用者自行擴充動畫標籤及實現此標籤之動畫元件的機制。當使用者開發一個新的動作時，可以同時讓所有線上使用者接受動作的擴充，而不需要重新啟動 IMNET 系統，或是手動的安裝。本論文是以 OSGi Framework 來建立動畫程序元件下載、安裝及執行的機制，以達到動態擴充新元件的目的。我們將以實例說明此設計的功能及執行的程序。在未來我們希望能透過這個機制實現語意虛擬環境(Semantic Virtual Environment)，讓虛擬人物能與未定環境中的人物進行互動。

關鍵詞：多人虛擬環境系統、語意虛擬環境、OSGi、延展性動畫腳本

一、前言

多人虛擬環境(Multi-user Virtual Environment，簡稱 MUVE) 是一個可供多人同時上線，透過文字或 3D 虛擬人物在特定應用中達到互動效果的虛擬世界。在多人虛擬環境裡面，使用者透過虛擬環境系統所提供的文字、圖像、語音、或動畫服務，進行各式各樣的互動與交流。近年來盛行的線上 3D 遊戲，便是特定虛擬環境系統的成功應用之一。這些遊戲除了透過華麗的角色和場景來吸引使用者之外，更提供各種富有挑戰性的劇情，可以讓人們在遊戲中擁有不同的趣味。一般而言，虛擬環境系統提供許多的虛擬角色模型，讓使用者可以選擇自己喜愛的動畫角色在虛擬環境中活動。除此之外，虛擬環境中的角色如何展現個人的風格，也是所有虛擬環境系統所極力希望達成的。另外，隨著 Web 2.0 時代的來臨，如何讓參與虛擬環境的使用者自行設計屬於自己的動畫行為，並與其他使用者分享，將是 3D 數位內容能否在虛擬環境中普及的重要關鍵之一。

然而，現有傳統虛擬環境系統的設計尚難達到上述目標，主要原因是一般虛擬環境中每個角色的行為是固定的(例如固定的行走或擺動方式)；即使這些動作是可以由使用者選擇切換的，所有動作的新增也必須由系統設計者加入後再安裝到各個客戶端程式。由於缺乏延展性，因此目前的系統尚難達到個人客製化動作並即時線上安裝執行的目標。

IMNET 是一個以 XML 為基礎的主從式虛擬環境系統，透過 XAML (eXtensible Animation Modeling Language)語言的設計讓使用者可以整合

```
<AnimItem model=”avatar” playMode=”seq”>
    <AnimHigh>Walk to Door</AnimHigh>
    <AnimImport src=”PushDoor” />
    <AnimIetm>
      <AnimPlugin><Audio src=”scream.wav” />
      </AnimPlugin>
      <AnimItem model=”door” DEF=”DoorOpening> …
        </AnimItem>
    </AnimItem>
    <AnimImport src=”Shock” />
</AnimItem>
```

圖 1 以 XAML 語言指定及擴充動畫腳本的範例

不同層次的動畫指令於同一腳本中，以操作虛擬環境中的人物模型[8]。XAML 另一個設計目標便是讓腳本的設計更具延展性。如圖 1 中，<AnimItem>是 XAML 中動畫元件的基本單位，而<AnimHigh>及<AnimImport> 是XAML中所提供以高階指令及檔案匯入的方式產生動畫的標籤。其中，<AnimPlugin>的標籤讓使用者可以嵌入系統未內建的動畫元件或應用模組，包含在此標籤中的 XML 碼將交由相對應的外部處理程式進一步處理，因此可以達到擴充的目的。但是，這些對應必須在程式開始執行前寫入組態檔中，方能使系統順利找到處理特定標籤的程式。如果系統要更新動畫程序或加入新的標籤，都必須重新啟動系統方式完成。

本研究的目的在提供多人虛擬環境系統一個動態載入動畫程序的機制，以達到讓使用者自行客製化虛擬角色之行為及動畫程序共享的目的。我們在 IMNET 的系統中加入 OSGi 的機制，讓虛擬環境的使用者可以動態安裝開發者提供的動作到虛擬人物上，在執行時間(run time)完成元件的整合，而不需要重新啟動系統。在實作設計上，當 IMNET 處理輸入的 XAML 時，如果遇到未知的 XAML 標籤時，能夠自行根據標籤指定的位置下載程式碼，並且動態安裝到 IMNET 內部的 OSGi Framework 下，然後呼叫並執行此程式碼。我們將以實例說明此設計的使用方式及此機制的未來延展可能。

在接下來的各節裡，我們將介紹虛擬環境系統、語意虛擬環境與 OSGi 等相關研究；第三節介紹如何在 IMBrowser 中加入 OSGi 機制；第四節提供一個實際的例子說明加入 OSGi 機制的方式；最後一節為結論及未來延展的可能性。

二、相關研究

在虛擬環境系統的文獻上，已有許多系統被提出，如 RING[5]、DIVE[4]、及 MASSIVE[6]。S. Benford 等人在 2001 年的協同式虛擬環境(Collaborative Virtual Environments)論文中提到如何從現有的網路通訊，利用更吸引人的 3D 環境，讓所有線上參與者彼此互動[2]。S. Benford 也指出在 CVEs 的研究上有許多具挑戰性的問題；其中最大的挑戰來自於如何在有限的網路流量下，讓線上參與者能同時在虛擬環境中作業。在 IMNET 系統中，我們利用高階的 XAML 動畫腳本語言，來減少虛擬人物動畫所需傳輸的資料量[7]，但是如前所述，先前的 IMNET 版本尚無動態定義動畫標籤及載入動畫產生程序的功能。此動態載入動畫程序的功能如果能在虛擬環境系統中實作出來，亦將有助於所謂「語意虛擬環境(Semantic Virtual Environment)」[12]的實現。

傳統虛擬環境系統在內容的表現和互動上大多是以人類為設計的焦點，而不是給電腦程式來使用的[11]。一般的虛擬環境利用場景樹(Scene tree)來存 3D 空間中的幾何資訊，然後經由 3D 瀏覽器顯示到使用者面前。人類可以輕易理解場景樹經過算圖後所呈現的圖像結果，但是對於電腦程式而言來說只能知道場景樹的幾何及繪圖屬性，而不會知道最後所表示的物體是什麼。比如說，人類看得出一個長方體與四個圓柱體，在算圖後是一張桌子，而機器卻無法理解這是桌子。此問題在一般網頁內容中一樣存在，這也是為何「語意網(Semantic Web)」[13]被提出的原因之一。

語意虛擬環境的研究提出以豐富的語意來描述虛擬環境。語意虛擬環境使用一個抽象的觀點來看待存在的虛擬環境，並且利用語意網(Semantic Web)的技術來描述世界模型和網路通訊模型的語意。使用語意網的目的在於讓機器也能夠了解在虛擬環境中，場景所代表的涵義。面對可能多變且複

雜的虛擬環境，語意虛擬環境提供一個在虛擬環境上開發新元件的方式，使得這些新元件可以在具有語意的虛擬環境下重複使用，突破不同虛擬環境間的侷限。

在語意虛擬環境的相關研究上，K. A. Otto 所開發的 SEVEN 是一個可利用軟體元件來開發 SVE 虛擬環境代理人的系統，發揮軟體的可重複使用的特性，使其能快速的將開發好的軟體套用在不同的虛擬環境上。本論文與 SEVEN 的不同之處在於，我們主要著眼於虛擬人物在行為上，而不是在虛擬環境的模型上；不過建置完整的語意虛擬環境也是 IMNET 系統目前努力的方向之一。

在動作的語意方面，T. Abacı 等人設計了一個與動畫需求有關的語意表達方式，使虛擬環境系統中的物件(Smart Object)不只可以提供幾何上的資訊，同時也能提供相關的動畫資訊[1]。如此一來，動畫中的物件可以彼此間有所互動。例如，虛擬人物在移動到目標地時，遇到一個具有語意描述的障礙物，藉由障礙物的描述，可以知道如何去搬運它。

實現上述語意虛擬環境的關鍵技術之一在於如何動態載入動畫軟體元件，並允許元件間透過預定或協商過的介面進行溝通。OSGi(Open Services Gateway Initiative)便是實現此一機制的技術之一，也是本研究所選用的機制。OSGi Framework 是由 OSGi Alliance 所制定的一套軟體元件或服務的佈署與執行平台[9][10]。OSGi 主要是希望能增加軟體的重複使用性、平台的獨立性、解決軟體複雜度日漸龐大問題、由物件導向到服務導向等等。OSGi 的制定也使得元件製作更小更簡單、元件獨立性高、可適應目前多變的產品應用、重複使用性高等優勢。Eclipse 的 Equinox 便是 OSGi 規範的一個實作，透過 Equinox 可以啟動一個 OSGi Framework，然後將符合 OSGi 規範的 bundle 安裝到 Framework 裡面[3]。本論文就是使用 Equinox 作為系統的 OSGi Framework。

三、系統架構設計

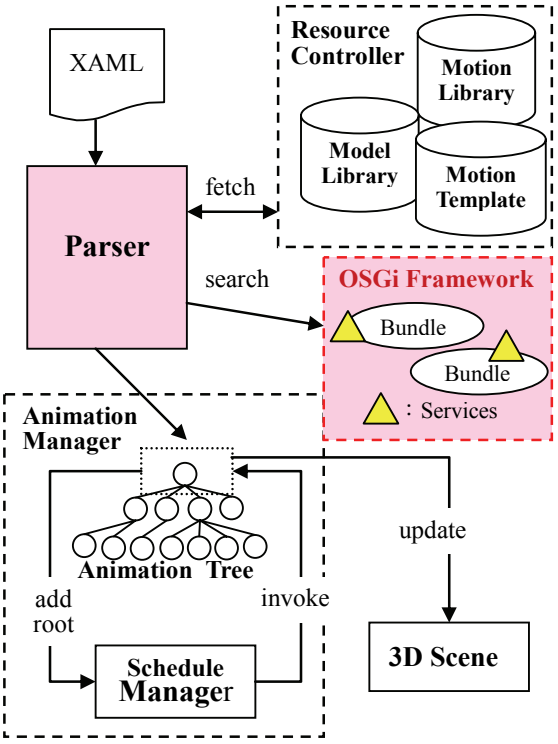

圖 2 XAML 動畫系統的架構

IMNET 虛擬環境系統中核心的軟體元件為 IMBrowser 3D 瀏覽器，而此瀏覽器的特色在於能剖析並處理以 XAML 指定的動畫腳本。在本論文所提出的系統中，我們主要是在 IMBrowser 上設計一個可以動態安裝新元件的機制，並且擴充原先的 XAML 語法，讓它可以指定一個位址來下載新的程式碼。在安裝新元件的機制上，我們在原先的 XAML 系統中內嵌 OSGi Framework，以用來處理新的元件，最後在 3D 瀏覽器上呈現執行結果。

圖 2 為 XAML 動畫系統(如 IMBrowser)的架構。此系統主要由 XAML 語法解析器(Parser)、動畫管理器(Animation Manager)、資源控制器(Resource Controller)和新增的 OSGi Framework 所組成。XAML 文件會先經由語法解析器來處理，轉換成內部使用的動畫結構樹，記載動畫優先權、播放時間和動畫內容等資訊，同時藉由樹狀階層的架構掌握動畫的播放時程。在語法解析器轉換 XAML 成為動畫結構樹的過程中，每一個 XAML 文件的標籤都會對照一套特定的轉換模式，轉換成相對應的動畫結構樹節點。基本的 XAML 標籤都有各自

的轉換模式，當使用延展性的語言標籤時，就必須事先將這個標籤的轉換模式註冊到語法解

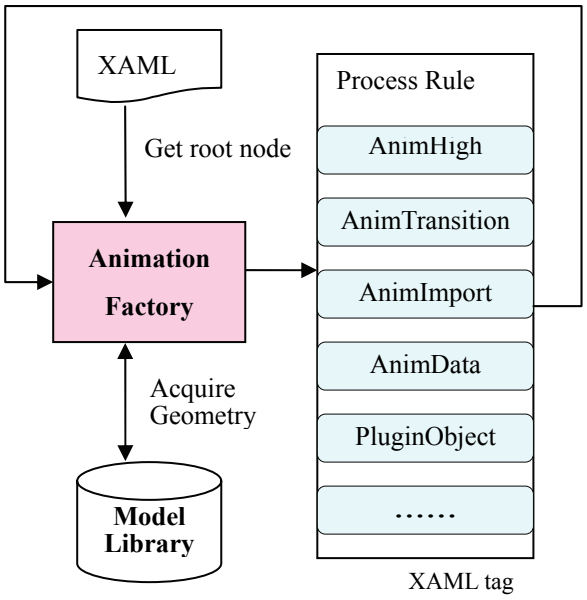

圖 3 XAML 語法解析器流程

析器中。本系統與先前版本不同的是當某個標籤並非基本的 XAML 標籤，也還沒有註冊到語法解析器的時候，本系統的語法解析器會試著在 OSGi Framework 中尋找可以處理這個標籤的轉換模式，或是經由標籤中的 codebase 屬性直接下載標籤的轉換模式，並且自動安裝到 OSGi Framework 中。

OSGi Framework 提供了一個 Symbolic Name 的管理機制，相同名稱且版本相同的 bundle 同時安裝到 OSGi Framework 中是不被允許的。另外，Equinox 會將安裝過的 bundle 都記錄下來，以便下次重新啟動 OSGi Framework 時，可以回復到上次關閉前的狀態。所以使用者即使重新啟動 IMNET，也不需要再重新安裝之前安裝過的轉換模式，而且在下載 codebase 所指定的轉換模式時，如果這個轉換模式已經存在 OSGi Framework 中，就不會再重複下載。

在 IMNET 中，處理 XAML 的部分主要是依靠內部的語法解析器，而我們主要的著手點在於修改語法解析器在處理 XAML 時的方式。在以下第一小節中我們會介紹原先解析器的處理流程；在第二小節裡則會介紹加入 OSGi 機制後的處理流程。

（一）原先解析器的處理流程

在 IMNET 中，使用者與虛擬環境互動的方式主要是透過 XAML 語言。當 IMNET 收到一段 XAML 時，語法解析器會以這段 XAML 的根節點為起點，然後將每一個處理到的元素轉換成 AnimData 的樹狀結構。元素的處理方式會因為其型態而有不同的轉換模式，所以 XAML 的開發者要為自己的元素撰寫一套專屬的轉換模式。在處理 XAML 的過程中，透過 Animation Factory 模組來管理元素的轉換，如圖 3 所示。轉換模式中包含了 AnimHigh、AnimTransition、AnimImport、AnimData、PluginObject 和其他有向 Animation Factory 註冊的轉換模式。在找到相對應的轉換模式後，就會被轉換成 AnimData 的資料節點，然後繼續對子節點做轉換。最後產出的動畫結構樹將交由動畫管理器來產生動畫，顯示到 3D 場景上。

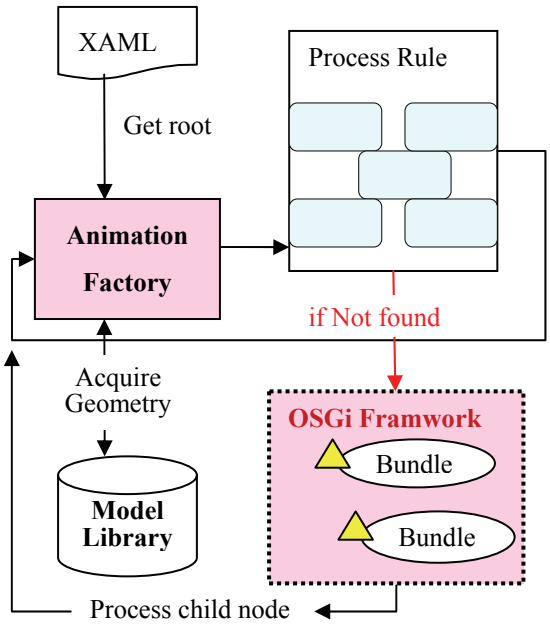

圖 4 加入 OSGi 機制後的處理流程

當我們在開發一個新的轉換模式時，設計好的轉換模式需要先手動安裝到 IMNET 系統下，然後完成註冊的手續。這種方式在單機的模式下還可

以接受，但是如果是在多人連線的情況下，新的轉換程式必須能在不重新啟動目前的系統的情況下，安裝到每個上線的 IMNET 系統上；而 OSGi 便提供了這樣的一個機制。

（二）加入 OSGi 機制後的處理流程

我們在原先的解析器中加入了 OSGi Framework，提供了另一種機制來處理新的轉換模式。在 IMNET 啟動的時候，會同時啟動一個 OSGi Framework，然後交由 Animation Factory 跟 OSGi Framework 互動。如圖 4，在 Animation Factory 處理 XAML 時，會先去尋找適當的轉換模式，如果找不到符合的模式，就會去搜尋 OSGi Framework 裡面有註冊過的服務，根據處理標籤的名稱來做比對。例如有一個名稱為 Bow 的標籤，這個標籤並不是 XAML 的標準標籤，而且也沒有跟 Animation Factory 註冊過，此時解析器就會去 OSGi Framework 中尋找名稱為 Bow 的服務，並且根據標籤中的 package 性質，來取得此服務的完整名稱。

在接下來的章節會提供一個實作的範例，來說明本系統的運作和利用 OSGi Framework 的機制來動態安裝 IMNET 的轉換模式。

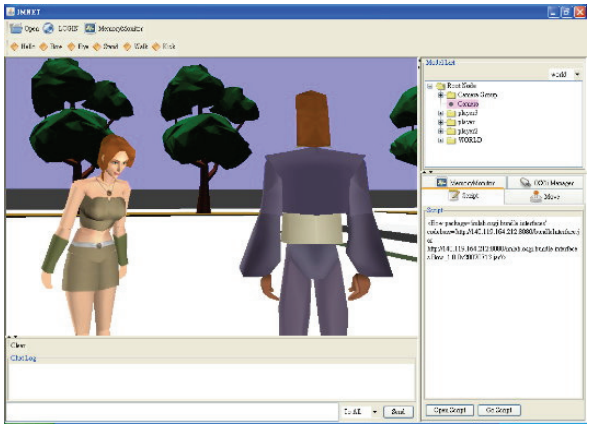

圖 5 鞠躬動作範例

四、系統實作與範例

在本節中我們將以一個使用者設計的「鞠躬動作」來說明本系統的運作方式，圖 5 所示便是在此範例中產生的鞠躬動作的截圖。在解釋這個範例時，我們將用程式開發者與使用者兩個角色來做說明。我們假設開發者負責實際做出符合規定的動畫產生 bundle，然後上傳到網路空間，並制定讓其他使用者執行的 XML 片段。使用者當接收到開發者所制定的 XML 片段時，將自動下載開發者的程式到 IMNET 系統中，並且執行結果。以下我們將逐項說明細節。

（一）製作 OSGi bundle

製作 bundle 的過程分為四個部份：介面、實作、bundle activator 及 OSGi 設定檔，如圖 6 所示。當 bundle 安裝到 Animation Factory 的時候，會呼叫介面中的 exec()函式，然後回傳一段包含 XAML 動畫腳本的 String 交由動畫引擎執行。

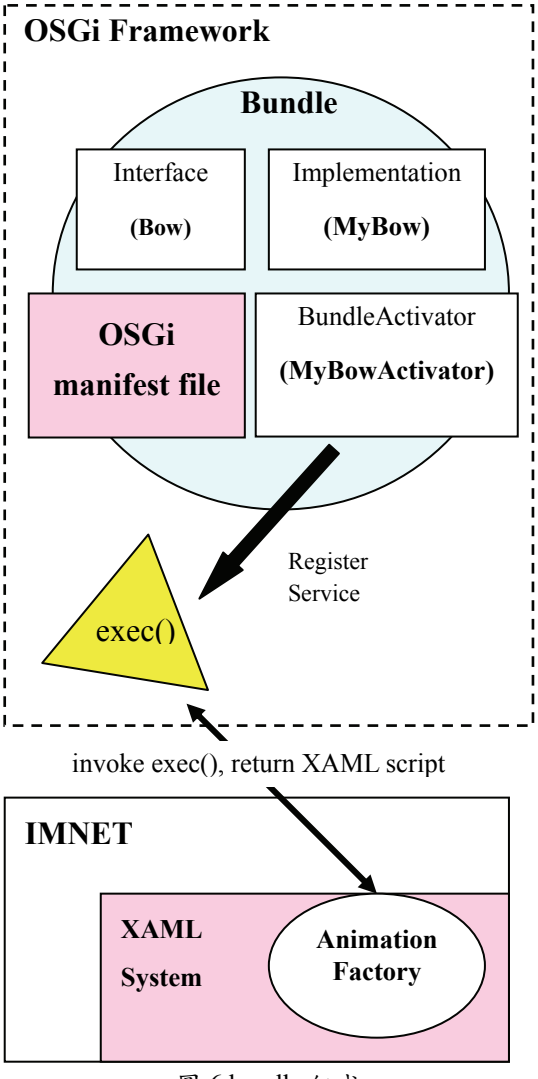

圖 6 bundle 組成

MyBow 實作了 Bow 介面，在 exec()函式中回傳一段 XAML 動畫片段(在本例中為鞠躬的動作)，如此一來在呼叫 Bow 服務的時候，就會產生鞠躬的動畫。另外，我們還需實作 BundleActivator 介面。BundleActivator 是標準的 OSGi 介面，當 bundle 在 OSGi Framework 中被啟動(Activate)時，就會自動執行 bundleActivator 中的 start()函式。在範例中，我們用 MyBowActivator 來實作這個介面。在 MyBowActivator 的 start()函式中，我們宣告了一個 bower 物件，以 Bow 為型態且 MyBow 為實作。然後使用 OSGi 的標準函式 registerService()將 bower 物件註冊成 OSGi 的服務，並將這個服務命名為 imlab.osgi.bundle.interfaces.Bow。

最後，我們還需完成 OSGi 的設定檔，此設定檔的內容大致如圖 7 所示。藉由這個設定檔 OSGi Framework 可以知道這個 bundle 的名稱和版本(bow_bundle_1.0.0)，在同一個 OSGi Framework 中是唯一的值。在這個設定檔中我們設定 Bundle-Activator 是 imlab.osgi.bundles.MyBowActivator，然後這個 bundle 匯出 imlab.osgi.bundle.interfaces 這個 package。如此一來，在 OSGi Framework 中的其他 bundle 就可以使用這個 package 了。最後階段是將所有東西包裝起來。一般來說，一個 bundle 會包裝成一個 JAR 檔(bundle 或 JAR 檔是 OSGi Framework 中的基本單位)。

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: bow_bundle Plug-in
Bundle-SymbolicName: bow_bundle
Bundle-Version: 1.0.0
Bundle-Activator: im-
lab.osgi.bundles.MyBowActivator
Bundle-Localization: plugin
Import-Package:
org.osgi.framework;version="1.3.0"
Export-Package: im-
lab.osgi.bundle.interfaces
```

圖 7 OSGi 設定檔

（二）上傳 bundle 和制定 XML 片段

Bundle 的開發者需要將完成的 bundle 上傳到某個伺服器空間，或是上傳到 IMNET 伺服器所提供的空間。如果要將 bundle 上傳到 IMNET 所提供的空間，系統會依照個人的帳號，分配屬於自己的命名空間(Namespace)，以區別不同人所上傳的檔案。除此之外，開發 bundle 的人也可以自行指定 URL 在 codebase 屬性上，讓 IMNET 可以透過 HTTP 通訊協定將程式碼自動下載到使用者端。在此範例中，我們將做好的 bow.jar bundle 上傳到 http://imlab.cs.nccu.edu.tw/bow.jar 的位置上。開發者必須制定一段 XML 片段來說明他的 bundle 要如何使用與下載，如圖 8 所示。這段 XML 的用途在於讓使用者端的 IMNET 系統得知處理 Bow 標籤的程式碼資訊，如 package 名稱和下載點 codebase。

```
<Bow package='imlab.osgi.bundle.interfaces'
codebase='http://imlab.cs.nccu.edu.tw/bow.jar' />
```

圖 8 含有 bundle 資訊的 XML 片段

（三）使用 XML 片段、下載和執行

最後的階段是測試使用者端是否能下載並且執行開發者所開發的程式碼。在 IMNET 的使用者介面中，使用者可以透過動畫腳本視窗輸入動畫腳本，並即時傳送到其他的使用者。當使用者收到這個動畫腳本時，這段 XML 腳本會被送到 Animation Factory 處理。由於它不是 XAML 的基本語言，也沒有在 Animation Factory 中註冊過，因此 Animation Factory 會去搜尋 OSGi Framework 中有沒有一個服務叫做 imlab.osgi.bundle.interfaces.Bow，如果沒有的話，就會去 codebase 屬性所指定的位置去下載 bow.jar。在 OSGi 的規範中， OSGi framework 必須提供安裝 bundle 所需要的 install()函式，所以 Animation Factory 可以透過 Equinox 的 install 指令，將 bow.jar 安裝到 OSGi Framework 中，並且自動的啟動 Bow bundle。當 Bow bundle 啟動的同時，會去呼叫 bundleActivator 的 start()函式，而註冊 bundle 服務的程式碼就在這個函式裡。所以當 bundle 啟動後，Bow 服務就被註冊到 OSGi Framework 裡面。最後 Animation Factory 會去呼叫 Bow 服務裡面的 exec()函式。依照範例的設計，此 bundle 服務會傳回一段 XAML 片段，然後再交由

Animation Factory 去執行這一段 XAML，最後在 3D 場景中產生預期的動畫。

五、結論與未來延展

多人虛擬環境可以讓許多人透過虛擬人物與其他人或環境互動，但目前大部分的系統都只允許在系統設計者所設計的功能下進行。隨著語意網與 Web2.0 時代的來臨，多人虛擬環境系統必須具備必要的延展性方能讓 3D 數位內容的設計與分享蓬勃發展。在本論文中，我們提供了一個方式讓多人虛擬環境可以讓使用者在程式進行當中安裝新的元件，以擴充自己或他人的動作。在 IMNET 系統中，我們利用 XAML 的延展性及 OSGi Framework，提供使用者設計新的 XAML 標籤，在完成註冊後，其他使用者能自動下載安裝並執行。在本論文中，我們並透過實例說明此機制的可行性。

然而，此動態元件安裝與執行的機制只是實現語意虛擬環境的一個必要機制。在未來的研究上，我們將朝向實現語意虛擬環境的方向繼續設計改進 IMNET 虛擬環境系統。例如，如何將環境中物件的幾何、屬性、功能等以 Ontology 的方式描述，並設計標準的程式介面，讓使用者所設計的程式，透過與語意虛擬環境的溝通，能取得所需的資料，進行進一步的推理，採取適當的行為，並產生相對應的動畫。我們希望虛擬替身之間也可以透過標準的介面進行互動。因此，如何發揮語意虛擬環境的長處，並利用系統的延展性來實現共同創作的精神等，都是我們將深入研究的地方。

六、參考文獻

[1] T. Abacı, J. C´ıger, “Action semantics in Smart Objects”, *Proceedings of Workshop towards Semantic Virtual Environments* (SVE 2005), 2005.

[2] S. Benford, C. Greenhalgh, T. Rodden, and J. Pycock, “Collaborative virtual environments”, *Communications of the ACM*, vol. 44, no. 7, pp. 79–85, 2001.

[3] Equinox, http://www.eclipse.org/equinox/

[4] E. Frecon and M. Stenius, “DIVE: A Scalable network architecture for distributed virtual environements,” *Distributed Systems Engineering Journal (Special issue on Distributed Virtual Environments)*, Vol. 5, No. 3, p.91-100, September 1998.

[5] T. Funkhouser, “Network Topologies for Scalable Multi-User Virtual Environments,” *IEEE VRAIS’96*, p.222-228, April 1996

[6] C. Greenhalgh and S. Benford, “MASSIVE: a collaborative virtual environment for teleconferencing,” *ACM Trans. CHI*, Vol.2, No.3, p.239-261, Sep 1995.

[7] T.Y. Li, M.Y. Liao, J.F. Liao, “An Extensible Scripting Language for Interactive Animation in a Speech-Enabled Virtual Environment,” *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME2004)*, Taipei, Taiwan, 2004.

[8] T.Y. Li, M.Y. Liao, P.C. Tao, “IMNET: An Experimental Testbed for Extensible Multi-user Virtual Environment Systems”, *ICCSA 2005, LNCS* 3480, pp. 957-966, 2005.

[9] OSGi Alliance, http://www.osgi.org/

[10] OSGi Technical Whitepaper, http://www.osgi.org/documents/collateral/OSGITechnicalWhitePaper.pdf

[11] K. A. Otto, “The Semantics of Multi-user Virtual Environments”, *Proceedings of Workshop towards Semantic Virtual Environments* (SVE 2005), 2005.

[12] Semantic Virtual Environments, http://page.mi.fu-berlin.de/otto/sve/index.html

[13] Semantic Web, http://www.w3.org/2001/sw/