一個能有效管理資料的開放式虛擬環境系統
An Open Virtual Environment System with Efficient Data Management

李彥霖
國立政治大學　資訊科學系
臺北市文山區指南路二段 64 號
s8705@cs.nccu.edu.tw

劉怡麟
國立政治大學　資訊科學系
臺北市文山區指南路二段 64 號
g8901@cs.nccu.edu.tw

李蔡彥
國立政治大學　資訊科學系
臺北市文山區指南路二段 64 號
li@nccu.edu.tw

摘要

多人虛擬環境系統結合了分散式系統與虛擬實境的技術，讓使用者透過 3D 瀏覽器觀看擬真的虛擬場景，並操控虛擬人物與其他使用者互動、對話，以達到娛樂、商業、醫學或學術上的目的。然而，一個 Client-Server 架構下的虛擬環境系統，由於 Server 端比 Client 端負擔較大的計算量與網路流量，整體的效能往往受制於 Server 端的計算與傳輸能力，以致影響到系統所能提供的同時上限人數。因此，在這篇論文之中，我們改寫一個開放原始碼的虛擬環境系統(VNet)，將 XML(eXtensible Markup Language)納入此系統的訊息傳輸協定中，希望藉由 XML 標準格式的彈性，能提供多人虛擬環境系統更具擴充性的訊息傳輸方式。我們進而在此開放平台上設計了一套資料管理(VEDM)的方法，實做出有效節省資料傳輸的演算法，如 Data Filtering、Dead Reckoning、Update Scheduling 等，透過降低網路 I/O 的需求，提升整體虛擬環境系統的效能。另外，為了方便在 3D 虛擬環境系統中的測試與實驗，我們在此系統中設計了虛擬使用者的功能，以使虛擬環境的實驗更加便利，並增加未來加值應用的可行性。

關鍵詞：多人虛擬環境系統、VNet、XML、VRML、虛擬環境中之資料管理、Data Filtering、Dead Reckoning、虛擬使用者。

1. 概論

多人虛擬環境(Multi-User Virtual Environment, MUVE)系統經過許多年的研究與應用，目前已有成功且多元發展的例子。在應用方面，透過虛擬場景及輔助介面的設計，此類系統能給使用者一個教學、娛樂或商業的環境，讓使用者不必外出也能在虛擬環境中完成真實世界裡的應用；例如瀏覽美術館的展出文物、與遠端使用者共同完成 3D 模型的設計等。在學術研究方面，多人虛擬環境的系統架構、彈性的訊息傳輸模式、使用者端的即時互動、以及有效率的傳輸方式，一直都是多人虛擬環境系統研究的重點。配合網路技術和電腦軟硬體設備的長足進步，多人虛擬環境系統已經不再是具備高階電腦的使用者才可參與的；一般使用者以個人電腦便可以輕易的參與多人虛擬環境，體驗其中的樂趣。

目前網路上有許多開放原始碼的多人虛擬環境系統，其中 VNet[12]提供了一個完全以 Java[8]語言實作的 Client-Server 系統架構，讓程式設計師可以自行修改其原始碼，以設計符合需求的功能。在這篇論文中，我們首先藉由改寫 VNet 這套系統，將目前在業界與學術界被廣為應用的 XML 標準納入 VNet 系統的訊息傳輸中，提供更具擴充性的訊息傳輸模式。這種訊息傳輸的模式能使虛擬環境中動畫的撥放方式更有彈性，並使資料傳輸格式更具延展性。

在此開放式實驗平台上，我們設計了一套資料管理（Virtual Environment Data Management, VEDM）的方法，以減低資料傳輸量，並以實驗方式證明其有效性。這些方法裡面包括了文獻上已提出的演算法，如 Data Filtering 及 Dead Reckoning 兩個演算法外，我們也針對 Server 端更新使用者資料時之排程做改善，以提高虛擬環境系統的效能。對於這些方法，我們也以實驗的方式比較這些方法的利弊得失，以做為設計使用時的參考。

為了方便實驗的進行，我們在此開放式虛擬環境平台上，設計了虛擬使用者(Virtual User，又名 Bot)的功能，讓使用者端以程式的方式產生並控制多個使用者。有了這些功能，虛擬環境系統便能輕鬆控制一定數量的同時上線人數，以便評估資料經過管理之後所得到的效益。另一方面，虛擬環境系統加入虛擬使用者的功能，也對於日

後虛擬人群方面或其他加值的研究有很大的幫助。

在第二節相關研究中，我們會介紹多人虛擬環境系統的架構以及虛擬環境系統中常見的問題；第三節介紹整體的系統架構，說明資料管理系統如何與 VNet 系統作整合；在第四節中我們介紹我們所用提昇效率的方式；第五節介紹虛擬使用者的功能；第六節是展示我們的實驗成果以及數據的分析；最後是我們的結論與未來發展。

2. 相關研究

多人虛擬環境系統有許多不同的系統架構，而目前最常見且廣泛應用的，是屬於 Client-Server 架構的多人虛擬環境系統[6]。伺服器端會負責管理虛擬場景，將場景中的變化傳送至使用者端；而使用者端則透過 3D 瀏覽器及輔助介面、讓使用者能見到場景中物件的改變，並且與其互動。另外一種多人虛擬環境系統為點對點 (peer to peer)的架構，便沒有伺服器的概念，而是以使用者與使用者間建立各自的連線方式來進行。

XML[14]的興起，讓管理資料的人員可以設計有意義的標籤名，為資料做標記的動作，讓資料本身更具意義；利用 XML 文件格式的嚴謹定義，與標籤間的樹狀結構，可讓資料在規則下更有彈性。在多人虛擬環境中將 XML 標準做為訊息傳遞的協定，可讓訊息的結構更為彈性並具擴充性，改變此類系統在傳統傳輸協定上不彈性的訊息結構，進而讓虛擬人物的動畫有更多元的呈現。

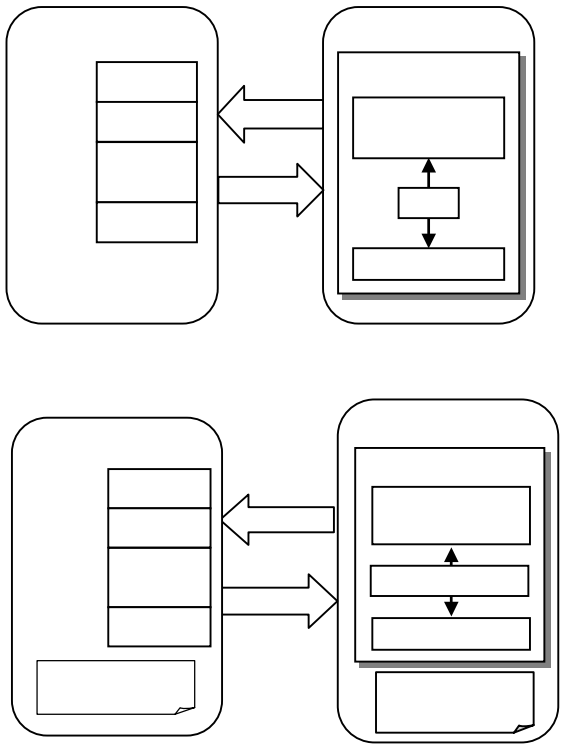
同步化問題[10]是虛擬環境中比較常發生的問題之一。造成不同步的原因又大致分成了 Client 和 Server 間的不同步(網路阻塞導致 Frame Rate 的變化量太大)與 Client 和 Client 間的不同步(硬體能力的差異)；如果忽略了同步化問題，使用者端所紀錄的資料便會有所差異，導致整個系統會出現物體動作不一致的問題。

網路傳輸能力的問題，也是在虛擬環境的研究中經常出現的[6]。為了配合網路的速度，減少傳輸的數量、讓傳輸更有效率等，這方面有許多相關的研究，像是 Level Of Detail（LOD）概念便是把使用者可以看到的物體做細微程度上的分級，當使用者的網路品質好，就能夠看到細微且精緻的 3D 模型；當網路品質較差時，看到的東西就比較粗略，不過卻能夠有效的解決延遲的問題。另外也有利用使用者在場景內的空間位置來做為傳送模型精細度的標準[7]，跟 LOD 有類似的精神。Dead Reckoning（DR）也是一個常用的方法，不過方法與上述兩種不太一樣，主要是以預測的方式來減少傳輸的次數，而並非降低每次傳輸的內容量，Data Filtering（DF）則是以過濾的方式來減少 Server 端傳輸的對象[5]。這一方面的研究最終都是希望能夠突破在網路傳輸方面的瓶頸。

3. 系統架構

我們在這一節中將深入說明整個虛擬環境系統與資料管理模組的架構。在本研究之中，由於考慮到系統的擴充性，我們選擇 VNet 這套開放原始碼的多人虛擬環境系統，因為它是以 java 語言來撰寫的，使用物件導向的觀念，具有較高的擴充性與延展性，方便我們把研究的結果加入系統之中。在以下小節裡，我們首先介紹在此系統中加入 XML 標準做為訊息傳輸協定，然後再介紹資料管理方法如何與虛擬環境系統整合，最後說明整個系統的階層關係。

3.1. 以 XML 標準傳遞訊息的 VNet

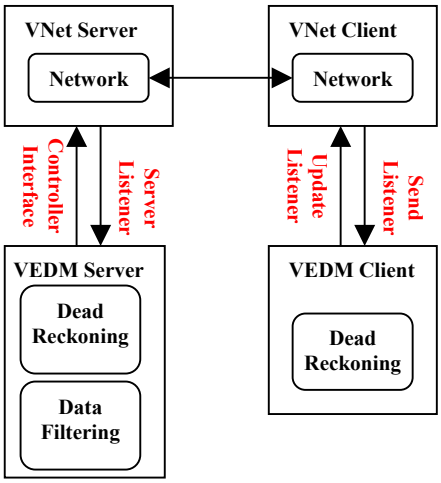


為了讓 VNet 系統的訊息傳輸模式能以

XML 標準作訊息傳遞，我們需要修改 VNet 系統本身訊息傳輸協定(VIP)[11]的設計，讓原本一個訊息種類對應到一種資料型態的訊息傳輸模式，變成能以 XML 標準傳輸標籤與屬性的結構[9]。在 Java 程式與 VRML 瀏覽器的溝通介面部分，我們也採用了另一種不同於 EAI (External Authoring Interface)的溝通介面。我們希望藉藉由結合 XML 與 VRML 的 X3D 而實作出的 VRML 瀏覽器，能讓 Java 程式對 VRML 的場景內容作完全控制。另外，為了讓我們的系統在未來能更有擴充性，我們使用 XML 文件做為系統的組態設定文件，藉由其標準格式提高系統未來在其他應用方面更高的相容度。基於上述幾項理由，我們修改了 VNet 系統(如圖二所示)。跟原本的 VNet 系統(圖一)比較，我們修改了訊息傳輸協定、使用者端與虛擬環境溝通介面以及加入 XML 文件做為 Server 端與使用者端的組態設定。

3.2. VEDM 與 VNet 的系統架構

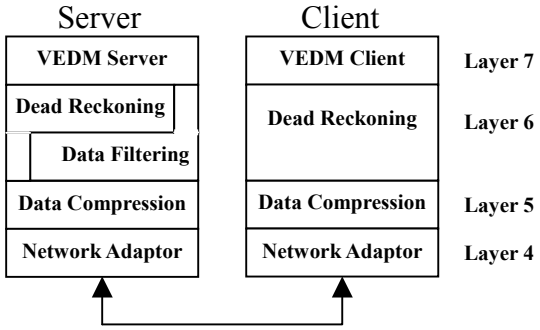
VEDM 對 VNet 而言，是一個新增的資料管理模組，負責告知 Server 端哪個 Client 該更新，以及在 Server 與 Client 端一起執行 Dead Reckoning 演算法。VEDM 和 VNet 之間溝通的方式如圖三所示。在 Server 是透過 Server Listener 和 Controller Interface 這兩個介面來達成，而使用者端則是 Send Listener 和 Update Listener。Server Listener 與 Send Listener 這兩個介面讓 VEDM 可以接收到 VNet 中使用者的動態，無論是使用者登入、登出，或是位置改變。Controller Interface 與 Update Listener 這兩個介面讓 VEDM 可以改變 VNet 中使用者的資料，以便把計算的結果套用至 VNet 中。



圖三. 系統架構圖

3.3. 系統階層關係

VEDM 設計的目的除了方便研究與測試之外，為了容易與 VNet 整合，我們採取階層式的設計，把實際網路傳輸的工作擺在底層(Layer 1~4)，然後在這之上才架構我們所設計出的管理方法。將來只須透過 VNet 來完成底層的工作，便可以把 VEDM 實作出來的成果套用至 VNet 之上。圖四便是 VEDM 的不同模組在 OSI 七層協定的位置圖。



圖四. 虛擬環境系統階層圖

根據此 OSI 七層協定來看，VEDM 屬於 Layer5~7 的部分。在 Server 端，最上層為應用程式解讀、利用的資料；在第六層部分可以選擇性地使用 Dead Reckoning 和 Data Filtering 這兩個模組；第五層則是提供一個壓縮的機制；第四層及以下則是負責把訊息傳送到 Client 端(Network Adaptor)。而當 Client 接收到一個訊息時，也是以類似的方式還原，最後以 VRML 的方式呈現在使用者面前。

4. 提升效率的方法

在這一節之中，我們將介紹在本篇論文之中我們所使用的資料管理方法，其中包含了 Data Filtering、Dead Reckoning、Data Synchronization 和 Update Scheduling。

4.1. Data Filtering

在原始的多人虛擬環境系統中，資料的傳輸是以廣播(Broadcasting)的方式來進行，亦即當使用者有資料更新到 Server 時，Server 就會把這個更新訊息傳送給目前系統中所有的使用者。很明顯地，我們知道這當中有許多不必要的傳輸，因為在虛擬環境的場景之中，使用者之間的距離可能很遠、障礙物的遮蔽或視野方向不同而看不到

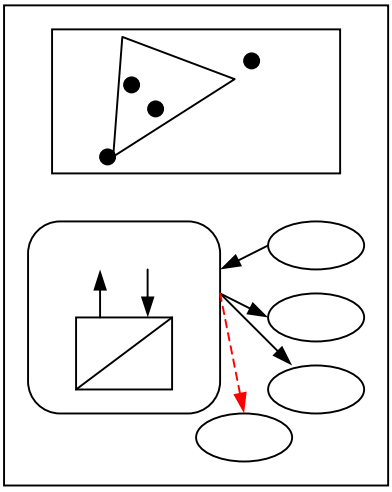
對方，因此有些使用者根本不需要接收某個使用者的最新動態，這時就可以利用 Data Filtering 來節省掉不必要的資料傳輸。

所謂的 Data Filtering，就是把資料作適當的過濾，但是要如何來過濾呢？根據多人虛擬環境系統的特性，一項更新的資料是否對某個使用者有用，跟這個使用者的視野有絕對的關係。所以我們依據此特性找出適合多人虛擬環境系統的過濾方法，而這些過濾的工作是在 Server 端完成的。Server 每次接收到使用者更新的訊息之後，會執行下列的動作：

(1) 利用此 Client (U)的 X ,Y ,θ(視野目前所在角度)，可見深度(D)以及可見廣度(W)，找出目前的視野範圍(R)，然後找出在此範圍內的 Clients，把這些 Clients 的資料 Update 給 U。

(2) 利用上述尋找視野的方法，檢查其他的 Clients，如果 U 在某個 Client 的視野之內，則把 U 的資料更新給這個 Client。

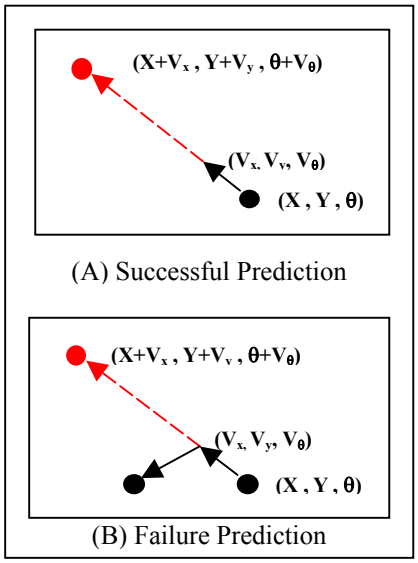
原本廣播的方法，每個 Client 要更新資料時就要傳給所有的使用者。當虛擬環境中有 n 個使用者時，server 便要傳送 n^2 筆資料；使用 Data Filtering 之後，當一個 Client 要更新資料時，只需要更新視野內的 Clients 即可；而視野內的使用者數目有限，只是一個常數倍，所以傳輸量從原本的 $O(n^2)$降低為 O(n)。



圖五便說明了如何利用 Data Filtering 來做到節省資料的傳輸：首先，User1 移動到最新的位置，向 Server 更新最新動態。Server 接收到 User1 的更新之後，便檢查目前虛擬環境內的所有使用者的所在位置，結果發現 User2 和 User3 在 User1 的視野之內，於是便把他們的資訊更新給 User1。如此一來，本來 User4 也要一起更新的，使用 Data Filtering 後就不需要傳送給 user1 了。由這個例子我們可以知道，當虛擬環境系統的使用者越多的時候，Data Filtering 所達到的效果會越好，因為一個使用者能看到的其他使用者數目有限，把他看不到的使用者過濾掉將能夠降低很多不必要的資料傳輸。

4.2. Dead Reckoning

Dead Reckoning 是一套廣為使用的演算法則。在虛擬環境方面的研究上，可被用在各種不同型態的資料上，例如使用者位置的預測、或虛擬人體的關節運動預測等。它的原理很簡單，但是在於節省資料傳輸方面，可以得到極好的效果。在虛擬環境的應用中，使用者的運動多具有一定的規律性，因此我們可以透過預測的方式來知道其它 Clients 的位置而不需隨時更新位置。



圖六. Dead Reckoning 流程

Dead Reckoning 的原理（如圖六所示），就是利用一個使用者的原始位置(X, Y, θ)，加上這個使用者前進的速度(V_x, V_y, V_θ)（或加速度），來預測他之後所在的位置。這個預測的動作是在每個使用者端都要做的，以此來計算沒有被更新資料的使用者位置。而在 Server 端也需要做預測，因為當一個使用者更新位置時，Server 必須要先算出目前場景內所有使用者的真正位置，才能正確的做 Data Filtering 的動作。使用 Dead Reckoning 這個演算法在節省傳輸方面具有絕對的好處，因為如果預測的路徑是正確的，便不需要更新被預設者的資料，只有在預測錯誤的時

候，才需要去更新最新的動態，因此即是在最壞的情況，也就是每次都預測錯誤，更新資料的次數也只跟沒使用 Dead Reckoning 時一樣。

但是，如何才能夠知道一個使用者的位置是正確的呢？我們在實作的時候，發現如果由 Server 來找出位置錯誤的使用者將較複雜，因為每個使用者算出其他使用者的位置可能不一樣，Server 必須隨時更新紀錄每一種使用者的狀態。因此，我們的作法是由每個使用者自己來負責錯誤的更新。因為預測會發生錯誤的情況，在於某個使用者的方向、速度發生改變，所以當一個使用者方向或速度改變時，便主動告知 Server，要 Server 通知其它使用者做更新的動作。

4.3. Data Synchronization

當使用 Data Filtering 和 Dead Reckoning 之後，資料傳輸明顯降低許多，但是出現了一些原始虛擬環境系統所沒有的問題：

(1) 每個使用者所計算的其他使用者資料可能不一致。
(2) 每個使用者所保有的其他使用者資料可能是錯誤的。

上面兩個問題會產生，主要是因為使用 Dead Reckoning，每個使用者各自在計算其它使用者的位置；再加上使用 Data Filtering，使用者更新的訊息不一定會傳送給所有的使用者，於是有些沒收到更新訊息的使用者就會用舊的資料預測其它使用者的位置，以致會產生資料的不一致性而發生錯誤。這兩個問題可能會造成使用者間所見到的場景不一致，以及看到不該出現的使用者。

Data Filtering 與 Dead Reckoning 減少了資料的傳輸，卻犧牲了資料的準確性。為了要改善這個問題，我們必須採取一個折衷的解決方法。如果在使用 Data Filtering 之時，能夠定時的作全面性的廣播，便可以解決上述的問題。但是在某段時間之內，使用者還是有可能會看到錯誤的場景。這似乎是無可必避免的，因為我們必須在資料傳輸量與場景正確度之間做一個取捨，才能讓虛擬環境系統有更好的效率。經過我們的實驗發現，如果在固定周期內使用一次廣播的方式做訊息的更新，整個虛擬環境系統便可以保有很高的正確率。

4.4. Update Scheduling

我們之所以要在虛擬環境系統之中加入 Data Filtering 和 Dead Reckoning，無非是希望降低資料的傳輸量，以減輕網路方面的負擔。另一方面，我們發現在真實網路傳輸的時候，網路延遲所帶來的問題。因為 VNet 中每個使用者和 Server 均是以 Socket 方式連結，所以每個 Socket 的流量(Throughput)並不一樣，有些較流暢，有些就可能會阻塞。但是在原始的虛擬環境系統中，當 Server 在更新訊息的時候，並不會把這些因素考慮進去，每個使用者被更新的機會是均等的，如此一來就會出現網路流量較快的使用者必須要等待流量較慢的使用者，直到資料傳送完為止，於是我們設計了一個更新排程(Update Scheduling)的方法。

為了增加獨立性和擴充性，我們在實作 VEDM 的時候，把網路傳輸部分設計成一個獨立的模組，以循環的方式輪流更新虛擬環境系統中所有的使用者，並且把每個使用者網路流量的差異加入考量之中。我們設定一個參數來控制在一個循環之中，哪些使用者有更新資料的權力，哪些沒有。首先我們先測量出 Server 和每個使用者間網路的線路狀態，測量的方式可以用 Server 送出一個訊息封包(message)，然後計算使用者回應所花的時間；而這個數據可以動態的去更新，以便適應網路流量的變化。有了線路狀態的數據後，便可以決定每個使用者要過多久才有權進行資料的更新。如此一來，網路狀態良好的使用者便可以每個循環都有權可以更新資料，狀態較差的使用者可能就要幾個循環才能夠更新一次了。

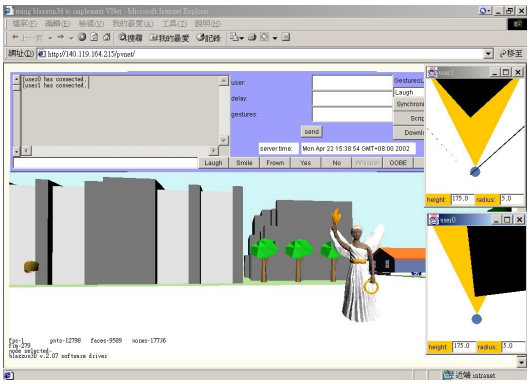
使用這種方式之後，降低了使用者整體的等候時間，而且網路狀態好的使用者有較高的服務品質；相對地，網路狀態差的使用者更新次數便減少了。不過由於問題是在網路流量，所以即使不斷的更新，資料還是卡在網路上。所以使用者如果想要取得好的服務品質，還是要先改善網路的品質。

5. 虛擬使用者

實驗的過程需要多位使用者共同參與，以達到多人的效果；所以為了降低實驗所需花費的人力資源，我們參考了 ActiveWorlds [1]的虛擬使用者(bot)設計方式，為 VNet 系統設計了一套關於虛擬使用者的軟體開發工具(SDK)。利用此 SDK 所創造的虛擬使用者在功能上將與真實使用者相近，再透過程式操控其行為(更新位置、方向)，便可達到模擬多位使用者在虛擬環境中的行走

的情況。

首先，在虛擬使用者的設計上，要採用與真實使用者相同的連線方式，讓伺服端無法分辨登入的使用者是真實或虛擬的。因此，我們使用 socket 與伺服端建立連線，並使用與真實使用者相同的訊息傳輸模式，傳輸以 XML 標準包裝的訊息。如此一來，就可讓虛擬使用者在資料傳輸方面與真實使用者相同。接下來，只要待虛擬使用者的連線成功後，便可利用程式不斷地更新虛擬使用者狀態的程式，讓真實使用者在虛擬場景中見到宛如有真人在背後操控的虛擬人物。



如圖七所示，我們簡化了虛擬使用者的介面，僅利用俯視的 2D 圖像(右邊視窗)，在管理者介面上個別顯示每位虛擬使用者所能看見的範圍與其他使用者的狀態(例如，位置、方向)。另外，為了瞭解資料管理的正確性，在每位虛擬使用者的介面中使用一些文字訊息，來顯示資料管理時的資訊。

虛擬使用者的資料管理與真實使用者相同，都各有一個管理訊息傳送與接收的介面。因為原本的 VNet 系統是以廣播的方式將訊息傳送至每個使用者端，所以若是在人數眾多的情況下，用程式一直更新並傳遞每位虛擬使用者的狀態訊息至伺服端，將會使伺服端因處理訊息的派送，降低了執行的效率。因此，在虛擬使用者的資料管理介面上，我們利用 XML 的延展性，加入了一個資料彙整傳送的功能，減少真正傳遞至伺服端的訊息量，提高伺服端的執行效率。詳細的實驗數據將在第 6.3 節敘述。

在實驗方面使用虛擬使用者的優點，除了可節省人力資源外，還可提供將來 VNet 系統更多元的發展。例如，在虛擬場景中的場景管理員、虛擬商店的售貨員、模擬熱鬧的商店街人群等，都是可用虛擬使用者取代真人在背後操控的虛擬人物。而讓 VNet 系統在功能面上更加完備。

6. 實作與實驗

6.1. 建立 2D 模擬平台

包含 VEDM 在內的整個虛擬環境系統都是以 Java 語言實作。由於多人虛擬環境系統是一個 Client-Server 架構的系統，在進行資料管理實驗的過程中，不管在時效或偵錯上都較複雜耗時。因此為了方便實驗起見，我們設計了一個 2D 模擬平台，利用 thread 的方式，把整個虛擬環境系統在單機上面模擬。此系統保有主從式(Client-Server)的架構，但原本虛擬環境系統的 3D 場景，則均以 2D 平面圖來取代。為了方便知道所有使用者的動態，Applet 除了畫出場景之外，也會畫出使用者的所在位置。如圖八所示，透過左邊的視窗便可清楚的透過第三人稱上視圖看到所有使用者的運動。

這些使用者的運動，是由電腦去隨機模擬的，經過適度的調整，讓它們的運動模式最接近人類運動的狀態。如果我們想要控制其中某一個使用者，只需要在欲控制使用者所屬的視窗上用滑鼠前後左右拖曳，便可做移動與選轉的動作，另外還有一些參數的設定像是身高、體重，都可以自由設定，讓這些虛擬的使用者有不同的狀態、形體，更具真實性。左邊的 Applet 視窗即是整個虛擬環境的狀況，黑色部份代表障礙物，而藍點則是代表一個使用者，紅色的三角形部份代表的是這個使用者的視野。

6.2. 和實際虛擬環境系統的整合

我們以 VNet 這套系統作為基礎，並將 VEDM 所研究出來的成果整合到 VNet 之中。VNet 是利用 Java 語言和 VRML[13]實作出的虛

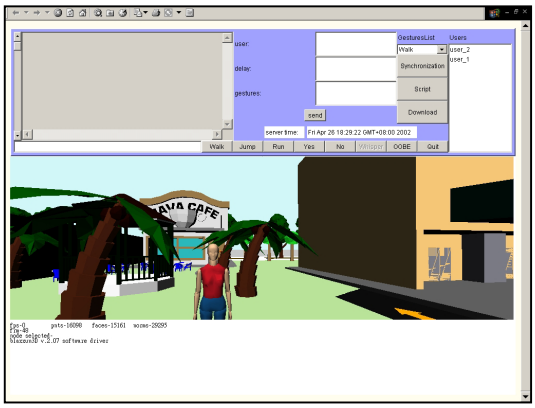
擬環境系統。由於它是以物件導向概念設計出來的系統，所以具有較佳的擴充性。透過模組化的設計，在系統之中加入新功能並不複雜。在使用者端則是使用 Blaxxun 3D 瀏覽器來呈現，另外附加一個 Java Applet 的程式來控制，使用者只需安裝好 VRML 瀏覽器，開啟普通的網頁瀏覽器如 IE、Netscape，便可連上虛擬環境系統。

我們利用 Event Driven 的方式把 Vnet 和 VEDM 整合起來：當有使用者登入、登出或更新資料時，Vnet 便會透過 Server Listener 這個介面通知 VEDM Server。VEDM Server 經過適當的處理之後，利用 Controller Interface 呼叫 Vnet，更新某些使用者的資料。在這個更新的動作中，VEDM 是以一個獨立的 Thread 來處理，並且支援一次同時更新許多使用者的資料，來降低傳送時的多餘資料(Overhead)。

上述的整合方式，不光是適用於 Data Filtering 這個功能，Dead Reckoning、Data Synchronization 與 Update Scheduling 都是以此種模式跟 VNet 結合，只要在介面中提供適當的函式，便可以輕鬆的把模組新增到系統之中。

6.3. 實驗

我們以實驗的方式比較本系統與原本的虛擬環境系統間的差異。我們實驗的環境為一般個人電腦、PentiumⅢ-700 的處理器、256Mb 記憶體、Windows 2000 作業系統、JDK1.3、Microsoft Internet Explorer、Blaxxun3D VRML Browser。



圖九. Vnet 使用者介面

VNet 的使用者操作介面如圖九所示。下方就是目前使用者所看到的虛擬環境狀況，包括了場景和其他的使用者(圖中的紅衣女子"Nancy"便是另一個使用者)，而上方則是提供使用者列表和對話的功能。使用者透過滑鼠的拖曳或是鍵盤的上下左右移動，便可以在場景內走動。

另外我們測量在單位時間內，Server 對所有使用者更新的總次數。我們分別測試了 5,10,15,20 個虛擬使用者同時在虛擬環境內活動。透過虛擬使用者的控制視窗，我們可以改變任何一個使用者的位置；同時，其他沒被操控到使虛擬使用者，也會隨機地做前進和旋轉的動作。我們比較以下四種狀況：

(1)沒使用 Data Filtering 與 Dead Reckoning。
(2)使用 Data Filtering，沒用 Dead Reckoning。
(3)使用 Dead Reckoning，沒用 Data Filtering。
(4)同時用 Data Filtering 與 Dead Reckoning。

其數據如圖十：

Client Number	No DF No DR	DF No DR	No DF DR	DF DR
5	372	112	102	23
10	1056	178	154	38
15	2625	255	496	52
20	3742	309	714	73

圖十：傳輸次數(Transmission / Sec)

實驗的結果顯示，使用 Data Filtering 與 Dead Reckoning 之後，效果會比都沒使用時好了 15~50 倍之多，而且使用者越多時，效果會越顯著。當我們觀察第 1 種狀況和第 4 種狀況的時候，發現傳輸的數量有平方的關係，證實了第 4.1 節所提到傳輸量從 $O(n^2)$降為 O(n)的關係。

由於使用 Data Filtering 與 Dead Reckoning 會產生資料不一致，所以我們特別分析四種狀況下的錯誤率，而錯誤率是由每秒發生錯誤次數除以每秒畫面更新總次數所計算出來的。

Client Number	No DF No DR	DF No DR	No DF DR	DF DR
5	0.02	5.87	4.66	33.75
10	0.06	4.84	6.78	46.31
15	0.10	4.54	4.36	36.13
20	0.22	7.28	3.52	39.41

圖十一：錯誤率(%)(錯誤次數/畫面更新次數)

如圖十一所示，以原本的傳輸方式傳送資料的話，不管有多少使用者都不會有錯誤發生，但是卻有龐大的傳輸量。使用 Data Filtering 與 Dead Reckoning 之後，必定會產生資料的不一致性，因而導致視覺上的錯誤發生。不過經過我們使用 Data Synchronization 這個補救的措施之後，即使同時使用 Data Filtering 與 Dead Reckoning，錯誤

率維持在一定的比率，但卻明顯的降低了傳輸的次數，在資料傳輸量與資料正確性之間的取捨，就要看虛擬環境系統的需求去衡量了。

7. 結論與未來延展

經過實驗測試之後，我們知道在使用 Data Filtering 與 Dead Reckoning 之後，的確能夠有效的降低資料的傳輸量，不過比較值得注意的是由於使用上面兩種方法，使得系統的錯誤率升高了，這是需要改進的。另外 Data Synchronization 能夠定時確保資料的正確性，而 Update Scheduling 則降低了 Server 端更新資料時的平均等候時間。整體而言，這些方法解決了 Server 端在網路方面的頻頸，進而提昇了系統的擴展性；所以對虛擬環境系統在提昇效率方面有正面的幫助。

根據上述的研究結果，我們認為在未來還有幾個方向可以繼續研究：

1. 人體動畫部分資料的最佳化：像是使用 Level of Detail 或是採取階層式的傳輸等。
2. 資料的壓縮：不過這裡的壓縮我們強調的是應用層或邏輯上面的壓縮，根據虛擬環境或者是 Animation 的特質來做一些邏輯上的分析，進而達到壓縮的效果。一般的壓縮方法雖然有一定的效果，但是並非研究的重點。
3. 降低錯誤率：使用資料管理方法後，資料所產生的不一致性而導致錯誤，可以在使用者介面更新畫面時，設計一套偵錯機制以降低錯誤率。
4. 虛擬人群的研究：虛擬環境系統已經具備虛擬使用者的功能，接下來便可進行人群的研究。透過模擬人群的移動，可以使虛擬環境系統內增加真實性，也讓使用者感到更多的互動。

參考文獻

[1] ActiveWorlds URL:<http://www.activeworlds.com>.

[2] J. Barraquand, L. Kavraki, J.C. Latombe, “Robot Motion Planning: A Distributed Representation Approach,” *Intl J. of Robotics Research*, 10:628-649,1991.

[3] Blaxxun, URL:<http://www.blaxxun.com>

[4] EAIspec, URL:<http://www.web3d.org/WorkingGroups/vrml-eai/>.

[5] T. Funkhouser, C. Sequin, and S. Teller, “Management of Large Amounts of Data in Interactive Building Walkthroughs,” *Proc. of 1992 Workshop on Interactive 3D Graphics*, pp. 11—20, 1992.

[6] R. Gossweiler, R. J. Laferriere, M. L. Keller, and R. Pausch, “An Introductory Tutorial for Developing Multi-User Virtual Environments,” Technical report, Computer Science Department, University of Virginia, 1994.

[7] T. Hijiri, K. Nishitani, T Cornish, T. Naka, S. Asahara, “A spatial hierarchical compression method for 3D streaming animation,” *Proc. of the Web3D-VRML 2000 Fifth Symposium on Virtual Reality Modeling Language*, February 2000.

[8] Java, URL:<http://java.sun.com>.

[9] T.Y. Li and Y.L. Liu, “An Open Multi-User Virtual Environment System with XML-based Message Delivery,” *Proceedings of the 2001 National Computer Symposium*, Taipei, Taiwan, 2001.

[10] Oldfield K.Y. So, “Issues in a Very Large Scale Distributed Virtual Environment,” Room 1026 Technical Report RM1026-TR97-1205, http://blackhole.cse.cuhk.edu.hk/~paper/report/fall97.ps.

[11] The VRML Interchange Protocol (VIP), URL:<http://ariadne.iz.net/~jeffs/vnet/VRML_Interchange_Protocol.html>.

[12] VNet, URL:<http://ariadne.iz.net/~jeffs/vnet/>.

[13] VRML97spec, URL:<http://www.vrml.org>.

[14] XML, URL:<http://www.xml.org>.