行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※※

※ **以 PC 群組建構環場視校空間之同步控制** ※

※ **演算法研究** ※

※ Study of the synchronization mechanism for ※

※ the CAVE system on a PC Cluster ※

※※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：☑個別型計畫 □整合型計畫

計畫編號：NSC 90-2213-E-032-015

執行期間：90 **年** 8 **月** 1 **日至** 91 **年** 7 **月** 31 **日**

計畫主持人：黃俊堯

計畫參與人員：白華勝，杜怡璋，須冠文

本成果報告包括以下應繳交之附件：

- **□赴國外出差或研習心得報告一份**
- **□赴大陸地區出差或研習心得報告一份**
- **□出席國際學術會議心得報告及發表之論文各一份**
- **□國際合作研究計畫國外研究報告書一份**

執行單位：淡江大學資工系

中 華 民 國 91 **年** 7 **月** 30 **日**

以 PC 群組建構環場視校空間之同步控制演算法研究

Study of the synchronization mechanism for the CAVE system on a PC Cluster

計畫編號：NSC 90-2213-E-032-015-
執行期限：90 年 8 月 1 日至 91 年 7 月 31 日

主持人：黃俊堯　淡江大學資工系
計畫參與人員：白華勝，杜怡璋，須冠文　淡江大學資工系

一、中文摘要

本計畫的目的是要利用多台個人電腦所構成的個人電腦群組環境(PC Cluster)**來建構一個高解析度但是低成本的環場視效空間系統**(CAVE system)**，而每台個人電腦之間是採用一般的乙太網路**(Ethernet)**來溝通。本計畫將以個人電腦群組來取代昂貴的工作站級電腦，藉由分散式運算的技術來達成環場視效空間的多螢幕電腦顯像、高解析度畫面、**3D **場景運算以及即時運算等特性。**

本計畫的貢獻在於提出環場螢幕同步技術，而此技術的優點包括：(1)**低成本**--**可使用多台個人電腦所組成的電腦群組環境來建構環場視效空間；。**(2)**良好的擬真效果，藉由分散式平行計算技術來達到高速複雜計算的需求。**(3)**具模組化與規格化的系統架構，可以在現有** VR **系統上擴充至多部電腦的螢幕同步。**

關鍵詞：環場視效空間, **個人電腦** CAVE **系統**, **平行運算**, **虛擬實境**, **虛擬環境**.

Abstract

This project is to study a method to construct a high-resolution yet low-cost CAVE system on a cluster of PCs that are interconnected by Ethernet. Since the goal of a CAVE system is to display a high-resolution image on multiple screens, its computation includes synchronizing multiple screens, rendering a high-resolution image, and managing the 3D virtual world. Since all of these tasks must be computed in real-time, they are beyond the computing capability of a single PC. Hence, the solution is to distribute these tasks among a cluster of PCs to perform distributed computing.

The contribution of this project is to presents a synchronization mechanism for the PC CAVE system that has following benefits: (1) low-cost - it can construct a CAVE system using a cluster of PCs; (2) a high-fidelity simulation effect - it can achieve real-time computing by distributed computing; (3) Modular and standardized system architecture - it can expand to multiple display synchronization on an existing VR system.

Keywords: CAVE (CAVE Automatic Virtual Environment), PC-CAVE, Parallel Computing, Virtual Really, Virtual Environment.

二、緣由與目的

環場視效空間(CAVE)**第一次出現是在** SIGGRAPH' 92 **的展示場上，是由芝加哥的伊利諾州立大學** EVL **實驗室**(Electronic Visualization Laboratory)**所設計**[1][2]。**為了提供快速的模擬效果以及同步的功能，該系統採用高等級的工作站級電腦及特殊電腦硬體設備來建構其** CAVE **系統。在成像系統上，該系統採用了四台** Silicon Graphics Inc **公司的工作站級電腦來作為顯示處理硬體，每台電腦皆對應一台投影器材來顯示**

成像結果，其中三台電腦利用背投影式投影機來顯示左方、前方及右方的畫面，另一台電腦則負責顯示地板的畫面。

在設計上，環場視效空間的建構有三個問題須解決，分別為(1).以使用者為中心的觀測投影法(Viewer-centered perspective projection)、(2).投影機及 Tracker 限制所產生的問題(Overcoming any resulting projector and tracking limitations) 以及(3).多螢幕的同步(Synchronization of displays)。

其中，第 1 點是由於環場視效空間允許使用者在此空間中自由移動，如此使用者的觀測位置將不會在螢幕的正前方，如圖 1所示。因此系統必須要採用 off-axis 的透視投影法，才能產生正確的畫面。

第 2 點是由於 Tracker 的延遲問題，因此在追蹤使用者的位置時會造成誤差，進而產生顯像上的錯誤。另外，在投影時須避免產生陰影，例如使用者的影子，因為地板的投影是使用前投影機由上往下投影的方式，因此在設計上必須小心影子不能夠遮蔽到虛擬場景的畫面。以及投影螢幕之間，投影畫面不能夠重疊或有間隙，否則會造成畫面的不連貫。

第 3 點的多螢幕的同步問題，通常是使用一台高階工作站級電腦負責處理與輸入設備之間的通訊，以及作為整個系統的控制主機，來負責協調四台工作站級的繪圖電腦，電腦之間是採用 reflective memory 來達到資料傳遞及畫面同步更新的工作。

但是以 EVL 實驗室所設計的環場視效空間而言，其價格太過昂貴。隨著電腦科技的進步，今日的個人電腦所擁有的計算速度已接近、甚至超過昔日工作站等級的電腦。為了提供低成本的環場視效空間，本計畫將以個人電腦群組(PC Cluster)來取代昂貴的工作站級電腦。由於環場視效空間的研究重點是多螢幕顯像、高解析度畫面、3D 場景運算的即時運算特性，因此如果能夠善加利用分散式平行運算技術(Parallelism)來將整個系統的工作分散至數台電腦，藉由平衡系統負擔以達到環場視效空間的需求，使得能夠利用個人電腦群組來達到工作站級電腦的運算效率。

當工作分散至多台電腦上時，要如何協調這些電腦的工作，也就是所謂的同步問題(Synchronization)，此將會是一個很嚴重的問題。由於在分散式的環境下時，每台電腦都是個別獨立運作，如果沒有一個良好的同步機制去協調這些獨立的電腦。將會造成系統的效率降低(throughput)以及系統的反應時間過長(respond time)，將會讓使用者無法忍受系統的反應。而在多螢幕顯像上，則會產生更嚴重的問題，當螢幕不一致時，將會造成使用者視覺上的混亂。

因此本計畫的目的是要設計一個多螢幕電腦顯像同步機制以應用於由個人電腦群組所組成之環場視效空間，而本計畫的研究方向則是研究如何分散工作以及螢幕之間的同步問題。

三、互動式模擬系統

如圖 2所示，在一個互動式視覺化模擬(Interactive Visual Simulation)系統中，包含了 4 個主要的子系統：使用者介面(user interface)、應用程式的相關處理(Application-Dependent Code)、3D 場景管理(包含 3D 場景資料庫)及繪圖引擎(Render Engine)。其中使用者介面子系統會傳輸使用者的動作資訊給應用程式子系統，應用程式子系統會將處理結果傳輸給 3D 場景管理，3D 場景管理子系統則會將場景資訊存入 3D 場景資料庫中，而繪圖引擎子系統將會透過存取 3D 場景資料庫來得知所要繪製的資訊，最後繪圖引擎會將繪製完成的畫面傳輸給顯示設備，並顯示在使用者的面前。

而在互動式視覺化模擬系統中，一個最主要的課題就是系統能否與使用者即時地產生互動(interactive)，也就是說，當使用者透過輸入裝置產生一個動作訊息，則整個模擬系統必須要很快地將此動作反應出來，讓使用者能夠依據這個訊息決定下一個動作，此即 Human-in-the-loop 的精神，否則系統將會發生一些嚴重的錯誤，例如畫面的跳動(jitter)，會讓使用者無法忍受系統的反應。而根據 Taylor 等人[4][5]的定義，從使用者輸入一個動作到模擬系統將此動作的反應顯示在螢幕上，稱為 end-to-end lag time。所謂的 Lag 是指一個處理流程從資料輸入到工作完成所需要花費的時間，因此 lag 是本計畫中決定工作分散的一個很重要的依據。

根據 Wloka [6]及 Taylor[4] [7]所提出出的 Lag model，互動式模擬系統中共通的各種 lags：User Input Device lag、Rendering lag、Application-Dependent Processing lag 、synchronization lag、frame-rate-induced lag 及 Network lag 等。

並且根據 Taylor 等人的分析測試指出，在互動式視覺化模擬系統中，影響系統最大的 lags 為 Application-Dependent Processing Lag、Rendering lag 及 Network lag，而 Synchronization lag在整個系統的總 lag 時間佔了最多，約為 50%。

而 Wloka 及 Taylor 都同時提出了平行化 (parallelism) 的方法來解決 Application-Dependent Processing Lag 及 Rendering lag。透過平行化可以將整個互動模擬系統(interactive simulation loop)分散至多台電腦上，使得各個處理階段分散至不同處理器來提高系統的效率。

四、互動式模擬系統的平行化

在各種圖學系統，如 scientific visualization、CAD、vehicle simulation 及 virtual reality 上，都需要龐大的浮點運算(floating point)及記憶體的頻寬，這些已經遠超過單一處理器的能力。因此建立一個高效率的繪圖系統時，平行化(parallelism)變成一個很重要的方法。平行化分為二種型態：functional parallelism(又稱為 pipelining)及 data parallelism。

管線化(pipelining)是將一個工作依功能分割至多個處理器，每個處理器負責一部分的工作，每一個處理器會等待前一個功能處理完其資料才開始運作，然後將自己處理完的資料交給下一個，類似工廠的生產線。並且這些處理器可以同時運作，只要將資料依序輸入，處理時間就可以重疊(overlapping)，而達到加速運算的效率。

在分散式的環境下，整個互動模擬系統分割成數個 LPs(Logic Processors)，並將 LPs 分別置於不同的電腦上執行，這些 LPs 之間就會有著溝通上之先後順序(precedence)問題(也就是 LPs 之 schedule 問題)。如果採用不好的排班演算法(schedule algorithm)可能會導致過多的 synchronization lag，嚴重地影響到系統的即時反應，如顯像速率下降而造成畫面跳動或是系統不能迅速地對使用者的輸入動作產生反應，令使用者感到不自然，真實性也就會跟著下降。

而 data parallelism 則是將資料分散至多個功能相同的處理器，來增加資料的產生量，最常見是應用在 Parallel rendering 技術。Parallel rendering 是將繪圖資料分散至多個功能相同的處理器，來增加圖形資料的處理量。Parallel rendering 根據所處理資料型態的不同，可以分為 object parallelism 及 pixel(image) parallelism。parallel rendering 的方法是根據圖學資料從 object space 轉換到 screen space 之間何時去做 sorting，可以分為三種動作方式：sort-first、sort-middle。以及 sort-last。

在繪圖工作中，最主要的目的是計算出每個 primitive 對於螢幕上每個圖素的影響(或貢獻)。由於每個 primitive 會經過 modeling 及 viewing transformations 的轉換，最後可能會落在螢幕上任何位置，因此必須將這些 primitive 經過排序(sorting)才能知道在螢幕上的位置。而在 fully parallel rendering 上，因為 primitive 及 pixel 的處理工作已經被分散至多個處理器上，所以 sorting 的工作還必須包含將這些 primitive 重新分散 (redistribution) 至適當 processor 上的步驟。根據何時計算螢幕位置及重新分散 (sort)，parallel rendering 可以進一步分為三種方法：在執行 geometry processing 之前 (sort-first) ；在 geometry processing 及 rasterization 之間 (sort-middle) ；或在 rasterization 之後(sort-last)。sort-first 是將未處理的 primitive 重新分配至適當的 processor，也就是還不知道螢幕位置之前。sort-middle 則是將已經轉換至 screen space 的 primitive 重新分配至適當的 processor。而 sort-last 則是將已經計算出來的圖素重新分配至適當的 processor。

表 1為此三種 sort 方法的比較，根據這個比較，將作為本計畫選擇採用哪種方法的依據。由於本計畫的硬體環境希望能夠利用現有的 PC 設備，以減低建構成本。而在 PC 環境下，繪圖晶片及繪圖系統都將 geometry processor 及 rasterization processor 設計成一個連續的處理步驟，無法分割。

並且網路環境是採用一般 100Mb 的乙太網路，因為 sort-last 必須要利用大量的頻寬，因此本計畫無法採用 sort-last 技術，而改採用 sort-first 來設計。

五、PC-CAVE 系統設計

本計畫的方法首先是將整個 simulation loop 的處理階段分為 3 個模組，分別為 Application-dependent Code 及 I/O 處理模組、Server site 模組及 Client Display Host 模組，每個模組交由不同的電腦負責，如圖 4所示。

第一個模組為〝Application-dependent Code 及 I/O 處理〞，Application-dependent Code 是根據所要模擬的系統而有不同的處理程序，例如飛行模擬必須要有航空動力學運算。各種的模擬系統通常都需要各種運算量非常龐大的處理工作，因此可能需要額外的電腦來特別處理此模組的工作，所以在本計畫中將其交由一台電腦來負責。

接下來將剩下的工作，依據 soft-first 的方法分成兩個模組，也就是從 Scene Management 之後的工作，這兩個模組將分為 Server/Client 的架構，主要由 Server 端處理場景的維護及更新工作，而將繪圖(Rendering)的工作分散至多台的 Client 端 - Display Host，每台 Display Host 各自負責不同的畫面，並且這些畫面互相不重疊，最後這些畫面會形成一個連貫式畫面，可以用來建構出環場視效空間的環境。

在這裡要討論的問題是要如何將場景資料傳遞給每個 Display Host，讓 Display Host 能夠繪出各自所負責的畫面。本計畫是利用 Culling 的技術，因為每個 Display Host 都有各自所需負責的畫面，也就是各自的可視範圍(View Frustum)，並且每個 Display Host 的畫面是互相不重疊，因此每個 Display Host 都必須要去計算各自的 View-Dependent Culling。如果將這個工作完全交由每個 Display Host 自己去計算，則每個 Display Host 都必須要將整個場景資料儲存在各自的記憶體中，這樣會使得每個 Display Host 為了繪製出各自所負責的畫面，而浪費了許多系統資源。並且 Display Host 為了維護與 Server 端場景資料的一致性，也必須要使用龐大的網路頻寬。

因此本計畫的方法是將 View-Dependent Culling 的工作由 Server 端負責，由 Server 端來替每個 Display Host 處理各自的 View-Dependent Culling 工作。本計畫將 Rendering Pipeline 分為兩個部分，如圖 5 所示將 Server 端及 Display Host 的工作分界設定在 Trivial accept/reject 工作之後，Trivial accept/reject 工作也就是 View-Dependent Culling 工作。

在前面的圖 4 所示，Server 端會替每個 Display Host 執行 View-Dependent Culling 工作，列出在 Display Host 所負責畫面中的物件，這個物件串列稱為 VF(View Frustum) Object List。而每台 Display Host 只要將 VF Object List 內的物件繪出即可產生出所要的畫面。這種方法使得場景一致性的問題可以有效的減少為只要維護 Server 端及 Display Host 之間的 VF Object List 一致即可，並且可以減少維護一致性所需要的網路頻寬，而且只需要 Server 端維護整個場景的最新狀態。

但是當繪圖工作分散至多台 Display Host 時，由於每個 Display Host 所要繪製的畫面複雜度不同，將會造成互相等待的情形。因此本計畫所使用之 Client/Server 架構將會增加額外的等待時間，因為這段等待的時間是由於處理不同的 Views 而需要耗費不同的時間所造成的，所以將這段時間稱之為 View-dependent lag。

View-dependent lag 是由於採用了 Asynchronous Synchronization 機制所產生的現象。Asynchronous Synchronization 機制是讓各處理階段完全獨立作業，各個處理階段都不需要等待其他的處理階段，以各自的最快速度執行所負責的工作，因此可以使得系統各處理階段的 throughput 達到最大。

在 Asynchronous Synchronization 演算法中(如圖 6 所示)，Server 端更新完第 i 個畫面(Frame)的場景，並且將每個 Display Host 的 pre-rendering 結果透過網路傳送給各個 Display Host 之後，緊接著 Server 端又可繼續執行第 i+1 個畫面的工作。而 Display Host 也不需要回應給 Server 端是否已經完成目前的畫面，並且 Display Host 只要一收到

Server 端所傳送過來的 VF Object-list，就可以立刻繪製新的畫面。在 Asynchronous Synchronization 演算法下，不管是 Server 端或 Display Host 都不需要互相等待其他電腦是否已經完成。

在 Asynchronous Synchronization 演算法下，若是 Server 端所需的執行時間比所有 Client 端短，使得每個 Client 端在每個 Frame 開始執行時都可以立刻得到這個 Frame 所需要的資料，因此每個 Client 端將會以各自最快的速度執行，而產生互相競爭的情形，如圖 7 所示。因為每個 Client 端對於每個 Frame 所負責畫面複雜度的不同(例如：物件數量、triangle 數量及貼圖數量)，而所需要的執行時間也會不一樣，造成在每個 Frame 中各自所負責執行的工作完成時間也不一致，使得處理下一個 Frame 的開始時間不同，也就是每個 Client 端所需要的資料從 Server 端送出到 Client 端收到並開始處理資料的時間不同，這段時間即是所謂的 Synchronization Lag。

因此，本計畫定義一個額外的 Lag：

View-Dependent Lag：此 Lag 是因為每個 Client 端所負責的畫面複雜度不同，使得在下一個畫面時，每個 Client 端的開始處理時間不一致，這段時間差稱之為 View-Dependent Lag。

由於這個 Lag 是因為每個 Client 端所負責畫面的複雜度不同所造成的，也就是跟所負責的畫面相關，因此將其稱為 View dependent Lag。並且也可以說是因為 Synchronization Lag 不同，而造成每個 Client 端開始處理畫面的時間差，所以此 View dependent Lag 也可以視為是 Synchronization Lag 額外延伸出來的 Lag。

在圖 7 中，可以發現 View-Dependent Lag 是無法預測的，因為每個 Client 端所需要的執行時間不同，而這個時間必須根據畫面複雜度，但是 Client 端所處理的每個畫面複雜度並不相同，使得 Client 端之間會產生互相競爭的現象。並且可以發現，在 i+1 的畫面時 Client 1 比 Client 2 早處理 i+1 的畫面，可是在 i+1 的畫面時卻是 Client 2 比較早處理 i+2 的畫面，這種不確定性會使得無法預測出每個畫面的 View-Dependent Lag。如果在環場視效空間下發生這種情況，將會造成每台 Client 端所顯示的畫面時快時慢，使得整個環場視效空間的畫面不連貫，讓使用者產生視覺上的混亂。

本計畫的同步機制將會利用 Waiting Rule 技術[8]，透過 Waiting Rule 可以有效地減少 View-Dependent Lag，並且達到 Client 端之間的工作同步。

要減少 View-Dependent Lag 有兩個問題要解決：首先是 View-Dependent Lag 的產生是因為每個 Display Host 所負責的畫面複雜度不一樣，使得所需的處理時間不同，造成每個 Display Host 開始處理同一個回合的時間會不一致，要解決這個問題就必須要使得每個 Display Host 要一起開始處理，不能任意執行繪圖的工作，要等待 server 端指示才能開始。

其次是要如何確定 Server 端送出通知訊號時，Display Host 已經處理完前一個的繪圖工作。在這裡必須要保證 Server 端送出前進訊號，所有 Display Host 已經完成所有的工作，並且正在等待 Server 端的訊號。所以必須要利用護航效應(Convey effect)的技術，護航效應通常在一些 simulation 系統上會盡量避免，但是在這裡卻是非常有用的一項技術，因為本計畫所要設計的環場視效空間系統必須要達到每個畫面顯示同步，畫面之間互相等待是一個必須的條件。因此本計畫必須要利用護航效應來達到環場視效空間系統的畫面同步。

並且 Server 端與 Display Host 之間也必須要同步，Server 端不能一直不停的產生新資料，如果 Server 端所需的處理時間比 Display Host 要少，會造成資料過多而使得系統反應遲鈍(當 Server 端與 Display Host 之間的 Buffer 大於 1 時)，或者畫面跳躍(當 Server 端與 Display Host 之間的 Buffer 為 1 時)。

因此本計畫採用 Waiting Rule 技術的兩個限制條件來達到上述的要求：

1. An LP waits to receive messages on all input lines whose clock values equal to the LP clock value.
2. An LP waits on all output lines on which there is a message to be sent.

在 Waiting Rule 的定義下，每個 LP 本身會有一個 clock，這個 clock 代表的是 LP 自己的邏輯時間，在處理完一筆資料以

後，邏輯時間就會跟著前進一個單位。而每筆資料送出時，也會伴隨一個 clock 值來代表資料送出的時間。因此第一個條件是指一個 LP 必須要等收到所有的輸入資料，並且所送來的資料 clock 值必須要大於 LP 自己的 clock 值，才能進入下一個回合。而第二個條件是指一個 LP 必須要等到它所送出的資料都到達接收端，才能進入下一個回合。也就是說，LP_{i+1} 必須要等到 LP_i 的輸出到達之後，才能開始執行下一個畫面的運算；並且當 LP_i 輸出的計算結果傳給 LP_{i+1} 後，必須等到 LP_{i+1} 接收到且準備處理時，LP_i 才能繼續執行下一個畫面的運算。

首先利用 Waiting Rule 的第一個條件來達到 Display Host 之間的同步(Marching the display hosts)。如圖 8 所示，Server 端與 Display Host 之間以 Pipeline 的方式，將工作重疊，Server 端會比 Display Host 先前進一個 Frame 的工作。而 Display Host 必須要等待 Server 端的通知才能開始前進至下一個回合，例如圖 8 中的 C_1，在做完第 i 個 frame 時，必須要等待 Server 端通知前進訊號，當 Server 端在第 i 個 frame 時，執行完第 i+1 個 frame 的處理後，開始將第 i+1 個 frame 的資料傳送給所有的 Display Host，並且開始通知所有的 Display Host 開始進入第 i+1 個 frame，也就是送出 clock 值為 i+1 的訊號，此時 Display Host 的 clock 值也為 i+1，因此所有的 Display Host 就會一起進入第 i+1 個 frame，同時開始處理第 i+1 個 frame 的畫面繪製的工作。如此就可以減少 View-Dependent Lag 的發生，Display Host 之間也就不會發生有忽快忽慢的現象。

雖然用了 Waiting Rule 的第一個規則來限制 Display Host 必須要等待 Server 端的前進訊號才能一起開始工作。但是只使用了第一個規則，並不能保證 Server 端送出前進訊號，所有 Display Host 已經完成所有的工作，並且正在等待 Server 端的訊號。因此將會利用 Waiting Rule 的第二條規則來達到護航效應。如圖 9 所示，與圖中的符號“？”所代表的一樣，這個符號所代表的意思是要如何得知所有 Display Host 都已經完成前一個 frame 的工作，所以必須要利用第二條規則，所有 Display Host 必須要回傳一個回應訊號給 Server 端，告知 Server 端自己已經完成前一個 frame 的工作，此時 Display Host 才能開始等待 Server 端的前進訊號，而 Server 端此時必須要收集到所有 Display Host 回應訊號才能送出前進訊號，讓所有 Display Host 前進到下一個 frame。

同步硬體

根據 Waiting Rule 可以知道本計畫必須要達到下列兩個功能：

1. Marching effect：必須要利用一個前進訊號(advance single)來驅動(trigger)所有的 Display Host 開始前進到下一個畫面的繪製工作，也就是讓所有的 Display Host 工作同步。
2. Convey effect：Server 端必須要收集(collection)所有 Display Host 的回應訊號(acknowledge single)，整個系統才能進入下一個畫面的工作。也就是必須讓處理較快的 Display Host 等待處理較慢的 Display Host。

因此，必須要利用某種通訊管道來傳送這兩個訊號，所以本計畫採用了同步硬體，同步硬體是同步機制不可缺的元件。由於網路互相干擾所造成的延遲，即使是在區域網路也是無法避免資料延遲問題，因此要在網路上單純的傳送簡單的驅動訊號以達到同步的效果也是很困難的。另外，利用時脈也是無法達到同步，因為即使相同的電腦硬體，其時脈的跳動也是不一致的，所以即使在系統啟動時有先做校時的動作，硬體時間仍然無法達到完全的同步。而本計畫所使用的同步硬體則是提供不同的管道來協調所有電腦的運作，例如 parallel port 或是 SCSI 的 channel。

在同步硬體的設計上分為 Client/Server，主要是能夠提供 Server 端及 Client 端們互斥的功能，此處互斥的功能是指是否擁有同步硬體的控制權？擁有控制權者才能夠對同步硬體下達指令。當 Server 端得到同步硬體的控制權時，Client 端們就不能夠控制同步硬體，而當 Server 端釋放控制權以後，Client 端們才能夠取得同步硬體的控制權，此時 Server 端必須要等到所有的 Client 端們釋放了控制權後，才能再次取得控制權；同樣地，此時 Client 端必須要

等到 Server 端再次釋放了控制權後，Client 端們才能再次取得控制權。這裡的 Server 端即是前面所提之負責處理 Scene Management 的 Server，而 Client 端則是指 Display Host。

本計畫所設計的同步機制主要是利用同步硬體上控制權的轉移來限制 Server 及 Client 之間的動作，可以將 Server 端釋放控制權的行為及 Client 端釋放控制權的行為分別重新定義如下，以達到前面所需要的 Waiting Rule 兩個功能：

1. 回應訊號(acknowledge single)：在這裡可以將 Client 端釋放同步硬體控制權的動作稱為送出一個回應訊號，代表 Client 端已完成所指定的工作，當所有 Client 端皆已完成之後，也就是收集(collection)到所有 acknowledge single 之後，Server 端將會得到同步硬體的控制權，也就是得到一個回應訊號。
2. 前進訊號(advance single)：並且將 Server 端釋放同步硬體控制權的動作稱為送出一個前進訊號，代表 Client 端可以進入下一個工作階段。

同步演算法

底下為出本計畫的同步演算法，如圖 10 所示，在 Server 端的工作可以整理如下(以第 i+1 個 frame 為例)：

1. 等待所有 Display Host 通知已完成第 i 個畫面的回應訊號 (acknowledge single)。
2. 送出前進訊號(advance single)，通知所有 Display Host 執行第 i+1 個畫面的計算。
3. 完成第 i+2 個畫面的計算。
4. 送出第 i+2 個畫面的計算結果至所有 Display Host。

而每個 Display Host 繪圖電腦的工作則整理如下：

1. 送出回應訊號(acknowledge single)，通知已完成第 i 個畫面的計算。
2. 等待 Server 端第 i+1 個畫面的前進訊號(advance single)通知。
3. 接收第 i+1 個畫面的場景資料。
4. 執行第 i+1 個畫面的計算。

透過這個同步機制，Server 端可以知道 Display Hosts 何時完成工作以決定是否能夠執行下一步工作，而 Display Host 也可以知道是否有資料可以處理。

並且為了讓所有畫面能夠同時出現在螢幕上，本計畫在實作上做了一些調整，將同步的時機，從 Display Host 開始處理下一個 frame 的時間，提前至上一個 frame 的顯示(display)時間，也就是圖 10 中的 F 方塊。這樣就可以達到同步之外還能讓畫面顯示一致，不會造成使用者的視覺錯亂。

圖 11 為本計畫的工作流程示意圖，在圖中說明了在 Display Host 完成繪製之後，必須要送出一個回應訊號(acknowledge single)通知 Server 端，而 Server 端收到所有 Display Host 的回應訊號(acknowledge single)之後，才會送出前進訊號(advance single)，此時所有 Display Host 將會同時顯示出第 i-1 個 frame 的畫面，並且開始處理第 i 個畫面的相關工作。而 Server 端也會同時開始處理第 i+1 個 frame 的場景更新及 View-Dependent Culling 的工作。

本計畫所提的同步演算法可以將 Server 端的工作及 Display Host 的工作加以重疊，並且可以將傳遞同步訊號(前進訊號及回應訊號)的時間與傳送 VF Object List 的時間重疊。因此本計畫的系統整體時間為：

$$\mathrm{Max}(T_{Computation}, T_{Graphics}) + \mathrm{Max}(T_{Communication}, T_{Signal} + T_{Acknowledge}).$$

本計畫所提的同步演算法主要是能夠縮短系統整體的處理時間，由上述公式可以推出本計畫透過將 Display 的工作平行分散並且利用 Waiting Ruler 的同步技術來達到使用者視覺上環場畫面的效果。

六、系統實作

本計畫在淡江大學資工系的實驗室實體架設一個環場視效空間系統雛形，本系統的架構分為 3 個部分，Application、Server 端及 3 個 Clients，如圖 12 所示。但是在實作上，並不會實作出 Application 這一部分，而是著重在 Server 端及 Client 端。Server 端分為 2 個模組，而 Client 端則分為 3 個模組。

Server 端的 2 個模組分別為 Scene Manager 及 Synchronization Layer，Scene

Manager 負責管理場景物件，並且要計算每個 Client 端的 View-Dependent Culling，並將 VF Object list 傳送給 Client 端。而 Synchronization Layer 則負責提供 Server 端的同步硬體功能(傳送前進訊號及接收回應訊號)，以及包裝網路程式，提供 Scene Manager 傳送 VF Object list 的功能。

Client 端的 3 個模組分別為 Object Table Manager，Synchronization Layer 及 Render Engine。Synchronization Layer 提供 Client 端的同步硬體功能(接收前進訊號及傳送回應訊號)，並且負責接收網路資料，提供 Object Table Manager 取得 VF Object list 的功能。Object Table Manager 負責將得到的 VF Object list 加以分析，將 FOV 內的物件最新資料儲存在 Object Table，並且提供給 Render Engine 資訊，這些資訊有 FOV 內包含哪些物件，以及這些物件的最新狀態。Render Engine 則負責將 FOV 內的物件繪製出來，並且將繪製出來的畫面暫存在後緩衝區(Back Buffer)，並且等待收到前進訊號後再顯示出來。

在硬體部分，整個系統由 4 台 Pentium-IV 1.7GHz 個人電腦所組成，這四台電腦之間透過 100MB 的乙太網路環境連接，每台電腦上配置 512MB 的記憶體，及 nVidia GeForce3 Ti500 的顯示卡。而每台電腦還透過平行阜與同步硬體連接，圖 13 為同步硬體的電路雛形。

而顯示設備方面，是利用懸掛布幕式的螢幕來顯示連貫畫面，如圖 14 所示。由於空間的因素，螢幕畫面是採用正投影的方式顯示在布幕上，並且為了避免投影光線被人阻擋，造成布幕上產生陰影，因此將投影機懸掛在天花板上，讓投影光線高於人的高度。投影環境如圖 15 所示，布幕的寬度為 187 公分，每個畫面的視角是 60 度，而使用者的位置是在前螢幕的垂直線上 162 公分的位置，投影機懸掛在布幕垂直線上 300 公分的天花板位置。投影機是採用 Mitsubishi 的產品，如圖 16 所示。

本系統每個螢幕可以產生 1024*768 解析度，因此本系統可以達到 3072*768 的解析度，並且 frame rate 可以達到每秒 30 個 frame 以上。

七、結論及未來發展

本計畫主要在探討如何在多台個人電腦所構成的電腦群組環境上建構一個高解析度低成本的環場視效空間。本計畫的解決方向為二：如何將模擬系統及繪圖的工作負擔分散至多台個人電腦，來有效提昇了系統效率；研究多台電腦螢幕之間的畫面同步問題，避免畫面顯示不一致，造成使用者視覺上的混亂。而在解決方法方面，本計畫藉由管線化(pipelining)的技術，將模擬系統的工作負擔分散至多台個人電腦，以及利用 Parallel rendering 技術將繪圖工作再分散，來達到多螢幕的功能。而多螢幕的同步問題則採用 Waiting Rule 的技術，來解決繪圖電腦之間的協調性。

本計畫的主要貢獻在於提出多螢幕的同步演算法。在提出此演算法之前，本計畫先研究在管線化的機制下所產生同步問題以及如何透過 Waiting Rule 的技術來解決此問題。進而研究在多螢幕的機制下，跟管線化也有著相同的問題，並且利用 Waiting Rule 的技術來解決多台電腦螢幕之間的畫面同步問題。

本計畫的後續研究發展方向可以有以下幾點:

1. 提供使用者在環場視效空間中移動的功能，由於目前並沒有提供 off-axis 的透視投影法，因此不管使用者在環場視效空間中如何移動，畫面都不會產生變化。並且目前沒有使用追蹤器(Tracker)來追蹤使用者的位置及頭部的方向，因此也不能夠提供資訊來作 off-axis 的透視投影法。這些功能將是未來必須繼續研究的課題。
2. 目前多人互動的研究非常熱門，例如遠距教學、視訊會議以及網路遊戲等等。而環場視效空間主要是提供一個虛擬環境，因此如何將此虛擬環境與其他的應用結合，是一個很有研究價值的方向。

八、參考文獻

[1] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon and J. C. Hart, “The CAVE: audio visual experience automatic virtual environment ”,

Communications of the ACM, Vol. 35 , No. 6, pp. 64-72, 1992.

[2] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, “Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE”, Computer Graphics (SIGGRAPH '93 Proceedings), Vol. 27, pp. 135-142, August 1993.

[3] Systran Corporation. http://www.systran.com/ .

[4] V.E. Taylor, J. Chen, T.L. Disz, M.E. Papka, and R. Stevens, “Interactive virtual reality in simulations: exploring lag time”, IEEE Computational Science and Engineering, Volume:3 Issue:4, Winter 1996 pp. 46 –54.

[5] V.E. Taylor, R. Stevens, and T. Canfield, “Performance Models of Interactive, Immersive Visualization for Scientific Applications”, Proceedings of the International Workshop on High Performance Computing for Computer Graphics and Visualization, 1995.

[6] M. Wloka, “Lag in Multiprocessor Virtual Reality”, Presence, 1995, 4(1): 50-63.

[7] V.E. Taylor, R. Stevens, and T. Canfield, “Performance Models of Interactive, Immersive Visualization for Scientific Applications”, Proceedings of the International Workshop on High Performance Computing for Computer Graphics and Visualization, 1995.

[8] K. Chandy, J. Misra, “Distributed Simulation: A Case Study in Design and Verification of Distributed Programs”, IEEE Transactions on Software. Engineering , Vol. SE-5, No. 5, 1979.

九、圖表

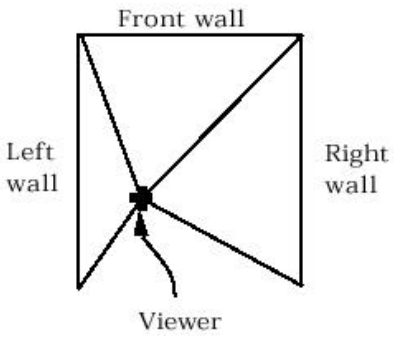

圖1. Off-axis perspective projection

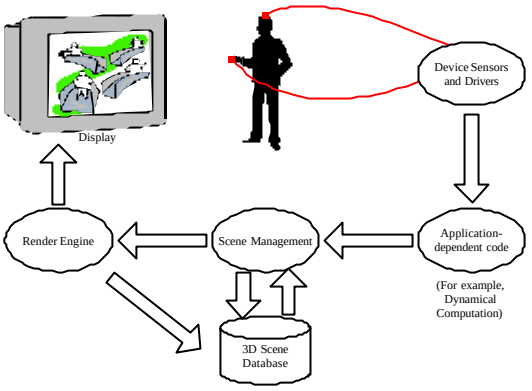

圖2. 互動模擬系統流程

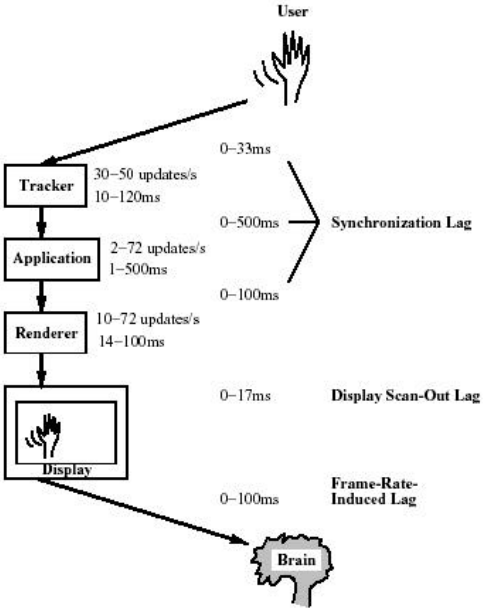

圖3. 互動式虛擬實境系統的 Lag model

表1. Sort 方法的比較

	優點	缺點
Sort-first	1.傳輸需求低。 2.硬體實作較容易，因為整個 rendering pipeline 可以設計成	1. 會產生 overhead

	一體。	
Sort-middle	1.傳輸需求低。 2.由於許多工作站級系統設計時，都將 geometry processor 及 rasterization processor 分散，因此 Sort-middle 分散 primitive 是在很自然的地方。	1.不適合 PC 環境，因為 PC 上的繪圖晶片是將 geometry processor 及 rasterization processor 設計成一個連續的處理步驟，無法分割。
Sort -last	1.硬體設計上比較容易，因為整個 rendering pipeline 可以設計成一體，除了影像合成的部分。	1.傳輸影像的圖素資料需要很高的頻寬。

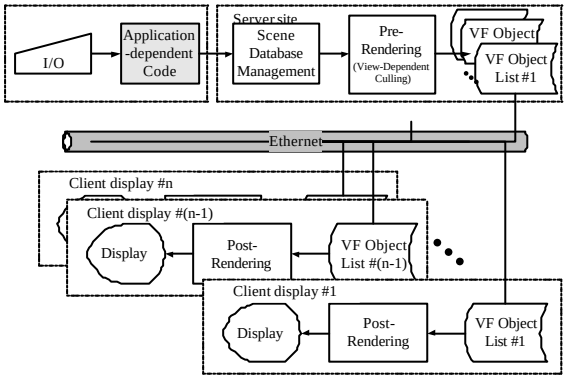

圖4. Rendering Parallelism 示意圖

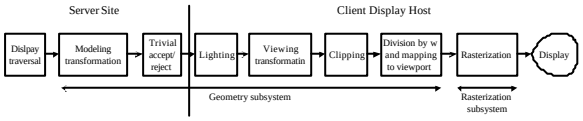

圖5. Server 端及 Display Host 的工作分界示意圖

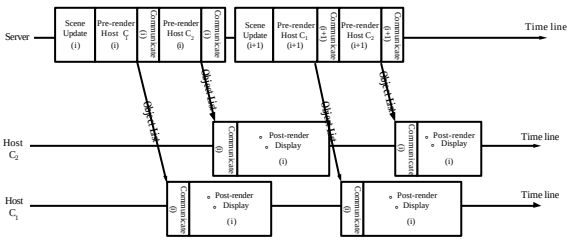

圖6. Asynchronous Synchronization 演算法

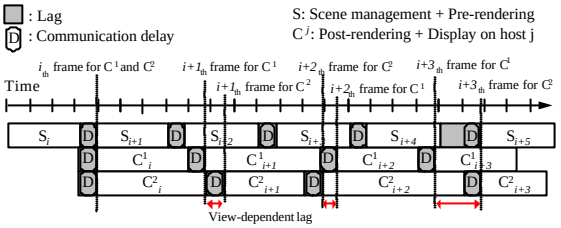

圖7. View-Dependent Lag 的示意圖

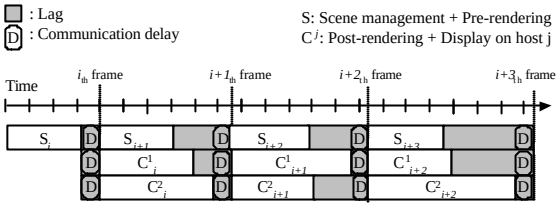

圖8. Waiting Rule 的第一個規則

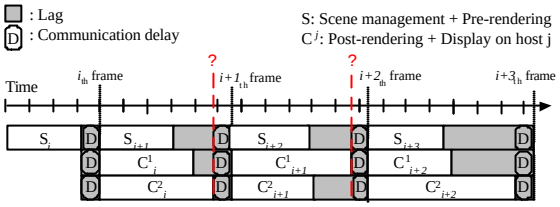

圖9. Waiting Rule 的第二個規則

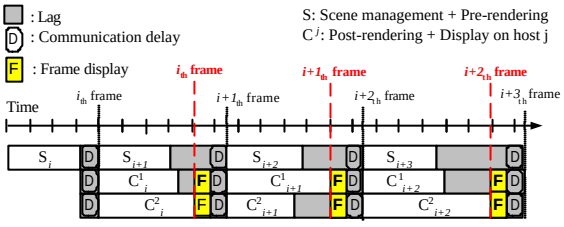

圖10. 同步演算法

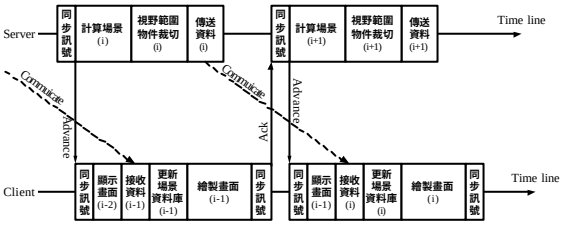

圖11. 同步演算法工作流程

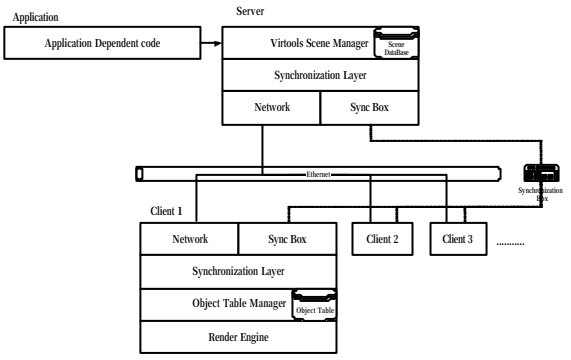

圖12. 系統環境架構

圖13. 同步硬體電路雛形

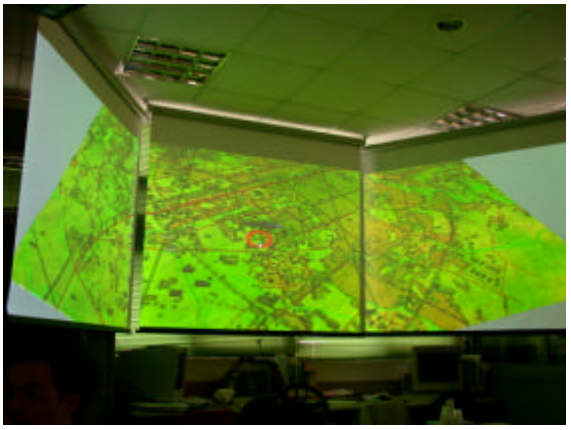

圖14. 懸掛布幕式螢幕

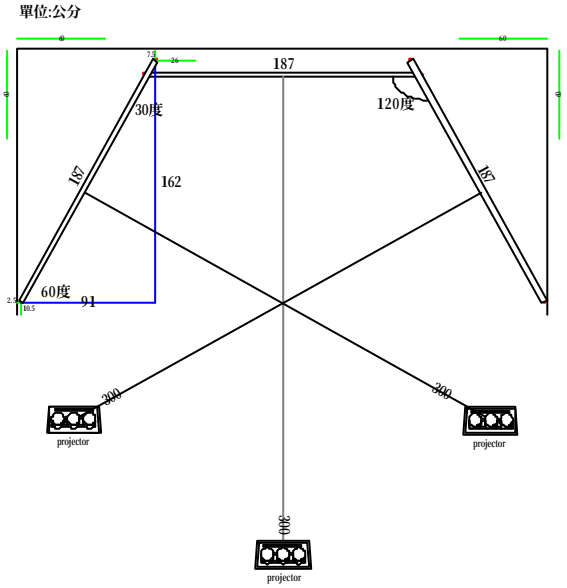

圖15. 投影環境

圖16. Mitsubishi 投影機設備