Experience of Building A High-Fidelity Mobile Crane Simulator with Cluster of Desktop Computers

*Jiung-Yao Huang and Hua-Hsen Bai
Department of Computer Science and Information Engineering
Tamkang University, Tamsui, Taiwan
**E-Mail : jhuang@mail.tku.du.tw*

Chi-fu Tai and +Chung-yun Gau
Institute of Occupational Safety and Health, Council of Labor Affairs, Executive Yuan, Taipei, Taiwan
+*E-Mail : gau@mail.cla.gov.tw*

Abstract

This paper presents the technique and experience of building a high-fidelity visual interactive simulator on a Cluster Of Desktop computers(COD). The COD is a fully distributed computation environment that is constructed by clustering several desktop computers to form a high-performance computation environment.

The contributions of this paper include proposing a method to build a high-fidelity interactive visual simulator on a cluster of desktop computers. A distribution socket, called Communication Backbone(CB), is designed and implemented to achieve this goal. In addition, the proposed architecture is used to construct a mobile crane simulator for training. The result of the implemented simulator is also presented and discussed in this paper.

1. Introduction

A virtual reality system is an interactive multi-sensual environment that tightly integrates real-time 3D computer graphics, wide-angle stereoscopic display, viewer tracking, hand and gesture tracking, binaural sound, haptic feedback, and voice input/output technologies to create a realistic synthetic environment for users. The interactive visual simulator is the most successful application of the virtual reality system. Different from the legacy virtual reality systems, the interactive visual simulator usually requires a more realistic simulating environment, including realistic mockup, for the user to fully immerse himself into the simulated scenario. Hence, a high performance mainframe is often used as the computer system of the interactive visual simulator.

A high fidelity virtual reality system, such as flight simulator or driving simulator, is capable of revealing all of the physics phenomena of the simulated entity and displaying the virtual image at the frame rate between 18 to 30 frames per second.[1] For example, when a user pushes the pedal, the fight simulator must recalculate the new position of the airplane according to its current position, velocity, acceleration, altitude, wind speed, and gravity. As the increasing demands of complexity and realism for the virtual scene, high performance supercomputer is often used as the computer system for the interactive visual simulator. Consequently, a legacy high fidelity virtual reality system often uses a multiprocessor mainframe-based system to construct its computer architecture. For example, the Ford company driving simulator[2], the IOWA drive simulator[3] and the VETT system[4] are all using a multiprocessor-based mainframe system as their computational platforms.

However, with the evolution of the computer technology, desktop computers have gained more computing power with less cost in these recent years. By carefully exploring the parallelism among the tasks of a virtual reality system, we can easily interconnect several computers by networking and employing pipeline techniques to design a high fidelity virtual environment with a cluster of desktop computers(COD).

There are two types of distributing multi-processor architectures. They are the loosely-coupled and tightly-coupled distribution.[5] In the rest of the paper, the tightly-coupled distributive computing environment is called the network-parallel environment. Two topologies are often discussed when the network-parallel environment is built. That is the server/client architecture and the fully distributive. The main differences between these two topologies lay on the protocol of forming the network-parallel environment at the initialization phase and of the message routing algorithm at the run-time phase. The work presented in this paper focuses on the second approach. For the rest of the paper, the technique of building an interactive network-parallel environment on the COD is presented first. The principle of designing a mobile crane simulator on the COD then follows. Finally, the implementation and result are presented.

2. The cluster of desktop computers(COD)

0-7695-1080-9/01 $10.00 © 2001 IEEE

The distributing computing technique presented in this paper is adopted from the High Level Architecture(HLA).[6] The HLA is a distributive simulation standard that is proposed and developed by Defense of Department(DoD), US, which has become IEEE 1516 standard recently. HLA standard is originally designed for multiple participant interaction over the Internet. This work presented in this paper adopts the concept from HLA with further enhancement to design the network-parallel environment on COD. Since there is no server for such a distributed environment to coordinate messages among distributed tasks, an asynchronous message passing mechanism by Candy[7] is referenced to perform the distributive simulation.

2.1 Principle of a fully distributed COD environment

The basic concept of designing a fully distributed interactive simulating environment with a cluster of desktop computers(COD) is that each process on the COD environment runs as a standalone program. Each process only needs to convey its event message to the communication kernel on every computer without knowing the existence of other processes.[8] Hence, under the COD architecture, each computer can be executed at its own pace and parallelisms among distributed tasks are then automatically explored. In order to achieve this goal, as shown in Figure 1, a transparent communication layer, called Communication Backbone(CB), is designed to provide seamless communication among distributed tasks.

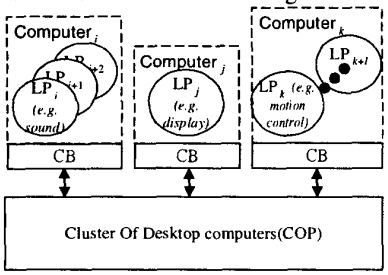

Figure 1. The infrastructure of the COD environment

Each computer of this COD environment executes CB as its communication layer, and Logical Processes(LPs) run on different computers communicating with each other via CB. Each LP of COD does not have to concern about the existence of other LPs. One or many LPs can run on a computer, depending upon the computational load of each LP. Each LP only needs to register to its resident CB upon its execution by issuing supported service calls. The CB will be responsible for the scheduling message flow among distributed tasks, no matter that the corresponding LP is in the same machine or across network. With this capability, a heterogeneous computing environment can be constructed, and different tasks can be easily plugged to form different types of simulation environment.

2.2 The virtual channel for pipelining execution

The virtual channel mechanism is an important notion about CB for COD. Conceptually, virtual channels are the pipelines that seamlessly interconnect LPs to form a simulating environment. Physically, as shown in Figure 2, a virtual channel is an entry mapping between CBs. That is, after an LP registers to CB as a publisher or subscriber, CB will respectively record LP's information in its Publication table or Subscription table. During the initialization phase, when a publisher is matched with a subscriber, an entry of its Publication table will be "linked" to the corresponding entry of Subscription table of that subscriber.

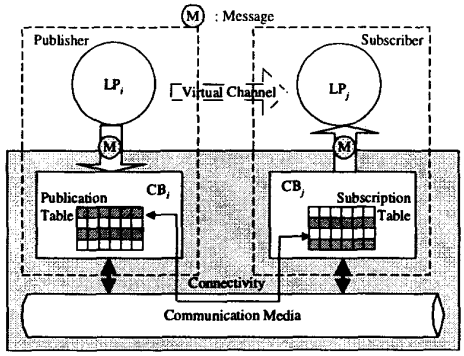

Figure 2. The virtual channel between LPs

Essentially, the virtual channel is the important mechanism that makes the transparency capability among distributed LPs possible. As discussed in the previous subsection, pair of CBs establishes a virtual channel during the initialization phase and the protocol of forming this virtual channel is presented in the next section.

2.3 Communication protocol

The communication protocol employed by the COD to schedule the message flow among LPs is discussed in the following subsections.

Initialization protocol. The message scheduling among the tasks on COD is realized in the form of virtual channel, and, as presented in the previous section, the virtual channel is established during the initialization phase. To provide transparency connection among LPs, CB must have an initialization protocol to build a virtual channel based upon publishing and subscribing information registered by LPs. Since Publish Object Class and Subscribe Object Class are two services defined in HLA to initialize a distributive environment, they are adopted to design our initialization protocol.

When an LP, says LP_j, registers to CB_j as a subscriber, CB_j will then continuously broadcast this subscription message at a constant time interval until an ACKNOWLEDGE message is received from another CB, says CB_i. Upon receiving Acknowledge message, CB_j will send a CHANNEL CONNECTION message with necessary information to the acknowledge issuer, i.e. CB_i, to construct a virtual channel between the two parties. An ACKNOWLEDGE message will be received again if such a virtual channel is successfully built.

On the contrary, when an LP, says LP_i notifies its CB_i as a publisher, CB_i will do nothing but listening to the network until a SUBSCRIPTION message is received. Upon receiving a SUBSCRIPTION message, CB_i will check its subscription table first to see if its LP is the producer of this subscription request. If it is, CB_i will then respond with an ACKNOWLEDGE message to start a virtual channel connection process with SUBSCRIPTION message issuer, says CB_j. Notice that CB_i will continuously listen to the network even while it is executed. With this protocol, an LP(an extra display, for example) can be dynamically added to the system without restarting the entire system.

Message routing. After the virtual channel is established, the publisher LP, says LP_i, can continuously send its data to its CB, says CB_i, through UPDATE ATTRIBUTE VALUE service. CB_i then routes the received data to appropriate CB, says CB_j, via the virtual channel that is established during the Initialization phase. Finally, CB_j notifies the arrival of new data with REFLECT ATTRIBUTE VALUE to its subscriber, says LP_j. That is, in terms of the push and pull model, the publisher continuously "pushes" the data through virtual channel and the subscriber "pulls" the required data from the virtual channel.

3. Design of the mobile crane simulator

3.1 The infrastructure

The mobile crane simulator is a project sponsored by Employment and Occupational Training Administration, Council of Labor Affair, Executive Yuan, Taiwan, to build a training device to train the worker to safely manipulate the mobile crane. The mobile crane is a highly dangerous yet effective lifting device in the construction site. Due to the nature of its hazard, occupational disaster often occurs. The mobile crane simulator aims to relieve this situation by providing a safety and repeatable training environment to train the construction laborer to safely manipulate the device. In order to provide a realistic simulated environment for the training, the mockup and the handling device of the trainer have to realistically duplicate the actual mobile crane. In addition, a motion platform is also required to emulate the hazard environment.

The mobile crane simulator is designed on the fully distributed cluster of desktop computers(COD). Each computer of the COD executes the Communication Backbone(CB) as its transparent communication socket to form an interactive distributed simulation platform. The CB is stemmed from the MUDS (Multiple User Distributive simulating) project.[8] Under COD environment, we can easily design a modularized interactive visual simulating system with CB. The software infrastructure of the mobile crane simulator is as shown in Figure 3.

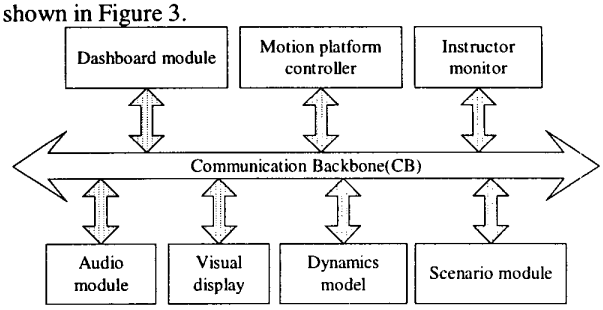

Figure 3. The infrastructure of the mobile crane simulator

The mobile crane simulator is composed of seven modules, Dashboard module, Motion platform controller, Instructor monitor, Scenario module, Dynamics model, Visual display and Audio module. With the help of the publisher/subscriber and push/pull models, each module is executed independently. In the following subsections, the function of each module is elaborately introduced.

3.2 The dashboard module

The dashboard module is part of the I/O device for the simulator.

Figure 4. The dashboard inside the mockup

As shown in Figure 4, there are two categories of input/output devices on the mobile crane simulator. One is called sensory device which is composed of a surround view of display, audio module and motion platform controller. The other is a mockup for the trainer to fully immerse himself into the training scenario. The

instruments on the dashboard include various indicators, meters, steering wheel, gas pedal and brake. The dashboard module is a program that monitors the signal of each instrument on the dashboard and translates the received signal into message to be passed to other modules. In addition, the dashboard module receives messages from the Instructor monitor and transforms them into signals to drive the meters and indicators on the dashboard.

Specifically, the input devices built in the mockup include steering wheel, brake, gas pedal and two joysticks to control the motion of the derrick boom and the plumb cable of the mobile crane.

3.3 The instructor monitor

Since the mobile crane simulator is designed as the training vehicle, an instructor monitor module is an important interface for the instructor to monitor the operation of the trainee. Two windows are designed for the instructor to supervise the trainee. The first one is called the Status window, as illustrated in Figure 5, which is a two-dimensional display to simulate the status of the mobile crane motion. The sub-windows display, respectively the current swinging angle of the derrick boom, the raising degrees of the derrick boom, the current length of the plumb cable, and the elongate length of the derrick boom. The information displayed on each window will also be printed on the small dialogue boxes next to the bottom-middle area. The alarm signals are also added to signal the misconduct of the operator if one occurs. For example, if the derrick boom overshoots the safety zone, the second alarm will be lighted to signal the possible danger.

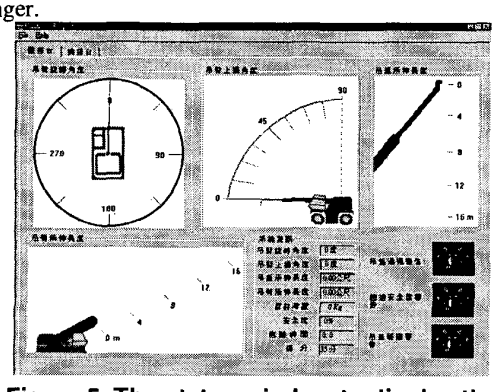

Figure 5. The status window to display the situation of the mobile crane

The second instructor window is the pictorial duplication of the instruments on the dashboard, called the Dashboard window, as shown in Figure 6. The dashboard window is the complete duplication of the dashboard inside the mockup. It aims to allow the instructor to oversee the training procedure of the trainee. In addition, since the dashboard module can send signals to the dashboard, the instrument display may be used for trouble shooting training when the instructor clicks any of the indicators or switches to signal an accident going to occur.

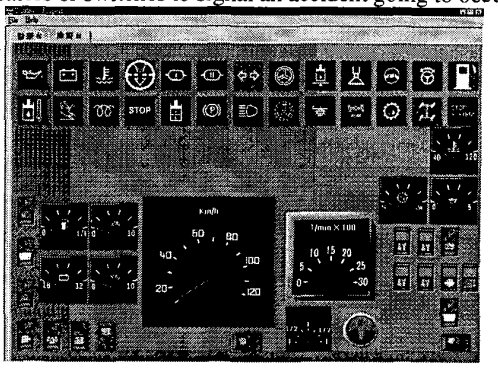

Figure 6. The dashboard window is the pictorial duplication of the dashboard inside the mockup

3.4 Motion platform controller

The motion platform controller is the module to manipulate the six-manipulator motion platform. The motion platform is another important sensory device for the trainee to fully immerse himself into the training scenario. The six-manipulator motion platform was first presented by Stewart[9] in 1965 and is called Stewart Platform Based Manipulator thereafter. As shown in Figure 7, The Stewart Platform Based Manipulator is constructed by using six parallel manipulators to connect the platform with the base. These six manipulators can be expanded and contracted individually to control the gesture of the platform.

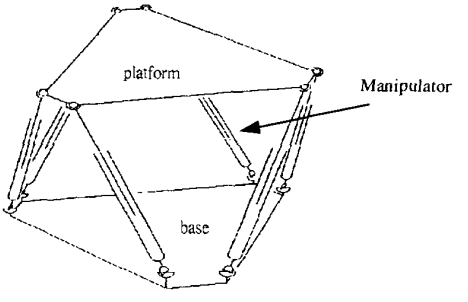

Figure 7. The stewart platform based manipulator

In order to realistically simulate the motion of the mobile crane inside the virtual environment, the motion platform controller must smoothly transform the posture of the platform between the consecutive statuses. In addition, the frequency of this interpolation should be synchronized with the visual display in order not to disorder the sensorium of the user. Otherwise, for example, the user may visually see the mobile crane going downhill while the motion platform is still in uphill posture.

Finally, the vibration of the motion platform is also another important factor to realistically simulate the mobile crane. Since the mobile crane is a heavy industrial instrument, it will create noisy sounds and vibration while its engine is ignited. The motion platform controller constantly generates a random up-and-down vibration to realistically simulate this situation.

3.5 Scenario control module

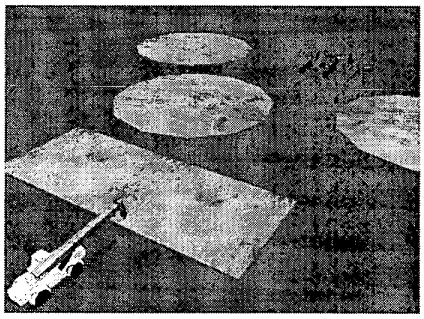

Figure 8. Bird eye view of the training scene

This module is designed to manage the state change inside the virtual world and, hence, to evaluate the performance of the trainee. Since this mobile crane simulator is designed as the vehicle for training as well as licensure, a scenario is designed to evaluate the trainee. As shown in Figure 8, the scenario includes driving the mobile crane from the starting point to a designated location. In order to increase the difficulty of training, the designed scenario includes the driving of the mobile crane through the sand land and hills.

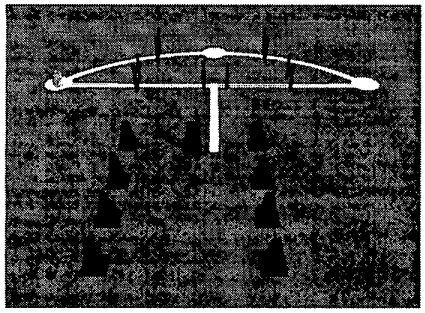

Figure 9. The trajectory of licensing exam

After arriving the testing ground, the trainee is asked to control the derrick boom to lift a cargo and move that cargo according to specific trajectory. As shown in Figure 9, bars are placed on the path of the trajectory to obstruct the movement of that cargo. The examinee requires to lifting the cargo located in the white circular zone at the left side of Figure 9 and moving the cargo to its right side along the trajectory and back to its original position. Score will be deducted if the bar is collided and the score will be dynamically displayed on the status window that is shown in Figure 5.

3.6 The dynamics module

The dynamic module increases the realism of simulation by calculating various physical phenomena, for example, inertia oscillation of the lift hook. The dynamic computation designed for the mobile crane simulator includes the inertia oscillation of the rope, collision detection and terrain following.

When the mobile crane and its lift hook are moved in the virtual environment, the dynamic computation uses the multi-level collision detection algorithm[10] to effectively perceive the collision if there is any. If a collision is detected, the dynamic module first animates the collision event and then sends messages to the sound module and the visual display module to playback a collision sound and display the scene respectively.

In addition, in order to realistically simulate the mobile crane, the dynamic module also computes the inertia oscillation of the lift hook during and after the derrick boom is moving. When the derrick boom is moving, the dynamic module computes the inertia of the left hook acts on the cable based upon the moving direction, speed and weight of the cargo. When the derrick boom is stopped from moving, the same computation of the inertia will be repeated and the cable is oscillated until a full stop.

Terrain following is another important factor for the realism of simulation. Since its center of gravity is higher than that of other types of vehicle, driving the mobile crane is also a dangerous process. The terrain following mechanism along with the motion platform provides a method to train the operator to handle the mobile crane properly.

3.7 The visual display and audio modules

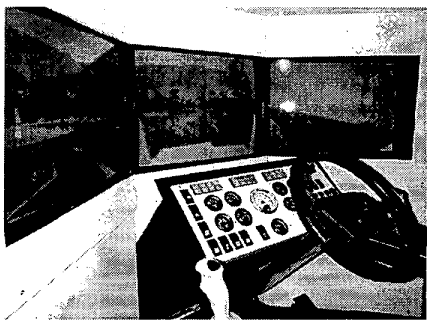

Figure 10. Three synchronized monitors to provide a surround view of scene

The visual display and audio modules are another two important sensory devices for the user to fully immerse himself in the virtual environment. As depicted in Figure 10, three monitors are used to provide around 120 degrees of surround view. This surround view system is fully

synchronized with each other so that a consistent view will be displayed.[11] With this surround view system, the trainee can fully immerse himself into the virtual environment.

The audio module is responsible for producing, the static sound, such as the background noise, as well as the dynamic sound effect, such as collision sound or motor working noise. The Microsoft DirectSound library is used to implement the sound module. With this sound module, a realistic training scenario can be provided for the trainee.

4. Implementation and result

A cluster of eight desktop computers constructs the computer system for the mobile crane simulator. These eight computers are networked into a local area network environment and are mounted on a racked frame as shown in Figure 11.

Figure 11. Rack-mounted computers to form a COD

The top three computers control the displays of three monitors inside the mockup to create the surrounded view. Each display computer is equipped with a TNT2 M64 3D graphic acceleration card. The fourth computer from the top is the synchronization server that synchronizes the frame rate of the above three graphical computers to generate the surrounded view. Due to the overhead of the synchronization among the three graphical computers, the frame rate of the surrounded view is 16 frame-per-second with totally 3235 polygons inside the virtual scene. Figure 12 shows the appearance of the mobile crane simulator.

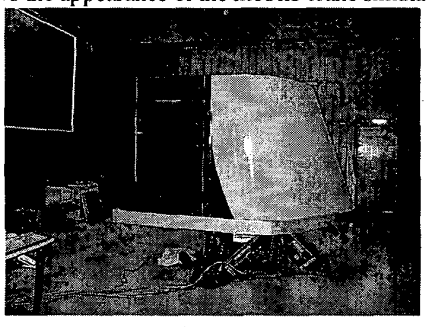

Figure 12. The mobile crane training system

5. Conclusion and future works

This paper presents a work of designing a high-fidelity simulator on a cluster of desktop computers. The legacy simulator systems were using a multiple processor mainframe as their computer system. Many studies have been proceeded on building cluster of network computers to perform complicated computation, which usually require a mainframe-based multiprocessor system. With the proper design of the communication layer, the networked desktop computers can gain the effect of parallel computation on distributed computers. This work presented in this paper successfully proves this point.

Although the current frame rate only reaches 16 fps, further accelerating of the frame rate is possible and current under investigation. With the success of this project, the cost of building a high-fidelity simulator can be significantly reduced. In addition, since the entire simulator system is modularized, more functionality can be added as required.

References

[1] G. Burdea. and P. Coiffet, *Virtual Reality Technology*, John Wiley & Sons, Inc., 1993.

[2] J. A. Greenberg and T. J. Park, "Driving Simulation at Ford", Automotive Engineering, pp.37-40, September 1994.

[3] J.S. Freeman, et al, "The IOWA Driving Simulator: An Implementation and Application Overview", technical paper, available at http://www.ccad.uiowa.edu/research/ids/technical-papers.

[4] D. Zeltzer, N. J. Pioch, and W. A. Aviles, "Training the Officer of the Deck", IEEE Computer Graphics and Applications, November 1995, pp.6-9.

[5] A. S. Tanenbaum, *Operating systems: Design and Implementation*, Prentice pub. 1987, ISBN 0-13-637406-9.

[6] E. Guckenberger, R. Whituecy and G. Hall, "Riding the Third Wave with DIS and HLA" , MS&T, May 1996, pp.28-36.

[7] K. Chandy and J. Misra, "Asynchronous Distributed Simulation via a Sequence of Parallel Computations", CACM, Vol. 24, No. 11, April 1981, pp198-206.

[8] J.Y. Huang, C.T. Fang-Tson, S.J. Wang, and W.C. Wang, "A Model and Design of a Fully Distributed Computing Environment for Virtual Reality", Real-Time Computing Systems and Applications (RTCSA), Taiwan, October 1997, pp.160-168.

[9] Stewart, "A platform with six degree of freedom", Proc. of Institute of Mechanical Engineering, 180, pp.371-386, 1965.

[10] M. Moore and J. Wilhelms, "Collision Detection and Response for Computer Animation", Proceedings of the 15th annual conference on Computer Graphics, Vol. 22, No. 4, pp.289-298, August 1988.

[11] "*Researching and developing the dangerous machinery virtual reality system – Manufacturing and popularizing the VR training system and multimedia education system of crane*", Project Report, Division of Occupational Safety, Institute of Occupational Safety and Health, Council of Labor Affairs, Executive Yuan, Taiwan, 1999.