Network And User Interface For Pat Dome Virtual Motion Environment System

J. W. Worthington, K. M. Duncan W. G. Crosier
Krug Life Sciences
1290 Hercules, Suite 120
Houston, Texas 77058

ABSTRACT

The Device for Orientation and Motion Environments Preflight Adaptation Trainer (DOME PAT) provides astronauts a virtual microgravity sensory environment designed to help allieviate the symptoms of space motion sickness (SMS). The system consists of four microcomputers networked to provide real time control, and an Image Generator (IG) driving a wide angle video display inside a dome structure. The spherical display demands distortion correction. The system is currently being modified with a new graphical user interface (GUI) and a new Silicon Graphics IG. This paper will concentrate on the new GUI and the networking scheme.

The new GUI eliminates proprietary graphics hardware and software , and instead makes use of standard and low cost PC video (CGA) and off the shelf software (Microsoft's Quick C). Mouse selection for user input is supported.

The new Silicon Graphics IG requires an Ethernet interface. The microcomputer known as the Real Time Controller (RTC), which has overall control of the system and is written in Ada, was modified to use the free public domain NCSA Telnet software for Ethernet communications with the Silicon Graphics IG. The RTC also maintains the original ARCNET communications through Novell Netware IPX with the rest of the system. The Telnet TCP/IP protocol was first used for real-time communication, but because of buffering problems the Telnet datagram (UDP) protocol needed to be implemented. Since the Telnet modules are written in C, the Ada pragma "Interface" was used to interface with the network calls.

INTRODUCTION

The main purpose of the Preflight Adaptation Training (PAT) laboratory at the Johnson Space Center (JSC) is to help reduce or eliminate Space Motion Sickness (SMS) symptoms and to improve neurosensory performance during the initial days of flight as well as helping in readaptation to earth's gravity following a space flight. The JSC PAT laboratory has two trainers; a Tilt-Translation Device (TTD) and the Device for Orientation and Motion Environments (DOME). This paper will discuss the networking scheme and graphical user interface (GUI) for the DOME system.

The DOME system creates a virtual reality environment to simulate some of the sensory rearrangements that astronauts experience on-orbit. A computer generated scene is projected onto the dome surface. Projection onto the dome surface requires distortion correction. The trainee can experience passive motion stimuli or can control his/her own motion with a six degree of freedom hand controller. Head movements can also be enabled to drive the scene. The microcomputer kinematics control can be set to simulate various motion environments e.g. microgravity.

Four microcomputers (PCs) and two Silicon Graphics (SGI) Crimson VGXT graphics computers, used as Image Generators (IG), drive the DOME system. The PCs communicate with each other over an ARCNET network

using calls to a proprietary virtual token ring network through the Novell Netware IPX kernel. One of the PCs also communicates with the SGI Crimsons using NCSA Telnet network calls over an Ethernet connection. The Ethernet also connects the SGI machines with each other.

The instructor/operator selects various training environments using one PC which is known as the Instructor Operator Station (IOS). The IOS employs a graphical user interface with mouse selection.

The other PCs are: the Real-Time Conroller (RTC) which is the 'executive' or has basic control of the system and is the PC which communicates with the SGI Crimsons; the Real-Time Positioning System which accepts trainer device input from the trainee or instructor and provides kinematics processing; and the Data Logging and Display System (DLDS) which can display selected signals as on an oscilloscope.

Problems with special hardware and proprietary software led to rewriting the software and changing the display for the IOS. The fact that the SGI machines use Ethernet for network communication required that the RTC be able to support an Ethernet connection as well as maintain the ARCNET communication with the rest of the system. Budget constraints forced a low cost solution for these items.

GRAPHICAL USER INTERFACE

History

The original IOS used Multisynch monitors with a 768x1024 resolution for the user display. In order to generate this display special graphics boards were installed. These special graphics boards were accessed by making calls to special software written in FORTRAN; therefore, the original IOS was written in FORTRAN.

The display output could best be described as pages rather than windows since they were static in nature and selection was a lot like turning to a page in a book. Several pages were provided by the subcontractor; they were quite detailed and the resolution allowed much information per page. Some of the pages were supposed to allow the user to change some of the parameters in the system's computer kinematics and input control.

Problems

The subcontractor had originally expected to allow the user to select items by using a mouse. However, for some reason perhaps because of memory conflicts with the graphics boards and mouse driver, or interfacing the FORTRAN code with assembler calls to Int33H, the mouse never worked correctly. Our efforts in getting the mouse to work within this setup also failed. It was relativley time consuming to use cursor keys on these large display pages waiting for the cursor to arrive at the desired location.

Changes that the user made to the DOME parameters on the IOS exhibited inconsistent results when sent to the RTC. Values were wrong or missing which meant the desired change was not made. The major problem appears to have been the difference in data representations in FORTRAN on the IOS machine transferring to Ada on the RTC machine. We made modifications that helped, but there were still inconsistencies.

The special graphics cards failed. This would have left the DOME inoperable except that we had written a very terse non-graphic IOS using Ada which allowed system operation. This failure pointed out the dangers of relying on special hardware items. The manufacturer was a local business so the boards were repaired in less time than if they would have to have been shipped. Also, the graphics cards and software were geared to the Multisynch monitors only. Even though we have two of these monitors, each one failed and needed repair within a few weeks of each other. This again pointed out how much we were relying on special hardware.

As trainer use increased we found parameters that we would like to add to the IOS to allow the instructor/operator to select. The acquisition of the SGI IGs along with Paradigm Simulation's VisionWorks software provided some new capabilities that the IOS needed to let the user select. The software code as received

from the subcontractor and the myriad FORTRAN graphics calls did not lend itself easily for modification. Also, most of the pages that were originally provided were now useless.

Solution

When the monitors failed it was time to find a new way to support the IOS. The terse Ada program was a place to begin, but interfacing it with C code graphics calls caused conflicts and did not work. We were also on a very limited budget at this time, so we needed to use materials we had on hand. We had a CGA monitor and graphics card and Microsoft Quick C which provides an extensive graphics library. A short test program proved that using Quick C with CGA would be fairly easy and very economical.

Mouse support was easily implemented by using Int33H functions called from the Quick C code.

By this time the requirements for what the user would need to be able to select from the IOS were firmly established, although these requirements do change over time. The pages that allowed selection of unused items were eliminated. New pages were added to provide support for the new capabilities provided by the VisionWorks software and SGI IGs e.g. 'floating' models and 'spinning'. The display should still be thought of as pages, but with a modular programming structure new pages can be programmed quickly. The worst limit posed by the CGA graphics is the resolution. Selections have to be relatively large so items per page are limited more severely than with the previous system or EGA.

The system allows user direct entry for some parameters. There has been no problem 'packaging' the C structured variables into a network packet that the Ada code on the RTC retrieves and uses, or vice versa.

The current IOS greatly decreased the time required for the instructor/operator to set up a session in the DOME. The mouse selection is much faster than the cursor keys. By making the pages pertain to what is required, the instructor/operator upon selecting an option (eg 'spinning' the scene) allows the system to immediately respond only to that item, whereas before all parameters were passed to the RTC. Having modularized and well-commented code in a familiar language has made this possible. This also allows for easy modification - adding or deleting pages for desired functions, and there is no 'wasted' code for unecessary pages.

Future

In the near future we hope to incorporate the IOS onto a Macintosh. This will allow true windowing rather than displaying pages. The choice of the Macintosh over Windows on a PC also stems from what is on hand. We already have development tools and experience in Macintosh programming, but we would have to purchase the Windows Development Kit and learn it. Also, our end users (the neuro lab scientists) use Macintosh computers at their desks and are already familiar with its menu scheme, dialog boxes, scroll bars, etc.

Modular programming structures will continue to be used; or perhaps object oriented class structures. Higher display resolution will be gained. It is also possible that we could combine the functions of the IOS and one of the other systems onto this single machine.

NETWORKING DOME COMPUTERS

History

The DOME system is set up to use networked microcomputers to allow a degree of parallel processing for the real-time operation. The Real Time Positioning System (RTPS) receives input from the hand controllers and helmet position. It performs calculations relating to the orientation of the hand control to the subject, thresholding, and kinematics. Its output is a six degree of freedom vector of changes in position (Delta Position Vectors or DPVs) in body coordinates which it sends to the Real Time Controller (RTC) over the network. The RTC takes the DPVs from the RTPS and transforms them into world coordinates, integrates these into a new world position vector (eyepoint), and sends these eyepoints to the IG for display. Therefore, the RTPS can be

processing the next subject input while the RTC is finishing with the previous input. This turns out to be a low cost method of implementing parallel processing of the math intensive equations necessary for the real time operation of the trainer. Our original image generator had a PC front end which was connected to the same network in order to receive the eyepoints from the RTC. Additionally we have a Data Logging and Display System (DLDS) PC which can be used to show various signals in real-time. This PC is also connected to the network.

The network architecture chosen by the subcontractor uses ARCNET cards and coaxial cable at the physical level. The network uses Novell Netware IPX as the kernel. The applications make calls to a library of functions which implement a proprietary virtual token ring network called Micronet developed by Microsim. The ARCNET has a throughput of about 2.5Mbps.

Ongoing problems with the original IG ultimately led to the acquisition of the two SGI Crimsons.

Problems

The specifications for acquiring the new SGI machines required that communication between them and our system be implemented with an Ethernet connection using the TCP/IP protocol. It was determined that the RTC would be the PC which would connect to the SGI. This raised several questions. What software would we need for Ethernet and TCP/IP on the PC? Would the RTC be able to maintain its ARCNET capabilities and take on the Ethernet connection? If not, would we need a bridge or some other interconnection device; or, as an alternative, would it be worth the time, effort and cost to modify the software in all of the PCs in order to have one Ethernet network? Would the RTC Ada code interface with the network software if it was not written in Ada? What would be the cost to modify the system?

Solution

Knowing the possibility that memory conflicts and Interrupt Request (IRQ) conflicts could occur we decided to go ahead and try to support both networks on the RTC. If it worked, fine; if not, we would still need the Ethernet and TCP/IP software for another PC, which would be used solely for communication with the SGI machines, or if we modified the system to use only Ethernet.

An Ethernet card was ordered. It is supported by the NE2000 driver, could support both thin-net or thick-net cabling, and had four selectable IRQ lines. A transceiver for adapting from the AUI Ethernet port on the SGI to a BNC connection for thin-net was secured.

The networking software required would only need to be a small subset of what was available. It would need to support creating a socket, opening a connection, sending packets, and receiving packets. It turned out that NCSA at the University of Illinois, Champaign-Urbana, had TCP/IP software avaliable in the public domain by the name of NCSA Telnet. The source code was provided and downloaded over the Internet. Only the necessary modules needed for the DOME system were compiled from the C source files and placed into a library along with the application functions that referenced the Telnet functions.

In order for the Ada code in the RTC to access the network calls written and compiled into a C library, package ETHNET was created. This package contained Ada procedures for the Ethernet network and used pragma INTERFACE and pragma INTERFACE_NAME to access the C functions. The Ada code also had to be linked with the C library file.

With the software installed and the hardware in place, a test was conducted between the RTC and an SGI Crimson. The proper IRQ line was finally set on the Ethernet card, a good address for installing the NE2000 driver was selected, and the communication between the machines worked fine. Expanding the test to include the ARCNET calls to the other machines also worked fine; we did not have any conflicts by using two different network cards in the RTC PC.

While testing in real-time it was discovered that the SGI code was taking the TCP/IP packets and buffering them about three at a time. This slowed down the real-time performance and caused 'jerky' motion. It was finally decided to change from the TCP/IP protocol to the UDP (datagram) protocol because UDP is a 'don't care if received' type protocol and SGI would not buffer these packets. The NCSA Telnet code when compiled for the PC included the datagram service functions so this change was made quickly. The datagram service solved the real-time problem and is working very well.

Future

The current system PCs are 286 and 386 based machines running at 8 to 10 MHz. We hope to minimize the reliance on networking by combining the fuctionality of the RTC, RTPS, and IOS into one or two PCs since the power of PCs has greatly increased over what the current system has.

Futhermore, we plan to have a single network supporting the datagram service and TCP/IP over the Ethernet. This will eliminate the proprietary network and enable greater control over the networking software.