FULLY IMMERSIVE TEAM TRAINING: A NETWORKED TESTBED FOR GROUND-BASED TRAINING MISSIONS

James Parsons, Don Lampton†, Kimberly Parsons, Bruce W. Knerr†, David Russell, Glenn Martin, Jason Daly, Bryan Kline, Matthew Weaver

Institute for Simulation & Training
University of Central Florida

†*US Army Research Institute*
Orlando, Florida

ABSTRACT

The Fully Immersive Team Training Testbed was developed to study the methods for using Virtual Environment (VE) technology for training dismounted infantry teams. The testbed allows multiple trainees networked together on different computers to be immersed simultaneously and produces a compelling sense of presence; a powerful feeling of being immersed in the VE. A wide variety of parameters can be configured for inclusion in a training scenario including tools, weapons, dynamic environmental objects, and number and skill level of opposition forces. In addition to providing a simulation arena for multiplayer interaction, the testbed captures all aspects of a mission, including radio communication, visuals with unrestrained placement of camera, and environmental audio and are made available for use in after action critiques.

This paper describes the implementation methods used for creating the complex simulation testbed. The environment is scalable and supports the networking of trainees located in different cities. Specifics on custom hardware development, software structure, body sensor deployment, locomotion method and networking solutions are provided. In addition, the implementation of a training scenario is described, and results are presented.

Biographical Sketch

James Parsons is a research faculty member working in virtual environment (VE) R&D in IST's Visual Systems Lab (VSL). He is currently working on the ARI VE Testbed project as a software engineer. In addition to his Masters degree in Computer Science, James also has a Bachelor of Fine Arts degree from the State University of New York at Purchase, where he studied scenic design for theater, film and television. Affiliations include Upsilon Pi Epsilon, IEEE, and the Association for Computing Machinery.

Fully Immersive Team Training: A Networked Testbed for Ground-Based Training Missions

James Parsons, Don Lampton†, Kimberly Parsons, Bruce W. Knerr†, David Russell, Glenn Martin, Jason Daly, Bryan Kline, Matthew Weaver

Institute for Simulation & Training
University of Central Florida

†*US Army Research Institute*
Orlando, Florida

INTRODUCTION

Military exercises conducted as distributed battlefield simulations have been acknowledged as a viable complement to real world training since the days of SIMNET in the early 1980's. Joint simulation exercises from SIMNET's time, through the development of DIS, and up to and including more recent HLA implementations have featured a wide variety of combat vehicles interacting together while the dismounted infantryman has for the large part been excluded from participation. Virtual Environment (VE) technology represents a possible means for dismounted soldiers to participate in simulated wargames. The Fully Immersive Team Training (FITT) testbed has been designed to investigate the use of VE technology to train dismounted soldiers.

This paper will explore some of the technical hurdles encountered in the development of a robust and flexible testbed for exploring the feasibility of using VEs to train soldiers in team tasks. The current testbed implementation is the result of several years of research and development conducted jointly by the Institute for Simulation & Training at the University of Central Florida and the Army Research Institute. The work that led to the testbed's current version can be reviewed elsewhere (Parsons and Russell, 1998). This paper will focus on the process used to implement FITT.

In section 2 below, the initial team training mission implemented on the testbed is described so that testbed features presented in later sections can be more clearly illustrated in reference to this baseline mission. Then in section 3, the overall hardware and software configuration of the testbed is considered. Next, attention is turned to sensing human movement and locomotion, then translating that data into a representational avatar. Section 5 addresses the networking issues encountered as the testbed was built, including starting and stopping the machines used for the simulation in a synchronized fashion. The implementations of the data capture client and mission playback application are detailed in section 6. Computer generated Opposition Forces (OPFORs) are described next. Finally, some concluding remarks are offered.

2. MISSION SCENARIO – TEAM TASKS

In designing the experimental scenario for the first FITT implementation, the need for tasks that required rapid decision making in complex environments, situational awareness, and teamwork were paramount. Implementing these requirements with

current VE technology while representing the critical components of military tasks in an experimental configuration that required a minimum of prerequisite training was an important challenge of this project.

After considerable study of both military and civilian special weapons and tactics training manuals, urban search and rescue procedures, and infantry field manuals, the experiment design team settled on activities that would routinely be performed during search/rescue missions, hostage retrieval missions, and emergencies involving hazardous materials (HAZMAT) missions.

FITT's first research effort was organized as a search mission wherein a two person immersed team under the direction (via radio communication) of a mission commander (the experimenter) canvases the rooms of a building looking for canisters containing hazardous gas. Each team member is equipped with a chemical hazard protective suit, a timer which shows remaining minutes of oxygen available to the suit's breathing apparatus, and a hand gun which fires tranquilizer darts used to render OPFORs harmless. The search is complicated by the threat of interference by OPFORs.

Team member #1 is the designated Team Leader. The Team Leader directs team movement to provide an efficient search while maintaining team security. The leader radios reports of deactivations and encounters with the enemy to an off-site mission commander. In addition to a tranquilizer dart gun, the Team Leader is equipped with a paint marker used to mark rooms that have been searched. Team member #2, the HAZMAT Equipment Specialist, carries a dart gun and a device to detect and deactivate active gas canisters.

The missions are situated in a ten-room building. Computer-generated OPFOR and innocent bystanders move through hallways and rooms. Different types of OPFOR, lightly armed looters and heavily armed terrorists, require the trainee teams to prioritize targets. The presence of both OPFOR and neutral forces require rapid shoot/don't shoot decisions.

Mission instructions specify the amount of air available for each mission. Team members must remember to periodically check their remaining air supply indicators, and must decide when to begin to exit the building. This is not an easy decision for the team to make in that leaving too soon wastes search time. However, underestimating the time needed to exit the building results in mission failure.

Procedures include rules for: the order in which rooms are searched, team formation for room entry, actions on contact with OPFOR and innocent bystanders, and format and content of radio reports. Successful performance of some of the procedures requires the team members to coordinate their movements and actions to within a second of each other.

A thirteen-page training manual includes a mission overview, learning objectives, task descriptions with graphics, and ends with a mnemonic to help the trainee remember the procedures. A paper and pencil knowledge test is administered after the participant completes the manual.

The mission tasks and procedures served as the functional hardware and software requirements for the development of the Fully Immersive Team Training system.

3. TESTBED DESIGN

Hardware Configuration

The basic testbed incorporates a number of plugable components to provide a flexible framework for developing networked team training applications. The primary components of this system are the individual combatant simulator (ICS), a computer generated entity server (CGES), a commander/experimenter stealth display (Stealth), and a data collection system. Along with these primary testbed components, an audio system has also been implemented to simulate radio

communication between the participants during an exercise.

The ICS provides a link between the subject and the networked virtual environment. This simulator provides a view into the shared virtual world via an image generator and a viewpoint-tracked display device. The ICS also provides a means for interacting directly with the virtual world through a variety of I/O devices that can be selected according to their applicability to the mission scenario. Finally, it is the role of the ICS to translate the real world gestures and movements of the subject into appropriate avatar responses.

The CGES provides support for all of the simulated entities in the virtual environment. Environmental attributes that can be expressed as a boolean state variable such as doors left open or closed, lights turned on or off, etc. can be rapidly configured and implemented in the FITT testbed. In the mission detailed in section 2 of this paper gas cannisters encountered in the VE can be found in one of two states: leaking or secure. The cannisters are an example of a boolean environmental attribute maintained by the CGES. In addition to simple two-state environmental elements, more sophisticated entities such as computer generated opposition forces can also be served by CGES. While configuration of these types of dynamic elements are not as quickly implemented as boolean elements (they generally require the development of custom behavior code) allowances have been made for a broad range of expression on the part of the mission designer. The testbed's capabilities in this area are touched on later in section 7 of this paper.

The Stealth display allows the experimenter to watch the movement of the subjects through the virtual environment from a third person overhead perspective. Along with the graphical display of the players' positions in the virtual environment, a number of statistics are reported on the stealth display which quantify the actions of the participants and assist the experimenter in determining the efficiency of the team as they move through their required tasks.

The last piece of the system is the data collection service. During any run through the world, the data collection system records all of the network traffic. This recording can be used to replay events as they unfolded during a particular mission, or it may be analyzed to produce a numerical summary for performance evaluation. Along with recording the network traffic, the data collection system also records all of the voice communications among the participants in the exercise. When the data stream is played back for an after action critique, the audio of those communications is also played to provide greater insight into the thought processes involved and the actual interactions among the members of the team.

The team training testbed consists of the following hardware components:

ICS #1:

1. 8 Processor / 3 Pipe Silicon Graphics Onyx RE2 (Team Leader and CGES)
2. 6 Tracker Ascension MotionStar
3. Virtual Research VR4 Head Mounted Display

ICS #2:

4. 4 Processor / 1 Pipe Silicon Graphics Onyx RE2 (Specialist)
5. 6 Tracker Ascension Flock of Birds
6. Virtual Research VR4 Head Mounted Display

7. Silicon Graphics Indigo2 High Impact (Stealth)
8. Silicon Graphics Indy (Data Collection)
9. Dell Pentium 90 (Audio Capture)
10. Dell 486 (Startup Browser)
11. Misc. handheld input devices
12. Misc. video equipment for observing player views and videotaping after action critiques.

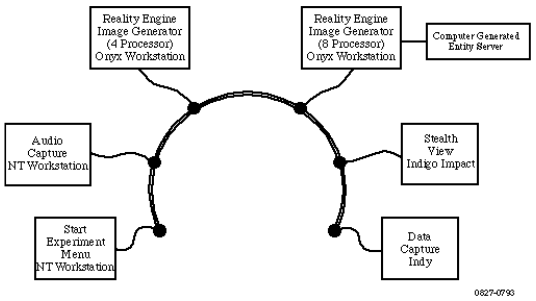

Figure 1. Networked Hardware Configuration

This hardware list is used for the current two player configuration of the testbed in support of the mission scenario described earlier in section 2. The testbed is easily scalable, and extra ICS's could be added as

necessary to support more players in the virtual environment. Note that the extra processing power of the 8 processor Onyx allows it to run the CGES component of the FITT testbed, in addition to its duties as an ICS.

Software Hierarchical Structure

The software for the multiplayer system has a distinct hierarchical structure that allows for all of the necessary interactions between the players and the virtual environment. Each object performs some portion of the necessary function of the overall system. By separating the workload in a logical manner, the software may more easily be extended or modified to add or remove functionality. The diagram of objects and their relationships in Figure 2 shows the relational structure of the software. Despite the complicated appearance, the actual complexity of the individual objects remains relatively low.

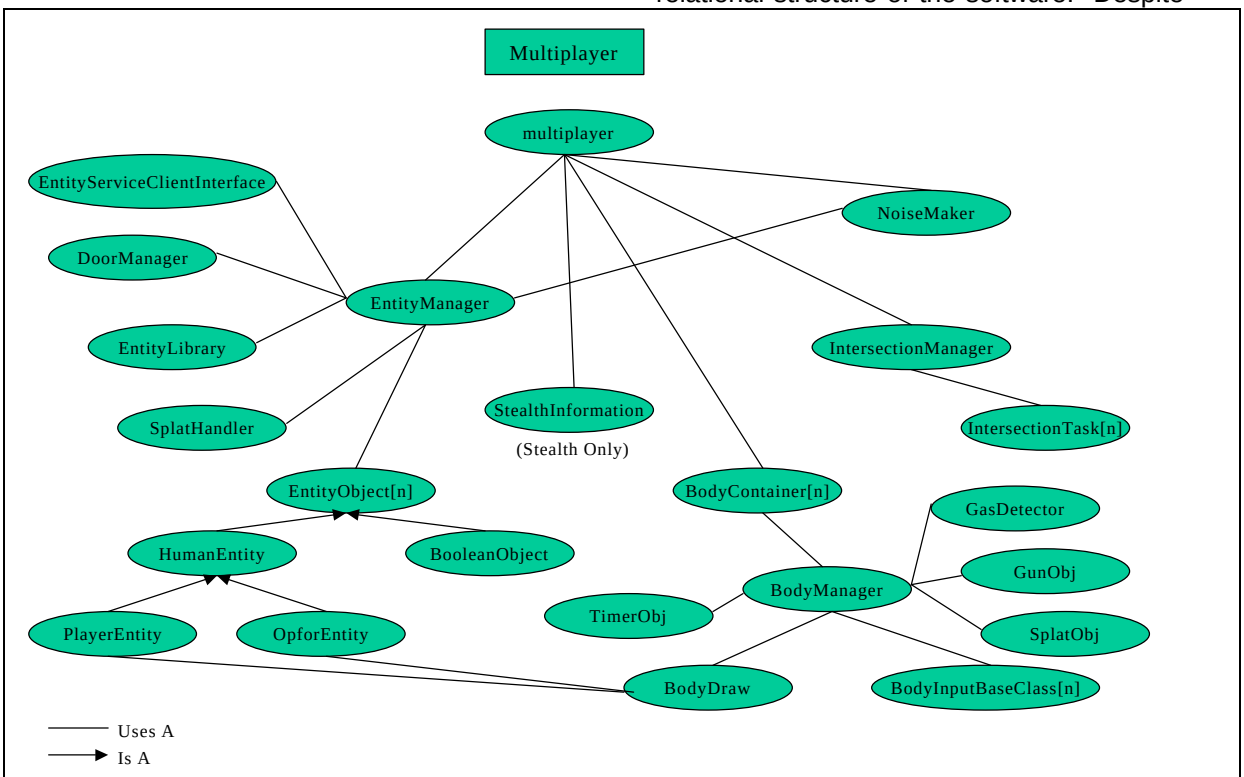

Figure 2: Hierarchical Model of the Multiplayer Software

The two principle components of the software hierarchy are the EntityManager and the main multiplayer object. These two objects control the interactions among several of the other objects which represent either the local player or remote objects and entities which must be depicted locally. These mechanisms take up the majority of processing time after the processing of the input devices.

The object-oriented hierarchical structure provides an excellent framework for development and maintenance of the software. Each piece may be developed separately by different programmers as long as the interface between the objects has been agreed upon. In the same manner, changes to the underlying methods used by each object need not affect the way two objects communicate.

PLAYER AVATAR

FITT provides a means for representing participants within the shared virtual world as avatars. The avatar model used for a particular player is configurable through the testbed menu interface to allow for gender and ethnicity issues. A default gender and racially neutral avatar can also be loaded to represent a participant.

Sensor Configuration

The 3D-avatar model incorporates 45 degree of freedom beads allowing for realistic deflection of limbs and torso. Setting the proper rotation angles for these degree of freedom beads is the responsibility of the ICS.

Each team member is suited up in 6 position sensors used to determine body position, view, and locomotion. (See Figure *3*). The sensor mounted on the HMD determines gaze direction. Body direction is measured by the position of a sensor mounted on a lightweight wooden backpack worn by each subject. This sensor determines which direction the subject's torso is facing, and which direction he or she will walk when moving forward.

Locomotion Method

The ankle trackers are used for locomotion through the virtual environment. In order for the subject to move around the virtual world, a stepping motion is made similar to everyday walking; walking in place. Raising and lowering the feet provides a smooth gait through the virtual environment. The software object responsible for ambulating the participant watches the step height by comparing the height of the ankle trackers relative to the ground. When the tracker crosses a software defined threshold, a step is initiated and the height of the step is translated into pitch rotations for the hip and knee joints, allowing the avatar to raise and lower its leg as the player walks. Means have also been included in the locomotion model for backing up as well as forward movement. This movement method is a refinement of a previous method developed at the Lab (Singer, Ehrlich, and Allen, 1998).

Articulation

The right arm of each player is tracked in order to articulate the arm of the virtual avatar. This requires two sensors working in conjunction with the back sensor. The position of the player's shoulder joint is determined by offset from the back tracker. This phantom shoulder position is used with a sensor strapped to the elbow to derive a vector for the upper arm. An additional sensor mounted on the input device and held within the hand of the player provides a way of deriving the vector for the lower arm position.

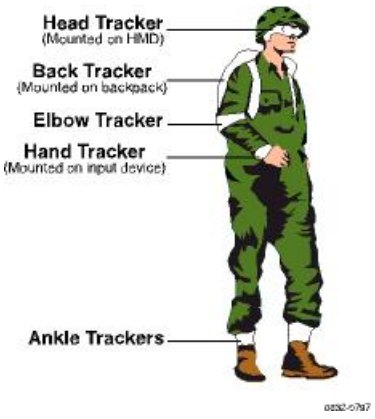

Figure 3
Sensor Configuration

NETWORKING ISSUES

The FITT Testbed is a networked simulation environment that uses the Distributed Interactive Simulation (DIS) protocol to pass messages back and forth. A subset of DIS protocol version 2.0.4 has been implemented for this project. Fully articulated human figures are not explicitly supported by the DIS protocol. For FITT,

body and limb position are transmitted via articulated parts structures attached to the Entity State PDU. The following DIS PDUs are used during the application:

1) Entity State PDU This PDU is the primary PDU used for the player avatars and the computer generated opposition forces. The articulated parts array is used to broadcast the position and posture of the avatar. A total of 47 articulated part parameters are available for modification during the simulation.
2) Fire PDU – Detonate PDU These two PDUs are used for the paint marker tool and the Gas Cylinder de-activator tool, as well as for the hand guns available to team members.
3) Action Request PDU Door opening and closing events are broadcast using these PDUs.
4) Start PDU – Stop PDU Used by the menu software for starting and stopping the simulation.
5) Collision PDU Used by the Data capture hardware to log collisions for after action analysis.

One important distinction that must be mentioned about the FITT software is that the DIS PDU traffic is handled by an Entity Services application that runs as its own process on each network machine. All message queueing and dead reckoning for all PDUs are encapsulated within this Entity Service running on the host machine. Applications wishing to use the Entity Service log on as a client.

Entity Server Implementation

In order to enable multiple players in the experiment, a method for network communication was needed. During the years from 1992 to 1995, IST performed work for U.S. Army STRICOM to build a networked dynamic terrain testbed. As part of that work, a package known as the Entity Service was created to facilitate DIS communication. The same package was used for this experiment.

The Entity Service was built with the concept of a possible unifying semantic called the "shared environment". The Shared Environment concept represents a flexible, highly configurable, representation of the state of the environment built upon well-defined semantics, strong abstractions and explicit decoupling of components (Lisle, Altman, Kilby, Sartor, 1994). A key point of this concept is the segregation of responsibility between the components that simulate the environment and the network protocol in use. The clients of the Entity Service no longer are concerned with the details of communication protocols, implementations or hardware. They communicate at a higher level of abstraction by message passing with the shared environment.

One final word about our choice of the Entity State PDU as the primary avatar state vehicle: At the start of this experiment, an evaluation of the human figure problem was made. The graphical representation of the human figure, in contrast to a vehicle, requires a relatively large number of articulated parts. For example, the current system supports fourteen different joints of the human body each with three rotations. In general, there are two methods that can be used for transmitting human figures within DIS. An Entity State PDU with all the articulated parts can be sent, or a Data PDU can be used to transmit the human articulated parts. Technically, the DIS standard does not support human figure articulated parts in its articulated parts enumeration (the enumeration focuses more on the parts of vehicles such as a turret). However, we chose to use Entity State PDUs because it was felt that articulated part information really should go along with all the other entity information even if the DIS standard did not include human figures in its enumerations (some arbitrary articulated part types were used).

Simulation Startup From Browser

Multiple startup problems inherent in a networked simulation are nothing new to the distributed simulation community. We believe, however, that our solution may be. For our immersive team training simulation we use 6 computers, all of which have their own unique startup sequences depending on which options the user wishes to run. To deal with experiment startup complications, we developed a multi-layered menu program in Java to select the various options available and launch the simulation from one terminal.

The menu allows the user to control any part of the simulation from one terminal. The menu has options to place the machines into or take them out of MCO (Multi Channel Option) mode (allows for stereo view). It can be used to set up information for data capture, start/stop a simulation, process data captured, playback a previously run simulation, email the collected data, and to bring the systems off the network as a restricted subnet for maintaining a distinct and stable experimental environment.

The menu program is a client/server combination. The server is a Java application that can be run on any one of the computers used for the simulation. The server receives commands and options from the client interface. Then, depending on the options, the server uses the Java System.exec() method to remotely start up the parameters of the simulation needed on the various machines. After executing the commands the server updates the state information of the client.

Originally, the client interface was designed as an applet which would run through an Internet browser. It was changed to an application to get around the security restrictions imposed by applets and browsers. Specifically, the browser would not allow an applet to communicate with an application unless the application was running on the same host that the applet was loaded from. Thus our options were to either set up the machine running the application as a web server, or to write the client as an application as well. We chose to implement the client as an application, but hope that the improved java security model available in JDK 1.2 set for FCS this year, will ease some of these restrictions in future browser releases.

Due to the reboot process incurred while taking the machines on and off the labs general development network, the menu needed to be robust. The client and server are both able to survive if the other is killed. If the client is killed the server simply waits for a new client to connect and then updates the new client with the current state of the simulation. If the server is killed (due to system reboot while taking the simulation off the network), the client will display a "Waiting for server message" and attempt to reconnect to the server. Once the server is running again, it will connect to the machines involved and determine the current state of the simulation and then wait for the client to reconnect.

PLAYBACK/AFTER ACTION CRITIQUE

FITT allows for the recording and playback of audio communication as well as player activity for all missions undertaken on the testbed.

Data Capture Coherency

Data recording is accomplished by capturing the PDU traffic from all participating machines (both player machines and the CGES server) into a terse binary file. The data capture program runs on an SGI Indy workstation attached to the experiment network. The start of an experiment is signaled by a Start/Resume PDU, and the end is signaled by a Stop/Freeze PDU. These are sent by the experimenter via the Stealth machine’s keyboard, to all machines at the beginning of the experimental trial. All PDUs are time stamped by the sending machine and all machines are time slaved to one designated master time server during each experimental run. This assures that all data can be reconstructed in the proper sequence during playback.

Audio For Radio Com And Environment

Audio Communication is an essential ingredient of FITT. Figure 4 shows the audio layout for the implementation of the mission described in section 2.

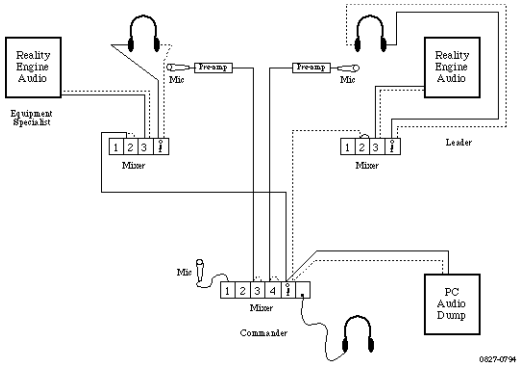

Figure 4
Audio Diagram

The audio for each player must integrate environmental sounds such as gun shots, door openings and closings, collision sounds, etc. as well as communication with the other two members of the team. Microphones at the Leader and Specialist positions feed their signals to the master mixing console, where the Commanders microphone signal is included in the mix. Note that the Commander can talk with each of the other team members separately in private if the need arises. The signal from the master mixer is then sent back to each player where a small mixer integrates it with environmental sounds generated from audio files on the ICS machines. These final signal mixes are then delivered to the headsets of the HMDs.

Audio Capture and playback is performed by a separate machine, a Dell Optiplex 560 PC running Windows 95. The PC's sound hardware (a Sound Blaster AWE64) is connected to an intercom system that allows both players and the experimenter to talk to each other as if communicating via radio. The PC is also attached via serial cable to both the SGI Onyx running the playback system, and the SGI Indy running the data capture program. During data capture, the PC receives its recording commands from the Indy. The Indy first instructs the PC to ready itself for audio capture and informs it of the current trial's unique ID number. Next, it instructs the PC to begin audio capture. At this point the PC begins "listening" on its audio input port for audio communication. When the audio level rises above a preset threshold, the PC saves the current time (from its internal multimedia timer) and begins recording audio into a .wav file. The threshold code acts as a software VOX circuit, and reduces the size of the .wav files when no communication is occurring. After the audio level falls below the threshold for a fixed amount of time, recording stops. The PC then calculates the length of the audio clip and saves the ending time of the clip. It then returns to "listening" mode. Finally, the SGI Indy informs the PC when to stop recording audio. All audio clips with their associated time stamps are then saved into a file in the same directory as the .wav files. This allows them to be recalled later by the playback system.

After Action Viewer Application

The Playback/After Action Review system is designed to allow experimenters and subjects to view a complete audio/visual computer-generated reenactment of an experimental trial. Features of this system include:

- The ability to view any instant of the experiment from any viewpoint
- The ability to advance or rewind the reenactment at any desired rate of speed
- The accurate representation of environment changes (doors opening, paint markers on walls, etc.) both during normal playback and fast/slow motion
- The recreation of audio cues (e.g door sounds) and player audio communications (radio com) synchronized to the visual playback

The Playback system runs on either of the SGI Onyx RealityEngine2 machines used as ICS stations during the mission. The software is built from the same libraries used to develop the experiment. In effect, it acts as a stealth station with a variable

viewpoint, but reads PDU's out of a data capture file rather than fresh off the network. In addition, the same PC used to record the subjects' audio communication is used to play back this audio, allowing the experimenters and subjects to both hear as well as see how the mission progressed. The control panel for the viewer has familiar VCR-style controls allowing play, stop, rewind, and fast-forward at the click of a mouse. There is also a shuttle dial that allows for variable speed playback, both slower and faster than normal. The panel provides a real-time clock and a PDU counter to indicate where in the recorded mission the playback system is currently pointing.

The playback system begins by first reading in the requested PDU file, sorting the PDU's chronologically (by their saved time stamp), and pre-loading the appropriate body and environment models. It then establishes communications with the audio playback PC, instructing it to prepare to play the appropriate trial's audio data. When the play button is pressed on the Playback control panel, the PC is sent a "play" command with the current time index (in milliseconds). The PC then computes which .wav file to play based on this time index. If the time happens to fall in between two .wav files in its list, it waits until the correct time to play. If the time is somewhere within a .wav file, it begins playback from an appropriate offset into the file. Audio is only played while the system is in real-time playback mode; therefore, if any control other than "play" is selected on the control panel, the audio PC is instructed by the SGI to stop playback.

The simulation time is computed by the SGI running the visual portion of the playback system. If the playback system is in "play" mode, the system's real-time clock is used to update the simulation time each frame. This provides an accurate representation of the passage of time. If it is in a fast-wind or slow-motion mode, the time is arbitrarily incremented or decremented. Each frame, the SGI reads all PDUs from the file that happened prior to (or if rewinding, subsequent to) the current simulation time. As each PDU is read, the appropriate entities are updated, the appropriate weapons fired, and the appropriate doors opened or closed. After all necessary PDUs are processed, the next frame is drawn and the cycle repeated. Originally, a method was to be implemented that would continuously synchronize the graphics data to the audio, because the audio capture PC is not time slaved to the primary simulation machines. Preliminary testing demonstrated that this was not necessary, however, as both the PC's and the SGI's internal clocks were independently accurate enough to control the timing of their respective playbacks. That is, no discrepancy in the audio and visual data was perceived over the length of any trial's playback.

7. Opposition Forces – CGF

In addition to maintaining coherency of dynamic objects within the shared world, the CGES application also serves OPFOR positions and behaviors. OPFORs are present to keep the mission participants alert and coordinated, for any room or hallway could possibly contain one or more of these characters. The OPFORs can be programmed with various levels of hostility and ability, ranging from an armed military officer with pinpoint firing accuracy and a high degree of visual acuity, to an unarmed innocent bystander that should not be neutralized upon sighting. The OPFORs can also be set to stand in place and guard an area, or can be given a sentry path to follow and search for intruders (i.e., the participants). Every OPFOR reacts to being shot by the players' tranquilizer pistols by slumping to the ground, and not moving for the remainder of the simulation. Configurable parameters for OPFORs include range of peripheral vision, response time after sighting a mission participant, sentry path, and accuracy with a weapon.

Conclusions

The first experiment currently being conducted with the FITT system is examining instructional strategies for team training in VE. Trainee teams study the mission procedures training manual, and then practice the missions in VE. During this practice, the trainees are given guidance either 1) before each mission with a demonstration, 2) during each mission with coaching, 3) after each mission with after action critique available from the playback application, or 4) not at all. Performance measures include: speed and accuracy of search, communications, and security procedures; and self-ratings of simulator sickness. The objective of the experiment is to determine how and when to give guidance when training in a VE.

Preliminary testing with FITT indicates that participants with no previous experience with VE can rapidly adapt to the FITT interface for walking and for selecting and operating equipment. Initial team performance is related to previous experience with relevant real-world procedures and team performance improves with practice across subsequent missions.

We look forward to extending our network for team training to support a planned exercise between team members in Houston, Orlando, and Montreal later this year.

9. References

Parsons, Kimberly Abel, Russell, David, A Research Testbed for Virtual Environment Training Applications for Dismounted Soldiers, IMAGE 1998 (In Press)

Singer, M.J., Ehrlich, E.A., & Allen, R.C. (1998) Effect of a Body Model on Performance in a Virtual Environment Search Task (Technical Report), U.S. Army Research Institute for the Behavioral and Social Sciences, Alexandria, Va.

Lisle, Altman, Kilby, Sartor. (1994) Architectures For Dynamic Terrain And Dynamic Environments In Distributed Interactive Simulation, 10th DIS Workshop, Orlando, Fl. March 1994