

An Application of Shared Virtual Reality to Situational Training

Sharon Stansfield
Nadine Miner

Sandia National Laboratories
Albuquerque, NM 87185

Dan Shawver
Dave Rogers

University of New Mexico
Albuquerque, NM 87106

Abstract

This paper presents current research being undertaken at Sandia National Laboratories to develop a distributed, shared virtual reality simulation system. The architecture of the system is presented within the framework of an initial application: situational training of inspectors and escorts under programs to verify compliance with nuclear non-proliferation treaties.

1 Introduction

A primary goal of the research being carried out by our group is the development of a virtual reality (VR) platform to support generic applications in situational training. In situational training, a student is taught to handle a variety of different situations, or scenarios. These scenarios may vary from routine events that the student will encounter constantly, to emergency situations that may arise only rarely in the actual job. Toward this end, we are currently carrying out research in several areas of multi-participant, distributed virtual reality and intelligent simulation. Our main emphasis is on applications that require "close quarters" situational training. Close quarters situations involve personnel and action at the individual or team level. Participant's perceive and interact with each other as people, rather than as machines, such as aircraft, or logical groupings, such as battalions. Therefore, it is critical that participants be visible to one another as complete human figures. Also important is the granularity of their movements. For example, participants might need not only to see one another, but also to determine what is being done with arms, hands, or heads. Therefore a VR platform for situational training must provide fully articulated graphical representations of each participant. It must also allow interactions with both the simulation and with other participants which are of a finer detail and greater complexity than do current simulations. SIMNET, for example, allows only gross actions such as maneuvering a vehicle through the environment or pointing and firing a weapon.

The remainder of this paper presents our preliminary system architecture and implementation using a current project in non-proliferation training as an example application. In this application, an instructor and a trainee share a common virtual environment.

The system automatically monitors participant's actions and reports when a security infraction, which should have been prevented by the trainee, has occurred. The system allows participants to manipulate objects in the environment, with actions such as grasping, lifting, placing, and dropping. Object behaviors, such as falling, are then invoked.

Related work includes SIMNET [8] and NPSNET [10], both of which are distributed, heterogeneous simulation systems for large-scale battlefield training; the extensive body of work related to the development and use of vehicle simulators, especially flight simulators [7], and NASA's work in using VR for training astronauts to handle repair of the Hubble space telescope [5].

2 Application Description

Nuclear non-proliferation treaties often provide for facility inspection to allow verification of compliance. For each participating nation, this requires two types of personnel, facility inspectors and personnel to escort these inspectors. U. S. inspectors must be trained to inspect foreign facilities to determine whether the hosting nation is complying with the terms of the treaty. Foreign inspectors do the same for U. S. facilities subject to such treaties. The time which an inspector is allowed within any foreign facility is limited and, therefore, training the inspector to become familiar with the facility ahead of time would make him or her more effective. The responsibilities of the escort are twofold. U. S. escorts of foreign inspectors (inspecting U. S. sites) must be familiar with the facilities in order to comply with treaty regulations concerning the foreign inspector's right of access. In addition, they must prevent any attempts at espionage by the foreign inspector as the inspection proceeds.

We are exploring virtual reality as a training tool for both inspectors and escorts. To address the problem of limited access to foreign sites, virtual models of foreign facilities can be created to allow an inspector to become familiar with the site before the actual inspection. Availability of information concerning foreign sites is the primary limiting factor.

Access to U. S. sites by escort/trainees is also limited for many reasons. These facilities are geographically scattered, they are secure facilities, and they are

often hazardous environments, due to the operations carried out within them. Virtual models of these U. S. sites can be used to familiarize the escort with a facility before the arrival of an inspector. This is especially important given the fact that short notification of an upcoming inspection is possible.

Both of the above applications rely heavily on the use of virtual environments for architectural walk-through – allowing a participant to visualize a facility without being physically present in that facility. The training of escorts, however, provides an additional problem domain for the application of VR. The escorts must not only be familiar with the facility, but they must also be trained to detect and prevent security infractions by the foreign inspector. It is this component of the training system on which we are now concentrating.

The escort training system is being developed concurrently with the generic VR platform for situational training applications. This application provides a test example during the implementation and verification of various platform components and provides a set of “real world” guidelines as to what features might or might not be useful in such a platform. Below, we describe the components of the platform in more detail.

3 The VR Platform for Situational Training

3.1 Geometric Modeling

The virtual environment (VE) used for the escort training work is a model of the hot cell laboratory, which is part of Sandia National Laboratories’ Reactor Engineering Center. The facility consists of two laboratories, one containing *glove boxes* and the other containing shielded *hot cells*. Glove boxes and hot cells allow experimenters to handle radioactive materials safely. Figure 1 shows the hot cell VE. This environment was created in three stages. First, the basic architecture was modeled from facility blueprints using Alias DesignerTM. A visit to the facility was then conducted and a video was made providing comprehensive coverage of the areas being modeled. In stage two, this video was used to model and place other structural details, such as overhead piping, as well as furnishings, such as instrument racks, lab tables, etc. In stage three, the same video was used to create texture maps for walls, floors, signs, instrument dials, and so on. These textures were taken from the digitized video, manually processed using a paint program, and then applied to the hot cell model. Other objects relating to the application were also modeled and placed in the virtual environment, including books, cups, and other everyday objects found in labs, as well as *shrouds* which cover classified items, and unclassified representations of such items.

Modeling of the environment is an independent component of the system. As each virtual environment is built, its model components are added to a reusable object library. This allows us to reuse objects and object components in building other VEs. Architectural items which can be reused are stored in a separate architectural library. The VR platform

can import geometric models in several different “standard” file formats, such as DXF. A more comprehensive internal format is used that includes texturing and other surface properties, as well as more complex model definitions.

3.2 VR Hardware

A participant may interact with the virtual environment in one of two modes. *Active* participants are entities present in and known to the simulation. They have an associated *avatar*, or graphical body, whose movements are slaved to their own. Active participants interact with the simulation and may cause changes in the state of the virtual world by carrying out actions such as handling objects. Currently, VR gear for active participants consists of a Virtual Research EyeGen3TM headmounted display for immersive viewing, four Polhemus FASTRAKTM magnetic trackers for position and posture tracking, and hand-mounted switches for indicating locomotion and the state of the hand (opened or closed.) In the escort training application, there are two active participants: the escort/trainee and the inspector/adversary. *Transparent* participants may view and move through the virtual environment, but they are not visible to the simulation or active participants. A transparent participant might be a visitor watching the training session, or an instructor controlling the scenario from “behind the scenes.” We currently provide transparent participants with an immersive viewing capability via the Fakespace BOOM3CTM which is a full-color, stereo viewer with mechanical tracking of the head and buttons to control motion through the virtual environment. A flatscreen monitor and mouse may also be used by the transparent participant. Figure 2 shows two active participants suited up in the VR gear. The platform also provides audio, in the form of simulation-related sounds and vocal feedback of simulation status, using the sound capabilities of the Silicon Graphics, Inc. (SGI) IndigoTM. We refer to the system component which updates and renders a participant’s view, including independent changes in world objects and other participants, as the *VR Station*. We currently use SGI platforms with RealityEngineTM graphics to drive our VR Stations. This allows us to utilize texture mapping to increase the realism of our virtual environments.

3.3 Representing Participants Within the Virtual Environment

We call the graphical representation of an active participant his or her avatar. Avatars are modeled and controlled using a version of the *Jack*[®] software developed by the University of Pennsylvania [2]. The *Jack*[®] software is a constraint-based graphical human figure controller originally developed for ergonomic analysis. It provides a complete and highly articulated graphical human model. The human figure software used for this work is a version of the *Jack*[®] software that separates the underlying human simulation and modeling component from the general-purpose display and control. This separation allows us to import the human figure model into our software environment

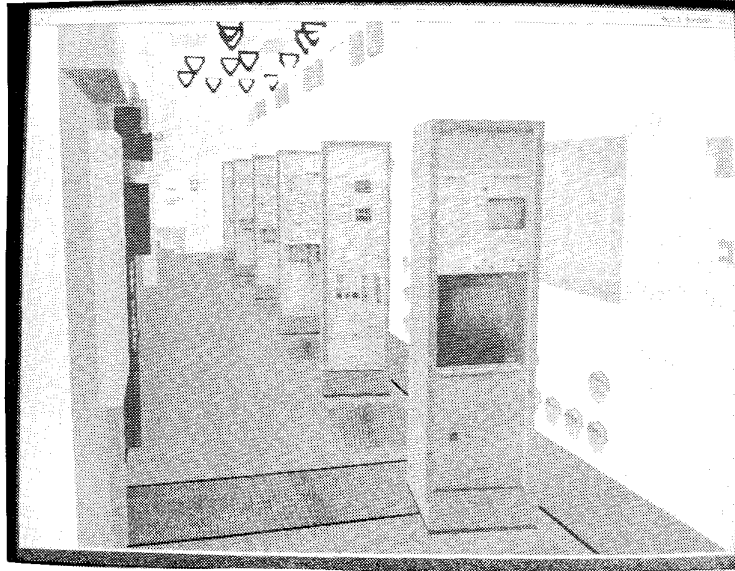


Figure 1: Hot cell lab virtual environment.

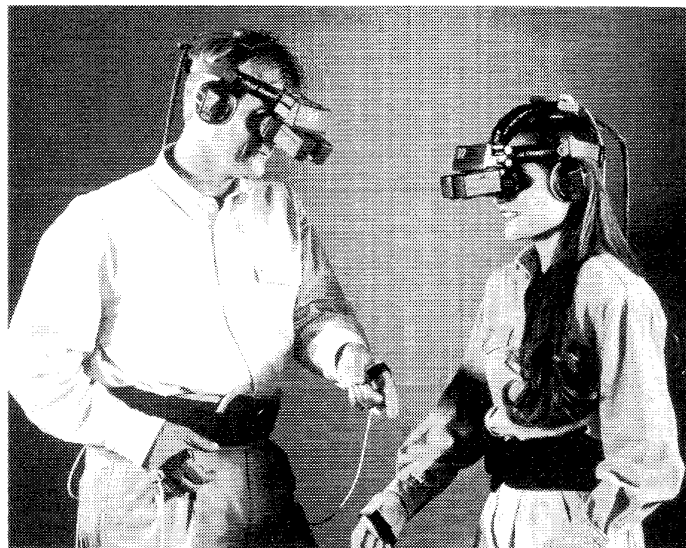


Figure 2: VR gear for active participants.

and to drive this figure using the *Jack*[®] controller as a server, residing on a separate processor. The posture and position of an avatar is controlled using the input from four Polhemus trackers mounted on the participant's head, each hand, and lower back. As a participant moves, changes in position are measured by the trackers. The *Jack*[®] server then generates the transforms for the avatar's new posture [1]. Figure 3 shows an avatar within the hot cell virtual environment. The escort training task requires two *Jack*[®] servers, one for the escort/trainee and one for the inspector/adversary.

3.4 Object Behaviors and Simulation Intelligence

Simulation and system intelligence within the generic VR platform may be distributed across any number of heterogeneous processes. The most important of these processes is the *world engine*. At a minimum, the world engine is responsible for consistency and coordination of object interactions, for updating the state of the virtual world at each simulation time-step and communicating these changes to other processes, and for coordinating execution of other simulation components. The architecture of the generic VR platform allows the world engine to be any software module capable of carrying out the indicated tasks. We are currently implementing a world engine using the BE Software Co.'s Behavior Engine[™] (BE) software [4]. The BE is an extended, object-oriented software platform that allows behaviors to be defined, stored, and instantiated in the same way as objects in languages such as C++. Our intention is to build a library of reusable behaviors within the BE which may be applied to any application domain. These behaviors include, for example, *grasped* (the behavior of an object when held by a manipulator) and *falling* (the behavior of an object when not supported.) In addition, the BE is used to create application-specific behaviors. One example, in the escort training application, is the *security violation* behavior, which is attached to all classified objects. This behavior logs illegal contact with the object for end-of-session performance review, and vocally informs the participants of the infraction. In our current implementation, the BE-based world engine is also responsible for collision detection and interpretation of some incoming sensor data.

The world engine drives the actions and events within the virtual world. The intelligence required to reason about these actions and events and to invoke the proper responses may reside wholly within the world engine, or it may be distributed among any number of other modules, with the world engine serving as command and control. Within the escort training system, we have chosen to add an additional reasoning module to the system. This reasoner is written in CLIPS (C Language Integrated Production System), a rule-based expert system shell [3]. When events, such as collisions between objects, cause the state of the virtual world to change, the world engine updates its state information and communicates the new states to the reasoner. The reasoner uses this new information to determine the consequences of these

events and communicates the resultant state changes back to the world engine, along with requests for execution of any associated behaviors. The world engine updates its own states, carries out the requested behaviors, and communicates the results to all other simulation modules.

Take, for example, the action of a participant grasping an object: the world engine reports to the reasoner that the hand is in contact with the object and that the hand is closed. The reasoner checks for conditions such as the object being liftable and not being held by another participant, etc. It then determines the new state of the hand and object and communicates this back to the world engine along with a request to invoke the grasp behavior (essentially affixing the hand and object.) The world engine also determines the new position of the object as the hand moves and communicates this to all VR Stations.

3.5 Communication Within the Distributed System

Figure 4 shows the configuration of the training platform. The platform components are distributed across multiple processors and computers. Many of these components, such as the VR Station and the *Jack*[®] servers, may have multiple instantiations for a particular application. For example, the escort training application has a VR Station and a *Jack*[®] server for each of the two participants, along with the world engine and reasoner, for a total of six processes. It may also have one or more transparent viewer VR Stations.

Each instance of a simulation component (eg. world engine and *Jack*[®] servers) must cooperate with all other components to "cover" its own part of the simulated environment for output to the VR Station. This is accomplished by having an indexed space defined by the world description (currently loaded from a common file by each process at start-up.) Each simulation component is given exclusive use of a portion of the index space. This is done by providing, in a standard order, a common set of shared, unique names for transforms which describe the position of an object in the world. Ethernet multicasting of datagram packets is used to communicate these transforms, in the shared index order, to the VR Stations. This index scheme allows all instances of the VR Station to display a local view of the same virtual world. In addition, it allows a process to communicate only those transforms which have changed since its last communicated update. This minimizes network traffic and processing by any processes receiving this information.

Similarly, sensors are grouped into logical sources, one for each active participant. Each such source has a standard set of potential sensors, with a standard naming order scheme, providing each source with its own index space for multicasting. A simulation process receives sensor input from the source or sources providing its required data (i.e. the *Jack*[®] server controlling avatar1 receives data from the position trackers attached to participant1.)



Figure 3: Avatar in hot cell VE.

4 Summary and Future Work

This paper has presented preliminary work in developing a generic VR platform for situational training applications. The primary goals of this work are development of a networked VR system that will handle close quarters situational training and the integration of intelligence and behavior-oriented design into the simulation process. We have illustrated the components of this system through an example application: a system to train escorts of foreign inspectors under non-proliferation treaty compliance. This application allows an instructor and a trainee to share a common virtual environment. The system automatically monitors the actions of the participants and reports when a security infraction, which should have been prevented by the trainee, has occurred. The system allows participants to manipulate objects in the environment, for example grasping, lifting, placing, and dropping a cup or book. The proper object behaviors, such as falling, are then invoked. This current implementation is both primitive and limited. It provides enough utility to demonstrate the use of VR for close quarters situational training and to show how it would be applied to a real world application, but it is by no means complete. The number of states and objects is limited, as is the simulation intelligence; behaviors of objects are primitive (eg. no acceleration of a falling object due to gravity); and the simulation cycle and event handling are simple. Developing both the capability of the platform and the complexity of the escort training application which utilizes it is our primary research agenda for the future. In addition, we will be exploring other applications for the platform, such as the

training of battlefield medics on the synthetic battlefield and the redesign of an earlier training system for robot operators which uses SILMA's CimStationTM platform as the world engine [6]. A related goal is the integration of a hypermedia component to allow trainees access to relevant information, such as texts and videos, while using the VR platform for training [9].

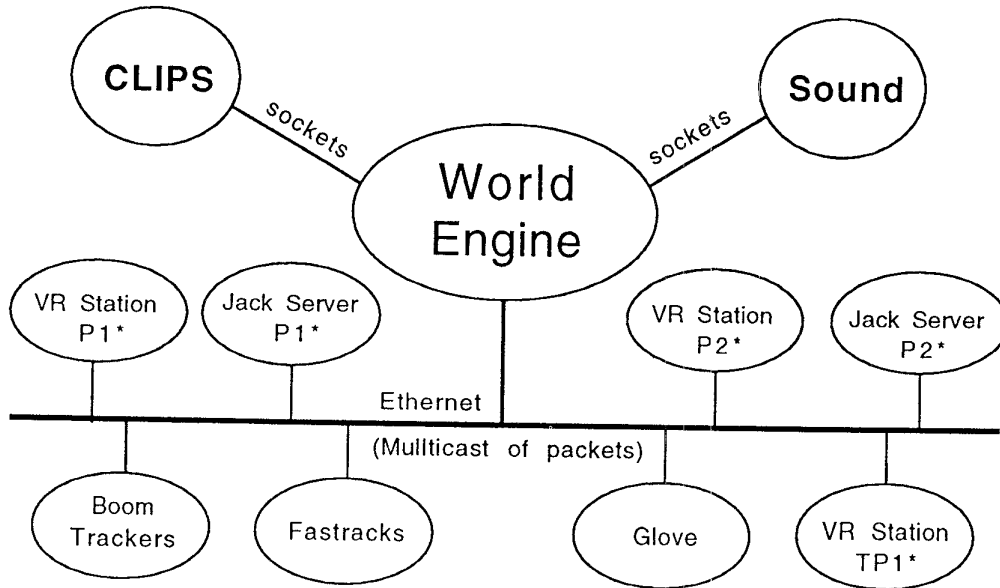
Acknowledgements

This work was performed at Sandia National Laboratories and was supported by the U.S. Department of Energy under Contract DE-AC04-94AL85000.

We are pleased to acknowledge the valuable contributions made to this project by the following people: Jim Pinkerton, Lydia Tapia, Meisha Collins and Sofia Pastoriza-Nunez (Sandia National Laboratories); Norman Badler and Mike Hollick (University of Pennsylvania); Chris Goad (BE Software Co.)

References

- [1] "Real-Time Control of a Virtual Human Using Minimal Sensors," N.I. Badler, M.J. Hollick, and J.P. Granieri, *Presence*, Vol. 2, No. 1, Winter 1993.
- [2] "Simulating Humans: Computer Graphics, Animation and Control," N.I. Badler, C.B. Phillips, and B.L. Webber, Oxford University Press, 1993.
- [3] "CLIPS Reference Manual, Version 6.0" *Technical Report*, Number JSC-25012, Software Technology Branch, Lyndon B. Johnson Space Center, Houston, TX, 1994.



* P1 = Active Participant 1, * P2 = Active Participant
 2* TP1 = Transparent Participant 1

Figure 4: Configuration of the VR platform.

- [4] "The Behavior Engine and BEF: A Technical Overview," C. Goad, *Technical Report*, BE Software Co., 1631 NW Johnson St., Portland, OR, 1994.
- [5] "Virtual Environments in Training: NASA's Hubble Space Telescope Mission," R. Bowen Loftin and Patrick J. Kenney, Robin Benedetti, Chris Culbert, Mark Engelberg, Robert Jones, Paige Lucas, Mason Menniger, John Muratore, Lac Nguyen, Tim Saito, Robert T. Savely, and Mark Voss, to appear in *Proceedings of the 16th Interservice/Industry Training Systems and Education Conference*, Orlando, FLA, 1994.
- [6] "An Interactive Virtual Reality Simulation System for Robot Control and Operator Training," N. E. Miner and S. A. Stansfield, *Proceedings of the IEEE Robotics and Automation Conference*, San Diego, CA, May, 1994.
- [7] "A Literature Survey for Virtual Environments: Military Flight Simulator Visual Systems and Simulator Sickness," R. Pausch, T. Crea and M. Conway, *Presence*, Vol. 1, No. 3, Summer 1992.
- [8] "The SIMNET Network and Protocols," A. Pope, *BBN Report No. 7102*, BBN Systems and Technologies, Cambridge, MA, 1989.
- [9] "A Computer-based Training System Combining Virtual Reality and Multimedia," S. A. Stansfield, *Proceedings of the NASA Conference on Intelligent Computer-aided Training and Virtual Environment Technologies*, Houston, TX, May 5-7, 1993.
- [10] "The Software Required for the Computer Generation of Virtual Environments," M. J. Zyda, D. R. Pratt, J. S. Falby, C. Lombardo, and K. M. Kelleher, *Presence*, Vol. 2, No. 2, Spring 1994.