Sharon Stansfield
Ithaca College
Ithaca, NY 14850

Daniel Shawver
Annette Sobel
Monica Prasad
Lydia Tapia
Sandia National Laboratories
Albuquerque, NM 87185

Design and Implementation of a Virtual Reality System and Its Application to Training Medical First Responders

Abstract

This paper presents the design and implementation of a distributed virtual reality (VR) platform that was developed to support the training of multiple users who must perform complex tasks in which situation assessment and critical thinking are the primary components of success. The system is fully immersive and multimodal, and users are represented as tracked, full-body figures. The system supports the manipulation of virtual objects, allowing users to act upon the environment in a natural manner. The underlying intelligent simulation component creates an interactive, responsive world in which the consequences of such actions are presented within a realistic, time-critical scenario. The focus of this work has been on the training of medical emergency-response personnel. BioSimMER, an application of the system to training first responders to an act of bio-terrorism, has been implemented and is presented throughout the paper as a concrete example of how the underlying platform architecture supports complex training tasks. Finally, a preliminary field study was performed at the Texas Engineering Extension Service Fire Protection Training Division. The study focused on individual, rather than team, interaction with the system and was designed to gauge user acceptance of VR as a training tool. The results of this study are presented.

1 Introduction

The call comes in: a terrorist bombing at the airport, mass casualties require immediate medical response. But the terrorists have raised the stakes. They have also released a biological warfare agent that's capable of killing not only the bombing victims, but the emergency responders themselves. How do our medical first responders train for such an event? Even in mass casualty situations without the added complexity of a bio-warfare hazard, training beyond traditional, exposition-based courses is often limited, and—although "experience is the best teacher"—error and misjudgment on the part of a responder can have dire consequences in the field. In addition, medical crisis response requires more than an understanding of the clinical aspects of a patient's condition: it requires situation assessment and critical thinking that must take place in a chaotic and often dangerous environment, which is a much different scenario than a doctor's office, or even an emergency room in a well-equipped hospital.

It has been shown that experiential, problem-based learning offers a way to

Presence, *Vol. 9, No. 6, December 2000,* 524–556

teach situational problem-solving skills, and studies have suggested that this type of learning provides greater realism (Albanese & Mitchell, 1993) and promotes better knowledge retention (Norman, 1992) than do traditional forms of training. To date, however, the only widely applied form of experiential learning within the medical first response community has been live exercises. In a live exercise, also called a *moulage*, a mock disaster is staged and responders practice their skills by treating "patients" who have been made-up to represent those injuries that are common to the type of crisis that is the subject of the exercise. Such exercises are a valuable and successful form of training, and it is important to state up front that simulation-based systems are not yet at the point at which they can reproduce enough of the chaos and complexity of a medical crisis to be considered as a replacement for live exercises. Having said that, however, live exercises also have many limitations:

- They are expensive and logistically complex to stage, hence:
  - They are not held frequently (perhaps once every one to five years.)
  - It can be difficult, if not impossible, for small communities to participate or hold exercises of their own.
- Responders must travel to the place of the exercise. This is especially critical in small communities, where it can impact the availability of responders in the event of an actual emergency.
- Sustainment training is difficult, which results in knowledge decay between exercises, especially for those situations that responders do not encounter regularly in the field (such as the release of a bio-warfare agent.)
- While they reproduce much of the situation that might be encountered in an actual event, live exercises do not address all of the elements of such a situation. For example:
  - Moulage permits highly realistic mock casualties, but, because they are real people, responders cannot actually treat their injuries. Usually they tell an instructor what they would do, and the instructor provides feedback. Hence, the immediacy of seeing the response to one's actions (and errors) is diluted.
- Scenarios in which there is a hazard, such as the release of a bio-warfare agent, can create additional complexity as well as danger for the responders themselves. Components of such a situation simply cannot be reproduced in a live exercise. For example, although responders may wear actual protective gear during an exercise (thus teaching them how difficult it can be to maneuver in these suits), the consequences of accidentally tearing a hole in this gear—exposure to the agent and its impact on performance and health—cannot be reproduced.

In this paper, we present a virtual reality (VR) system that intends to address many of these issues by providing a simulation-based system in which first responders can experience scenarios such as an act of bio-terrorism, practice their decisions and actions, and understand the consequences. They can treat a patient and immediately see the results reflected in the state of the patient, and they can do this within a full scenario that includes environmental and situational aspects that could effect both their decision-making and its outcomes (good or bad.) In such a VR system, they can be exposed to a much broader range of scenarios and factors than they might ever experience in live exercises. And, technology development permitting, they may soon be able to do so more often, at a lower cost, and without having to travel far from the communities that need them.

The military has long used simulation-based experiential training systems to augment live exercises (for example, individual flight simulators or the SIMNET distributed battlefield simulation.) These military systems, however, have not focused on training tasks in which the individual must act directly and manually on the environment (rather than through the operation of a vehicle) nor in which the responses to an action may be subtle (such as a change in skin color) rather than explosive. It is under these latter circumstances that medical first responders must operate, and it is this problem that the research presented in this paper seeks to address. Doing so required that we design and implement several interaction and simulation methodologies, from

the implementation of distributed-system support for high-fidelity simulations to the development of clinically realistic virtual patients and, perhaps most importantly, to the creation of techniques that permit a user to act naturalistically upon the virtual environment.

The remainder of this paper presents this research and is structured as follows: section 2 discusses related work in the areas of simulation-based medical and distributed training; section 3 introduces the training task and presents the content of the BioSimMER scenario; section 4 presents the components and architecture of the VR platform; section 5 discusses in technical detail the underlying methods that were developed to permit naturalistic interaction with the virtual world; and section 6 describes the virtual human development and the virtual patient and avatar drivers. BioSimMER was field-evaluated at the Texas Engineering Extension Service Fire Protection Training Division, and the results are presented in section 7. Finally, in section 8, we summarize the work and discuss our goals for the future development of the system.

2 Related Work

Satava and Jones (1997) present a categorization of virtual environments in medical education and training that is analogous to that defined by the military's use of simulation-based training environments. Based upon application, they divide medical VEs into one of the following three categories: individual training, medical crisis training, and medical virtual prototyping. These categories correspond to, for example, flight simulators, distributed battlefield environments, and simulation-based acquisition models. Although the work presented in this paper most readily falls into the second category, the majority of medical applications of VR address the first. These task-specific individual medical trainers—which we also refer to as partial trainers—seek to train a single (or a limited) set of skills within a simulation that is highly realistic and anatomically correct. Kaufman and Bell (1997) discuss the potential of VR-based partial trainers for teaching and assessing task-specific clinical skills. The application of VR to the development of partial trainers for medical procedures is a promising area and is being addressed by researchers working in a number of diverse clinical specialties. The following citations are representative of this work.

- Delp et al. (1997) present a trainer for debridement of a gunshot wound to the leg. This trainer accurately models the effect of the wound on the anatomy of the leg, including the musculature and circulatory systems.
- Kuppersmith et al. (1997) present a VR-based simulator for temporal bone dissection.
- Robb (1997) and Wiet et al. (1997) present virtual endoscopy simulators.
- Gibson et al. (1996) present a trainer for arthroscopic knee surgery that integrates a volumetric graphical representation of the anatomy with a haptic device providing related force feedback.
- Dinsmore et al. (1997) have developed a simulation for training palpation of subsurface breast tumors, which incorporates a force feedback model of the haptic response of healthy and tumorous tissue.

Common to all such partial trainers is their focus on a specific task and anatomical region.

The use of VR-based trainers for structured task training is also being explored in other application areas. In addition to the long-standing use of partial trainers for training vehicle operation (aircraft, tanks, helicopters, and the like) within both the military and civilian sectors, researchers have begun to explore VR as a means to train other, fairly structured, tasks. Tate et al. (1997) explored VR-based training of shipboard fire fighting. Interaction was via joystick, and tasks consisted primarily of navigation and equipment location, although participants could also open and close virtual doors. Johnson et al. (1998) are developing a VR system for training equipment operation, with an emphasis on incorporating intelligent agents for tutoring and feedback. Interaction with the system is through a virtual control panel consisting of simple buttons and toggles (as with the actual equipment). More-complex processes were explored by Loftin and Kenney (1995)—who developed an immersive VR system to train the flight team on the procedure for repairing the Hubble Space Telescope—

and by Stansfield (1998), who describes a VR system for training teams in the disassembly of a subcomponent of a nuclear weapon. Each of these systems, while providing an experiential learning environment, is still focused primarily on the training of the specific steps that are involved in accomplishing a fairly structured procedure. Errors or variation from this procedure on the part of the trainee usually result in failure, halting of the procedure, and/or intervention by a human trainer or intelligent agent.

Training systems that focus on critical decision-making in unpredictable environments fall into the second category. Again, outside of the medical arena, the military has taken the lead in developing simulation-based experiential training systems. The SIMNET/DIS architecture (Calvin et al., 1993) for training and rehearsing battlefield operations has been in use now for many years. Closely related systems, such as the High-Level Architecture (Dahmann, Calvin, & Weatherby, 1999) and NPSNET (Macedonia et al., 1994), continue the research into large-scale, distributed mission training systems for warfighting. Bell (1999) surveys the various measures of effectiveness for these and similar systems, pointing to studies that show generally positive results in their application to training.

Historically, distributed mission training systems have not focused on the individual or team level, but rather on large, theatre-of-war operations. This has permitted the networking of large numbers of geographically distributed players, who interact within the shared simulation environment. The shared information, communicated in the form of DIS protocol packets, has been relatively simple, such as the locations and headings of adversaries within a user's field of interest. Although simulated humans have been incorporated into such environments (Reece & Kelly, 1996), their behaviors have likewise been simple, permitting only a few "canned" behaviors, such as standing, walking, and shooting. This simplicity has primarily been the result of limitations of the DIS protocol itself.

Much less has been achieved in the development of experiential medical-crisis training systems than in either medical partial trainers or distributed mission-training systems. Perhaps the greatest development focus in this area has been on the creation of interactive CD- and multimedia-based systems (such as that developed by Kizakevich et al. (1998) to teach and assess prehospital triage and stabilization skills). In this type of system, scenarios are represented by a set of images, discourse, and other traditional computer-based presentation methods. Users view the scenario on a monitor and interact with it via mouse and keyboard input. Such systems, although valuable teaching tools, tend to limit the actions that the user can take, and so do not provide a full experiential training environment. In addition, one might argue that they are several layers of abstraction away from the "learning by doing" experiences of live exercises and actual events.

Moving from multimedia to simulation-based trainers, Small et al. (1999) present an emergency medical trainer that's akin to a flight simulator: a customizable mannequin with realistic anatomical features represents the patient, whom users act upon using physical instruments that interface the mannequin to a computer control system that drives the appropriate clinical state and response. Such systems, although highly sophisticated, can be expensive and limited in their programmability. In addition, because the system does not provide an overall environment, a physical re-creation of an emergency room or disaster location would be required. Other work, including that presented here, takes the approach of creating the entire training scenario via software. Stytz et al. (1997) present work in the development of a virtual emergency room (VER). The VER is a distributed system that permits multiple users to train in scenarios that are relevant to level I and II emergency rooms (the equivalent of a Mobile Army Surgical Hospital (MASH) unit). A virtual patient dynamically displays changes in his physical condition, which is driven by a set of scripts. Users interface with the system via a combination of mouse-selecting and 3-D menus. A virtual control panel tethered to the user's view is employed to move through the environment and to view menu displays and status. Again, interaction with the scenario and other users is not naturalistic: the trainer interface is imposed on the experience. Chi et al. (1996) also present a simulation-based approach to emergency medical training. Again, a dynamic virtual patient pre-

sents its changing physical condition and responds to the medic/trainee. The user interacts with the system via a series of menus. In this case, however, both the virtual patient and the virtual medic are presented to the user in a third-person view. When the user specifies an action via menu-selected input, an animation is presented that shows the virtual medic performing the action on the virtual patient. The system uses PaT-Nets to model the state of the patient and its changes, both spontaneously over time and in response to input from the user. MediSim (Stansfield, Shawver, & Sobel, 1998) was a precursor to the BioSimMER system presented here, and it utilized this PaT-Net patient model in an immersive, interactive virtual environment that was aimed at providing a more natural, less intrusive simulation-based experience. In this case, the animated medic was replaced by a first-person avatar that was driven by sensors worn by the user.

Finally, current applications of VR to education—as represented by the work of Allison et al. (1997), Johnson et al. (1999), and Salzmann et al. (1999)—have provided some insight into the development of context-oriented conceptualization and critical reasoning skills within the new learning paradigm represented by VR. Although these projects focus on a very different kind of knowledge acquisition, the idea of a VR-based component to life-long learning, from grade school education through specialized training, is indeed exciting.

Figure 1. *Airport VE.*

Figure 2. *Airport VE with casualties.*

3 BioSimMER Application Description

3.1 The Scenario

The training scenario for BioSimMER is that terrorists have taken over a small airport and are holding hostages. They claim to have released a bio-warfare (BW) agent, but they haven't indicated what it is. After a standoff of several hours, law enforcement raids the airport. The terrorists set off a bomb, which causes conventional injuries and additional dispersion of the unknown agent. The trainees enter the scenario at the point just after the raid and explosion. Figure 1 shows part of the airport VE developed for this work, and figure 2 shows another view with casualties.

The biological agent released in this scenario is Staphylococcus Enterotoxin B (SEB), a biological toxin which is known to have been weaponized (Sidell, Patrick, & Dashiell, 1998.) The scenario provides for aerosol dissemination via the explosion of the terrorist bomb. To enhance the realism of the scenario, the dissemination of the SEB through the airport VE was modeled, and the results were used to determine casualty exposure rates and facility contamination levels (for future training of tasks such as the placement of sensors). The SEB dispersion was modeled using the

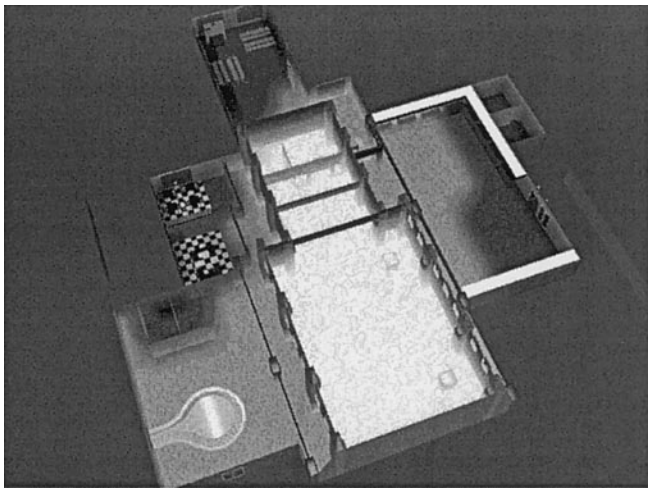

Figure 3. *Visualization of the dissemination of SEB through the airport VE.*

Figure 4. *Protective gear worn by emergency responders in contaminated environments.*

CONTAIN software, which accurately models the dispersal and disposition of aerosols within the interiors of facilities (Murata et al., 1997). Figure 3 shows an overhead view of one time step in the visualization of this dispersion.

3.2 The Training Tasks: Assessment and Stabilization

The BioSimMER training task is the field assessment and stabilization of casualties. First responders arriving at an event with multiple casualties usually first triage all patients following, for example, the START (Simple Triage and Rapid Treatment) triage system described in the *First Responder Chem-Bio Handbook* (Tempest Publishing, 1998.) START involves quickly evaluating respiration, circulation, and mental status of the patient. Based upon these evaluations, patients are placed into one of four categories: deceased, immediate care, delayed care, and minor or no injury. After all patients have been triaged, the medical treatment of individual patients can begin, with patients in the "immediate" category taking precedence. This treatment involves a more in-depth patient assessment (ABCDE: assessment of *A*irway, *B*reathing, *C*irculation, *D*isability or neurological, and *E*xpose/examine entire body for injury). Proper assessment and diagnosis should result in what is referred to as an *intervention,* a preliminary treatment with the goal of stabilizing the patient in preparation for transport to a hospital or some other, better-equipped facility. The content of BioSimMER's current training focuses on this stage of response and includes assessment procedures and interventions (such as inserting an IV, administering medication, and applying a dressing). BioSimMER does not address surgical procedures or other advanced medical treatments.

In the case of a suspected exposure to a chemical or biological agent, two additional tasks must be performed during initial triage: protective breathing apparatus must be applied, and the casualty must be decontaminated. In addition, the medical responders must wear protective gear to prevent their own exposure to the agent. This gear is cumbersome and difficult to work in, due in part to a decreased visual field from the helmet and a lack of tactile feedback caused by bulky gloves. (As an interesting aside, these also happen to be limitations of the current state of VR hardware. This usually negative feature of current VR gear is hence turned into a positive, because it replicates the real-world difficulties inherent in the application.) Figure 4 shows a photo of the actual protective gear worn by responders.

BioSimMER is meant to be an experiential learning

tool, augmenting live exercises and/or actual experience. It has no tutorial component or "expert guide." Trainees carry out their tasks within the scenario, the only feedback being the changing condition of the virtual patient. BioSimMER has a recording capability that stores the high-level actions taken by the trainee, along with a time stamp. Evaluation of performance and feedback from an instructor can therefore be done in a post-training session. It is assumed that trainees come into the system having received traditional training and having some familiarity with their tasks. BioSimMER then permits them to have frequent and varied experience with a range of potential scenarios, continuously practicing, honing their performance and critical thinking skills, and broadening their range of experience.

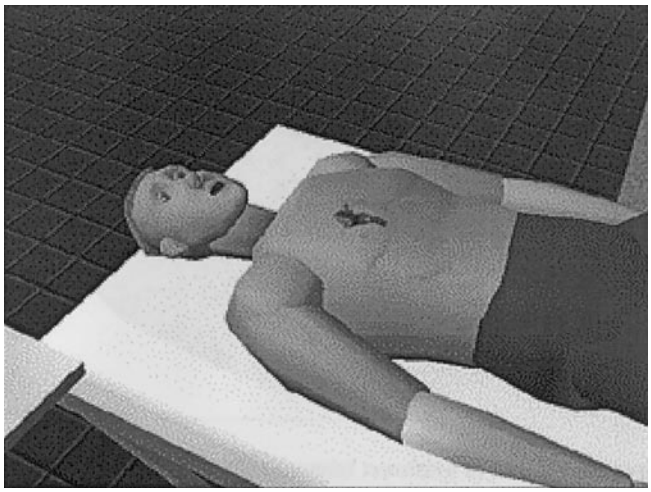

Figure 5. *Virtual patient with chest wound resulting in tension pneumothorax.*

3.3 Initial Injury Set

The BioSimMER system currently supports four injury types: SEB exposure, tension pneumothorax, cerebral contusion, and catatonia due to psychological shock. All of these injuries are consistent with our scenario of the terrorist act with an explosion and the release of a bio-warfare agent. We describe these injuries in more detail in the following sections, and we include some of the symptoms that a first responder would need to recognize during assessment, as well as the recommended intervention. All injury models are reusable and can be (and have been) utilized to build additional training scenarios.

3.3.1 Exposure to SEB via Inhalation. ***Description***: SEB is a biological toxin produced by the bacteria Staphylococcus. Although SEB causes illness at extremely low doses, a high dose is usually required to cause death. Symptoms usually begin to present three to twelve hours after the initial exposure.

Symptoms and assessment: Symptoms of SEB exposure include high fever, chills, headache, vomiting, and muscle aches/general discomfort.

Intervention: Only supportive care can be provided, such as starting an IV. A mask placed over the nose and mouth will help prevent further exposure.

3.3.2 Tension Pneumothorax. ***Description***: A tension pneumothorax results when an injury to the chest causes a hole in the pleura ("sacks" between the chest wall and the lung) which acts as a one-way valve, allowing air to enter the space between these pleura, but not allowing it to escape. As the pressure in this space increases, it begins to push on the lung. This results in respiratory distress and decreased flow of blood to the heart (and, ultimately, to the rest of the body). The tension pneumothorax modeled for this work is severe enough to result in death if not treated.

Symptoms and assessment: Among the symptoms that may be seen during assessment of tension pneumothorax are respiratory distress as reflected in the vital signs, breathing, and chest motion; a drop in blood pressure due to decreased blood flow; changes in skin color due to lack of oxygen (pink tone to red, blue, and finally to gray as the patient goes into shock); and loss of consciousness.

Intervention: An occlusive bandage is placed over the wound to prevent more air from entering the chest, and a needle aspiration is performed to remove air from the chest cavity and relieve the pressure on the lung, permitting the patient to breathe. Figure 5 shows the virtual patient with chest wound resulting in a tension pneumothorax.

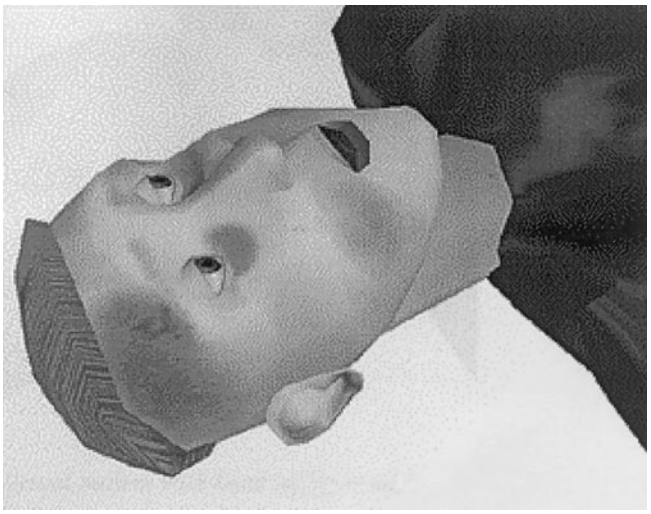

Figure 6. *Virtual patient with head injury resulting in brain contusion.*

3.3.3 Cerebral Contusion. *Description*: A cerebral contusion is caused by a head trauma resulting in damage to a portion of the brain. It may also be accompanied by brain swelling and bleeding resulting in the formation of clots. Figure 6 shows the virtual casualty with a head wound resulting in a cerebral contusion.

Symptoms and assessment: Assessment of a cerebral contusion should focus on neurological symptoms. Among those that may be seen during assessment of a cerebral contusion are a loss of consciousness, a confused or no verbal response to questions, an inability to perform voluntary movement (such as following a finger with the eyes, or moving a limb), and unilateral pupillary response to light. As the injury progresses, increasingly severe symptoms may arise, such as difficulty breathing, seizure, and coma.

Intervention: Important steps to be taken in the case of a brain contusion are administering a sedative to prevent seizure, and keeping the airway open and the patient breathing normally, as oxygen must reach the brain in order to prevent additional brain injury.

3.3.4 Psychological Trauma Resulting in a State of Catatonia. *Description*: Psychological trauma can present itself in varying degrees of hysteria or catatonia. A patient suffering from psychological trauma is consistent with the terrorist scenario and is an important component of emergency responder training: patients suffering from psychological shock may hinder rescue efforts or further injure themselves or others. In the case of a catatonic patient, the trauma may present symptoms that are consistent with actual injuries, such as head trauma or exposure to a toxic agent, even if the patient is uninjured. We have modeled a virtual patient in a state of catatonia for these reasons.

Symptoms and assessment: The patient shows no physical signs of injury but is not responsive to verbal questions; the patient does not display voluntary motor control (he or she does not move arms or legs when asked, and is unable to perform the occular motility test.) Involuntary pupillary response to light does occur.

Intervention: The patient does not require immediate medical action, which is a key point for responders to recognize.

3.4 Medic Actions Supported

BioSimMER currently supports a set of actions that are sufficient to perform the primary assessment and the desired interventions for the injury types modeled. As with the injuries, this set of actions is both reusable and extensible as further training tasks and injury cases are added to the system. Table 1 lists the current set of medic actions, and figure 7 shows the user (via her sensor-driven avatar) applying a sterile bandage to a virtual patient.

4 VR System Architecture and Components

Figure 8 diagrams the VR system architecture developed for this work. The system immerses multiple users in a shared virtual world that consists of themselves (represented graphically as avatars), the environment, virtual casualties, and other application-relevant virtual objects. Users are able to interact with the virtual world naturalistically, by handling and using objects and speaking to the simulated patient. The virtual world responds to the user's actions appropriately and changes state accordingly.

Table 1. *The set of actions the user may perform*

Action	Purpose
	Initial Exam
Ask patient what happened	Gather information; test coherence
Expose patient	Look for wounds
Request vitals	Blood pressure; respiratory rate; pulse rate; temperature
Squeeze fingertips	Check capillary refill rate
	Neurological Exam
Ask patient to move arms, legs	Check voluntary motor control
Test ocular motility	Voluntary motion of eyes
Test pupillary reflex	Involuntary pupil response to light
	Airway Intervention
Insert J-tube	Open airway
	Respiration Intervention
Perform needle aspiration	Relieve pressure on lung
	Circulatory Intervention
Start intravenous therapy (IV)	Increase, balance fluids
	Neuro-trauma Intervention
Apply cervical collar	Prevent further damage to spine
	Biological Exposure Intervention
Apply wipe to body	Decontamination
Place mask over nose, mouth	Breathing barrier against external toxins
	Wound Care
Apply sterile bandage	Protect wound
Apply occlusive bandage	Protect wound, prevent air intake
	Medications
Auto injection	Inject sedative (head wound) Inject atropine (agent exposure)

This section presents the architecture of this VR system. (The platform and many of the underlying components have been used to develop additional applications, such as that presented by Stansfield (1998).)

4.1 System Hardware

The VR hardware used in this study is commercially available, off-the-shelf equipment. A minimum-

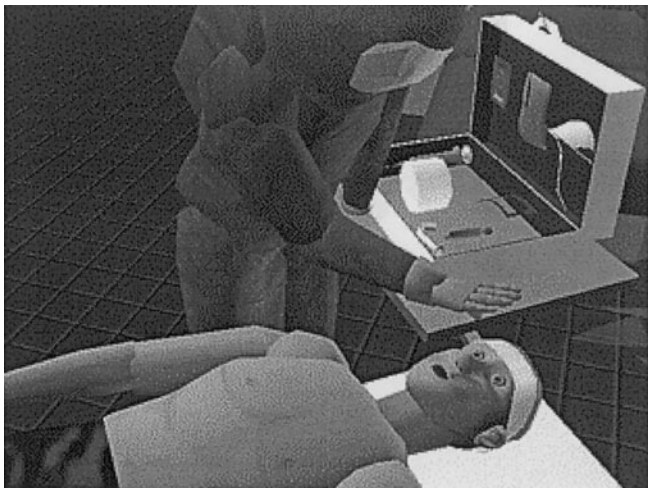

Figure 7. *User applies a sterile bandage to patient's head wound.*

computer configuration requires one graphics pipe or card for each user (view) desired. In general, we have also run with at least one processor per user. Additional processors to offload drivers and external simulation modules are desirable, but not required. The system is distributed and runs across a LAN, so the equipment must be networked. (Ethernet and multicast hardware and system software are used.) Our current configuration uses SGI workstations (Octanes with MXE graphics.) We are in the process of porting the VR display software (VR_Station) to a PC platform. When this is completed, the system will be capable of running on a PC LAN, which should considerably reduce the cost of the hardware platform, making it more accessible to end users.

The VR gear worn by each user currently consists of a head-mounted display (HMD); four position trackers worn on the head, the back of each hand, and the small of the back; a microphone; and (optionally) earphones. Our current configuration consists of Optics 1 PT_01 HMDs and/or Virtual IO i-glasses and Ascension magnetic position trackers with extended range capability. Sound input and output utilize the hardware available on the SGI workstations. The system is not VR device dependent. Figure 9 shows users wearing the VR gear that is used to support immersion within the BioSimMER scenarios.

4.2 Graphical Models

The virtual environment is represented by a hierarchical world model, which is loaded from a set of files that, taken together, specify the scene graph used by the display module. These files are explained below.

- The .hi file specifies the hierarchical relationships and the transformations used to position geometry. The two types of transforms within the model are *static* transforms (these remain unchanged during a simulation) and *dynamic* transforms (which may be changed during a session, usually by multicast data indicating updates in relative position).
- A set of geometry files define the subcomponents of the world at the leaf level of the hierarchy. The geometry files can be in any of a number of graphical file formats (such as Inventor, DXF, and so on).
- If the graphical file format does not permit the association of properties, such as material and textures, then a separate .props file may also be created.

Graphical models for BioSimMER were developed using the Alias/Wavefront commercial modeling software. Other applications have imported models from a variety of modeling and CAD/CAM packages. Because the display module must be capable of displaying images at a high-enough frame rate for HMD presentation, the geometry of the models is kept as simple as possible. To create more detail, texture maps are often used in place of model subcomponents. Level-of-detail (LOD) modeling also helps to keep the polygon count down. A separate model (set of geometry files) is created for each LOD that is used to represent an object. LOD changes are specified in the .hi file.

4.3 VR_Station Display Driver

VR_Station drives the individual displays for each user/view. We currently have a Performer-based version that runs on SGI workstations and are in the process of developing an OpenGL-based version that will run on PC platforms under Windows. Typically, there are multiple instances of VR_Station, each running on a dedicated graphics pipe/board and (prefer-

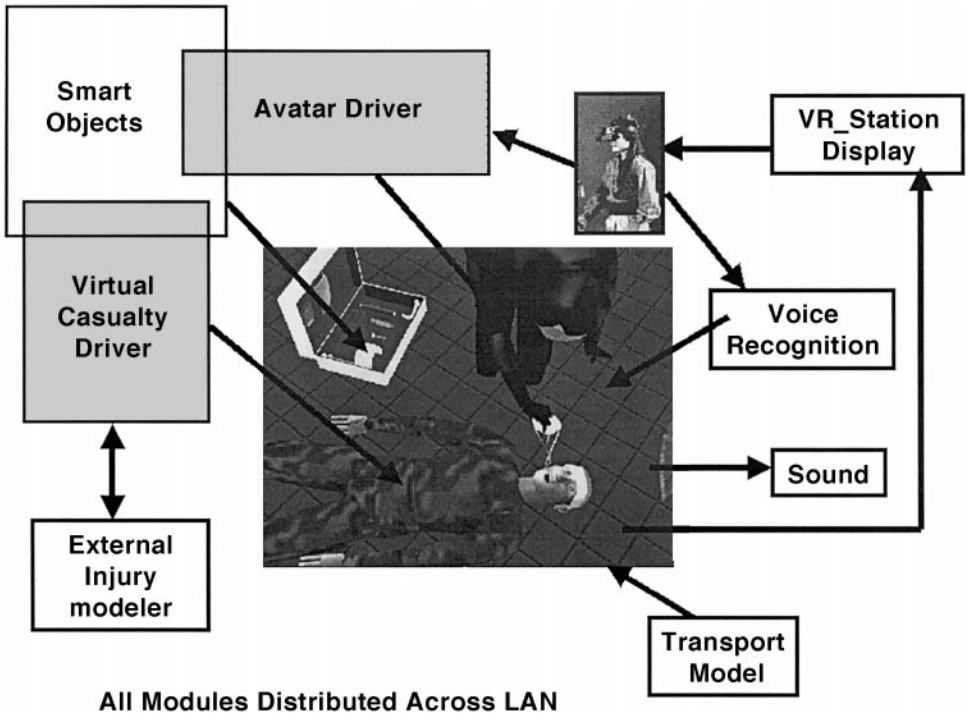

Figure 8. *The BioSimMER system architecture.*

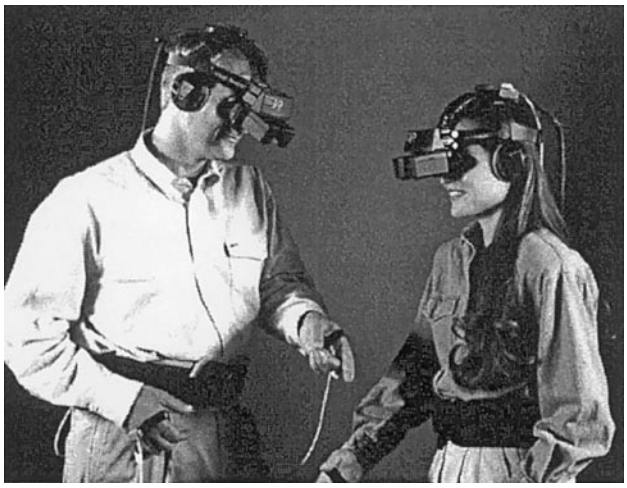

Figure 9. *Users wear "VR gear" consisting of HMD, position trackers, and microphone, which immerses them in the virtual world.*

ably) processor. Each user has an instance of VR_Station that allows him to independently control his viewpoint and motion within the virtual world. Real-time updates of the view of objects in the world are either remotely driven by sensors worn by the user (in the case of immersion) or are locally driven via mouse and keyboard.

VR_Station captures the view associated with the world in a class called VR_cam, which provides a view maintenance and update action. The inputs to a VR_cam object include the specified viewer's head position, the state of any buttons or other input devices associated with the VR hardware, and keyboard values that permit additional input (such as selecting modes of movement). Current inputs provide a forward and backward locomotion and modes that allow locomotion to be either free (fly mode) or constrained to maintain consistent height above a ground plane (walk mode). These modes allow the user to move within the virtual world beyond the physical limitations imposed by the range of motion of the associated VR tracking devices. Through subclassing, the VR_cam class allows a polymorphous set of viewing devices to be supported within VR_Station. Currently, in addition to the HMD and magnetic tracker viewing utilized by BioSimMER users,

FakeSpace BOOM and monitor/mouse-driven viewing modes are also available. For the latter two modes, direction of locomotion is based on direction of view; for HMD viewing, the orientation of the user's body, rather than of the head, determines the direction of movement. This permits simulated walking and upper-body movement reflecting the user's body posture. In addition to these features, BioSimMER has the additional capability to directly place both the avatar and its associated camera/view within the virtual world via a menu interface that acts through the avatar driver.

4.4 VR_Multicast

Our typical VR environment consists of multiple instances of several types of modules: an instance of VR_Station for each user and/or viewer and an instance of the avatar driver for each user. There may also be multiple instances of other simulation modules, such as the virtual patient driver. All VR_Station instances "see" the same world, although each VR_Station can show a different view of it. Therefore, the communication from simulation modules to VR_Station must allow this sharing. This is accomplished via Ethernet multicasting of UDP datagrams, and is implemented within VR_Multicast.

At start up, each VR_Station loads the world model independently by instance from the .hi file. For those transforms within the hierarchy that are dynamic (such as components of the avatar model), the associated simulation module instance (such as the avatar driver) is responsible for sending information about changes to those transforms (such as changes due to user motion as reflected by changes in the tracker data). Each such simulation module must also cooperate with all other modules to "cover" only its own part of the simulated environment. This is done by having an indexed space defined by the world model (in a file of type .oj) in which each simulation module gets exclusive use of a segment of that index space.

The receiving process, a VR_Station, creates an object of class joint_reader, through which it receives transform updates. The joint_reader object creates and manages an auxiliary thread of control, which allows timely response to network input without causing the main thread to block. This auxiliary control thread buffers incoming data for the main thread to pick up at the appropriate time, which ensures minimum impact on the update-render cycle by communication overhead. The sending process, a simulation module, creates one or more objects of class joint_publisher, each with its own list of joint (transform) names that it will output. The class internally maps the transform data passed to it for each object from its own internal order to the appropriate network-order indices (as specified in the .oj file.) The receiving object also maps indices, from network order to internal order.

The network packet format provides the corresponding network-order index for each transform sent. This allows sender processes to send as few transforms as necessary to handle whatever has changed since the last update was sent. It is also immaterial whether incoming data to a VR_Station originated in one simulation process or several; each simulation process will be allowed to exclusively handle its part of the joint index space by specifying its own subset of the joint names, and the receiver will simply update joints as data are made available.

The sensor_publisher and sensor_reader classes within VR_Multicast support the multicast communication of sensor data in the same way that the joint_publisher and joint_reader classes support the communication of transforms. The sensor_publisher communicates with the tracker system to obtain the current positions of the trackers worn by the users. The sensor_reader class obtains this information from the network for those modules that require it to update their portions of the virtual world. The top-level class interface for sensors does not require the ordering flexibility needed for joints, so a standard ordering of sensors, on a per-avatar basis, is assumed. Each avatar constitutes a separate logical communication channel. It is possible for the publisher or receiver to concern itself with only a subset of the possible sensors.

4.5 Voice Recognition

BioSimMER contains a voice-recognition component that permits the user to speak to the virtual casu-

alty (although this capability is currently rather limited). The user can also interact with the simulation, requesting information (such as vital signs) and commanding certain actions (such as exposing a patient by removing his shirt). The Entropic GrapHvite speech-recognition software supports the voice-recognition component. This software was chosen because it is speaker independent and does not require that each user train the system before use. It is necessary, however, to familiarize the user with the command vocabulary that the system will recognize. This vocabulary is functionally simple and has been developed to permit a number of variations on each command.

4.6 Sound Server

The sound server serves the various sounds used within the simulation, which include background sounds, such as sirens, as well as responses from the virtual patient to questions asked by the medic and responses from the simulation to requests for information such as vitals.

4.7 World Engine and Smart Objects

A driving force behind the design of the VR platform was the desire to make user interactions with the virtual world as naturalistic as possible (within the constraints of current VR hardware technologies). Rather than using a menu-driven or purely voice-activated interaction paradigm, we wanted users to be able to grasp and apply their virtual instruments (sometimes in more than one step) and to otherwise act upon the virtual world in the same way that they would upon the real one. Of course, an exact replication of the real world is not possible, due to such factors as the lack of force feedback and imprecision/noise in the tracker data. Therefore, we have designed and implemented a virtual world that aids the user in these interactions. The World Engine (WE) is the component of the virtual environment that drives the simulation. It orchestrates object interactions with and between entities, handles communication with external modules, and drives the subsequent changes to the state of the virtual world at each time step. Classes of objects within the World Engine, which we call Smart Objects, monitor user intent and communicate with each other to create the desired actions and effects. Smart Objects are discussed in detail in section 5.

4.8 Virtual Humans

4.8.1 Avatar Driver. Training applications such as BioSimMER require that users be represented within the virtual environment with a high degree of fidelity. It is important, for example, that multiple users be able to see each other as full human figures, so that the position and actions of each user can be determined. This is true even when team members are not performing coordinated tasks, but are acting individually, with coordination of only high-level actions such as selecting which casualty to triage. In addition, VR imposes several additional requirements on the representation of the user. First, and most important, the behavior of the participant's virtual self, which we call his/her avatar, must be updated at near real time to reflect the immediate actions of the user. In addition, the number of sensors/trackers used to obtain information concerning the participant should be small to minimize the amount of data that must be collected and processed and also to avoid encumbering the user. Our avatar driver uses input from four trackers and represents the user as a fully articulated figure, as is described in detail in section 6.2.

4.8.2 Virtual Patient Driver. The virtual patient is a dynamic, interactive simulation that correctly presents the clinical symptoms of the modeled injury and whose state changes realistically over time, both spontaneously (due to the injury) and in response to the actions of the user. Two important aspects of the virtual patient are the clinically correct simulation of condition and the techniques used to realistically impart this information to the user (for example, by changes in skin color, chest motion, and the like). Our virtual patient driver is described in section 6.3.

5 World Engine and Smart Objects: Interacting with the Virtual World

To create realistic training environments, users must be able to act upon and affect their environment. To enable this capability, the virtual objects in the world must have one or more of the following characteristics:

- They must be manipulatable, providing support for grasping and handling and for applying objects to other objects (such as tools and instruments).
- They must be autonomous, invoking behaviors that appropriately simulate their dynamic states. (For example, a grasped object must move with the grasping hand.)
- They must be interactive, changing their state appropriately in response to the actions of the user. (For example, a rolled bandage unrolls when placed on the head.)

We accomplish this by representing entities in the world that participate in manipulation and interaction as sets of Smart Objects. The mechanism for this interaction is *sites*, which are anchored to an object and capture both a local point and orientation on that object. Proximity detection between sites is used to determine, at any given time, those objects that may interact and the nature of that interaction. This information is used when the object is to be grasped or to have another object applied to it. The location of the site represents the place on the object at which this action will occur; the orientation of the site provides the correct orientation for the grasping or applied object. The grasping or applied object also has a site at the point of contact, which allows for the contact to be correctly placed and oriented. The method of using sites to correctly place grasped objects in the hand has been utilized by others, for example in the Jack system (Badler, Phillips, & Webber, 1993). In our work, the positioning and orientation functionality has been extended to include real-time reaction capabilities related to sites. This provides mechanisms for interaction with and between objects within time-critical simulation scenarios.

Within BioSimMER, the site-based mechanism provides the following functionality:

- ***Attach***: Fix one object to another (used to implement grasping).
- ***Activate***: Trigger object response by "touch" (that is, momentary contact).
- ***Dock***: Apply an object, which may remain in place and/or may trigger a subsequent response.
- ***Stow***: Place an object in a "home" position and orientation.

The Smart Object construct provides the infrastructure for the World Engine simulation driver. It has been designed using an object-oriented approach and is implemented primarily in C++. Below, we provide the technical details of this design and discuss how it is used to drive interaction with the virtual world.

5.1 World Engine Class Structure

Figure 10 shows the basic class hierarchy for Smart Objects. The BASIC_SIM class is the root of the class hierarchy for all simulation entities that are simple (such as an auto-injector) and time varying (such as the virtual patient). Within the former, the manipulatable/interactive entities (medical instruments) are all members of the BASIC_REACTIVE_OBJECT class or its subclasses. These classes represent the characteristics of an entity that permit it to participate in interaction. Other aspects, such as geometry and other visual attributes, are represented outside this object system, within the scene graph. (It is certainly possible to keep all information about an object within the same representation. The BioSimMER system keeps the scene graph as a separate structure to permit the graphical representation to be easily ported to different modeling and display software independent of the simulation engine.)

Simulation time steps are managed by the BASIC_SIM class, which provides function entry points that in turn call the time-varying objects of the class. To accommodate the output of transforms and other such functions that are best accomplished in one step, there are two additional classes—PROXIMITY_DETECTOR and JOINT_MANAGER—both of which have a single instance within

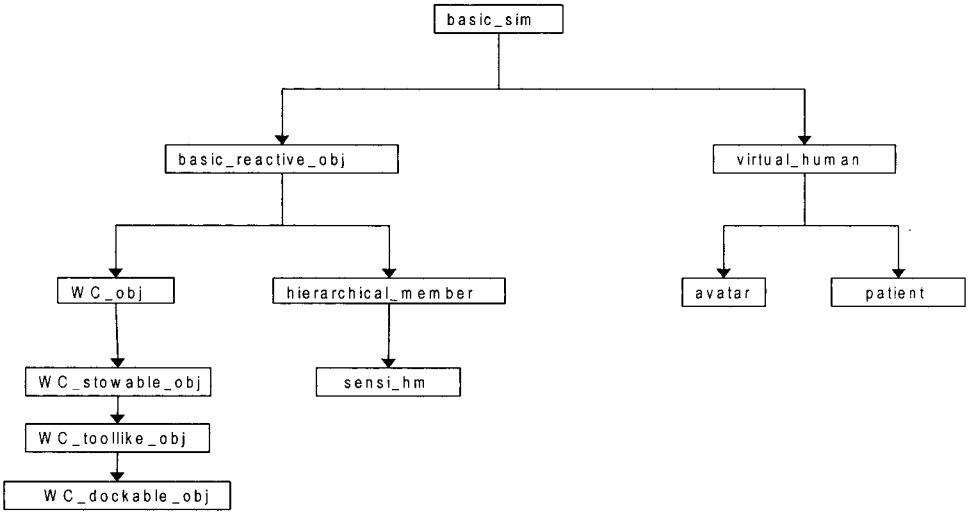

Figure 10. *The Smart Object class hierarchy.*

the world. The PROXIMITY_DETECTOR instance provides the functionality suggested by its name. (Its interaction with individual objects is further discussed in a following section.) The JOINT_MANAGER instance is an auxiliary global object that collects and stores output data until the end of the update phase of a time step. This allows output to be multicast once per time step in as few packets as possible, and supports position updates for individual objects (that is, traversal of the scene graph occurs in the joint manager.)

5.2 Handling and Using Virtual Objects

Grasping, docking, and activation are based on object-to-object communication. For example, grasping and grasped objects cooperate in maintaining the state of being grasped: the grasped object updates its position as required by any change in the position of the object that is holding it. Entry into being grasped is discussed below. Exit from this state is handled by either the grasping object telling the grasped object to detach itself, or by the grasped object forcing a detach action and informing the grasping object via a callback.

Objects may make themselves *active* objects to act upon other objects or *responsive* objects if they wish to be subject to the actions of other objects. The PROXIMITY_DETECTOR object allows an object to make itself subject to actions (grasp or activate). (This is usually done when the object is first constructed.) During simulation, the object controls its current responsiveness by setting internal flags. Based on the values of these flags, the object responds appropriately to inquiries from the proximity detector at each simulation time step.

The proximity detector also allows an object to indicate that it may (often) be active. The avatar's hand is an example of such an object. These objects also control their dynamic active state by setting internal flags, and they keep the proximity detector informed by responding appropriately to its inquiries. An instrument object would be active only when held and not yet applied to another object; a hand would be active only when empty. Object state information is summarized in Figure 11.

Docking is largely initiated and maintained by the dockable object itself, rather than the target object to which it docks. To initiate docking, the dockable object must have one or more hotspots (sites such as a needle's tip) and one or more possible targets (other objects' sites of interest, such as the patient's arm.) It must also be active. To maintain docking, the dockable object must actively update its position based on the position of the target site where it is docked.

Object State
REQUESTED_GRASPABLE
REQUESTED_ACTIVATABLE
GRASPABLE
ACTIVATABLE
CURRENTLY GRASPED OR ACTIVATED
OBJECT DOING THE GRASP OR ACTIVATION
LIST OF SITES AND THEIR AFFORDANCE TYPES
... and if needed
IF STOWABLE: *HOME FRAME*
IF TOOLLIKE: *THE SET OF TARGETS*
IF DOCKABLE: *OBJECT TO WHICH DOCKED*

Figure 11. *Object states.*

5.3 Object Actions

Object communication occurs when the state of objects changes—as when objects activate, attach, detach, or dock. During docking, for example, this communication might take the form of a call to a callback object at the site to which the object is attempting to dock. Object communication also occurs at each time step to update the positions of objects. For example, if one object is attached to another, if an activating object is present, or if two objects are docked, then position information about the relevant site must be queried and potentially used to update positions and/or states of activation. Inverse kinematic callback objects are also called at each time step, in order to update the joint positions of the objects to which they are attached.

The kinematic hierarchy (scene graph) is maintained with both permanent and temporary attachments (links). The scene graph and its permanent links are created at initialization and are available to all components of the simulation. The temporary links involve nonhierarchical objects that are in the WC_OBJ class or its subclasses. These objects may be moved and handled, and may change state to implement a temporary link—as in the attached or docked relationships. These relationships may make an object logically subordinate to another in the hierarchy, but no explicit link is created for this, because the mechanisms of attachment and docking effectively maintain the relationship.

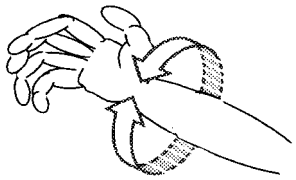

Figure 12. *The "let go" gesture.*

The permanent links require using HIERARCHICAL_MEMBER objects for those parts of the hierarchy that must participate in object interactions, because not all nodes in the scene graph need to have Smart Object representations. Those that do are represented by HIERARCHICAL_MEMBER objects if they are not at the top of the graph hierarchy and by WC_OBJ if they are.

5.4 Grasping and Releasing

In grasping, closeness of the user's hand to a graspable object initiates the grasp. To release the object, the user performs a "let go" gesture (rotating his/her hand briskly about the axis of the forearm), which is recognized by the avatar driver as meaning any attached object should be released. Figure 12 illustrates the "let go" gesture. Proximity-based grasping and a "let go" gesture are required because the user wears only a single tracker on each hand, and this doesn't provide enough information to deduce intent from hand posture. (Available gloves and other hand-posture tracking devices are currently unreliable, require frequent calibration, and generate a great deal of data to be communicated across the network. Thus, software-aided grasping provides, at least for the present, a more reliable and user-independent approach.)

The initiation of a grasp is based on the following algorithm. (The hand must be empty in order to grasp an object.)

- If the empty hand is active, the proximity detector searches for nearby graspable objects on its behalf.
- If graspable objects are found, then either the closest or all such objects are reported to the hand object.
- If a reported object has an appropriate affordance type for the site being considered, then it may be grasped. Grasping succeeds unless the object to be grasped refuses to attach. The hand object saves the identity of the object held (so it can force updates), and the held object saves the identity of the holder object so that it can keep itself attached when it updates its position.

A release may be initiated by either the holder object or the held object. The holder object can tell the held object to detach itself, or the held object can call a releaser callback and perform its own detach. The detach accordingly changes the state information for held and holder objects. To prevent released objects that remain at their position of release from being immediately (and incorrectly) regrasped, a hysteresis effect for the held object has been implemented: the holder object cannot regrasp the same object while it is still in the immediate vicinity. This action is controlled by the held/formerly held object.

The posture of the empty or grasping hand is changed by modifying the pose (open or closed) of the hand model. This change in hand pose is invoked by the objects involved (the avatar palm and the held/released object.)

For contacted or approached objects to respond, they must be registered with the proximity detector and must be currently graspable or "activatable." The acting object controls whether it wants the closest object or an exhaustive list of objects to test for suitability. The acting object selects for the type of affordance to be presented. If multiple sites are within reach, the acting object must decide which (if any) to interact with. Currently, only the location of the site on the acting and subject objects matter for proximity detection. However, the action taken usually makes use of the orientation of the sites as well. The orientation of sites is defined to make the grasped or docked state look correct. Figure 13 illustrates how sites are oriented and positioned during a grasp.

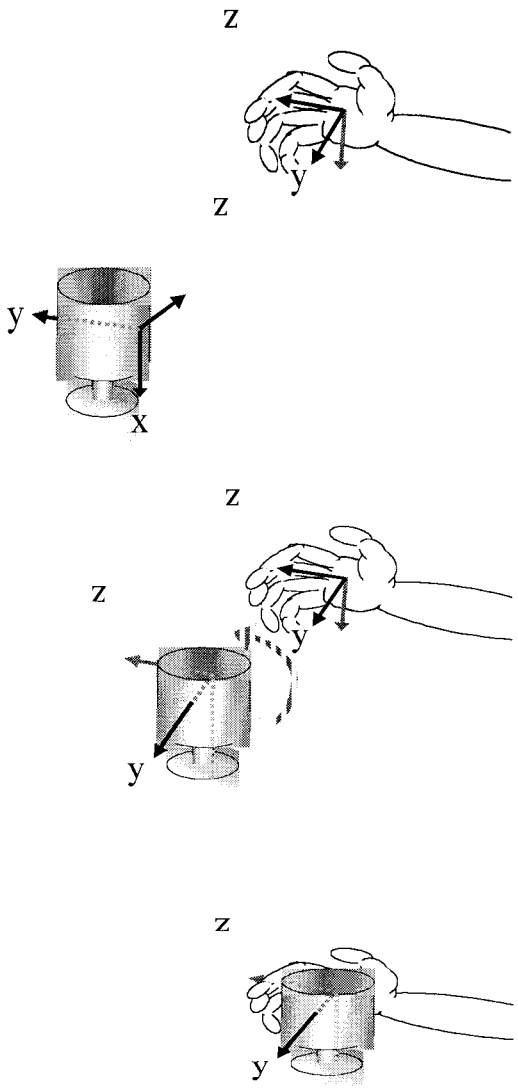

Figure 13. *Orienting and positioning of sites during a grasp.*

5.5 Right- and Left-Handed Grasps

Objects may be grasped with either the left or the right hand. The hand involved in a grasp is implicit, because it is the current acting object being tested by the proximity detector. This may not always result in a grasp that looks or feels natural, because the approach, initial contact, and final hand-object orientation may be different depending on whether it is a right- or left-handed grasp. These considerations are especially important when a system is intended to be used by many different

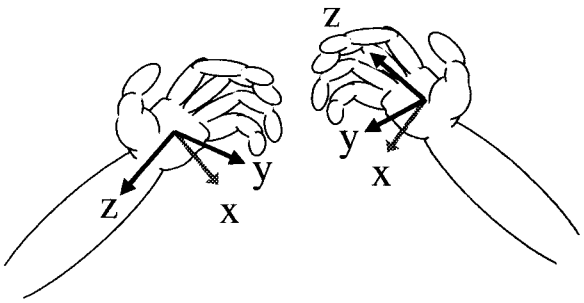

Figure 14. *Left and right grasp sites.*

people who may be either left- or right-hand dominant. To permit users to manipulate with either hand, as well as to accommodate dominant-hand differences between users, the hand object can also explicitly allow only sites of interest to the right hand (or sites of interest to the left hand) to be reported by the proximity detector. The object to be grasped can characterize its affordance sites as being for both hands or for the left or right hand only. Thus, an object can be modeled with sites for left-hand grasping and sites for right-hand grasping, and only the appropriate sites will be used as affordances by that hand. The orientation of the grasp sites on the right and left hands is shown in Figure 14. The *x* and *y* axes of each hand are mirror images of the other. The difference in the *z* axis between hands maintains the appropriate hand orientation for all site frames. Using this convention, many objects can use the same sites to fit either hand appropriately.

5.6 Tools, Instruments, and Docking Behavior

When an object is being held in a grasp, it may itself become active. This permits the user to apply one object to another, such as a tool to a work piece or a needle to a patient. The TOOLLIKE_OBJ class and its subclasses provide this capability by maintaining a list of targets of interest on other objects. Some TOOLLIKE_OBJ objects need only to touch the target of interest to activate it. Others, represented by the DOCKABLE_OBJ class and its derived classes, change state to become attached to a target. The docked object may continue to be grasped, or it may force the grasp to end. If it continues to be grasped, inverse kinematics is used to maintain the grasping object's position. For example, inverse kinematic calculations for the arm of the avatar are invoked by a callback object that calls the avatar driver when the avatar must maintain its grasp of an object during a docking action. This is required because the entire arm (or possibly the torso and arm) must be adjusted to provide a solution. An example is the use of the auto-injector by the BioSimMER user: the needle docks to the appropriate site on the virtual patient and maintains the dock and grasp for the amount of time it takes to inject the full contents of the injector into the virtual patient. It then becomes free, and the user may release it.

BioSimMER has implemented the following actions for a docking object:

- The object pops into place (inserting the J-tube into the patient's mouth).
- The docking object determines how long it stays in place (the injection example above).
- The docking initiates a sequence of actions (bandaging the virtual patient's head, for example, consists of the grasped bandage—when placed near the head wound—unrolling and placing itself on the patient's head to cover the wound).
- The docking makes a subordinate object ready to take action. (To start an IV, the first action is to place the IV bag on the patient's chest; only then does the IV needle become graspable and permit insertion into the virtual patient's arm.)

Figure 15 shows an avatar grasping a J-tube, which is a docking object. The docking position is the patient's mouth/throat. When the avatar has moved the J-tube to the proper position, it pops into place (inserted in the patient's throat).

6 Virtual Humans: Avatars and Virtual Patients

BioSimMER contains two types of virtual humans: avatars, which are the graphical representation of the user; and virtual patients/casualties, which are autono-

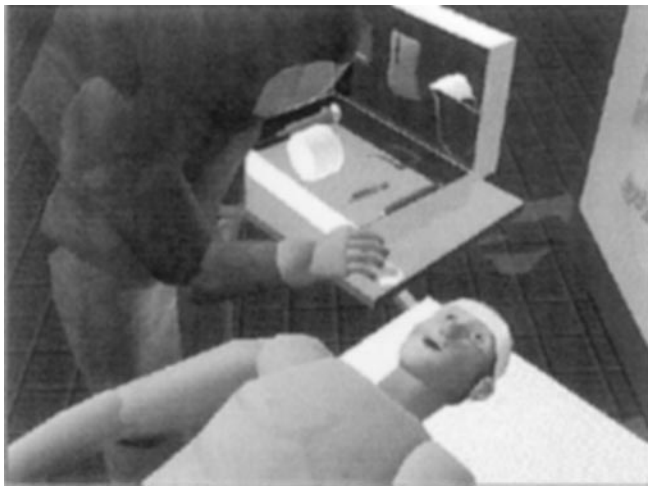

Figure 15. *Avatar holding J-tube to be inserted into the virtual patient's throat.*

mous simulations of the injured person that the user trains on. Both types of virtual human have the same kinematics, and both are driven via a hybrid combination of Smart Objects and external input. In the case of the avatar driver, the external input comes from the trackers worn by the user, and Smart Objects are used to represent those parts of the body that permit the user to interact with the virtual world (that is, to grasp, manipulate and apply virtual tools/instruments). In the case of the virtual patient driver, the external input comes from an external injury modeler that drives the course of the injury. Smart Objects are used to represent parts of the patient's body that the user must act upon or that might present certain medical conditions and responses of the patient. In this section, we first briefly present the graphical model of the virtual human. The avatar driver and virtual patient are then described in detail.

6.1 Virtual Human: Joint Set and Kinematic Hierarchy

Our standard human figure has 49 joints, including the transform that places the figure in the world (that is, that places the pelvis, which is the root of the figure). Of these joints, 21 are in the main torso, arms, legs, and feet, and 28 are in the two hands (fingers and thumbs). In the case of the avatar, the joints defining the torso, arms, legs, and feet are continually modified as the driver attempts to pose the avatar to conform to the tracker data representing the motion of the user. (This is described in more detail in section 6.2.) The avatar hand joints are changed by setting standard closed and open poses for the appropriate hand, depending on whether or not an object is being grasped. The avatar executes a reach by placing the hand using forward kinematics of the arm through the wrist joint. As detailed in section 5, the palm and target object then cooperate to permit the grasp.

In the case of the virtual patient, the joints are changed only when it is required that the patient move. Such motions might be cyclic (such as trembling) or invoked only once (such as moving an arm in response to a request by the user). Currently, these motions are stored as animations and invoked as required by the injury model. The animations are crafted and generated using software developed in-house and are defined in terms of the human figure hierarchy and its joint set. In addition to the standard joint set, the virtual patient also contains transforms that place and move the eyeballs and eyelids. This will be discussed in section 6.3.

6.2 The Avatar Driver

The avatar driver developed for this work utilizes the data from the four trackers worn on the user's body (top of the head, lower back, and backs of left and right hands). Inverse kinematics is used to determine pelvis and leg position, and a table look-up and interpolation are then used to position the torso and head. Finally, the arms and hands are positioned. In placing the avatar's hands, the driver takes the action of the user into account: if the user is not holding an object or is holding an object that is free to move in space (such as a penlight), then the avatar driver positions and poses the avatar body so that the body parts that are tracked coincide as closely as possible in position and orientation to the tracker data. If the avatar hand is grasping an object that is not free to move (for example, an auto-injector that has been inserted into the patient), then the avatar

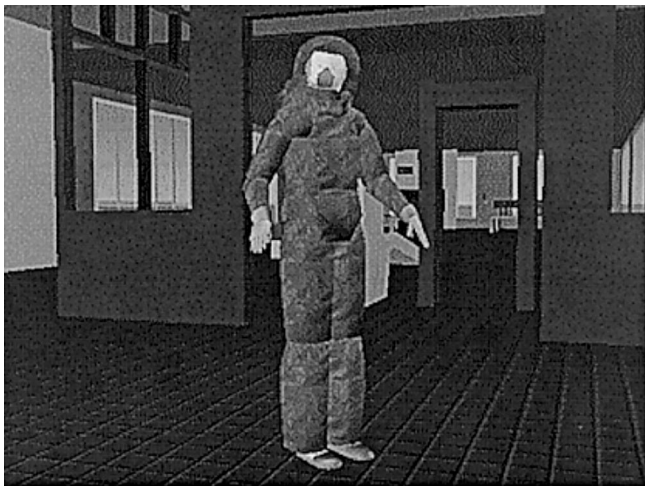

Figure 16. *The emergency responder avatar in protective gear.*

must remain bound in position to that object, even though the user may move her hand. In this case, inverse kinematics keeps the hand as close to the object as possible by replacing the actual sensor data with the frame that is required to keep the hand in contact with the object site at which the grasp occurs.

We impose this latter constraint to make it easier for the user to utilize virtual instruments. Noise in the sensor data and delays in communication make it difficult to keep the avatar hand still, even when the user's hand is not moving. If the hand were not bound to the fixed object, the user might see her avatar hand moving in relation to the object. Because we are not using force feedback, this movement would make it difficult for the user to determine whether or not her hand was actually in contact with that object. Figure 16 shows the avatar developed for this work. In the case of BioSimMER, the avatar's visual appearance reflects the protective gear worn by actual responders in a contaminated environment.

6.2.1 Placing and Posing the Avatar. The avatar body is placed and posed using the following strategy:

1. Place the pelvis appropriately. The pelvis is constrained to be level from side to side and will point in the correct direction in terms of the horizontal plane. The actual pelvic tilt (forward and backward) will be set in conjunction with the torso placement in step 3.
2. If necessary, do one of the following: for a bent-knee standing position, bend the legs and keep the feet on the floor; for a kneeling position, place the knees and toes on the floor, if possible. The joints involved are allowed only to rotate, without stretching (translation).
3. Place the head and adjust the torso (spine) to allow head placement subject to the pelvis placement in step 1. This is essentially an inverse kinematics solution to placement of the head center as the end effector relative to the pelvis as the root of the kinematic chain. This step is more fully described in section 6.2.2.
4. Position the arms so that the hands are as close to the positions of their respective sensors as possible. A closed-form inverse kinematics solution is used that places the hand base (the wrist joint) relative to the shoulder position determined by the torso placement in step 3. The offset of the wrist from the actual sensor position on the back of the hand is applied to the sensor data to yield the wrist joint data for the inverse kinematics. The wrist rotations are then adjusted so that the hand is oriented to match the orientation of the user's hand as indicated by the sensor data.

6.2.2 Posture of the Torso. The inverse kinematics solution mentioned in step 3 of section 6.2.1 uses table look-up and interpolation of the table values, an approach suggested by the work of Wiley and Hahn (1997). In our application of this technique, however, we start with spherical (that is, 3-D polar) coordinates for the head relative to the pelvis root (thus giving us three dimensions) and add another dimension for head tilt. The table is generated by placing a simulated figure in various combinations of bending, twisting, and leaning to the side, and tilting of the head forward or backward, rather than from data captured from sensors. The joints involved are allowed only to rotate, without stretching. This generally creates a plausible torso bend

and twist. When the table is used for look-up, the joint values are obtained by quadrilinear interpolation. The head is then oriented as required to match the orientation of the head sensor.

We use four dimensions to define avatar posture. The first is distance from pelvis to head. The second is the angle of the pelvis-to-head vector with respect to the vertical. The third is the angle of horizontal projection of the pelvis-to-head vector with the straight-ahead axis of the pelvis. (Strictly speaking, we actually use the lateral component of that horizontal projection, rather than the angle itself). The first and second dimensions determine how the torso must bend forward, and the third determines how the upper torso must twist to allow the shoulders and head to comfortably face in the appropriate direction. The fourth dimension takes the head tilt into account, so that the neck base joint will tilt the head backward or forward relative to the upper torso. If the avatar should be standing with the head tilted forward to look down, we try to assure that the torso will be back far enough so that it will not block the line of sight of the user. However, if the user looks up, tilting the head back, this torso pose would be awkward; in this case, the upper (and mid-) torso are allowed to come forward so that the neck can tilt backward.

6.2.3 Calibrating the Avatar to the User.

Currently, the avatar figure is calibrated only to the overall height of the user; this is due primarily to constraints in the amount of accurate information about the user's full proportions available to us. Two ways to overcome this difficulty would be to keep an accurate database of measurements for each user, or to customize an avatar for each known user by taking those measurements up front. Semwal, Hightower, and Stansfield (1998) present an avatar driver that utilizes this approach. Unfortunately, this requires that each new user undergo a tedious measurement phase, and so it works best if the set of users is fixed or grows only slowly. Another way to address this problem would be to have the user wear a larger number of sensors on his body and to automatically calibrate all proportions based on this large set of readings. In our work, we prefer not to require users to go through a lengthy measuring stage, and so we use an automatic calibration routine that utilizes only four trackers for each user. The most prominent difficulty that arises from this method is that we cannot guarantee that the reach of the user and the reach of the avatar are the same. However, we have not found that this discrepancy is so great that it limits the user's ability to use the system; the avatar's reach has been sufficient for the large number of users that have used the system, both formally and informally. Another resulting discrepancy is that sometimes the avatar's arm posture does not exactly match that of the user. We are studying the look-up table methodology that is currently used to pose the torso as a possible solution to this problem.

6.3 Virtual Patients

A complete virtual patient model consists of the following components: the geometric and other visual attributes that make up the appearance of the patient for each possible state (such as skin color and wounds); the behaviors of the patient, both voluntary and involuntary (such as blinking and limb motion); and the injury model that drives the state of the patient, both spontaneously over time and in response to the actions of the user. We use our standard human figure as the basis for all of the virtual patients. To provide a richer training experience, we have varied certain aspects of their appearance, such as hair, eye and skin color, clothing, and the like. To avoid having to redefine the kinematic hierarchy, however, we do not drastically vary the body shape of the virtual patient. The varied clinical appearance of the virtual patient was developed with input from medical subject-matter experts and is as accurate as possible. Patient behaviors are created using the kinematic hierarchy to define motions and a set of Smart Objects to permit interaction with the user and presentation of medical conditions. Finally, the dynamic clinical state of the virtual patient is driven by an external injury modeler based on finite-state automata (FSA).

6.3.1 Patient Appearance. Creating a realistic appearance for the virtual patients was an incremental

task. Some of the injuries could be represented using texture maps that were created from photographs of either actual injuries or moulage participants in emergency-response live exercises. Other injuries and clinical appearance (such as the correct color of the skin for a cyanotic patient) were developed by hand with input/acceptance from our medical expert. Figures 5 and 6 in section 3 show two of the virtual patients created for BioSimMER. To represent changes in the size of the patient's pupils, textures were created for normal, dilated, and constricted pupils, and were dynamically swapped at the appropriate time.

With the exception of eye movements (discussed in section 6.3.3) and breathing, patient motions were created using animations. Key frames were first set for the appropriate joints; and then a linear interpolation between these key frames was used to produce the appropriate motion. Again, motions were deemed acceptable when agreed to by our expert. The movement of the chest during breathing was created via morphing, with the rate of breathing driven by the injury model.

6.3.2 Patient Behaviors. As with the avatar, the patient may be thought of as a geometric human model and a collection of Smart Objects, with each Smart Object handling its aspect of the patient's behavior. In the case of the virtual patient, the Smart Objects must handle both state changes from object-object interactions and states that can change asynchronously—because the patient's condition may deteriorate or improve either as a result of the actions of the user or spontaneously as the course of his injury progresses over time.

Changes that are the result of medic actions (often mediated through medical instruments) are modeled using subclasses of the BASIC_REACTIVE_OBJECT class. For example, performing a needle aspiration should register with the patient model as a release of pressure on the lung. The effect of this action may take time and be dependent on various other aspects of the patient's condition. The state-machine based injury modeler discussed in section 6.3.4 determines the behaviors that result either spontaneously from the injury itself or as a result of treatment.

6.3.3 Eye contact and movement behaviors. Two behaviors that require special handling are the patient's eye contact and saccadic movements. Both of these specific eye behaviors involve action at a distance and so are exceptions to the pattern of object interaction that we have previously described. Each can, however, be based on the available site data that our models provide.

A conscious patient is modeled so that he or she blinks and has random eye movements, with periods of eye fixation on the nearby medic if present. This eye movement adds realism to the virtual patient as representing a human. A patient's symptoms may also require that the blinking rate change from normal: for example, a catatonic patient blinks at a much slower rate. The mechanisms for eye movement and blinking are both based on Perlin noise functions (Perlin, 1985, 1995). The noise is evaluated to determine when to change the eyes' gaze and/or to blink. If the gaze should fix on an object, such as the face of the medic, the object will respond with one or more frames returned from the appropriate inquiry function. If there is no appropriate object upon which to fix the gaze, and it is time for the gaze to shift, then a direction within the patient's field of view is chosen randomly.

Neurological assessment also requires more-specific eye movements and/or response, and we have implemented two of these tests: ocular motility and pupillary response to light. During the ocular motility test, the patient is asked by the user to follow the user's finger. Finger-following involves two stages:

- ***Acquisition***: Inquire a suitable hand site and determine whether it is within the field of view; if so, enter an eye-tracking state. In this state, focus the eyes on the point where the hand site is.
- ***Tracking maintenance***: At each time step, reacquire the hand site if possible; otherwise, exit the eye-tracking state (because the hand has left the field of view).

To test pupillary response to light, the user needs to be able to direct a penlight toward either eye and observe whether that pupil constricts. This is implemented by checking whether each eye lies within the beam of

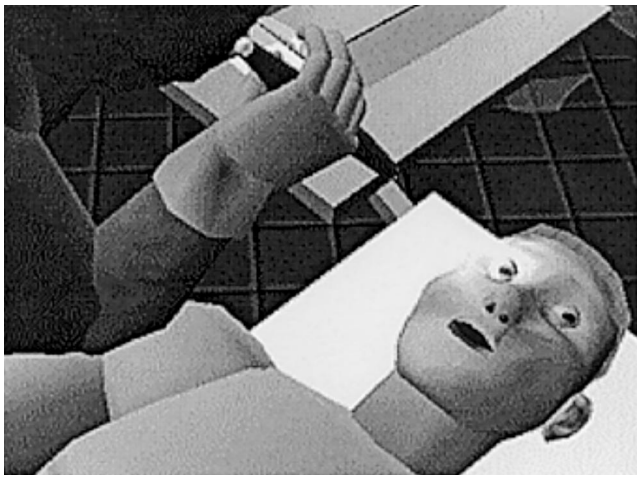

Figure 17. *Pupillary response test.*

the virtual penlight being manipulated by the user. The beam is modeled by the frame of the penlight and an angle within which the beam is concentrated, together with a distance over which the beam is effective. Thus, it is possible to inquire of the penlight Smart Object what the beam strength is at the position of the eye. If there is sufficient light shining into the patient's eye, then a pupillary response may occur, depending on the current state of the patient (obtained via the external injury model). Figure 17 shows the pupillary response test.

6.3.4 External State Machine-based Injury Model. If BioSimMER is to be a training system, then it is crucial that the presented injuries and the changes in patient state (both spontaneous and in response to the actions of the user) be realistic and medically sound. To ensure this accuracy, we worked throughout the project with subject-matter experts in both medicine and medical training who provided the necessary input to accurately model the virtual patient. In addition, rather than develop the injury models in-house, we collaborated with Tekamah, a company that develops computer-based medical training modules. Tekamah's training modules are based on decision trees that map the most likely outcomes for each patient. These decision trees describe the initial condition or state of each patient, the anticipated user/medic actions, and the results of these actions on the state of the patient—all of which are modeled using FSA (Thompson & Allely, 1997). These FSA do not, of course, represent the entire human physical state in all its complexity, but rather the critical states of a patient's condition and the responses that are necessary and sufficient to provide the training experience. The FSA developed and implemented by Tekamah for this work provides the injury models for the injuries described in section 3. It runs as a separate, external module that communicates with the World Engine to drive the dynamic injury state of the virtual patients. To accommodate this, a communication server was developed to permit the external injury modeler (FSA) and the World Engine to communicate via ASCII string messages. The message syntax uses SNOMED (Systematized Nomenclature of Human and Veterinary Medicine) codes, which provide a potentially extensible set of standardized names for medical conditions, treatments, anatomy, and the like.

A single instance of the external injury modeler handles all patients within the simulation in the same time step; therefore, only one message channel has been created to communicate between the external injury modeler and the World Engine. Messages from specific virtual patients on the World Engine side are multiplexed into this one channel. Messages coming to the World Engine from the external injury modeler are dispatched to the intended virtual patient, so that each virtual patient deals only with messages that are relevant to itself. The external injury modeler associates a case name with each individual patient and defines separate internal objects to model each of the patient's physiological systems (such as neurological, respiratory, and so on). The World Engine interprets input from the external injury modeler in terms of these internal objects and then dispatches the proper messages to the appropriate virtual patient.

User actions affecting the patient are also handled by the World Engine: Avatar and/or medical instrument Smart Objects call methods in the appropriate virtual patient object. In most cases, the virtual patient object will then contact the external injury modeler to determine what response to make to the user's action and to allow the injury modeler to input the specific action into

its FSA to update its own internal states. In general, any patient response or state that is controlled by the injury modeler is activated (for a response) or changed (for a state) within the World Engine only when an appropriate message is received from the injury modeler.

The World Engine processes all messages received from the injury modeler during a time step (by invoking the appropriate patient behavior). In the case of patient response to a user's action, the virtual patient driver sends a message to the external injury modeler informing it of the action. For example, if the user asks the patient to move his left arm, the virtual patient driver sends an inquiry message to the injury modeler to determine whether, in his current state, the patient is able to move that arm. This message is sent either by the Smart Object for the specific affected body part of the virtual patient or by a callback object installed in the Smart Object for the body part. If, at some future time step, a message is received from the external injury modeler indicating that the arm is able to move, then the appropriate animation routine is called. If the arm is not able to move, no message is returned from the injury modeler, with the result that there is no arm motion on the part of the patient.

In the case of spontaneous changes in patient state that may need to be reflected visually or in response to inquiries (such as a change in skin color or vitals), the external injury modeler spontaneously sends messages that the World Engine uses to update the state of the virtual patient. The virtual patient driver keeps track of its current state and invokes appropriate behaviors, such as changes in appearance or vocal recitation of vitals. An exchange of messages is not needed in this case, but it is instead assumed that update messages will come from the external injury modeler whenever the patient's state changes.

7 Initial Field Evaluation

In this section, we describe the experimental results of a preliminary field test and study conducted at the Texas Engineering Extension Service Fire Protection Training Division (TEEX FPTD) in College Station, Texas. The purpose of the study was to assess predictors of effectiveness for the use of VR-based systems such as BioSimMER in training emergency response personnel. The focus of the study was on the individual and the tasks that he or she might perform. Team coordination tasks were not studied.

7.1 Selection and Sources of Subjects

The TEEX FPTD recruited 23 subjects from 18 to 45 years old for this study. The FPTD sent a letter asking for volunteers to the chiefs of fire departments within the state of Texas and within driving distance of the FPTD. Subjects were recruited from Bryan, Blinn College, College Station, Copperas Cove, Ft. Worth, Austin, and Houston. Previous participation in the TEEX FPTP was sufficient for consideration for study inclusion. Visual acuity not correctable to 20/50 in both eyes, abnormal color or depth perception, or a history of disabling symptoms of motion sickness were exclusion criteria. Subjects were not compensated. Anonymity of the subjects was protected by having each subject anonymously complete a questionnaire. Assessment of subject demographics was used in an aggregate manner to compare significance of variables within- and between- groups in the assessment.

7.2 Experimental Procedure

The experiment was performed July 28 through 30, 1999, in the Computer Science Department Computer Lab at Texas A&M University in College Station, Texas. A total of 23 subjects were run for the entire experiment. Personnel from Sandia National Laboratories performed the experiments and provided technical support.

Subjects were run consecutively, and, for each subject, the experiment consisted of the following stages:

1. Subjects were given a consent form to read and sign. (These forms are on file at Sandia National Laboratories.) A chart-based vision test was given to ensure acceptable visual acuity.
2. Subjects watched an introductory video, which began by describing the purpose of the study. The

Table 2. *Questionnaire Part I: Demographics*

Part I: Demographics
1. Your occupation(s): Primary: Secondary: 2. Years of experience in job(s): Primary: Secondary: 3. Years of experience with computers/computer workstations: 4. Years of experience with flight simulators (if applicable): 5. Experience with a "Virtual Reality" system (please describe): 6. Susceptibility to motion sickness (self-rated): LOW MEDIUM HIGH

Table 3. *Questionnaire Part II: System Assessment*

Part II: System Assessment:
1. Applicability of this form of training (i.e. fully immersive virtual reality) to your activities as a first responder: 2. Applicability of the selected scenarios to your responsibilities as a first responder: 3. Fidelity of virtual environment (i.e. resemblance of this scenario/simulation) to the operational (first responder) environment: 4. Level of difficulty of scenarios selected: 5. Relevance of decision-making to your responsibilities as a first responder: Importance of the following information/visual displays to your decision-making in the selected scenarios: 6. Patient Appearance: 7. Vital Signs: 8. Environmental cues external to patient: 9. Importance of tactile (touch-based) information to your decision-making in the selected scenarios: 10. Importance of auditory information to your decision-making in the selected scenarios: 11. Satisfaction with field-of-view: 12. Satisfaction with visual/image updates: 13. Satisfaction with image resolution: 14. Satisfaction with ability to handle/use virtual objects 15. Satisfaction with vocal commands/vocabulary: 16. Confidence in this training system to support training-of-interest: 17. Acceptance of this form of training system to augment existing training (i.e. field exercises, computer-based instruction): 18. Suggested additions/modifications to the type of information presented that you believe will assist in decision-making tasks as a first responder:

VR hardware (HMD, trackers, and microphone) was introduced, and their use and purpose were explained. The methods for interacting with the VR were introduced and demonstrated (picking up and letting go of virtual objects, voice recognition, and vocabulary). The experimental process was explained and subjects were told that they could stop the experiment at any time. Questions were answered.

3. Subjects then went into the testing area where they put on the VR gear and were "calibrated" in a process that consisted of standing in a particular posture (at attention) while the computer reads the trackers and resizes the avatar body to match that of the subject.
4. Subjects were then permitted a ten-minute practice session during which they interacted with the VR system by manipulating virtual objects using the SEB exposure virtual patient. (The SEB exposure was chosen because the symptoms and treatment are not time critical, nor could the subject drastically change the state of the casualty via treatment.) This permitted the subject to become familiar with the system without scenario-related pressures.
5. Subjects then participated in a simulation scenario. One of two scenarios (tension pneumothorax or head trauma) was randomly assigned. Each of

Table 4. *Subject demographics*

Grouping A: years of occupational experience as a paramedic	
10 years or more	10 subjects
Less than 10 years	13 subjects
Grouping B: VR/flight simulator experience (one or both vs. neither)	
None	13 subjects
Some experience with one or both	10 subjects
Grouping C: occupation (noninstructor vs. instructor)	
Paramedic/EMT	20 subjects
Instructor	3 subjects

these scenarios requires the subject to recognize and address specific symptoms in a timely manner to assess and treat the patient. In addition, each of these scenarios is a common trauma injury that is familiar to experienced paramedics, thus ensuring that the subjects were not exposed to a totally unfamiliar injury. The scenario ended when the virtual casualty either was stabilized or died.

6. Subjects removed the VR gear and returned to the briefing area, where they filled out an anonymous questionnaire asking them to assess the system and its effectiveness for training tasks relevant to their responsibilities. The questionnaire was computer-based and consisted of two parts: Part I collected demographic information, and part II collected system assessment information. With the exception of question 18, responses to the questions in part II were in the form of a ranking from 1 (low/not applicable) to 5 (high/highly applicable). Subjects could choose not to answer any question. Tables 2 and 3 contain the study questions.

7.3 Experimental Results

7.3.1 Demographics. The study included a homogeneous group of 23 volunteer subjects that was characterized by limited experience with virtual reality systems. All were highly motivated to participate and reported extensive operational experience in field emergency medicine. For the purpose of additional analysis, the subjects were grouped into three separate demographic subcategories, as shown in table 4. Questions can be categorized as either task-oriented (questions 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15) or application-oriented (questions 1, 2, 4, 5, 16, and 17).

Table 5. *Mean and standard deviation of responses over all subjects*

All subjects		
Question #	Mean	Standard Deviation
1	4.261	0.864
2	4.261	1.009
3	3.696	0.822
4	2.522	1.123
5	3.870	1.140
6	4.478	0.994
7	4.652	0.647
8	3.609	1.118
9	3.957	0.878
10	4.318	0.780
11	3.435	1.037
12	3.957	0.562
13	4.000	0.739
14	3.261	1.176
15	4.000	0.739
16	4.391	0.656
17	4.348	0.775

7.3.2 Analysis over All Subjects. Table 5 shows the mean and standard deviation of responses over all subjects for each question in part II, while Figure 18 displays the mean responses graphically. As is indicated in the questionnaire, responses were numeric values between 1 and 5. In general, subjects reported high acceptance of perceptual cues, such as auditory and visual feedback reflecting patient status. Field of view was considered moderately satisfactory. Vital sign indicators were considered critical and dis-

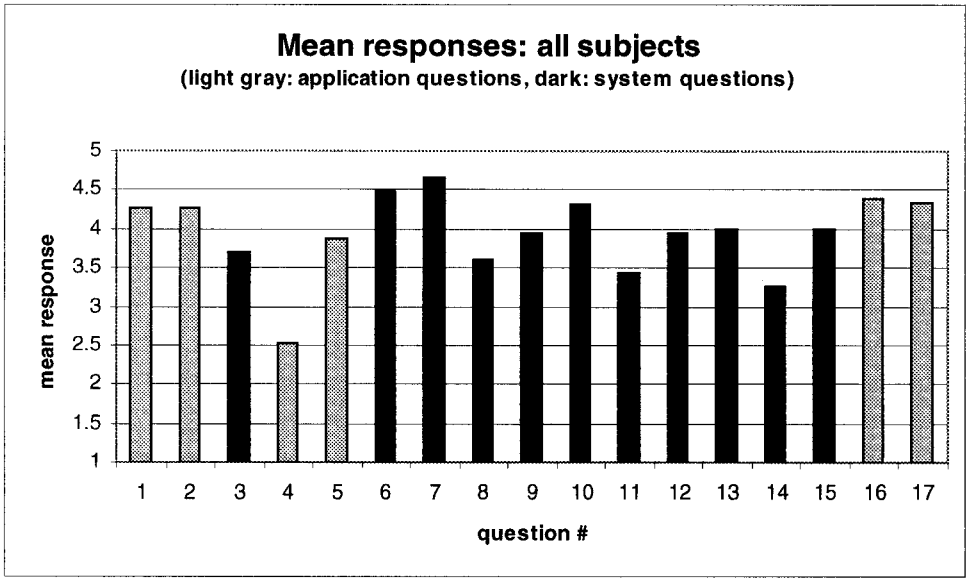

Figure 18. *Mean of responses: overall subjects.*

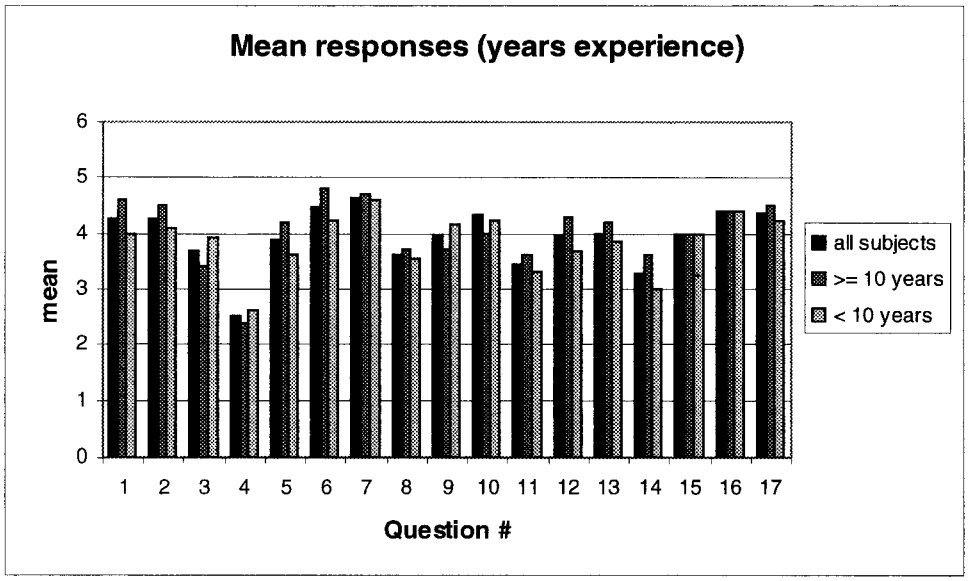

Figure 19. *Mean of responses: occupational experience groups.*

played appropriately. Moderate satisfaction was reported for the ability to manipulate objects. However, tactile information was determined as critical to the realism of the simulation and highly relevant to clinical decision-making. The content of the medical scenarios demonstrated were assessed as low in difficulty, but high in familiarity/relevance as a training set for emergency responders.

7.3.3 Analysis between Groups. Between-group analysis was performed for each demographic

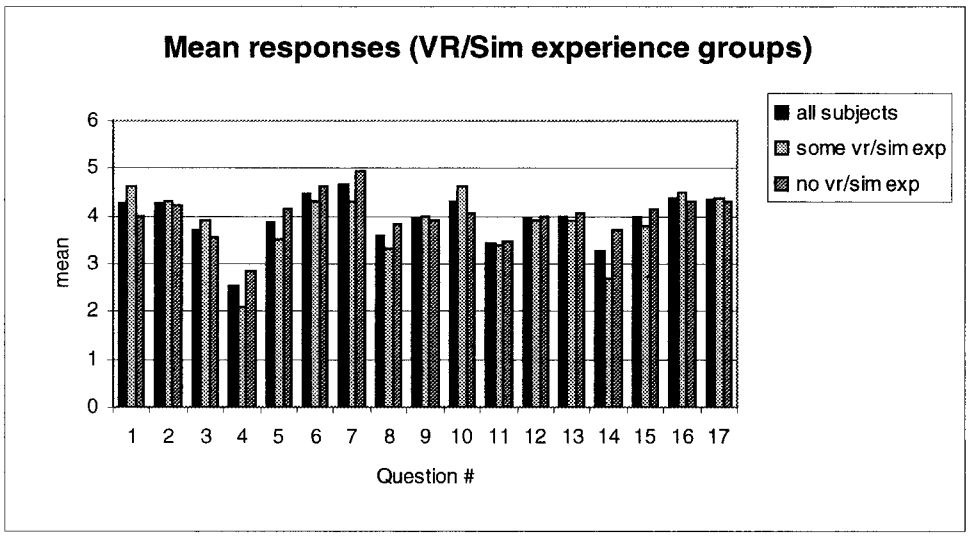

Figure 20. *Mean of responses: VR/flight simulator groups.*

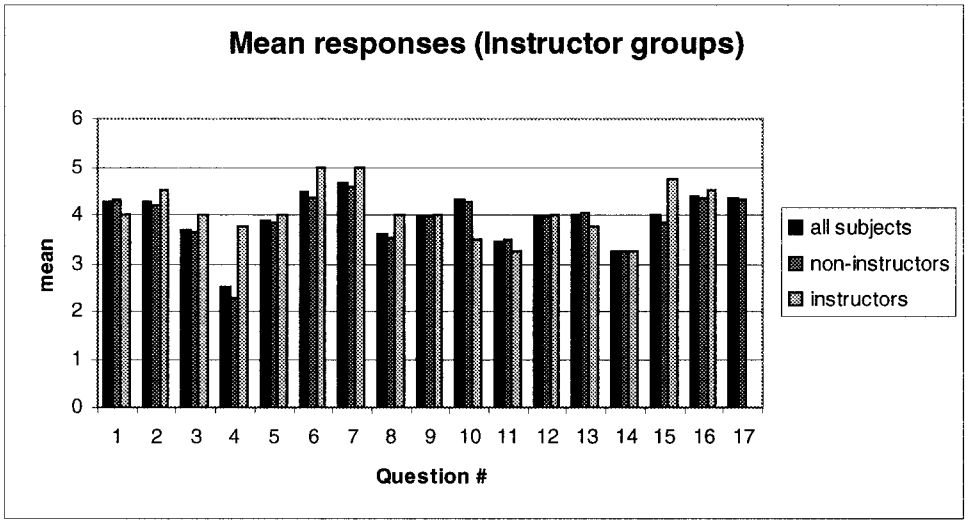

Figure 21. *Mean of responses: instructor groups.*

group listed in table 4. Figure 19 shows the mean responses for all subjects, along with those for subjects in the respective occupational experience groups. Figure 20 shows this data for subjects in the VR/simulator experience groups, and figure 21 for subjects in the instructor/noninstructor groups.

In addition, a single-factor analysis of variance (ANOVA) was performed between groups in the occupational experience, VR/simulator experience, and instructor/noninstructor groups. Table 6 shows the *p*-values for these analyses. In all cases, the responses do not seem to indicate any significant between-group differences.

7.3.4 Study Conclusions. The study group was satisfied and accepting of VR training systems as a mo-

Table 6. *p-values for between-group analysis*

	p-value (single factor ANOVA)		
Q#	Occupational experience groups	VR/sim experience groups	Instructor/ non-groups
1	0.0997	0.0997	0.5193
2	0.3307	0.8749	0.6138
3	0.1334	0.3066	0.4281
4	0.6590	0.1161	0.0123
5	0.2311	0.1785	0.8097
6	0.1791	0.4625	0.2574
7	0.7639	0.0182	0.2459
8	0.7397	0.2545	0.4539
9	0.2271	0.8405	0.9162
10	0.6526	0.1244	0.4183
11	0.5154	0.8916	0.7045
12	0.0068	0.6825	0.8695
13	0.2642	0.5810	0.4692
14	0.2335	0.0418	0.9843
15	1.0000	0.2642	0.0215
16	0.9571	0.4989	0.7246
17	0.4220	0.7844	0.6760

dality for first-responder training. Little, if any, significant between-group differences were found for subjects with different experience levels, previous or no exposure to VR and/or flight simulators, nor for instructors versus practitioners. Multisensory perceptual cues (visual, auditory, and tactile) were deemed critical to patient-management simulations. Comments collected from the questionnaire indicate the need for improvements in input devices, naturalistic handling and use of virtual objects, and for increased/improved vocal and audio feedback before an optimal training experience can be provided. For example, many subjects stated the need for breathing sounds and for more verbal interaction with the patient. Training scenarios also appear to be most satisfying when consisting of complex patient scenarios embedded in information-rich environments. Several of the subjects felt the scenario should be more in-depth and challenging. Additional pretreatment information in the form of a "dispatch" or eyewitness statements was also suggested.

In addition to the data collected from the questionnaire, the study also collected the timing and high-level actions of each subject as they performed the given patient-management tasks. Although it was not within the intended scope of the study to formally evaluate this data (we felt that the system development was not sufficiently advanced to permit reliable performance or training transfer evaluation), we offer some examples below. Tables 7 and 8 show, respectively, the actions of subject 16 treating the tension pneumothorax and subject 20 treating the head trauma, both with successful outcomes. Table 9 shows an instance in which deficiencies in the system presentation of the scenario most likely led to an incorrect action: Subject 18 initially performs a needle aspiration on the head-trauma casualty, then recovers and auto-injects the anti-convulsant, thus ending the seizure. It seems likely in this case that BioSimMER's use of two different types of needle, a standard needle for aspiration and an auto-injector for medication, led to the misstep.

8 Summary and Future Work

This paper has presented the design and implementation of a distributed VR platform that was developed to support multiple users carrying out complex tasks in a collaborative environment in which situation assessment and critical thinking are primary components of success. The system is fully immersive and multimodal. Users are represented as tracked, full-body figures, permitting team members to interact. The system supports the manipulation of virtual objects, thus allowing users to act upon the environment naturalistically. The underlying simulation component creates an interactive, responsive world wherein the consequences of such actions are presented within a realistic, time-critical scenario. The focus of this work as been on medical emergency-response training. BioSimMER, an application of the system to training first responders, is an existing prototype built upon this infrastructure. An initial field evaluation of BioSimMER was performed at the Texas Engineering Extension Service Fire Protection

Table 7. *Subject 16 treatment sequence for tension pneumothorax*

Subject number 16	
Case is tension pneumothorax. Total treatment time: 218 seconds	
Treatment sequence:	
elapsed time 8 seconds	patient being exposed
elapsed time 15 seconds	subject requests vitals
elapsed time 28 seconds	patient receives chest bandage
elapsed time 62 seconds	patient receives intravenous therapy
elapsed time 89 seconds	patient receives endotracheal tube
elapsed time 100 seconds	patient receives mask
elapsed time 105 seconds	patient ocular motility to be tested
elapsed time 105 seconds	patient eyes following right finger
elapsed time 105 seconds	patient becomes unconscious
elapsed time 140 seconds	patient receives needle aspiration
elapsed time 140 seconds	patient regains consciousness
elapsed time 160 seconds	patient receives cervical collar
elapsed time 165 seconds	subject requests vitals
elapsed time 218 seconds	patient pupil response to be tested

Table 8. *Subject 20 treatment sequence for head trauma*

subject number 20	
Case is head trauma. Total treatment time 418 seconds	
Treatment sequence:	
elapsed time 28 seconds	subject requests vitals
elapsed time 49 seconds	patient being exposed
elapsed time 71 seconds	patient receives endotracheal tube
elapsed time 84 seconds	patient ocular motility to be tested
elapsed time 84 seconds	patient eyes following right finger
elapsed time 130 seconds	patient receives intravenous therapy
elapsed time 174 seconds	patient pupil response to be tested
elapsed time 189 seconds	patient becomes unconscious
elapsed time 211 seconds	patient starting seizure
elapsed time 249 seconds	patient receives head bandage
elapsed time 259 seconds	subject requests vitals
elapsed time 335 seconds	patient receives injection
elapsed time 346 seconds	patient ending seizure
elapsed time 361 seconds	subject requests vitals
elapsed time 418 seconds	subject requests vitals

Table 9. *Subject 18 treatment sequence for head trauma*

Subject number 18	
Case is head trauma. Total treatment time is 449 seconds.	
Treatment sequence is:	
elapsed time 20 seconds	patient being exposed
elapsed time 29 seconds	subject requests vitals
elapsed time 55 seconds	patient movement of limb to be tested
elapsed time 62 seconds	patient movement of limb to be tested
elapsed time 66 seconds	patient movement of limb to be tested
elapsed time 79 seconds	patient movement of limb to be tested
elapsed time 90 seconds	patient movement of limb to be tested
elapsed time 113 seconds	patient receives cervical collar
elapsed time 126 seconds	patient pupil response to be tested
elapsed time 164 seconds	patient ocular motility to be tested
elapsed time 164 seconds	patient eyes following right finger
elapsed time 189 seconds	patient becomes unconscious
elapsed time 209 seconds	patient starting seizure
elapsed time 215 seconds	patient receives intravenous therapy
elapsed time 222 seconds	subject requests vitals
elapsed time 257 seconds	patient movement of limb to be tested
elapsed time 268 seconds	patient movement of limb to be tested
elapsed time 297 seconds	patient receives endotracheal tube
elapsed time 308 seconds	subject requests vitals
elapsed time 325 seconds	patient receives needle aspiration
elapsed time 337 seconds	subject requests vitals
elapsed time 353 seconds	patient movement of limb to be tested
elapsed time 362 seconds	patient receives injection
elapsed time 373 seconds	patient ending seizure
elapsed time 377 seconds	subject requests vitals
elapsed time 402 seconds	patient movement of limb to be tested
elapsed time 409 seconds	subject requests vitals
elapsed time 443 seconds	patient receives head bandage
elapsed time 449 seconds	subject requests vitals

Training Division with very positive feedback from firefighter/paramedics.

Of course, much research remains to be performed in developing the underlying intelligent simulation and VR capabilities. Among those we are hoping to explore is the use of

- tactile feedback both for the manipulation and representation of the virtual environment (such as collapsed structures that inhibit movement);
- methods for imposing the environment on the user (such as the effects of a toxic exposure on the physical capabilities of the user);

- more complex, autonomous behaviors for our virtual humans (including better verbal interaction); and
- the development of techniques for interactive composition and control of complex scenarios, an area in which we have made some preliminary progress already (Shawver, 1997).

A primary goal of BioSimMER, its predecessors, and (hopefully) its successors is producing tools that will meet needs and fill training gaps in the real world. The system's application to emergency response could be extended to include rural telemedicine and graduate medical education (two additional areas we have begun to explore with faculty at the medical school of the University of New Mexico). To accomplish this goal, the development of additional injury models and scenarios must continue. The system must also be further developed to make it more robust and user friendly.

Acknowledgments

The work presented in this paper is a result of contributions made by a number of people over several years of development. We would like to thank each of them for their vital role in the success of this project. From Sandia National Laboratories: James Singer, who wrote the animation software and the sound server; Ron Hightower, who developed early components of the avatar driver; Nadine Miner and Jim Pinkerton, who contributed to VR_Multicast; and Dave Rogers and Tom Keller, who developed portions of the airport VE. From Tekamah Corporation: Eric Allely and Scott Thompson, who developed the external injury modeler. From TEEX: Rick Tye, who recruited subjects for our human performance study, and Dick Volz, who assisted us in setting up and performing the experiments (as well as the firefighters who volunteered their time to participate in the study). We would also like to thank John Silva, Shaun Jones, and Rick Satava (DARPA), and Bill Pugh (NHRC) for their support of this and previous work.

This work was sponsored by the Defense Advanced Research Projects Agency (DARPA). Previous work upon which it is built was sponsored by DARPA and the Naval Health Research Center. The work was performed at Sandia National Laboratories, a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the U.S. Department of Energy under Contract DE-AC04-94AL85000.

References

Albanese, M., & Mitchell, S. (1993). Problem-based learning: A Review of Literature on Its Outcomes and Implementation Issues. *Academic Medicine*, 68, 69–76.

Allison, D., Wills, B., Hodges, L., & Wineman, J. (1997). Gorillas in the Bits. *Proceedings of the IEEE Virtual Reality Annual International Symposium*.

Badler, N., Phillips, C., & Webber, B. (1993). *Simulating Humans: Computer Graphics animation and Control.* New York: Oxford University Press.

Bell, H. (1999). The effectiveness of distributed mission training. *Communications of the ACM*, *4*(9), 72–78.

Calvin, J., Dickens, A., Gaines, R., Metzger, P., Miller, D., & Owen, D. (1993). The SIMNET virtual world architecture. *Proceedings of the IEEE Virtual Reality Annual International Symposium*, 450–455.

Chi, D., Clarke, J., Webber, B., & Badler, N. (1996). Casualty modeling for real-time medical training. *Presence: Teleoperators and Virtual Environments, 5*(4), 359–366.

Dahmann, J., Calvin, J., & Weatherby, R. (1999). A reusable architecture for simulations. *Communications of the ACM*, *42*(9), 79–84.

Delp, S., Loan, P., Basdogan, C., & Rosen, J. (1997). Surgical simulation: An emerging technology for training in emergency medicine. *Presence: Teleoperators and Virtual Environments, 6*(2), 147–159.

Dinsmore, M., Langrana, N., Burdea, G., & Ladeji, J. (1997). Virtual reality training simulation for palpation of subsurface tumors. *Proceedings of the IEEE Virtual Reality Annual International Symposium*, 54–68.

Gibson, S., Samosky, J., Mor, A., Fyock, C., Grimson, E., Kanade, T., Kikinis, R., Lauer, H., McKenzie, N., Nakajima, S., Ohkami, H., Osborne, R., & Sawada, A. (1996). *Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback* (Technical Report TR96-19). Cambridge, MA: Mitsubishi Electric Research Laboratory.

Johnson, A., Moher, S., Ohlsson, S., & Gillingham, M. (1999). The Round Earth Project: Deep learning in a collaborative virtual world. *Proceedings of the IEEE Virtual Reality Annual International Symposium*, 164–171.

Johnson, W., Rickel, J., Stiles, R., & Munro, A. (1998). Integrating pedagogical agents into virtual environments, *Presence: Teleoperators and Virtual Environments, 7*(6), 523–548.

Kaufman, D., & Bell W. (1997). Teaching and assessing clinical skills using virtual reality. *Medicine Meets Virtual Reality / Studies in Health Technology and Informatics*, *39*, 467–472.

Kizakevich, P., McCartney, M., Nissman, D., Starko, K., & Smith, N. (1998). Virtual medical trainer: Patient assessment and trauma care simulator. *Medicine Meets Virtual Reality / Studies in Health Technology and Informatics*, *50*, 309–315.

Kuppersmith, R., Johnston, R., Moreau, D., Loftin, R., & Jenkins, H. (1997). Building a virtual reality temporal bone dissection simulator. *Medicine Meets Virtual Reality / Studies in Health Technology and Informatics*, *39*, 180–186.

Loftin, R., & Kenney, P. (1995). Training the Hubble Space Telescope flight team. *IEEE Computer Graphics and Applications*, *15*(5), 31–37.

Macedonia, M., Zyda, M., Pratt, D., Barham, P., & Zeswitz, P. (1994) NPSNET: A network software architecture for large-scale virtual environments. *Presence: Teleoperators and Virtual Environments, 3*(4), 265–287.

Murata, K., Williams, D., Tills, Jack, Griffith, R. O., Gido, R. G., Tadios, E. L., Davis, F., Martinez, G., Washington, K., & Notafrancesco, A. (1997). Code manual for CONTAIN 2.0: A computer Code for nuclear reactor containment analysis (Technical Report SAND97-1735). Albuquerque: Sandia National Laboratories).

Norman, G., & Schmidt, H. (1992). Psychological basis of problem-based learning. *Academic Medicine*, *67*(9), 566–568.

Perlin, K. (1985). An image synthesizer. *Computer Graphics*, *19*(3), 287–296.

———. (1995). Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, *1*(1), 5–15.

Reece, D., & Kelly, P. (1996). An architecture for computer generated individual combatants. *Proceedings of the 6th conference on Computer Generated Forces and Behavioral Representation*.

Robb, R. (1997). Virtual endoscopy: Evaluation using the visible human datasets and comparison with real endoscopy in patients. *Medicine Meets Virtual Reality / Studies in Health Technology and Informatics*, *39*, 195–206.

Salzmann, M., Dede, C., Loftin, B., & Chen, J. (1999). A model for understanding how virtual reality aids complex conceptual learning. *Presence: Teleoperators and Virtual Environments, 8*(3), 293–316.

Satava, R., & Jones, S. (1997). Virtual environments for medical training and education. *Presence: Teleoperators and Virtual Environments, 6*(2), 139–146.

Semwal, S., Hightower, R., & Stansfield, S. (1998). Mapping algorithms for real-time control of an avatar using eight sensors. *Presence: Teleoperators and Virtual Environments, 7*(1), 1–21.

Shawver, D. (1997). Virtual actors and avatars in a flexible user-determined-scenario environment. *Proceedings of the IEEE Virtual Reality Annual International Symposium*, 170–171.

Sidell, F., Patrick, W., & Dashiell, T. (1998). *Jane's Chem-Bio Handbook*. Alexandria, VA: Jane's Information Group.

Small, S., Wuerz, R., Simon, R., Shapiro, N., Conn, A., & Setnik, G. (1999). Demonstration of high-fidelity simulation team training for emergency medicine. *Academic Emergency Medicine, 6*(4), 312–323.

Stansfield, S. (1998). Applications of virtual reality to nuclear safeguards. *Proceedings of the Joint ESARDA/INMM Workshop on Science and Modern Technology for Safeguards*, 456–462.

Stansfield, S., Shawver, D., & Sobel, A. (1998). MediSim: A prototype VR system for training medical first responders. *Proceedings of the Virtual Reality Annual International Symposium*, 198–205.

Stytz, M., Garcia, B., Godsell-Stytz, G., & Banks S. (1997). A distributed virtual environment prototype for emergency medical procedures training. *Medicine Meets Virtual Reality / Studies in Health Technology and Informatics*, *39*, 473–485.

Tate, D., Silbert, L., & King, T. (1997). Virtual environments for shipboard firefighting training. *Proceedings of the IEEE Virtual Reality Annual International Symposium*, 61–68.

Tempest Publishing. (1998). *First Responder Chem-Bio Handbook: A Practical Manual for First Responders*. Alexandria, VA: Tempest Publishing.

Thompson, S., & Allely, E. (1997). Finite state automata as a model for casualty simulation (technical report). Rockville, MD: Tekamah Corporation.

Wiet, G., Yagel, R., Stredney, D., Schmalbrock, P., Sessanna, D., Kurzion, Y., Rosenberg, L., Levin, M., & Martin, K. (1997). A volumetric approach to virtual simulation of functional endoscopic sinus surgery. *Medicine Meets Virtual Reality / Studies in Health Technology and Informatics*, *39*, 167–179.

Wiley, D., & Hahn, J. (1997). Interpolation synthesis for articulated figure motion. *Proceedings of the IEEE Virtual Reality Annual International Symposium*, 156–160.