In G. Lakemeyer and B. Nebel (Eds.), *Exploring Artificial Intelligence in the New Millennium.* San Francisco: Morgan Kaufmann Publishers, 2002.

Extending Virtual Humans to Support Team Training in Virtual Reality

Jeff Rickel and W. Lewis Johnson
Information Sciences Institute & Computer Science Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292-6695
rickel@isi.edu, johnson@isi.edu
http://www.isi.edu/isd/carte

Abstract

This paper describes the use of virtual humans and distributed virtual reality to support team training, where students must learn their individual role in the team as well as how to coordinate their actions with their teammates. Students, instructors, and virtual humans cohabit a three-dimensional, interactive, simulated mock-up of their work environment, where they can practice together in realistic situations. The virtual humans can serve as instructors for individual students, and they can substitute for missing team members, allowing students to practice team tasks when some or all human instructors and teammates are unavailable. The paper describes our learning environment, the issues that arise in developing virtual humans for team training, and our design for the virtual humans, which is an extension of our Steve agent previously used for one-on-one tutoring.

1 INTRODUCTION

Complex tasks often require the coordinated actions of multiple team members. Team tasks are ubiquitous in today's society. For example, teamwork is critical in manufacturing, in an emergency room, and in rescue operations. To perform effectively in a team, each member must master their individual role *and* learn to coordinate their actions with their teammates. There is no substitute for hands-on experience under a wide range of

situations, yet such experience is often difficult to acquire; required equipment may be unavailable for training, important training situations may be difficult to reproduce, and mistakes in the real world may be expensive or hazardous.

In such cases, distributed virtual reality (Durlach and Mavor 1995) provides a promising alternative to real world training. Students, possibly at different locations, cohabit a three-dimensional, interactive, simulated mock-up of their work environment, where they can practice together in realistic situations. Empirical results have demonstrated the effectiveness of virtual reality for training in areas as diverse as harbor navigation for submarine commanders (Hays et al. 1998), repair of the Hubble Space Telescope (Loftin and Kenney 1995), and therapy for fear of flying (Hodges et al. 1996). Nonetheless, the full potential of distributed virtual environments for education and training has only begun to be explored.

The availability of a realistic virtual environment is not sufficient to ensure effective learning. Instructors are needed to demonstrate correct performance, guide students past impasses, and point out errors that students might miss. Yet requiring instructors to continually monitor student activities places a heavy burden on their time, and may severely limit students' training time. In addition, team training requires the availability of all appropriate team members, and may require adversaries as well. Thus, while virtual environments allow students to practice scenarios anywhere and anytime, the need for instructors and a full set of teammates and adversaries can provide a serious training bottleneck.

One solution to this problem is to complement the use of human instructors and teammates with intelligent agents that can take their place when they are unavailable. The agents cohabit the virtual world with human students and collaborate (or compete) with them on training scenarios. Intelligent agents have already proven valuable in this role as fighter pilots in large battlefield simulations (Hill et al. 1997; Jones et al. 1998), but such agents have a limited ability to interact with students. Our work focuses on a different sort of intelligent agent: a virtual human that interacts with students through face-to-face collaboration in the virtual world, either as an instructor or a teammate (Johnson et al. 2000). We call our agent Steve (Soar Training Expert for Virtual Environments).

Our prior work focused on Steve's ability to provide one-on-one tutoring to students for individual tasks (Rickel and Johnson 1997; Rickel and Johnson 1999; Rickel and Johnson 2000). Steve has a variety of pedagogical capabilities one would expect of an intelligent tutoring system. For example, he can point out student errors and answer questions such as "What should I do next?" and "Why?". However, because he has an animated body, and cohabits the virtual world with his student, he can provide more human-like assistance than previous disembodied tutors can. For example, he can demonstrate actions, use gaze and gestures to direct the student's attention, and guide the student around the virtual world. This makes Steve particularly valuable for teaching tasks that require interaction with the physical world. Johnson, Rickel, and Lester (2000) discuss these and other advantages of animated pedagogical agents.

This paper describes our extensions to Steve to support team training. Steve agents can play two valuable roles: they can serve as a tutor for an individual human team member, and they can substitute for missing team members, allowing students to practice team tasks without requiring all their human teammates. Steve's prior skills provided a solid

foundation for his roles in team training, but several new issues had to be addressed. Each agent must be able to track the actions of multiple other agents and people, understand the role of each team member as well as the interdependencies, and communicate with both human and agent teammates for task coordination. In the remaining sections, we describe our learning environment for team training, present our solutions to these new issues, and discuss related and future work.

2 RELATED WORK

Although several recent systems have applied intelligent tutoring methods to team training, none of them provide embodied virtual humans like Steve that can work with students in a three-dimensional, simulated mock-up. The PuppetMaster (Marsella and Johnson 1998) serves as an automated assistant to a human instructor for large-scale simulation-based training. It monitors the activities of synthetic agents and human teams, providing high-level interpretation and assessment to guide the instructor's interventions. It models team tasks at a coarser level than Steve, and is particularly suited to tracking large teams in very dynamic situations. AETS (Zachary et al. 1998) monitors a team of human students as they run through a mission simulation using the actual tactical workstations aboard a ship, rather than a virtual mock-up. AETS employs detailed cognitive models of each team member (including eye movements, individual keystrokes, and speech) to track and remediate their performance. However, the system does not use these models to provide surrogate team members, and the automated tutor provides feedback to students only through a limited display window and by highlighting of console display elements. AVATAR (Connolly et al. 1998) provides simulation-based training for air traffic controllers. It monitors and remediates their verbal commands to simulated pilots and their console panel actions. However, the automated tutor and pilots have no planning capabilities, so scenarios must be more tightly scripted than in our approach. Perhaps the closest work to ours is the Cardiac Tutor (Eliot and Woolf 1995), which trains a medical student to lead cardiac resuscitation teams. The system includes a simulation of the patient, as well as simulated doctors, nurses and technicians that play designated team roles. However, these teammates do not appear as virtual humans; they are heard but not seen. Medical protocols, expressed as linear sequences of actions, play the role of the task knowledge that guides our agents, and the system has some abilities to dynamically adapt protocols to the stochastic simulation and to errors by the student. However, the representation of task knowledge appears less general than ours, since it is tailored particularly to trauma care. Unlike our system, where any team member could be a student or agent, their system is limited to a single student playing the role of the team leader.

Our work is also closely related to other research on embodied conversational agents (Cassell et al. 2000) and, more specifically, animated pedagogical agents (Johnson et al. 2000). Cassell and her colleagues have built several sophisticated systems that support face-to-face conversations between a pair of virtual humans (Cassell et al. 1994) and between a human user and a virtual human (Cassell and Thórisson 1999; Cassell et al. 2000). However, their work does not allow multiple agents and people to collaborate on tasks in a 3D virtual world, and they do not address any pedagogical issues. Lester and his colleagues have built two animated agents that can tutor students through tasks (Lester et al. 1999; Lester et al. 1999). However, their agents do not support team training, and they only appear as 2D characters in a 2D virtual world. More recently, they built a 3D

pedagogical agent that performs tasks in a 3D virtual world (Lester et al. 1999). However, the agent does not collaborate with students on tasks; the student specifies high-level tasks via menus, and the agent carries them out. André et al. (2000) developed multiple animated agents that collaborate to present information to a user. The domain knowledge they give to agents is similar to our approach in representing roles and communication among agents playing different roles, but their agents do not collaborate with humans on tasks, and they do not address pedagogical issues. Bindiganavale et al. (2000) developed a training system that allows multiple virtual humans to collaborate on tasks in a virtual world. However, in their system, the student learns by giving natural language instructions (task knowledge) to the agents and viewing the consequences; he can't participate in the scenario directly, and the virtual humans do not include any tutoring capabilities.

3 THE LEARNING ENVIRONMENT

Our learning environment is designed to mimic the approach used at the naval training facility in Great Lakes, Illinois, where we observed team training exercises. The team to be trained is presented with a scenario, such as a loss of fuel oil pressure in one of the gas turbine engines that propels the ship. The team must work together, guided by standard procedures, to handle the casualty. At Great Lakes, the team trains on real, operational equipment. Because the equipment is in operation, the trainers have limited ability to simulate ship casualties. For example, they mark gauges with grease pencils to indicate hypothetical readings, and they use cardboard cutouts to indicate fires. In our learning environment, the team, consisting of any combination of Steve agents and human students, is immersed in a simulated mock-up of the ship; the simulator creates the scenario conditions. As at Great Lakes, an instructor (human or agent) accompanies each student to coach them on their role.

Each student gets a three-dimensional view of the virtual world through a head-mounted display (HMD) and interacts with the world via data gloves, as shown in Figure 1. Lockheed Martin's Vista Viewer software (Stiles et al. 1995) uses data from a position and orientation sensor on the HMD to update the student's view as she moves around. Additional sensors on the gloves keep track of the student's hands, and Vista sends out messages when the student touches virtual objects. These messages are received and handled by the simulator, which controls the behavior of the virtual world. Our current implementation uses VIVIDS (Munro et al. 1997; Munro and Surmon 1997), developed at the USC Behavioral Technology Laboratories (BTL), for simulation authoring and execution. Separate audio software broadcasts environmental noises through headphones on the HMD based on the student's proximity to their source in the virtual world. Our current training environment simulates the interior of a ship, complete with gas turbine engines, a variety of consoles, and their surrounding pipes, platforms, stairs and walls. A course author can create a new environment by creating new graphical models, a simulation model, and the audio files for environmental sounds.

Our architecture for creating virtual worlds (Johnson et al. 1998), developed in collaboration with our colleagues from Lockheed Martin and BTL, allows any number of humans and agents to cohabit the virtual world. While a single simulator controls the behavior of the world, each person interacts with the world through their own copy of Vista and the audio software, and each agent runs as a separate process. The separate software com-

Figure 1 Two students interacting with a virtual ship via head-mounted displays and data gloves. Despite their physical proximity to one another, they are operating in completely different parts of the virtual ship.

ponents communicate by passing messages via a central message dispatcher; our current implementation uses Sun's ToolTalk as the message dispatcher.

This distributed architecture has several important advantages. First, since each software component runs as a separate process, the various processes can run on different machines, possibly at different locations. Second, this modular approach makes it easy to replace any of the software components with new alternatives that satisfy the same external interface. For example, VIVIDS and Vista could be replaced by a commercial product that supports the creation of interactive virtual mock-ups. Finally, the approach is extensible. Since components communicate via the message dispatcher rather than by sending messages to one another directly, one component need not know which other components are expecting its messages. This approach greatly facilitates team training, where arbitrary combinations of people and agents must cohabit the virtual world. Our extension to team training would have been more difficult had we originally designed a more monolithic system geared towards a single student and tutor.

Humans and agents communicate through spoken dialogue. An agent speaks to a person (teammate or student) by sending a message to the person's text-to-speech software, which broadcasts the utterance through the person's headphones. Our current implementation uses Entropic's TrueTalk for speech synthesis. When a person speaks, a microphone on their HMD sends their utterance to speech recognition software, which broadcasts a semantic representation of the utterance to all the agents. The person activates speech recognition prior to each utterance by touching their two index fingers together; Vista detects the signal from the gloves and sends a message to activate that person's speech recognition software. Our current implementation uses Entropic's GrapHvite for speech recognition. Currently, Vista provides no direct support for human-to-human conversation; if the humans are not located in the same room, they must use telephone or radio to hear one another.

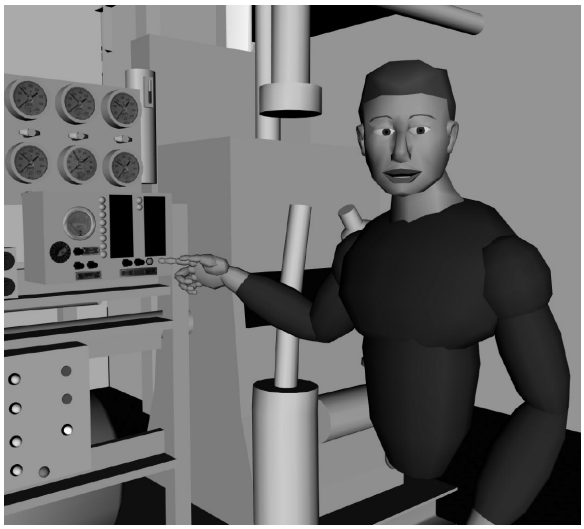

Figure 2 Steve describing an indicator light

For team training, teammates and instructors must be able to track each other's activities. Each person sees each other person in the virtual world as a head and two hands. The head is simply a graphical model, so each person can have a distinct appearance, possibly with his or her own face texture-mapped onto the graphical head. Each Vista tracks the position and orientation of its person's head and hands via the sensors, and it broadcasts the information to agents and the other Vistas. Each agent appears as a human upper body, as shown in Figure 2. To distinguish different agents, each agent can be configured with its own shirt, hair, eye, and skin color. Its voice can be made distinct by setting its speech rate, baseline pitch, and vocal tract size; these parameters are supported by the TrueTalk software. The agents, of course, do not need audio or visual cues to distinguish other agents and humans; each Vista and speech recognizer indicates in its messages which person it is tracking, and agents send out similar messages about their activities.

4 EXAMPLE

To illustrate our learning environment and Steve's current capabilities, consider a training scenario in which two human students, Jack and Jill, are learning their roles in handling a loss of fuel oil pressure in one of the gas turbine engines aboard a ship. Jack is serving as the Propulsion and Auxiliary Control Console (PACC) operator, and Jill is in the engine room. Each student is assigned a Steve agent as their tutor. In addition, three other Steve agents serve as their teammates: one serves as the Engineering Officer of the Watch (EOOW), one serves as the Electrical Plant Control Console (EPCC) operator, and one serves as the Shaft Control Unit (SCU) operator.

Jack's tutor looks at him and introduces the scenario: "Let me show you how to handle a loss of fuel oil pressure. First, when you detect it, inform the EOOW." Looking over at the

EOOW, the tutor continues, "We have a loss of fuel oil pressure in engine room one." The EOOW nods in acknowledgment and passes the message on to the engine room. Jack's tutor leads him over to the normal stop button, points at it, and says, "First, press the normal stop button to stop the turbine." The tutor presses the button, and Jack watches its indicator light up and the engine's power lever angle go to idle. "I will now transfer thrust control to the central control station," the tutor informs Jack. Jack, believing that he remembers the procedure, says, "Let me finish." "Okay, you finish," replies the tutor, who shifts to monitoring Jack's performance of the task.

Jack steps forward to the console and presses the wrong button. "No," the tutor comments while shaking his head. Jack, suddenly less sure of himself, asks, "What should I do?" The tutor replies, "I suggest that you press the CCS button to initiate the transfer." Jack presses the button and his tutor nods approvingly. As a result of Jack's action, the CCS button blinks on both the PACC and the SCU. The SCU operator, in the engine room, presses the blinking CCS button on his console to complete the transfer, and the button stops blinking and remains illuminated on both consoles.

Jack looks over at the EOOW and says, "Thrust control is now at the central control station." The EOOW nods in acknowledgment and instructs the EPCC operator to switch to generator one. Jack watches as the agent operator pushes a series of buttons and informs the EOOW when the switch is complete. Now the EOOW commands the engine room to investigate the cause of the casualty.

Upon receiving the command, Jill's agent says, "Let me show you how to check for the cause of the loss of fuel oil pressure. First, check that all suction valves are wide open. A partially closed valve in the suction line can increase the suction lift above the pump's capabilities." The agent guides Jill around the engine room and shows her the location of the valves. They check each one, but all are already wide open.

Next, the agent leads Jill over to the relief valve. As she gets close, she can hear the sound of the oil passing through the valve. Her tutor continues, "Next, check the relief valve set pressure. As you can hear from the sound of the oil passing through the valve, the set pressure is too low, causing the loss of fuel oil pressure." The agent shows Jill how to reset the relief valve lifting pressure, then reports back to the EOOW, "The cause of the casualty has been determined and corrected."

Although this scenario does not illustrate all of the capabilities of our learning environment, it highlights some of the most important. Jack and Jill cohabit a virtual mock-up of their work environment, along with Steve agents that serve as their tutors and teammates. They can navigate around the environment to learn the location of relevant equipment, often under the guidance of their agent tutor. Agents and students can manipulate objects in the virtual world and see their visual and auditory effects. Finally, students can collaborate with each other, as well as their agent tutor and teammates, to practice realistic training scenarios.

5 AGENT DESIGN

5.1 ARCHITECTURE

To collaborate with students and other agents in a virtual world, Steve must be able to perceive the state of the world, choose appropriate actions, and execute those actions to change the state of the world. Thus, as shown in Figure 3, each Steve agent consists of three main modules: perception, cognition, and motor control (Rickel and Johnson 1999). The perception module monitors messages from other software components, identifies relevant events, and maintains a snapshot of the state of the world. It tracks the following information: the simulation state (in terms of objects and their attributes), actions taken by students and other agents, the location of each student and agent, and human and agent speech (separate messages indicate the beginning of speech, the end, and a semantic representation of its content). In addition, if the agent is tutoring a student, it keeps track of the student's field of view; messages from the student's Vista indicate when objects enter or leave the field of view. The cognition module, implemented in Soar (Laird et al. 1987; Newell 1990), interprets the input it receives from the perception module, chooses appropriate goals, constructs and executes plans to achieve those goals, and sends out motor commands to the motor control module. Steve's cognition module can typically react to new perceptual input in a fraction of a second, so it is very responsive (Rickel and Johnson 2000). The motor control module accepts the following types of commands: move to an object, point at an object, manipulate an object (about ten types of manipulation are currently supported), look at someone or something, change facial expression, nod or shake the head, and speak. The motor control module decomposes these motor commands into a sequence of lower-level messages that are sent to the other software components (simulator, Vista Viewers, speech synthesizers, and other agents) to realize the desired effects. See Rickel and Johnson (1999) for more details on this architecture.

To allow Steve to operate in a variety of domains, his architecture has a clean separation between domain-independent capabilities and domain-specific knowledge. The code in the perception, cognition, and motor control modules provides a general set of capabilities that are independent of any particular domain. These capabilities include planning, replanning, and plan execution; mixed-initiative dialogue; assessment of student actions; question answering ("What should I do next?" and "Why?"); episodic memory; path planning; communication with teammates; and control of the agent's body (Rickel and Johnson 1999). To allow Steve to operate in a new domain, a course author simply specifies the appropriate domain knowledge in a declarative language. (Recent work has focused on acquiring the knowledge from an author's demonstrations and the agent's experimentation (Angros 2000; Angros et al. 2002)). The knowledge falls in two categories: perceptual knowledge (knowledge about objects in the virtual world, their relevant simulator attributes, and their spatial properties) and task knowledge (procedures for accomplishing domain tasks and text fragments for talking about them). For details about Steve's perceptual knowledge, see Rickel and Johnson (1999); the remainder of the paper will focus on Steve's representation and use of task knowledge.

Low-level animation of Steve's body runs as software that is linked into the Vista Viewers rather than into Steve. Before each graphical frame is rendered (about 15 to 30 times per second), Vista calls the animation code to update Steve's body position. The animation code is controlled by messages it receives from Steve's motor control module. Because the

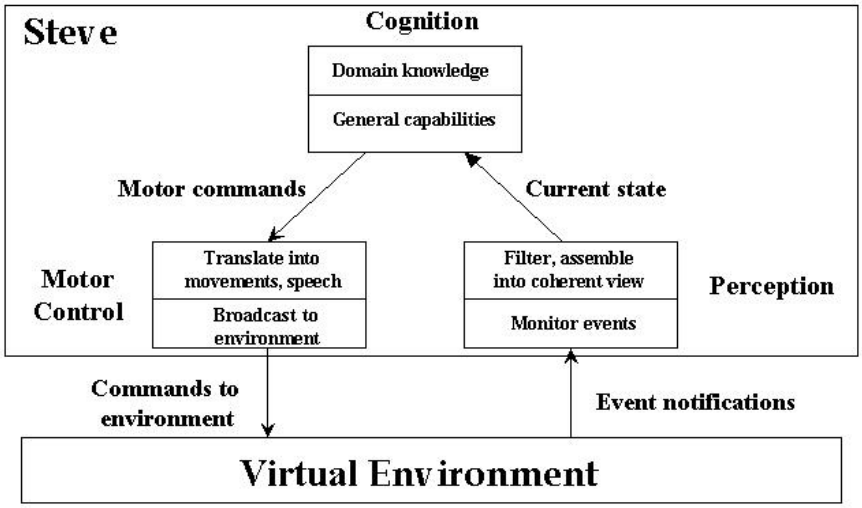

Figure 3 Steve's three main modules (perception, cognition, and motor control) and the types of information they send and receive.

animation code controls the dynamics of all body motions, the motor control module need only specify the type of motion it wants. The animation code generates all movements dynamically; there are no keyframes or canned animations. Movements involving different parts of the body can be performed simultaneously, and a new command to a body part interrupts any existing motion for that part. Steve's motor control module isolates the details of how to control the animation code, allowing the cognition module to send higher-level motor commands that do not depend on those details. This approach makes it easy to replace Steve's current body with a new one, such as a commercial product for human figure animation.

5.2 REPRESENTING TASK KNOWLEDGE

Most of Steve's abilities to collaborate with students on tasks, either as a teammate or tutor, stem from his understanding of those tasks. As the scenario unfolds, Steve must always know which steps are required, how they contribute to the task goals, and who is responsible for their execution. In order to handle dynamic environments containing other people and agents, he must understand the tasks well enough to adapt them to unexpected events; he cannot assume that the task will follow a pre-specified sequence of steps. Moreover, our goal was to support a declarative representation that would allow course authors to easily specify task knowledge and update it when necessary (Rickel and Johnson 1997).

Our representation for individual tasks, used in our previous work for one-on-one tutoring, satisfies these design criteria. The course author describes each task using a standard plan representation (Russell and Norvig 1995). First, each task consists of a set of steps, each of which is either a primitive action (e.g., press a button) or a composite action (i.e., itself a

Task transfer-thrust-control-ccs
Steps press-pacc-ccs, press-scu-ccs
Causal links
press-pacc-ccs achieves ccs-blinking for press-scu-ccs
press-scu-ccs achieves thrust-at-ccs for end-task
Ordering press-pacc-ccs before press-scu-ccs
Roles pacc: press-pacc-ccs; scu: press-scu-ccs

Figure 4 An example team task description

task). Composite actions give tasks a hierarchical structure. Second, there may be ordering constraints among the steps; these constraints define a partial order over the steps. Finally, the role of the steps in the task is represented by a set of causal links (McAllester and Rosenblitt 1991). Each causal link specifies that one step in the plan achieves a particular goal that is a precondition for another step in the plan or for termination of the task. For example, pulling out a dipstick achieves the goal of exposing the level indicator, which is a precondition for checking the oil level.

This task representation is suitable for structured tasks based on standard procedures. It would not be suitable for tasks that require creative problem solving, such as design tasks. Fortunately, many tasks in industry and the military have this type of structure, including operation and maintenance of equipment, trauma care (Eliot and Woolf 1995), and surgical procedures (Billinghurst and Savage 1996). Moreover, this representation need not be viewed as a fixed sequence of steps. Rather, it is a general causal network of steps and goals, and can be used by a planning algorithm to dynamically order the steps even in the face of unexpected events, as described in the next subsection (Using Task Knowledge).

To extend Steve to team training, we had to decide how to assign team members to task steps. Much of the research on multi-agent teams has addressed methods by which teammates dynamically negotiate responsibility for task steps. However, supporting such negotiation among a team of agents *and* people would require more sophisticated natural language dialogue capabilities than Steve currently has. Fortunately, many team tasks have well-defined roles that are maintained throughout task execution. We focus on this class of tasks.

Extending Steve to support such team tasks required one addition to each task description: a mapping of task steps to team roles. For example, Figure 4 shows a simplified task model for transferring thrust control to the central control station of a ship. Two roles must be filled: one operator mans the propulsion and auxiliary control console (PACC) in the central control station (CCS), and another operator mans the shaft control unit console (SCU) in the engine room. The PACC operator requests the transfer by pressing the CCS button on her console, which results in the CCS button blinking on both consoles. When the CCS button is blinking on the SCU, the SCU operator presses it to finalize the transfer. This last action achieves the end goal of the task, which is indicated in the task description by specifying its effect as a precondition of the dummy step "end-task." This representation for end goals is standard in AI planners (Russell and Norvig 1995).

Task loss-of-fuel-oil-pressure
Steps transfer-thrust-control-ccs, ...
Causal links ...
Ordering ...
Roles
eoow: (transfer-thrust-control-ccs pacc), ... ;
engrm: (transfer-thrust-control-ccs scu), ...

Figure 5 Specifying roles for a subtask

If a step in the task is itself a team task, it will have its own roles to be filled, and these may differ from the roles in the parent task. Therefore, the parent task specifies which of its roles plays each role in the subtask. For example, Figure 5 shows a partial description of a task for which the task in Figure 4 is a subtask. This task description calls for the engineering officer of the watch (EOOW) to play the role of the PACC operator and for the engine room officer (ENGRM) to play the role of the SCU operator for the transfer of thrust control.

Task descriptions (e.g., Figures 4 and 5) specify the structure of tasks, but they leave the goals (e.g., ccs-blinking) and primitive steps (e.g., press-pacc-ccs) undefined. The course author defines each primitive step as an instance of some action in Steve's extensible action library. For example, the step press-pacc-ccs would be defined as an instance of press-button in which the particular button to be pressed is pacc-ccs, the name of an object in the virtual world. The course author defines each goal by the conditions in the simulated world under which it is satisfied. For example, ccs-blinking is satisfied when the simulator attribute scu-ccs-state has the value "blinking." Thus, Steve is able to relate his task knowledge to objects and attributes in the virtual world.

5.3 USING TASK KNOWLEDGE

Any combination of human students, agent teammates, and human and agent instructors can participate in a training scenario. A request for a new scenario can come from a variety of sources, such as a human instructor or student (via a graphical user interface) or a curriculum sequencing software component. Regardless of the source, the request is sent to each Steve agent as a message. The message specifies the name of the task to be performed (e.g., transfer-thrust-control-ccs or loss-of-fuel-oil-pressure) and assigns a student or agent to each role in that task. If an agent was previously assigned a student to tutor, and the agent or student is assigned a role in the team task, the agent coaches the student on that role. If an agent has not been assigned a student to tutor, and is given a role in the task, the agent simply serves as the missing teammate. Currently, a single Steve agent cannot work with multiple students and cannot play one role while tutoring a student on another role. However, anyone (student or agent) can play multiple roles in a task; if an agent tutor or its student is assigned multiple roles, they work together on all those roles.

Given a request for a team task, each Steve agent involved in the task as a team member or instructor uses its task knowledge to construct a complete task model. Starting with the

task description for the specified task, each agent recursively expands any composite step with its task description, until the agent has a fully decomposed, hierarchical task model. Role assignments in the request are propagated down to subtasks until the task model specifies which team member is responsible for each step. For example, if the task is loss-of-fuel-oil-pressure (Figure 5), with Joe as the EOOW, then Joe will play the role of the PACC for the subtask transfer-thrust-control-ccs (Figure 4), and hence he is responsible for the step press-pacc-ccs. Typically, all agents have the same task knowledge, so each agent will construct the same hierarchical task model with the same assignment of responsibilities. (Although our approach would permit different agents to have different knowledge.)

For simulation-based training, especially for team tasks, agents must be able to robustly handle unexpected events. Scripting an agent's behavior for all possible contingencies in a dynamic virtual world is difficult enough, but the problem is compounded when each agent must be scripted to handle unexpected actions by any human team member. One option is simply to prevent human students from deviating from standard procedures, but this robs the team of any ability to learn about the consequences of mistakes and how to recover from them. Instead, we have designed Steve to use his task knowledge to adapt task execution to the unfolding scenario.

To do this, each agent maintains a plan for how to complete the task from the current state of the world. The task model specifies all steps that might be required to complete the task; it can be viewed as a worst-case plan. Agents continually monitor the state of the virtual world, identify which goals in the task model are already satisfied, and use a partial-order planning algorithm to construct a plan for completing the task (Rickel and Johnson 1999). This plan is a subset of the task model, consisting of the steps relevant to completing the task, the ordering constraints among them, and the causal links that indicate the role of each step in achieving the end goals. In our prior work, this plan would specify how an agent intended to complete a task. For team training, the plan specifies how the agent intends for the team to collectively complete the task, with some causal links specifying the interdependencies among team members (i.e., how one team member's action depends on a precondition that must be achieved by a teammate). Thus, agents dynamically interleave construction, revision, and execution of plans to adapt to the unfolding scenario.

The evolving task context (task model and current plan for completing it) guides the behavior of the agents. If an agent is serving only as a missing team member, it simply performs its role in the task, waiting when appropriate for the actions of its teammates, and communicating with them when necessary. In contrast, an agent serving as an instructor for a human student interacts with that student in a manner similar to one-on-one tutoring. The agent can demonstrate the student's role in the task, explaining each action it takes, or it can monitor the student as she performs the task, providing feedback on the student's actions and answering questions ("What next?" and "Why?") when the student needs help. Moreover, the agent instructor can easily shift between these two modes as the task proceeds; the student can always interrupt the agent's demonstration and ask to finish the task herself, and she can always ask the agent to demonstrate a step when she gets stuck. An agent instructor is responsible for handling student errors; currently, agent teammates do not provide any pedagogical feedback. The tutorial interactions between an agent instructor and its student arise from the agent's domain-independent capabilities for one-on-one instruction, detailed in our earlier papers (Rickel and Johnson

1999; Rickel and Johnson 2000), operating over the representation of the current team task context (described above) and the state of the dialogue between the agent and student (as described in the earlier papers). These tutorial capabilities only had to be extended to handle several new cases, such as students requesting help when they are not responsible for the next step.

5.4 TEAM COMMUNICATION

In team tasks, coordination among team members is critical. Although team members can sometimes coordinate their actions by simply observing the actions of their teammates, spoken communication is typically required. Team leaders need to issue commands. Team members often need to inform their teammates when a goal has been achieved, when they are starting an activity, and when they detect an abnormal condition. Because team communication is so important, it must be taught and practiced in team training.

To allow agents to communicate with their teammates and teach appropriate communication to their students, they must have a model of the appropriate communication. Agents must know what to communicate, when to say it, and how to say it. When a teammate says something, they must know how to interpret it and how it relates to the task. Addressing these issues required extensions to Steve's prior abilities.

We model team communication as explicit speech acts in the task descriptions. For the sort of structured tasks we have studied, this is natural; all the documented team procedures given to us specified when one team member should say something to another and how it should be said. To support this, we extended Steve's action library to include a new type of action: a speech act from one team member to another. Each speech act appears as a primitive action in the task description. This allows us to explicitly model its relationship to the task, including the role responsible for performing it, ordering constraints on when it should be said, and causal links that specify how its effect contributes to completing the task (i.e., which other steps depend on that result).

The definition of a particular speech act includes five components:

- It specifies the name of the attribute being communicated. For example, for a speech act informing the EOOW that thrust control has been transferred to the central control station, the attribute being communicated is thrust-location. The attribute may or may not be an actual simulator attribute.
- It specifies the value being communicated for that attribute. In the example above, the value is ccs (i.e., central control station).
- It specifies the appropriate text string. For the example above, the appropriate phrase might be "Thrust control is now at the central control station." Note that we do not want to leave this text string up to a natural language generator, because there is often a precise utterance that should be used.
- It specifies the semantic representation that will be returned by a student's speech recognizer when she says the utterance or some acceptable variant of it. Our speech recognition software currently maps each path through its grammar to a symbol that represents its content, but a more structured representation could also be used.

- It specifies the name of the task role to which the speech act is directed (e.g., EOOW). Recall that the task description containing the speech act specifies the role responsible for performing the speech act, so together these two pieces of information fully specify the appropriate speaker and hearer.

Given this representation for team communication, Steve agents can both generate and comprehend such utterances during task execution. When an agent's plan calls for it to execute one of these speech acts, it sends the text string to appropriate speech synthesizers for its human teammates to hear, and it broadcasts the semantic representation of the speech act for its agent teammates to "hear." When a human says the appropriate utterance, her speech recognizer identifies it as a path through its domain-specific grammar, maps it to an appropriate semantic representation, and broadcasts it to the agents. When an agent receives such a message from a person or another agent, it checks its plan to see if it expects such a speech act from that teammate at that time. If so, it updates the specified attribute in its mental state with the specified value, and it nods to the speaker in acknowledgment. If a speech recognizer fails to understand a student's utterance, or the utterance is not appropriate at the current time, the student's instructor agent is responsible for giving the student appropriate feedback.

There are several important points about this approach. First, it only applies to structured tasks for which the required team communications can be specified in the task description; it will not suffice for tasks that require more arbitrary communication. Fortunately, many well-structured team tasks, particularly in the military, include such a prescribed set of utterances. Second, since Steve does not include any natural language understanding abilities, all valid variations of the utterances must be added to the grammar for the speech recognizer. Again, this is reasonable for tasks with prescribed utterances. Third, note the difference between our approach and communication messages in a purely multi-agent system; a speech recognizer cannot tell to whom an utterance is intended, so agents must use the task model to determine whether the speaker is addressing them. Finally, each agent must treat a human student and their instructor as jointly performing a role; if either of them generates the speech act, it must be treated as coming from that role.

Although spoken communication is typically required for team tasks, nonverbal communication is also important. Human students can observe the actions of their nearby agent and human teammates, which is often required for proper team coordination. Agents look at a teammate when expecting them to do something, which can cue a student that she is responsible for the next step. Agents look at the teammate to whom they are speaking (Figure 6), allowing students to follow the flow of communication and recognize when they are being addressed. Finally, agents react to their teammates' actions; they look at objects being manipulated by teammates (Figure 7), and they nod in acknowledgment when they understand something a teammate says to them. All these nonverbal acts are generated from the agents' domain-independent capabilities (Rickel and Johnson 2000); they need not be included in task models. For tasks that require face-to-face collaboration among team members, such nonverbal communication is critical.

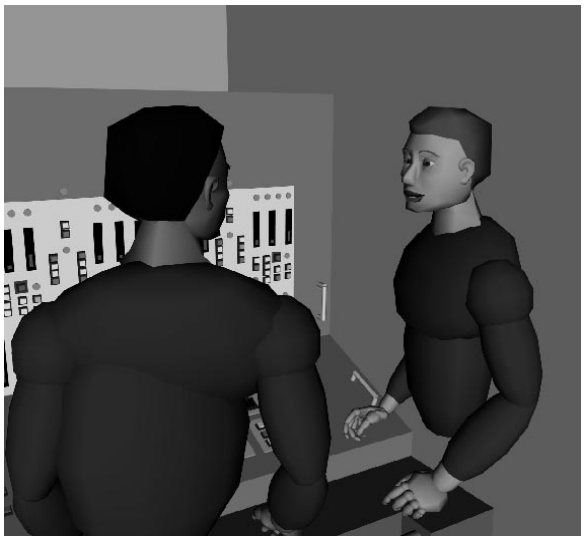

Figure 6 One Steve agent speaking to another

6 DISCUSSION

6.1 STATUS

Steve has been tested on a variety of naval operating procedures. In our most complicated team scenario, five team members must work together to handle a loss of fuel oil pressure in one of the gas turbine engines. This task involves a number of subtasks, some of which are individual tasks while others involve subteams. All together, the task consists of about three dozen actions by the various team members. Steve agents can perform this task themselves as well as in concert with human team members. All of Steve's algorithms were designed for efficient real-time performance; there are no noticeable delays in the agents' behavior, even when running several agents on the same computer. However, the Vista Viewer software requires a high-end Silicon Graphics computer to achieve a reasonable frame rate given the complexity of the 3D ship model and the number of agents it must animate. We have not yet evaluated the training effectiveness of our system for team training, but we hope to perform a formal evaluation with actual naval personnel in the future.

7 FUTURE WORK

Some important limitations in our system could be alleviated by incorporating recent research results from related areas. To go beyond tasks with prescribed utterances, we could leverage ongoing research on robust spoken dialogue (Allen et al. 1996; Smith and Hipp 1994). To handle tasks with shifting roles and unstructured communication among teammates, we could incorporate a more general theory of teamwork (Jennings 1995; Levesque et al. 1990; Tambe 1997). To handle tasks that involve simultaneous physical

Figure 7 One Steve agent watching another

collaboration (e.g., two people jointly lifting a heavy object), we will need a tighter coupling of Steve's perception and body control (Badler et al. 1993). Although research in these areas is still incomplete, many useful methods have been developed.

Steve agents serving as instructors should provide more information on team activities than they currently do. They should provide running commentary on relevant actions of teammates, which can be difficult because it requires the ability to synchronize verbal descriptions with real-time events in the virtual world (André et al. 2000). They should indicate perceptual cues that can help the student track teammates' actions. They should interleave their demonstrations of the student's role with descriptions of how and why the student's actions are needed by teammates and vice versa. Our representation of team tasks should support all these capabilities, but they have not yet been added to Steve's tutorial repertoire.

Another limitation in Steve agents is that they never make mistakes. This issue was raised by trainers at the Great Lakes naval training facility, who emphasized the importance of learning to recognize and recover from mistakes by teammates. Our approach of allowing students to make mistakes so that the team can learn their consequences and how to recover from them is consistent with this learning objective. However, a student can currently only learn from her own mistakes and those of her human teammates. To better support this learning objective, we must incorporate an appropriate model of task errors into Steve agents. For example, we might generate errors automatically by degrading an agent's perception (e.g., it fails to notice something in the virtual world) or task knowledge.

There is a growing understanding of the principles behind effective team training (Blickensderfer et al. 1997; Burns et al. 1993; Smith-Jentsch et al. 1998; Swezey and Salas 1992). Empirical experiments are beginning to tease out the skills that make teams effective (e.g.,

Figure 8 An interactive peacekeeping scenario featuring (left to right) a sergeant, a mother, and a medic

task skills vs. team skills), the basis for team cohesion (e.g., shared mental models), the best types of feedback (e.g., outcome vs. process), and the best sources of feedback (e.g., instructor vs. teammate). Because our approach allows us to model face-to-face interaction among human instructors and students and their agent counterparts, we are now in an excellent position to incorporate and experiment with a variety of these new ideas in team training.

We are currently addressing many of the issues discussed above in an ambitious new project (Rickel et al. 2001; Swartout et al. 2001). In contrast to our prior work, which focused on teaching well-defined tasks, our new project focuses on leadership training and decision making in stressful situations. Figure 8 shows a screen shot from a prototype implementation of an example application, in which a young lieutenant (human user) is being trained for a peacekeeping mission. In the current implementation, there are three Steve agents that interact with the lieutenant: a medic (front right) that serves as his teammate, a Bosnian mother (front center) whose boy has been accidentally injured by one of the lieutenant's vehicles, and a sergeant (front left) who serves as both a teammate and mentor. All other characters (soldiers and an angry crowd of locals) are simple scripted agents.

This new type of team training exploits all of Steve's prior capabilities, but it is pushing us in several new directions. First, we are aiming for more realism in the graphical bodies; in our latest implementation, shown in Figure 8, the bodies and animation algorithms for all the Steve agents and scripted characters were developed by Boston Dynamics Incorporated. Second, since the human user must collaborate with his agent teammates to formulate novel plans, rather than simply execute well-defined procedures, we are integrating state-of-the-art natural language understanding and generation algorithms into Steve, as well as extending those algorithms to handle multi-party conversations in immersive virtual worlds (Traum and Rickel 2002). Third, to model the behavior of teammates in stressful situations, as well as create virtual humans that can induce stress in the human user by reacting emotionally, we have integrated a computational model of emotions into Steve (Gratch and Marsella 2001). For example, the mother in our

peacekeeping scenario becomes increasingly angry at the lieutenant if his decisions thwart her goal of getting assistance for her boy. Our challenge as we integrate these extensions into Steve is to maintain the efficient real-time performance and ease of authoring of the approach described in this paper.

8 CONCLUSION

This paper has described a new approach to team training based on virtual humans in distributed virtual reality. The approach builds on our learning environment and Steve agent developed for one-on-one tutoring of individual tasks. In our new learning environment, any combination of human students, agent teammates, and human and agent instructors can cohabit a three-dimensional, interactive, simulated mock-up to practice team tasks. To support such training scenarios, Steve was extended in several ways. Rather than tracking the activities of a single student, Steve now tracks the actions of multiple other agents and people. Steve's representation for task knowledge was extended to include a representation for team roles, allowing each agent to understand the role of each team member as well as the interdependencies. Agents use their task knowledge to dynamically maintain a plan specifying how the team can collectively complete their task from the current situation. Because communication is essential for team coordination, we model such communication as explicit speech acts in the agents' task knowledge, allowing them to understand what to communicate, when and how to say it, and how the communication facilitates completion of their task. In addition, our use of embodied virtual humans supports the many types of nonverbal communication that are essential for face-to-face collaboration. Virtual humans and distributed virtual reality provide students the ability to practice team tasks in realistic scenarios anywhere and anytime, providing a training environment that is safer, cheaper, and more flexible than physical alternatives.

9 ACKNOWLEDGMENTS

This work was funded by the Office of Naval Research under grant N00014-95-C-0179 and AASERT grant N00014-97-1-0598. We are grateful to our collaborators who developed the other software components on which Steve relies. Randy Stiles and his colleagues at Lockheed Martin developed the visual interface software (Vista Viewer). Allen Munro and his colleagues at the Behavioral Technology Laboratories developed the simulator (VIVIDS). Ben Moore at ISI developed the speech recognition and audio components. Finally, Marcus Thiébaux at ISI developed the 3D model of Steve's body and the code in the visual interface software that controls its animation. Our new work is funded by the Army Research Office through the USC Institute for Creative Technologies under contract DAAD19-99-C-0046.

References

Allen, J. F., B. W. Miller, E. K. Ringger, and T. Sikorski (1996). Robust understanding in a dialogue system. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, San Francisco, CA, pp. 62–70. Morgan Kaufmann.

André, E., T. Rist, S. van Mulken, M. Klesen, and S. Baldes (2000). The automated design of believable dialogues for animated presentation teams. In J. Cassell, J. Sullivan,

S. Prevost, and E. Churchill (Eds.), *Embodied Conversational Agents*. Cambridge, MA: MIT Press.

Angros, Jr., R. (2000). *Learning What to Instruct: Acquiring Knowledge from Demonstrations and Focussed Experimentation*. Ph. D. thesis, Department of Computer Science, University of Southern California, Los Angeles, CA.

Angros, Jr., R., W. L. Johnson, J. Rickel, and A. Scholer (2002). Learning domain knowledge for teaching procedural tasks. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, New York. ACM Press. Forthcoming.

Badler, N. I., C. B. Phillips, and B. L. Webber (1993). *Simulating Humans*. New York: Oxford University Press.

Billinghurst, M. and J. Savage (1996). Adding intelligence to the interface. In *Proceedings of the IEEE Virtual Reality Annual International Symposium (VRAIS '96)*, Los Alamitos, CA, pp. 168–175. IEEE Computer Society Press.

Bindiganavale, R., W. Schuler, J. M. Allbeck, N. I. Badler, A. K. Joshi, and M. Palmer (2000). Dynamically altering agent behaviors using natural language instructions. In *Proceedings of the Fourth International Conference on Autonomous Agents*, New York, pp. 293–300. ACM Press.

Blickensderfer, E., J. A. Cannon-Bowers, and E. Salas (1997). Theoretical bases for team self-correction: Fostering shared mental models. *Advances in Interdisciplinary Studies of Work Teams 4*, 249–279.

Burns, J. J., E. Salas, and J. A. Cannon-Bowers (1993). Team training, mental models, and the team model trainer. In *Proceedings of Advancement in Integrated Delivery Technologies*, Denver, CO.

Cassell, J., T. Bickmore, L. Campbell, H. Vilhjálmsson, and H. Yan (2000). Conversation as a system framework: Designing embodied conversational agents. In J. Cassell, J. Sullivan, S. Prevost, and E. Churchill (Eds.), *Embodied Conversational Agents*. Cambridge, MA: MIT Press.

Cassell, J., C. Pelachaud, N. Badler, M. Steedman, B. Achorn, T. Becket, B. Douville, S. Prevost, and M. Stone (1994). Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. In *Proceedings of ACM SIGGRAPH '94*, Reading, MA, pp. 413–420. Addison-Wesley.

Cassell, J., J. Sullivan, S. Prevost, and E. Churchill (Eds.) (2000). *Embodied Conversational Agents*. Cambridge, MA: MIT Press.

Cassell, J. and K. R. Thórisson (1999). The power of a nod and a glance: Envelope vs. emotional feedback in animated conversational agents. *Applied Artificial Intelligence 13*, 519–538.

Connolly, C. A., J. Johnson, and C. Lexa (1998). AVATAR: An intelligent air traffic control simulator and trainer. In *Proceedings of the Fourth International Conference on Intelligent Tutoring Systems (ITS '98)*, Number 1452 in Lecture Notes in Computer Science, Berlin, pp. 534–543. Springer.

Durlach, N. I. and A. S. Mavor (Eds.) (1995). *Virtual Reality: Scientific and Technological Challenges*. Washington, D.C.: National Academy Press.

Eliot, C. and B. P. Woolf (1995). An adaptive student centered curriculum for an intelligent training system. *User Modeling and User-Adapted Instruction 5*, 67–86.

Gratch, J. and S. Marsella (2001). Tears and fears: Modeling emotions and emotional behaviors in synthetic agents. In *Proceedings of the Fifth International Conference on Autonomous Agents*, New York, pp. 278–285. ACM Press.

Hays, R. T., D. A. Vincenzi, A. G. Seamon, and S. K. Bradley (1998). Training effectiveness evaluation of the VESUB technology demonstration system. Technical Report 98-003, Naval Air Warfare Center Training Systems Division, Orlando, FL.

Hill, Jr., R. W., J. Chen, J. Gratch, P. Rosenbloom, and M. Tambe (1997). Intelligent agents for the synthetic battlefield: A company of rotary wing aircraft. In *Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence (IAAI-97)*, Menlo Park, CA, pp. 1006–1012. AAAI Press.

Hodges, L. F., B. O. Rothbaum, B. Watson, G. D. Kessler, and D. Opdyke (1996). A virtual airplane for fear of flying therapy. In *Proceedings of the IEEE Virtual Reality Annual International Symposium (VRAIS '96)*, Los Alamitos, CA, pp. 86–93. IEEE Computer Society Press.

Jennings, N. (1995). Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence 75*.

Johnson, W. L., J. Rickel, R. Stiles, and A. Munro (1998, December). Integrating pedagogical agents into virtual environments. *Presence: Teleoperators and Virtual Environments 7*(6), 523–546.

Johnson, W. L., J. W. Rickel, and J. C. Lester (2000). Animated pedagogical agents: Face-to-face interaction in interactive learning environments. *International Journal of Artificial Intelligence in Education 11*, 47–78.

Jones, R. M., J. E. Laird, and P. E. Nielsen (1998). Automated intelligent pilots for combat flight simulation. In *Proceedings of the Tenth Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, Menlo Park, CA, pp. 1047–1054. AAAI Press.

Laird, J. E., A. Newell, and P. S. Rosenbloom (1987). Soar: An architecture for general intelligence. *Artificial Intelligence 33*(1), 1–64.

Lester, J. C., B. A. Stone, and G. D. Stelling (1999). Lifelike pedagogical agents for mixed-initiative problem solving in constructivist learning environments. *User Modeling and User-Adapted Interaction 9*, 1–44.

Lester, J. C., J. L. Voerman, S. G. Towns, and C. B. Callaway (1999). Deictic believability: Coordinating gesture, locomotion, and speech in lifelike pedagogical agents. *Applied Artificial Intelligence 13*, 383–414.

Lester, J. C., L. S. Zettlemoyer, J. Gregoire, and W. H. Bares (1999). Explanatory lifelike avatars: Performing user-designed tasks in 3d learning environments. In *Proceedings of the Third International Conference on Autonomous Agents*, New York. ACM Press.

Levesque, H. J., P. R. Cohen, and J. H. T. Nunes (1990). On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, Los Altos, CA, pp. 94–99. Morgan Kaufmann.

Loftin, R. and P. Kenney (1995). Training the hubble space telescope flight team. *IEEE Computer Graphics and Applications 15*(5), 31–37.

Marsella, S. C. and W. L. Johnson (1998). An instructor's assistant for team-training in dynamic multi-agent virtual worlds. In *Proceedings of the Fourth International Conference on Intelligent Tutoring Systems (ITS '98)*, Number 1452 in Lecture Notes in Computer Science, Berlin, pp. 464–473. Springer.

McAllester, D. and D. Rosenblitt (1991). Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, Menlo Park, CA, pp. 634–639. AAAI Press.

Munro, A., M. Johnson, Q. Pizzini, D. Surmon, and D. Towne (1997). Authoring simulation-centered tutors with RIDES. *International Journal of Artificial Intelligence in Education 8*, 284–316.

Munro, A. and D. Surmon (1997). Primitive simulation-centered tutor services. In *Proceedings of the AI-ED Workshop on Architectures for Intelligent Simulation-Based Learning Environments*, Kobe, Japan.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Rickel, J., J. Gratch, R. Hill, S. Marsella, and W. Swartout (2001, March). Steve goes to Bosnia: Towards a new generation of virtual humans for interactive experiences. In *AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*.

Rickel, J. and W. L. Johnson (1997). Intelligent tutoring in virtual reality: A preliminary report. In *Proceedings of the Eighth World Conference on Artificial Intelligence in Education*, pp. 294–301. IOS Press.

Rickel, J. and W. L. Johnson (1999). Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence 13*, 343–382.

Rickel, J. and W. L. Johnson (2000). Task-oriented collaboration with embodied agents in virtual worlds. In J. Cassell, J. Sullivan, S. Prevost, and E. Churchill (Eds.), *Embodied Conversational Agents*. Cambridge, MA: MIT Press.

Russell, S. and P. Norvig (1995). *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall.

Smith, R. W. and D. R. Hipp (1994). *Spoken Natural Language Dialog Systems*. New York: Oxford University Press.

Smith-Jentsch, K. A., R. L. Zeisig, B. Acton, and J. A. McPherson (1998). Team dimensional training. In J. Cannon-Bowers and E. Salas (Eds.), *Making Decisions Under Stress: Implications for Individual and Team Training*. American Psychological Association.

Stiles, R., L. McCarthy, and M. Pontecorvo (1995, July). Training studio interaction. In *Workshop on Simulation and Interaction in Virtual Environments (SIVE-95)*, Iowa City, IW, pp. 178–183. ACM Press.

Swartout, W., R. Hill, J. Gratch, W. Johnson, C. Kyriakakis, C. LaBore, R. Lindheim, S. Marsella, D. Miraglia, B. Moore, J. Morie, J. Rickel, M. Thiébaux, L. Tuch, R. Whitney, and J. Douglas (2001). Toward the holodeck: Integrating graphics, sound, character and story. In *Proceedings of the Fifth International Conference on Autonomous Agents*, New York, pp. 409–416. ACM Press.

Swezey, R. W. and E. Salas (1992). Guidelines for use in team-training development. In R. W. Swezey and E. Salas (Eds.), *Teams: Their Training and Performance*, pp. 219–245. Norwood, NJ: Ablex.

Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research 7*, 83–124.

Traum, D. and J. Rickel (2002). Embodied agents for multi-party dialogue in immersive virtual worlds. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, New York. ACM Press. Forthcoming.

Zachary, W., J. Cannon-Bowers, J. Burns, P. Bilazarian, and D. Krecker (1998). An advanced embedded training system (AETS) for tactical team training. In *Proceedings of the Fourth International Conference on Intelligent Tutoring Systems (ITS '98)*, Number 1452 in Lecture Notes in Computer Science, Berlin, pp. 544–553. Springer.