Teaching Animated Agents in Virtual Worlds

Andrew Scholer, Richard Angros, Jr., Jeff Rickel and W. Lewis Johnson

Information Sciences Institute & Computer Sciences Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292-6695
ascholer@isi.edu - angros@isi.edu - rickel@isi.edu - johnson@isi.edu

Abstract

This paper describes the potential for instructing animated agents through collaborative dialog in a simulated environment. The abilities to share activity with a human instructor and employ both verbal and nonverbal modes of communication allow the agent to be taught in a manner natural for the instructor. A system that uses such shared activity and instruction to acquire the knowledge needed by an agent is described. This system is implemented in STEVE, an embodied agent that teaches physical tasks to human students.

Introduction

The use of embodied agents in simulated environments provides unique opportunities for communication with human participants in a natural manner. An embodied agent can both describe and demonstrate physical actions, use gaze and gestures to focus attention on a particular object, and provide feedback and conversational cues by speaking or through physical expression. This verbal and non-verbal interaction can allow agents and humans to build a collaborative dialog centered around shared work on a physical task such as maintaining a machine or operating a control console.

One area in which the use of this expanded communication between agent and human has been explored is the use of animated agents as instructors (Johnson, et al. 2000). Such work attempts to make learning tasks as natural as possible for human students by modeling the interaction between a human instructor and student. Much like a human instructor in the real world, embodied agents in a simulated environment can use both verbal and non-verbal means to provide instruction, demonstrations, and responses to student actions and questions. The various modalities of communication available allow the agent to tailor the delivery of a message to the demands of its content. When a student takes a step that progresses towards the end goal of their task, a simple head nod can provide encouragement in an unobtrusive manner. For a student that is thoroughly lost and asking for help, describing what they should be doing and then demonstrating how to do it may be more appropriate.

In addition to multiple modes of communication, a natural seeming dialog must allow for either party to drive the dialog. Building such a mixed initiative interaction between human and agent requires that either one should be able to take action that affects the shared environment. To maintain the cohesiveness of the dialog, the agent must be able to respond appropriately to any action that the student takes. The effects of actions by a student during a demonstration by the agent may change the state of the world in a way that requires the agent do extra work to successfully complete the task. Similarly, while trying to perform a task being learned, a student may make errors that change the steps required to complete the task. Because of the potential for interference through student actions, there is no way to predict an exact sequence of environmental states the agent will have to respond to, even in an environment with deterministic rules governing the effects of actions.

With a practically limitless set of environmental states and student actions to deal with, scripting all of the agent's actions is highly impractical at best. Instead, it is necessary that the agent have the knowledge to dynamically adapt its plans to new situations and be able explain that adaptation to the student. This kind of flexibility requires that the agent know a significant amount about its environment and the tasks it can perform. In addition to the perceptual and declarative knowledge of how to navigate in the world, manipulate objects, and talk about them, the agent must possess deep procedural knowledge about the task it is to teach. Generating valid plans and explaining the reasons behind those plans require that the agent know the goals of the task, the steps that can be taken to achieve those goals, and the constraints that govern how the steps can be ordered.

Although it is possible to manually encode such knowledge, such an approach can be time consuming and requires that a domain expert formalize the knowledge to be incorporated into the system. Even for experts, distilling practical knowledge into the kind of declarative knowledge needed by an agent can be a difficult process.

Other researchers have explored a number of ways to facilitate providing knowledge to an agent. Systems have been designed that use higher-level instruction languages to make knowledge engineering quicker and more easily understood (Badler, et al. 1998). Alternatively, some systems seek to eliminate traditional programming through instruction (Huffman & Laird 1994), example solutions

(Wang 1996) or experimentation (Gil 1992). Little attention however has been paid to learning through collaborative dialog based on activity in a virtual environment.

We believe that the same principles that guide an agent performing instruction in a virtual world should guide the way in which it learns. An agent should exploit the properties of its environment and its embodied nature to make instructing it as natural a process as possible. The human instructor should be able to teach the agent through a combination of interactive demonstration and instruction that closely resembles how they would work with a human student. For its part, the agent should function as an active learner, experimenting in the environment to supplement what it has been taught and utilizing verbal and non-verbal communication to communicate with the instructor about what it has learned and what gaps remain in its knowledge.

This paper describes our progress towards these goals. We have designed a system to acquire the knowledge needed by a pedagogical agent in a way that is consistent with how an instructor might teach a human student. This system, Diligent, has been implemented on an embodied agent, named STEVE (Soar[1] Training Expert for Virtual Environments) (Rickel & Johnson 1999, 2000), designed to teach physical tasks to students in a 3-D virtual environment.

System Overview

STEVE

STEVE (Figure 1) serves as an instructor within 3-D simulated environments. He can demonstrate physical tasks (demonstrations involve explaining steps as they are performed) or monitor a student performing a task. Students performing a task may ask the agent what action needs to be taken next and even ask the agent to show how to perform that action. If the agent is demonstrating a task, the student can ask the agent to stop and let the student attempt to finish it. Should a student simply take an action that helps or hinders STEVE while it is in the process of demonstrating a task, the agent uses a partial-order planning algorithm (Weld 1994) to adapt its actions and explanations to the new state of the environment. Any time a student is unsure why an action has been taken or needs to be taken, they can ask STEVE "Why?" The agent will describe the reasons behind the actions it has taken or suggested in terms of the goals that steps satisfy.

Human students interact with STEVE and the virtual environment through a simulator. This simulator has two interfaces: a traditional mode for use on a computer workstation and an immersive mode for use with a head-mounted display and data gloves. In the traditional mode, the mouse and keyboard are used to navigate the virtual world being displayed on the monitor and to manipulate objects within it. GUI menus can be used by the student to ask STEVE questions and to issue commands. In the immersive mode, the student uses the data gloves to manipulate objects and move around. Although an immersed student's head-mounted display does not display all of the GUI menus for communicating with STEVE, similar functionality is provided by speech recognition.

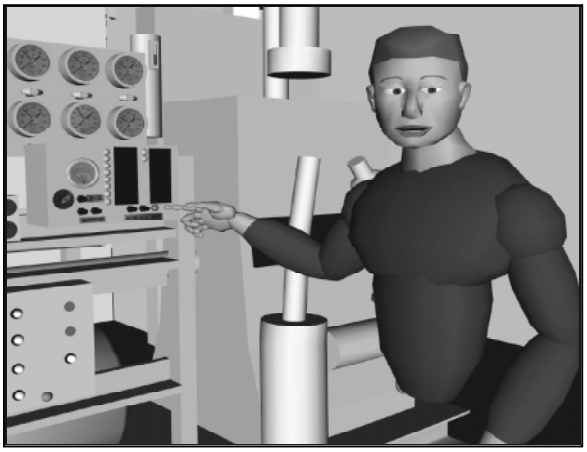

Figure 1: STEVE

A commercial speech recognition engine is used to recognize the commands and questions that STEVE is able to respond to. A student, whether immersed or working at the desktop, can issue verbal commands and queries to STEVE that the agent will respond to. In turn, STEVE's messages to the students are both printed to a window and made spoken by a text-to-speech program.

STEVE is designed to be portable between different simulated environments and even completely different simulators. For this reason, it is provided with no special access to know about the state of its environment or to effect changes to that state. Instead, STEVE relies on a perceptual-motor system that receives updates of attribute-value pairs from the environment, indicating changes in the world, and alerts the simulator to the agent's actions so that the world can respond appropriately. To teach tasks in a new environment, STEVE must acquire the knowledge needed by the perceptual-motor system to recognize actions, navigate, and manipulate objects in the virtual world as well as the knowledge needed to create, modify and explain plans for completing tasks.

Sources of Knowledge

STEVE agents have three primary sources from which to acquire the knowledge they need: a human instructor, the simulated environment, and a limited amount of initial domain knowledge. How each of these sources is used is governed by our goal of teaching embodied agents in a natural manner.

The instructors serve as the guiding force for knowledge acquisition. They are responsible for choosing a task to

[1] STEVE is implemented using Soar (Laird, Newell and Rosenbloom 1987).

teach STEVE and for providing a demonstration of the task. As learning continues, the instructor must troubleshoot the knowledge acquired and answer questions STEVE may have about the environment and task. However, there are limitations to the kinds and amount of information the instructor can be expected to supply.

It is assumed that the instructor is a domain expert with the knowledge to perform a task and answer questions about it. However, no assumptions are made that they possess any programming background or even a formalized representation of the knowledge being taught. For this reason, we wish to limit the kinds of instruction that they are expected to engage in; teaching STEVE should resemble working with a human student as much as possible rather than a programming or knowledge engineering task.

In addition, we assume that the instructor's time is valuable. Indeed, this is the motivating factor behind using an animated agent as an instructor in a simulated environment. If instructors' time were a limitless resource, we could simply have them work with students as they tried to learn tasks. Although it is reasonable for some time to be spent on teaching an agent that can turn around and instruct many students, we do not want to force the instructor to perform hundreds of demonstrations or waste time teaching things that STEVE can learn on its own.

While the instructor serves as the primary guide to knowledge acquisition, the simulated environment is the most prevalent source of that knowledge. It serves both as a place for the instructor to demonstrate procedures for the agent and as a place for the agent to improve its understanding of the world and a given task by experimenting in the environment.

As mentioned earlier, the environment is implemented wholly external to STEVE for reasons of portability. The agent lacks any ability to directly access or change the state of the world aside from taking physical actions. Instead, the simulator runs as a separate program with a well-defined API for message exchange with the agent (Rickel & Johnson 1999). The simulator passes the agent messages updating the values for attributes of the environment and the actors within it. The agent is responsible for receiving these messages and using them to maintain its own model of its environs.

From agent to simulator, two kinds of events may be passed: physical actions and a reset message. To manipulate objects or move its own body, the agent sends a message to the simulator, which updates the environment in the appropriate ways. The agent can use the reset message to restore the environment to a predefined state, allowing it to start experiments and student lessons from a consistent reference state.

Finally, the agent's initial domain knowledge allows it to specify to the simulator which attributes it is interested in monitoring. This requirement is largely a product of the simulator currently used for STEVE, which requires instructions as to which events to notify an agent about.

How the Agent Learns

STEVE begins learning about a task through a process of *programming by demonstration* (Cypher 1993). The human instructor tells STEVE to observe his actions, and then performs the task by manipulating objects in the simulated world. As the agent watches, it learns both necessary information about the environment and the procedural knowledge associated with the task.

Each time an instructor manipulates an object that is new to STEVE, the agent records the perceptual knowledge necessary to interact with it. It notes where the object is located, updating the information used by the agent's navigational system to find its way from point to point in the world. In addition, it uses information about where the instructor was standing in relation to the object and knowledge about its own body to infer where the front of the object is and where it should stand when manipulating it.

To acquire information that is not accessible through this observation of physical events, STEVE asks questions of the instructor. This is how it learns about the names of objects. Whenever the instructor interacts with an object STEVE is unfamiliar with, the agent generates a question to ask the instructor. For example, on seeing the instructor manipulate a button it is unfamiliar with, STEVE asks, "What is the name of that button?" and creates a GUI via which the instructor can respond. In keeping with our goal of making instruction as flexible and natural as possible, the instructor can either answer a question immediately or have STEVE hang on to it until later. If STEVE has been told to hold all its questions, it continues to generate them, but does not ask them immediately. Instead, the agent waits for the instructor to signal that they are ready to answer questions and then asks any it may have. Because questions that STEVE has been waiting to ask may be outside of the current context of discourse between instructor and agent, STEVE works to establish meaningful context for delayed questions by moving to the appropriate object and pointing it out as he asks for its name.

In addition to learning information about objects as the instructor performs each action, STEVE records the action itself, noting the state of the environment before and after the action. This information forms the basis for its understanding of the *operators* that govern what happens when actions are taken in the environment. Each operator relates changes caused by an action to the preconditions necessary for those effects to take place. For example, in one of STEVE's domains, the operation of turning a valve handle causes the effect of shutting the valve if the precondition that the valve is currently open is satisfied. Separate operators represent other effects this same action might have under different circumstance, like opening the valve if the valve is currently closed.

Each step in the task is recorded as an instance of some operator. By the time the task is complete, STEVE has learned a series of steps that can be used to perform the task. This list of steps is sufficient to perform the task, but not to allow the agent to adapt its plan of action or explain

why actions are necessary to a student. To carry out these more complicated activities, the agent must understand the goals of the task and the constraints and dependencies between steps.

To establish the goals of the task, STEVE examines the end state of each attribute that changed during the demonstration of that task. For example, given a task in which a valve was opened and a light was turned on and then back off, STEVE would conclude that the goal state is to have the valve open and the light off. After STEVE has derived its list, the instructor is allowed to review it and remove goals that are merely side effects. Once the task's goals are established, STEVE is prepared to learn how the steps relate to one another in satisfying those goals.

We have already noted that formalizing such knowledge sufficiently to provide direct instruction can be a difficult task for an instructor. It is much easier for STEVE to learn the deeper knowledge about the steps in a task by observing the consequences of performing those steps in alternative orderings. With the knowledge it has acquired about the correct ordering of steps, STEVE is capable of generating these alternative orderings for himself by experimenting with the task.

These experiments focus on understanding the demonstration the agent has seen rather than solving practice problems to ensure the agent acquires the knowledge it will need to teach the task. A demonstration has an initial state and a sequence of actions that lead to its final state. In contrast, practice problems consist of an initial state and a goal state but no specification of the steps needed to reach the final state. Although they are useful for learning general knowledge in systems (Gil 1992), practice problems are hard to solve with little domain knowledge. Such learning can require many demonstrations and practice problems, as seen in OBSERVER (Wang 96). In addition, practice problems may not be focused on understanding the dependencies between the actions in a particular plan for completing the task. Because STEVE has a very limited amount of initial domain knowledge and our goal is for the agent to learn the information necessary to teach a particular task, experimenting to learn about the demonstration provided by the instructor is more appropriate than solving practice problems.

STEVE's experiments on the demonstration consist of performing the task it is learning on its own, iteratively omitting a single step in each performance. Thus, STEVE skips the first step the first time it performs the task, the second step during the second performance, etc. Using this method, the agent has the opportunity to learn about how every step is affected by the absence of each step that comes before it. The experiments are of a manageable size for reasonably complex tasks, requiring the agent to perform on the order of n^2 steps where n is the number of steps in the task. In the domains we have studied even the longest tasks involve no more than 50 steps. Furthermore, large tasks can often be broken down into a hierarchy of subtasks, each of which can be experimented on separately.

Learning is performed by refining the preconditions of operators associated with each action. This is done through a modified version space (Mitchell 1978) that maintains bounds representing the most specific and most general combinations of preconditions possible for that operator. The state of the world before each action the agent takes, and the changes that occur afterwards, are used to create new operators and update old ones. Successful applications of old operators under new conditions can be used to broaden the most specific representation of the operator's preconditions. Similarly, actions that fail to replicate the changes of an old operator may be useful in narrowing down the most general set of preconditions for an operator.

Ideally, these two bounds will meet to define the exact preconditions of an operator. However, this process can be slow. Additionally, even after experiments with a task are complete, it is likely that the general and specific sets of preconditions for the operators involved will not have met to form a clear concept. Many facets of the environment will not change at all during experiments with the demonstration, making it unclear how they relate to the steps in the task being learned. For this reason, STEVE maintains a third set of preconditions, known as the heuristic set. This heuristic set is bounded by the general and specific sets and focuses on preconditions whose states change during the execution of the step or during the execution of a previous step in the task.

This set represents STEVE's assumption that the ordering of steps in a demonstration has significance. The agent assumes that the instructor has a reason for performing each step at a particular point – that effects of earlier actions are likely to be preconditions for later actions. Experiments refine the agent's understanding of the preconditions, giving the agent the knowledge it needs to decide which of these step orderings in a task are important to keep and why they are important. Thus the heuristic set does not speed the learning of a proven set of preconditions, but does provides a useful starting point for performing and learning about a task despite limited initial domain knowledge. (For a complete discussion of how STEVE uses demonstration and experimentation to learn procedural knowledge see Angros 1997, 2000).

Once the agent is finished with its experiments, it can create a task model (Figure 2) using a hierarchical partially ordered plan representation based on the operator rules learned. This representation, consisting of a set of steps and the ordering constrains and causal relations between those steps, has proven effective in a wide variety of research on task-oriented collaboration and generating procedural instructions (Delin et al. 1994, Mellish & Evans 1989, Young 1997). Causal links (McAllester & Rosenblitt 1991) record which steps establish the preconditions of other steps while ordering constraints insure that a step does not disturb the preconditions of a later step. For STEVE, this knowledge is sufficient to adapt its plan to unforeseen changes to the environment and to explain to a student why individual steps need to be performed (Rickel & Johnson 1999).

After the agent has finished experimenting and

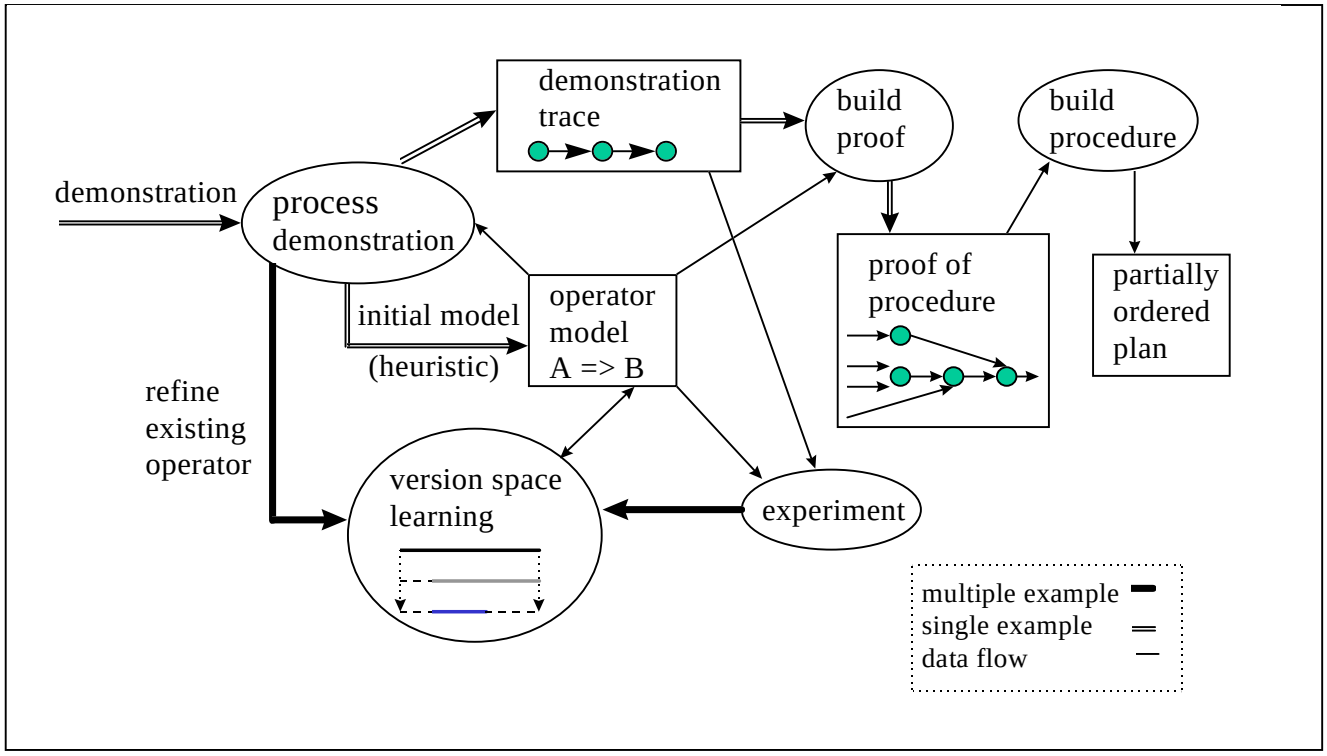

Figure 2: How the Agent Learns Procedures

generating this task model, the instructor can review the knowledge STEVE has acquired. The instructor may do so by either examining a graph of STEVE's partially ordered plan (Figure 3) or by allowing STEVE to demonstrate its understanding of the task by trying to teach it back to the instructor. The instructor (now playing the part of a student) can ask questions and take unanticipated actions to observe how STEVE responds as it demonstrates the task or monitors the instructor performing it. During this review, the instructor can use GUIs to change the way in which the agent manipulates objects (where it stands, how it grasps the object, etc.) and modify the graph of STEVE's plan to refine its understanding of the task by introducing or removing causal links or ordering constraints.

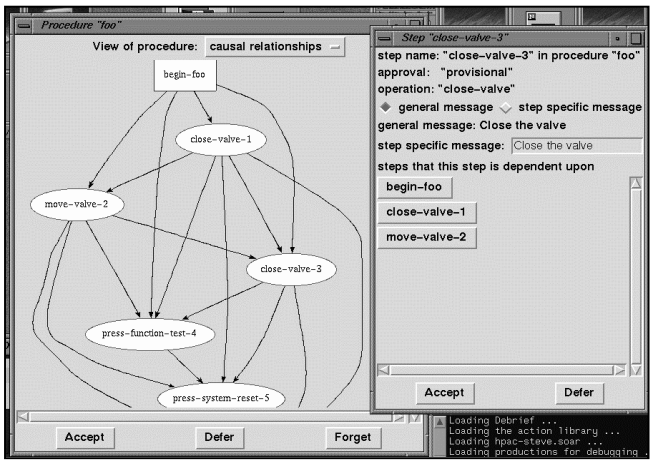

Figure 3: The Partial Order Plan

Towards a Richer Collaborative Dialog

STEVE currently makes use of many of the opportunities for learning that a dialog between human and embodied agent in a virtual world affords. Significant improvements however, are needed to allow STEVE to forge demonstration, experimentation, and instruction into a cohesive dialog that uses verbal and non-verbal communication. In deciding how this dialog should be formed, we intend to build on recent computational models of human collaborative dialog (Rich & Sidner 1998, Traum 1994, Lochbaum 1994) and to extend those models to account for the nonverbal interactions between collaborators in a shared environment. For this to happen, the agent's application of communication modalities such as verbal instruction, physical examples and gestures all must be made more flexible.

STEVE's ability to accept instruction is a good example of the ways in which the agent's present limitations prevent the smooth interleaving of different modes of communication. A human instructor is able to directly provide only some of the types of information that STEVE needs. Even for many of these types of knowledge, direct instruction is possible only when STEVE has asked a question about a subject. For example, there is no means for an instructor to provide the name for an object other than to manipulate it and wait for STEVE to ask for the object's

name. To foster a better dialog, the agent must be able to accept instruction about any type of knowledge it possesses and be a more active learner by demanding such instruction when it can identify gaps in its own knowledge. Furthermore, such interaction should be able to happen at any time. Although it is reasonable to make expectations based on context, and to require the instructor to adequately establish context for switches in the focus of dialog, context should not be enforced by limiting what a person can discuss with the agent at any given time.

Once the agent can use physical actions, instruction and gesture in a flexible manner, it will be possible to build a more natural and powerful dialog for instruction. Even abilities, such as watching demonstrations and performing experiments, that are already well developed in STEVE will profit from the ability of an instructor to mix instruction modalities at will.

As an example, consider demonstrations performed for STEVE. The agent watches the instructor's actions to learn about both physically performing steps and the rules governing the operators that those steps apply. Currently, there is no way for the instructor to focus STEVE's attention on any particular feature of the world – a very natural process in instructing humans. If the instructor and agent were able to mix verbal and non-verbal instruction into the demonstration, the instructor would be able to point out through words and gestures the features of the world worth particular attention. By telling STEVE "Watch that light" and gazing towards a particular one, the instructor should be able to specify that STEVE evaluate an action solely based on the effect it has on that particular light.

Not only would the instructor be able to use an improved dialog with STEVE to focus what the agent reasons about, but also to suggest what it should be doing. While the agent experiments with a task, the instructor should be able to focus the agent's experimentations by suggesting particular steps or relations to consider. If explicitly told that a set of steps could be performed in any order, STEVE could forgo the need to perform experiments that test for ordering constraints between those steps, using the instructor's advice to update its knowledge of the relationships between those steps. Such tailored instruction would be useful to STEVE both in its own learning and in guiding what it should emphasize when later on it instructs human students.

Conclusion

STEVE illustrates the potential for instructing animated agents through collaborative dialog in a simulated environment. Such dialogs meld verbal and nonverbal interaction with the performance of a task in the environment. This combination allows for a wide variety of complementary learning techniques such as demonstration, experimentation and instruction to be employed in ways that simulate the way that human instructors work with human students in shared environments.

Acknowledgments

This project is funded by the Office of Naval Research, grants N00014-95-C-0179 and N00014-97-1-0598. We gratefully acknowledge the efforts of our colleagues at Lockheed Martin under the direction of Randy Stiles that developed our virtual reality software and our colleagues at the Behavioral Technology Laboratory under the direction of Allen Munro that developed the simulator software. We also wish to thank Ben Moore for his work on acquiring perceptual knowledge for STEVE and Marcus Thiebaux for developing STEVE's graphical body and the code that animates STEVE's movements.

References

Angros, Jr., Richard; W. Lewis Johnson; Jeff Rickel. 1997. Agents that Learn to Instruct. *AAAI Fall Symposium on Intelligent Tutoring System Authoring Tools*. Menlo Park, CA: AAAI Press.

Angros, Jr., Richard. 2000. Agents That Learn What To Instruct: Increasing the Utility of Demonstrations by Actively Trying to Understand Them. Ph.D. diss., University of Southern California.

Badler, Norman; Rama Bindganavale; Juliet Bourne; Martha Palmer; Jianping Shi; and William Schuler. 1998. A Parameterized Action Representation for Virtual Human Agents. *Proceedings of the First Workshop on Embodied Conversational Characters*, 1-8.

Cypher, A. ed. 1993. *Watch What I Do: Programming by Demonstration*. Cambridge, Mass.: The MIT Press.

Gil, Y. 1992. Acquiring Domain Knowledge for Planning by Experimentation. Ph.D. diss., School of Computer Science, Carnegie Mellon Univ.

Huffman, S. B.; and Laird, J. E. 1995. Flexibly instructable agents. *Journal of Artificial Intelligence Research* 3:271-324.

Johnson, W. Lewis; Jeff W. Rickel; and James C. Lester. 2000. Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. *International Journal of AI in Education*.

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

Lochbaum, Karen E. 1994. Using Collaborative Plans to Model the Intentional Structure of Discourse. Ph.D. thesis, Harvard University.

Mitchell, T. M. 1978. Version Spaces: An Approach to Concept Learning. Ph.D. diss., Dept. of Computer Science,

Stanford Univ.

Rich, Charles; and Candace L. Sidner. 1998. COLLAGEN: A Collaboration Manager for Software Interface Agents. *User Modeling and User-Adapted Interaction* 8 (3-4):315-350.

Rickel, Jeff; and W. Lewis Johnson. 1999. Animated Agents for Procedural Training in Virtual Reality: Perception, Cognition, and Motor Control. *Applied Artificial Intelligence* 13: 343-382.

Rickel, Jeff; and W. Lewis Johnson. 2000. Task-Oriented Collaboration with Embodied Agents in Virtual Worlds. In J. Cassell and J. Sullivan and S. Prevost (eds.), *Embodied Conversational Agents.* Boston: MIT Press.

Traum, David R. 1994. A Computational Theory of Grounding in Natural Language Conversation. Ph.D. thesis, Department of Computer Science, University of Rochester.

Wang , X. 1996. Learning Planning Operators by Observation and Practice. Ph.D. diss., School of Computer Science, Carnegie Mellon Univ.

Weld, Daniel S. 1994. An Introduction to Least Commitment Planning. *AI Magazine*. 15(4):27-61.