Task-Oriented Collaboration with Embodied Agents in Virtual Worlds

Jeff Rickel and W. Lewis Johnson

1 Introduction

We are working toward animated agents that can collaborate with human students in virtual worlds. The agent's objective is to help students learn to perform physical, procedural tasks, such as operating and maintaining equipment. Like most of the previous research on task-oriented dialogues, the agent (computer) serves as an expert that can provide guidance to a human novice. Research on such dialogues dates back more than twenty years (Deutsch 1974), and the subject remains an active research area (Allen et al. 1996; Lochbaum 1994; Walker 1996). However, most of that research has focused solely on verbal dialogues, even though the earliest studies clearly showed the ubiquity of nonverbal communication in human task-oriented dialogues (Deutsch 1974). To allow a wider variety of interactions among agents and human students, we use virtual reality (Durlach and Mavor 1995); agents and students cohabit a three-dimensional, interactive, simulated mock-up of the student's work environment.

Virtual reality offers a rich environment for multimodal interaction among agents and humans. Like standard desktop dialogue systems, agents can communicate with humans via speech, using speech synthesis and recognition software. As in previous simulation-based training systems, a simulator controls the behavior of the virtual world**.** Agents can perceive the state of the virtual world via messages from the simulator, and they can take action in the world by sending messages to the simulator. However, an animated agent that cohabits a virtual world with students has a distinct advantage over previous disembodied tutors: the agent can also communicate nonverbally using gestures, gaze, facial expressions, and locomotion. In addition, students also have more freedom; they can move around the virtual world, gaze around (via a head-mounted display), and interact with objects (e.g., via a data glove). Agents can perceive these human actions; the position and orientation data used to track the users of a virtual reality system can provide agents with each user's location, field of view, and object manipulations. Thus,

virtual reality is an important application area for research on multimodal dialogues because it allows more humanlike interactions among synthetic agents and humans than typical desktop interfaces can.

To explore the use of animated agents for task-oriented collaboration in virtual worlds, we have designed such an agent: Steve (Soar Training Expert for Virtual Environments). Steve is fully implemented and integrated with the other software components on which he relies (i.e., visual interface software, a simulator, and commercial speech synthesis and recognition products). We have tested Steve on a variety of naval operating procedures; he can teach students how to operate and maintain the gas turbine engines aboard naval ships, including both individual tasks and team tasks. Steve is not limited to this domain; he can provide instruction in a new domain when given only the appropriate declarative domain knowledge. Despite the growing number of animated agents that converse with human users (André 1999; Johnson, Rickel, and Lester 2000; this book), Steve is unique in having domain-independent capabilities to support task-oriented dialogues situated in three-dimensional virtual worlds.

This chapter focuses on Steve's ability to integrate verbal and nonverbal communication to collaborate with students. Section 2 discusses the important roles that nonverbal communication plays in task-oriented collaboration. Section 3 illustrates Steve's current capabilities through an example interaction with a student. Sections 4 and 5 briefly review our architecture for virtual worlds and Steve's architecture; details are available in earlier papers (Johnson et al. 1998; Rickel and Johnson 1999a). Finally, section 6 describes the methods that govern Steve's communicative behavior, and section 7 provides conclusions and directions for future work.

2 Roles for Nonverbal Communication

While most of the previous research on task-oriented dialogues has focused on verbal interactions, an animated agent that cohabits the virtual world with students permits other types of interactions that play important roles in human task-oriented dialogues. These roles for nonverbal communication provide the primary motivation for using animated agents for task-oriented collaboration.

2.1 Interactive Demonstrations

An animated agent can demonstrate physical tasks, such as operation and repair of equipment. Demonstrating a task may be far more effective than trying to describe how to perform it, especially when the task involves motor skills and spatial relations, and the experience of seeing a task performed is likely to lead to better retention (Najjar 1998). Moreover, an interactive demonstration given by an agent offers several advantages over showing students a videotape. First, students are free to move around in the environment and view the demonstration from different perspectives. Second, they can interrupt with questions or even a request to finish the task themselves. Third, the agent can adapt the demonstration to different situations. For example, Steve is able to construct and revise plans for completing a task in response to changes in the virtual world, so he can demonstrate tasks under different initial states and failure modes, as well as help the student recover from errors.

2.2 Navigational Guidance

When a student's work environment is large and complex, such as a ship, one of the primary advantages of a virtual mock-up is to teach the student where things are and how to get around. In this context, animated agents can serve as navigational guides, leading students around and preventing them from becoming lost. For example, Steve inhabits a complex shipboard environment, including multiple rooms. The engine room alone is quite complex, with the large turbine engines that propel the ship, several platforms and pathways around and into the engines, a console, and a variety of different parts of the engines that must be manipulated, such as valves.

As Steve demonstrates tasks, he leads students around this environment, showing them where relevant objects are and how to get to them. Leading someone down a hallway, up a flight of stairs, around a corner, and through some pipes to the valve they must turn is likely to be more effective than trying to tell them where the valve is located. Our experience in training people using immersive virtual reality has shown that students can easily become disoriented and lost in complex environments, so animated agents that can serve as guides are an important instructional aid.

2.3 Gaze, Gesture, and Body Orientation as Attentional Guides

To draw a student's attention to a specific aspect of the virtual world, tutoring systems make use of many devices, including arrows, highlighting by color, and verbal referring expressions. An animated agent, however, can guide a student's attention with the most common and natural methods: gaze and deictic (pointing) gestures. Steve uses both methods to guide students' attention to objects in the virtual world, as well as to connect his verbal referring expressions to objects so that students learn their names. Argyle and Cook (1976) discuss the use of deictic gaze in human conversation, and the prevalence of deictic gestures in human task-oriented dialogues was noted in the earliest studies (Deutsch 1974).

Steve also uses his body orientation as a cue to his attentional focus. When he moves to an object to perform an action on it, he ends his locomotion with his body facing the object. Steve's immediate attention (indicated by his gaze) subsequently shifts back and forth between the student and the object as Steve describes the action, performs it, and finally discusses its results. During that time, however, the lower trunk of Steve's body remains oriented toward the object; Steve looks at the student when appropriate by turning his upper body, neck, and head. Thus, Steve's lower body orientation indicates to the student his focus on the object, and the orientation changes only as the focus of the task shifts to a new object, providing a cue to help the student recognize such focus shifts. Kendon (1972) observed a similar hierarchy of body movements in human speakers; while the head and hands tend to move during each sentence, shifts in the trunk and lower limbs occur primarily at topic shifts.

2.4 Nonverbal Tutorial Feedback

One primary role of a tutor is to provide feedback on a student's actions. In addition to providing verbal feedback, an animated agent can also use nonverbal communication to influence the student. Nonverbal feedback can reinforce a verbal comment. For example, Steve shakes his head when telling students that they made an error. However, nonverbal feedback can also provide more varied degrees of feedback than verbal comments alone. For example, nonverbal feedback may often be preferable because it is less obtrusive than a verbal comment. Steve uses a simple nod of approval when students perform correct actions to reassure them without interrupting. Similarly, human tutors often display a look of concern or puzzlement to make a student think twice about their

actions, especially in cases where either they are unsure that the student has actually made a mistake or they do not want to interrupt with a verbal correction yet. Graesser and his colleagues even found that human tutors often respond to student errors with a positive, polite verbal comment accompanied by a puzzled expression; the tutor, out of politeness, avoids directly criticizing the student but still communicates disapproval by nonverbal means (Graesser et al. in press).

2.5 Conversational Signals

When people carry on face-to-face dialogues, they employ a wide variety of nonverbal signals to complement their utterances and regulate the conversation. While tutorial dialogue in most previous tutoring systems resembles Internet chat or a telephone conversation, animated agents allow us to more closely model the face-to-face interactions to which people are most accustomed. Some nonverbal signals help regulate the flow of conversation, and would be most valuable in tutoring systems that support speech recognition as well as speech output, such as Steve or the Circuit Fix-It Shop (Smith and Hipp 1994). This includes back-channel feedback, such as head nods to acknowledge understanding of a spoken utterance (Duncan 1974). It also includes the use of eye contact to regulate turn taking in mixed-initiative dialogue (Argyle and Cook 1976). Steve employs these types of conversational signals.

Other nonverbal signals are closely tied to spoken utterances and could be used by any animated agent that produces speech output. For example, intonational pitch accents indicate the degree and type of salience of words and phrases in an utterance, including rhematic (i.e., new) elements of utterances and contrastive elements (Pierrehumbert and Hirschberg 1990); to further highlight such utterance elements, a pitch accent is often accompanied by a short movement of the eyebrows or a beat gesture (i.e., a short batonlike movement of the hands) (Ekman 1979). Steve does not currently employ such nonverbal signals. However, Cassell and her colleagues have developed agents that do (Cassell et al. 1999; Cassell et al. 1994), and we are working in that direction. Although people can clearly communicate in the absence of nonverbal signals (e.g., by telephone), communication and collaboration proceed most comfortably and smoothly when they are available.

2.6 Activities of Virtual Teammates

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 as a team, members must master their individual roles and learn to coordinate actions with their teammates.

Distributed virtual reality provides a promising vehicle for training teams; students, possibly at different locations, cohabit a virtual mock-up of their work environment, where they can practice together in realistic situations. In such training, animated agents can play two valuable roles: they 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. While verbal communication is often required for team coordination, the ability to track visually the activities of teammates is often indispensable. Although this chapter focuses on one-on-one interaction between Steve and a student, we have extended Steve to support team training, as described in an earlier paper (Rickel and Johnson 1999b).

3 Example

To illustrate Steve's capabilities, let us suppose that Steve is demonstrating how to inspect a high-pressure air compressor aboard a ship. The student's head-mounted display gives her a three-dimensional view of her shipboard surroundings, which include the compressor in front of her and Steve at her side. As she moves or turns her head, her view changes accordingly. Her head-mounted display is equipped with a microphone (to allow her to speak to Steve) and earphones (through which Steve speaks to her).

After introducing the task, Steve begins the demonstration. "I will now check the oil level," Steve says, and he leads her over to the dipstick. Steve looks down at the dipstick, points at it, looks back at the student, and says, "First, pull out the dipstick." Steve pulls it out (fig. 1). Pointing at the level indicator, Steve says, "Now we can check the oil level on the dipstick. As you can see, the oil level is normal." To finish the subtask, Steve says, "Next, insert the dipstick," and he pushes it back in.

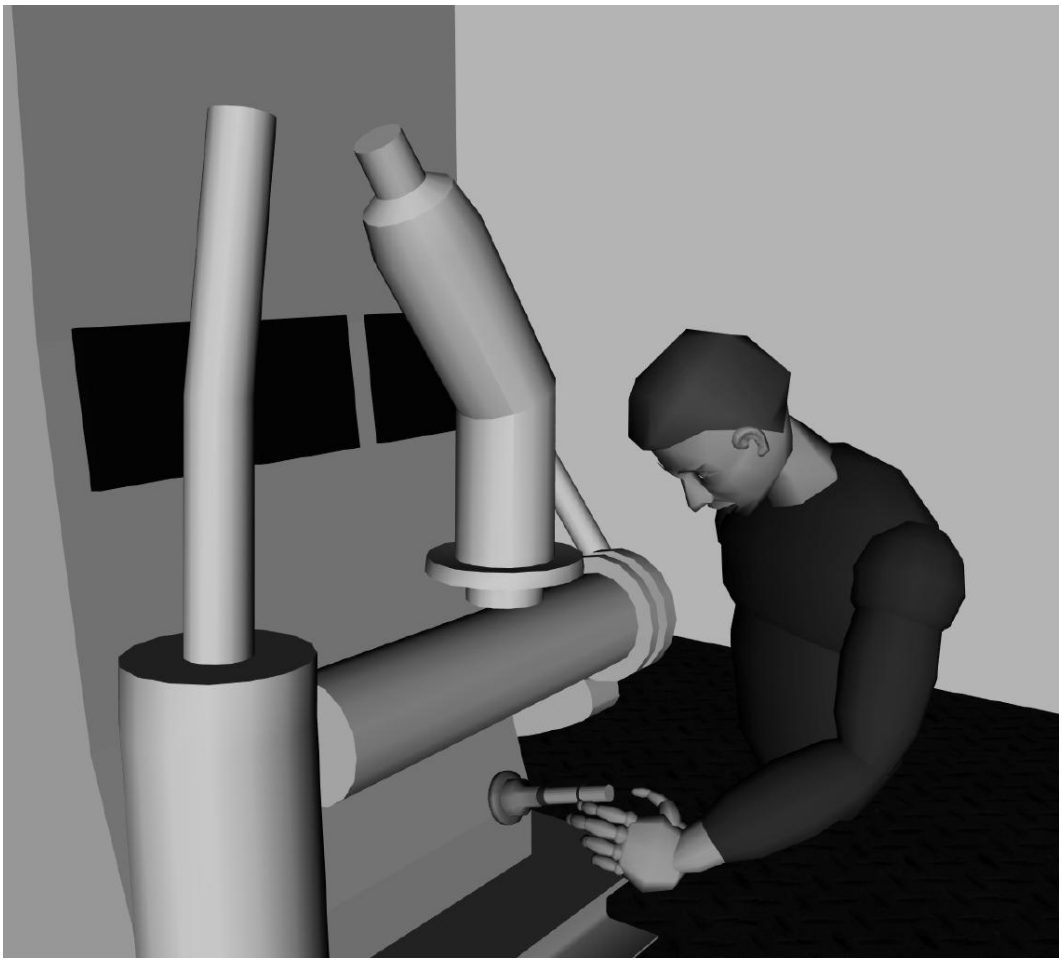

Figure 1: Steve pulling out a dipstick

Continuing the demonstration, Steve says, “Make sure all the cutout valves are open.” Looking at the cutout valves, Steve sees that all of them are already open except one. Pointing to it, he says, “Open cutout valve three,” and he opens it.

Next, Steve says, “I will now perform a functional test of the drain alarm light. First, check that the drain monitor is on.” Steve looks at the power light and back at the student. “As you can see, the power light is illuminated, so the monitor is on” (fig. 2). The student, realizing that she has seen this procedure before, says, “Let me finish.” Steve acknowledges that she can finish the task, and he shifts to monitoring her performance.

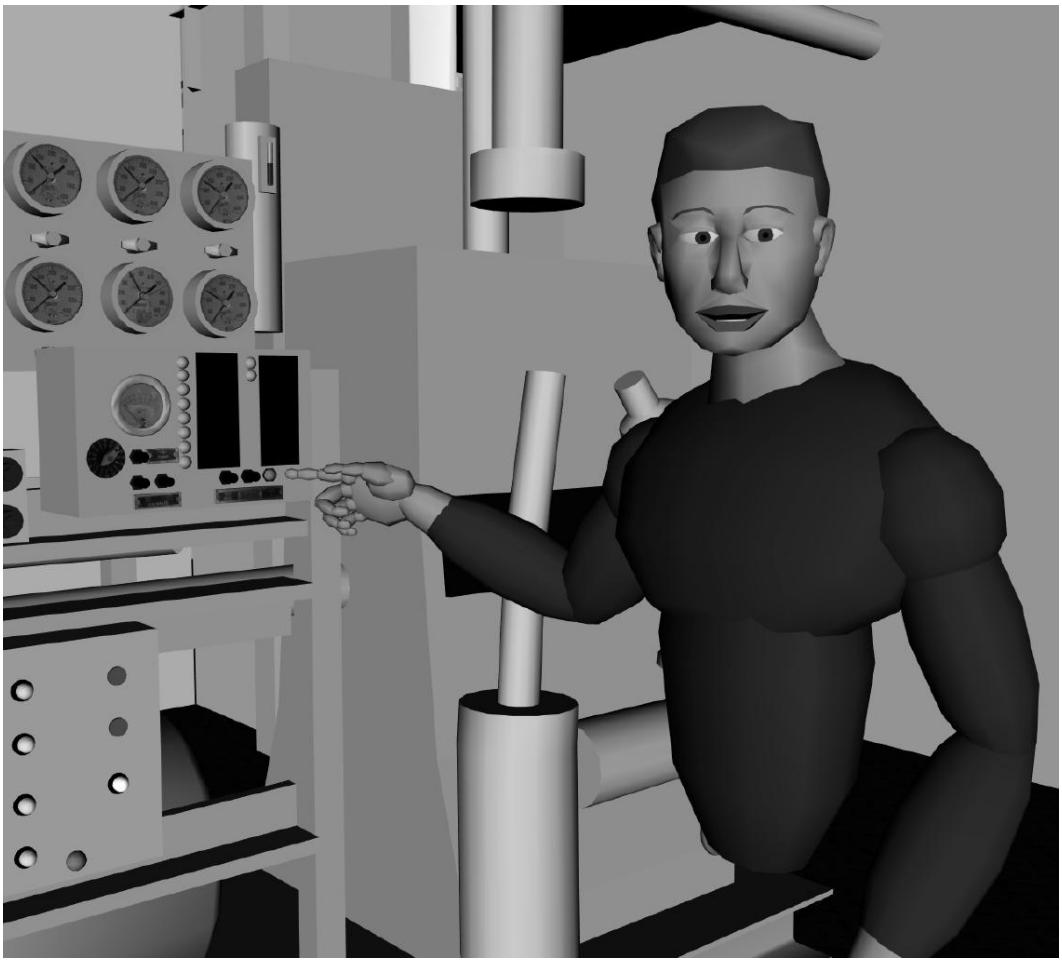

Figure 2: Steve describing a power light

The student steps forward to the relevant part of the compressor but is suddenly unsure of what to do first. “What should I do?” she asks. Steve replies, “I suggest that you press the function test button.” The student asks, “Why?” Steve replies, “That action is relevant because we want the drain monitor in test mode.” The student, wondering why the drain monitor should be in test mode, asks, “Why?” again. Steve replies, “That goal is relevant because it will allow us to check the alarm light.” Finally, the student understands, but she is unsure which button is the function test button. “Show me how to do it,” she requests. Steve moves to the function test button and pushes it (fig. 3). The alarm light comes on, indicating to Steve and the student that it is functioning properly. Now the student recalls that she must extinguish the alarm light, but she pushes the wrong button, causing a different alarm light to illuminate. “No,” Steve

responds as he shakes his head. Flustered, she asks Steve, “What should I do next?” Steve replies, “I suggest that you press the reset button on the temperature monitor.” She presses the reset button to extinguish the second alarm light, causing Steve to nod in approval, and then she presses the correct button to extinguish the first alarm light. Steve looks at her and says, “That completes the task. Nice job.”

Figure 3: Steve pressing a button

4 Creating Virtual Worlds for People and Agents

Before we can discuss Steve's architecture, we must introduce a software architecture for creating virtual worlds that people and agents can cohabit (fig. 4). With our colleagues from Lockheed Martin Corporation and the USC Behavioral Technologies Laboratory, we have designed and

implemented such an architecture (Johnson et al. 1998). For purposes of modularity and efficiency, the architecture consists of separate components running in parallel as separate processes, possibly on different machines. The components communicate by exchanging messages. Our architecture includes the following types of components.

- *Simulator:* A simulator controls the behavior of the virtual world. Our current implementation uses the VIVIDS simulation engine (Munro et al. 1997) developed at the USC Behavioral Technologies Laboratory.
- *Visual Interface:* Each human participant has a visual interface component that allows him or her to view and manipulate the virtual world. Several hardware devices connect participants to this component: a head-mounted display provides their view into the world, position sensors on their head and hands track their movements, and they interact with the world by "touching" virtual objects using a data glove. The visual interface component plays two primary roles. First, it receives messages from the other components (primarily the simulator) describing changes in the appearance of the world, and it outputs a three-dimensional graphical representation through each person's head-mounted display. Second, it informs the other components when each person interacts with objects. Our current implementation uses Lockheed Martin's Vista Viewer (Stiles, McCarthy, and Pontecorvo 1995) as the visual interface component.
- *Audio:* Each human participant has an audio component. This component receives messages from the simulator describing the location and audible radius of various sounds, and it broadcasts appropriate sounds to the headphones on the person's head-mounted display.
- *Speech generation:* Each human participant has a speech generation component that receives text messages from other components (primarily agents), converts the text to speech, and broadcasts the speech to the person's headphones. Our current implementation uses Entropic's TrueTalk text-to-speech product.
- *Speech recognition:* Each human participant has a speech recognition component that receives speech signals via the person's microphone, recognizes the speech as a path through its grammar, and outputs a semantic token representing the speech to the other components. (Steve agents do not have any natural language understanding capabilities, so they have no need for the recognized sentence.) Our current implementation uses Entropic's Graphvite product.
- *Agent:* Each Steve agent runs as a separate component. The remainder of the chapter focuses on the architecture of these agents and their capabilities.

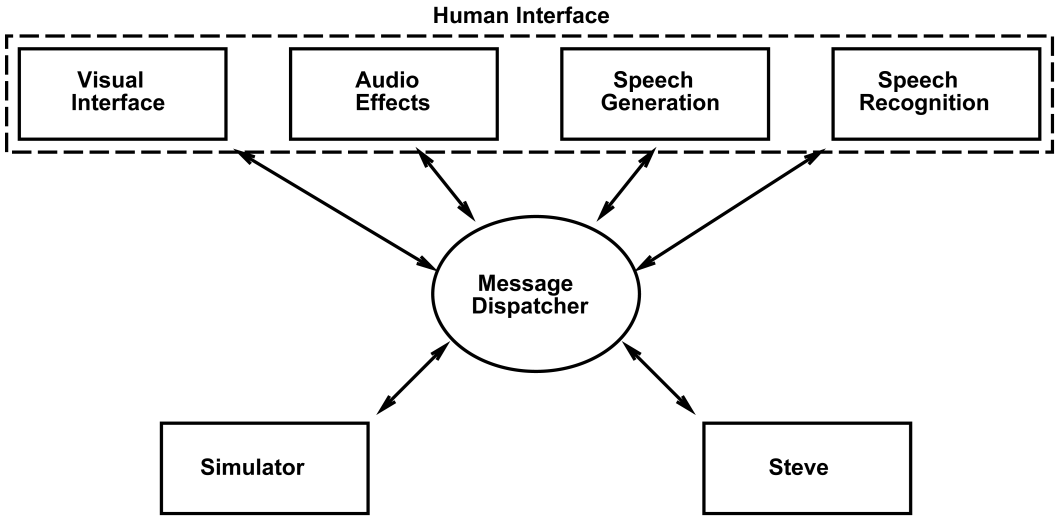

Figure 4: An architecture for virtual worlds. Although the figure only shows components for one agent and one human, other agents and humans can be added simply by connecting them to the message dispatcher in the same way.

The various components do not communicate with one another directly. Instead, all messages are sent to a central message dispatcher. Each component tells the dispatcher the types of messages in which it is interested. Then, when a message arrives, the dispatcher forwards it to all interested components. For example, each visual interface component registers interest in messages that specify changes in the appearance of the virtual world (e.g., a change in the color or location of an object). When the simulator sends such a message, the dispatcher broadcasts it to every visual interface component. This approach increases modularity, since one component need not know the interface to other components. It also increases extensibility, since new components can be added without affecting existing ones. It has been especially valuable in supporting team training, since it allows any number of students and agents to be connected to the virtual world. Our current implementation uses Sun's ToolTalk as the message dispatcher.

5 Steve's Architecture

Steve consists of three main modules: perception, cognition, and motor control (fig. 5). The perception module monitors messages from the message dispatcher and identifies events that are relevant to Steve, such as actions taken in the virtual world by people and agents and changes in

the state of the virtual world. Its main job is to provide a coherent view of the state of the virtual world to the cognition module. The cognition module interprets the input it receives from the perception module, chooses Steve's next actions, and sends out high-level motor commands to control the agent's voice and body. The motor control module decomposes these motor commands into a sequence of lower-level commands that are sent to other components via the message dispatcher. In our current implementation, cognition runs as one process, and perception and motor control run as a separate, parallel process. This chapter will focus primarily on Steve's cognition module; for details on the perception and motor control modules, see Rickel and Johnson (1999a).

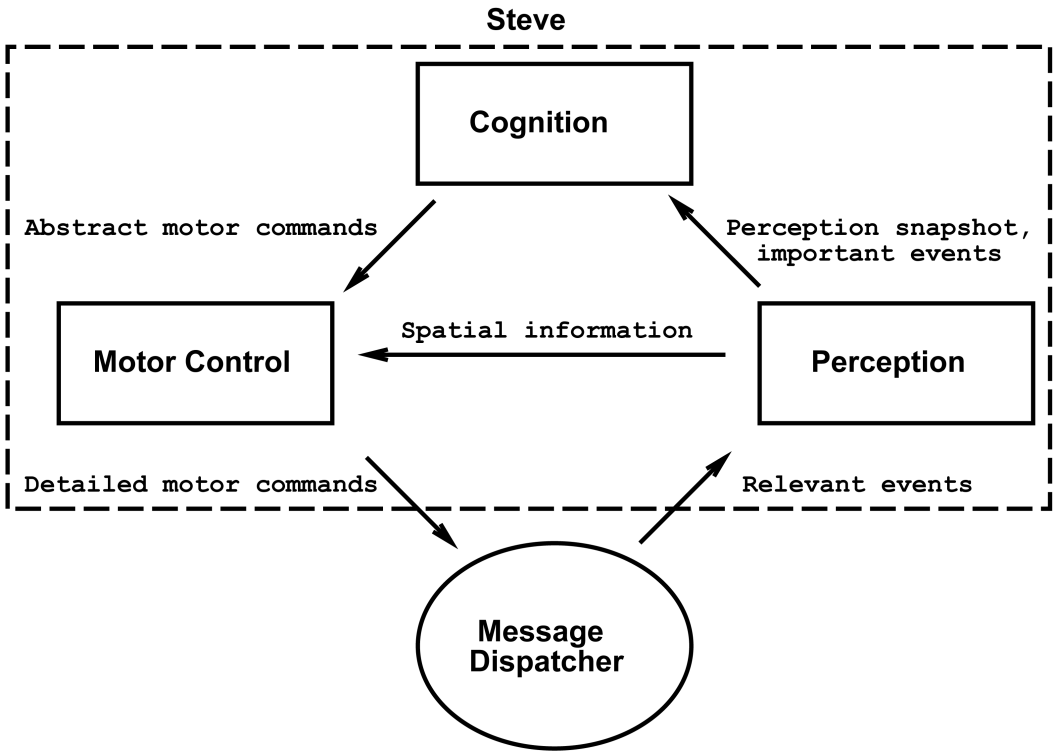

Figure 5: The three main modules in Steve and the types of information they send and receive

The cognition module is organized into three main layers. The lowest layer is Soar (Laird, Newell, and Rosenbloom 1987; Newell 1990). Soar was designed as a general model of human cognition, so it provides a number of features that support the construction of intelligent agents.

Soar's decision cycle is the feature most relevant to this chapter; we describe it below.

The next layer provides a set of domain-independent capabilities to support task-oriented collaboration. Soar is a general cognitive architecture, but it does not provide built-in support for particular cognitive skills such as demonstration and conversation. Our main task in building Steve was to design a set of domain-independent capabilities such as these and layer them on top of the Soar architecture.

The final layer of the cognition module provides Steve's domain-specific knowledge. To teach students how to perform procedural tasks in a particular domain, Steve needs a representation of the tasks. A course author must provide such knowledge to Steve. Given appropriate task knowledge for a particular domain, Steve uses his domain-independent capabilities to teach that knowledge to students. Our layered approach to Steve's cognition module allows Steve to be used in a variety of domains; each new domain requires only new task knowledge, without any modification of Steve's abilities as a teacher.

The cognition module operates by looping continually through a decision cycle. During each decision cycle, it receives new input from the perception module, it executes the *operator* that was selected at the end of the previous decision cycle, and it selects the operator to apply during the next decision cycle. Each operator is represented by a set of production rules that implement one of Steve's capabilities, such as answering a question or demonstrating an action. These serve as the building blocks for his behavior, and the process of operator selection serves as an arbitration mechanism for sequencing them. The timing of each decision cycle depends on the hardware on which Steve is running, and each decision cycle varies slightly in duration depending on Steve's cognitive activities. Steve typically executes about five to ten decision cycles per second on our current hardware (an SGI Onyx).

Low-level animation of Steve's body runs as software that is linked into the Visual Interface software (Vista Viewer) 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 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 currently supports locomotion (movement of the entire body from one location to another), facial movements (eyes, eyebrows, eyelids, and mouth), gaze (dynamic tracking of objects and people), head movements (nodding and shaking the head),

and arm and hand movements (including grasping, pressing, and pointing), all subject to motion limits. It 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.

6 Task-oriented Collaboration

Having discussed Steve's architecture and its interface to the other software components that make up the virtual world, we now focus on the domain-independent layer of Steve's cognition module, which provides his capabilities for task-oriented collaboration. First, we describe the actions that serve as the building blocks from which his dynamic behavior is constructed. Next, we discuss Steve's representation of the task and dialogue context. Finally, we examine how Steve uses the current context to choose his next action.

6.1 Behavioral Building Blocks

The cognition module generates Steve's communicative behavior by dynamically selecting his next action from a repertoire of behavioral primitives. To support the needs of task-oriented collaboration and exploit the roles for nonverbal communication outlined in Section 2, we currently generate Steve's behavior from the following building blocks.

• *Speak:* Steve can produce a verbal utterance directed at the student (or a teammate in team training). To make it clear to whom the utterance is directed, the motor control module automatically shifts Steve's gaze to the hearer just prior to the utterance. (Task-related events can cause gaze to shift to something else before the utterance is complete.) To make it clear that Steve is speaking, the motor control module automatically maintains a "speaking face" (eyebrows slightly raised and mouth moving) throughout the utterance. Steve has a wide range of utterances, all generated from text templates, ranging from a simple "OK" or "no" to descriptions of domain actions and goals. A message from the speech synthesis component indicates to Steve's perception module when the utterance is complete.

• *Move to an object:* To guide the student to a new object, Steve can plan a shortest path from his current location and move along that path (Rickel

and Johnson 1999a). To guide the student's attention, the motor control module automatically shifts Steve's gaze to his next destination on each leg of the path. In contrast, to simply follow the student around (e.g., when monitoring the student's activities), Steve shrinks and attaches himself to the corner of the student's field of view, so that he can provide visual feedback on their actions.

• *Manipulate an object:* To demonstrate domain task steps, Steve can manipulate objects in a variety of ways. Currently, this includes manipulations that can be done by grasping the object (e.g., moving, pulling, inserting, turning) or using his fingers (e.g., pressing a button, flipping a switch). To guide the student's attention, the motor control module automatically shifts Steve's gaze to the object just prior to the manipulation. State change messages from the simulator indicate to Steve's perception module when the manipulation is complete (e.g., a button's state attribute changing to "depressed").

• *Visually check an object:* Steve can also demonstrate domain task steps that simply require visually checking an object (e.g., checking the oil level on a dipstick or checking whether an indicator light is illuminated). This requires Steve to shift gaze to the object and make a mental note of the relevant property of that object.

• *Point at an object:* To draw a student's attention to an object, or connect a verbal referring expression to the object it denotes, Steve can point at the object. To further guide the student's attention, the motor control module automatically shifts Steve's gaze to the object just prior to pointing at it.

• *Give tutorial feedback:* To provide tutorial feedback on a student's action, Steve indicates a student error by shaking his head as he says "no," and he indicates a correct action by simply looking at the student and nodding. As discussed earlier, the motivation for shaking the head is to complement and reinforce the verbal evaluation, and the motivation for the head nod is to provide the least obtrusive possible feedback to the student.

• *Offer turn:* Since our goal is to make Steve's demonstrations interactive, we allow students to interrupt with questions (currently just "What next?" and "Why?") or to request to abort the task or finish it themselves. Although they can talk during Steve's utterances or demonstrations, Steve explicitly offers the conversational turn to them after each speech act (which could be several sentences) or performance of a domain action. He does this by shifting his gaze to them and pausing one second. Not only does this give students convenient openings for interruptions, but it also helps to structure Steve's presentations. (Prior to adding this feature,

users complained that Steve's presentations were hard to follow because he never paused to take a breath.)

• *Listen to student:* When the student is speaking, Steve can choose to quietly listen. This simply involves shifting gaze to the student to indicate attention.

• *Wait for someone:* When Steve is waiting for someone to take an action (either the student or a teammate in a team training scenario), he can shift gaze to that person (or agent) to indicate his expectation.

• *Acknowledge an utterance:* When a student or teammate says something to Steve, he can choose to explicitly acknowledge his understanding of their utterance by looking at them and nodding. The speech recognizer does not provide recognition of intermediate clauses, so Steve is currently limited to acknowledging understanding of entire utterances.

• *Drop hands:* When Steve is not using his arms and hands, he can drop them back down to hang loosely at his sides. Although there is evidence that such a move can convey a conversational signal (i.e., end of turn) (Duncan 1974), Steve does not currently use this behavior for that purpose; it simply means he has nothing else to do with his hands (such as pointing or manipulating).

• *Attend to action:* When someone other than Steve manipulates an object in his environment, Steve automatically shifts his gaze to the object to indicate his awareness. Unlike all the above behaviors, which are chosen deliberately by the cognition module, this behavior is a sort of knee-jerk reaction invoked directly by the perception module. Because an object manipulation is a very transient event, our design rationale was to react as quickly as possible.

6.2 Representation of Context

If Steve's environment were predictable, his behavior could be carefully scripted to ensure coherence. However, our goal is to support dynamic environments where Steve is not in complete control. To make the training experience as interactive as possible, the student has the freedom to speak or act on the environment at any time. To expose the student to a wide range of situations he or she might face, the simulator will create unexpected circumstances such as equipment failures. Rather than script Steve for each possible training scenario, our approach has been to give him a more general model of tasks that allows him to adapt dynamically to many variations in the task context. Team training adds additional dynamics to Steve's environment; the presence of other students and

autonomous agents makes the possibility of scripting his behavior impractical.

The key to maintaining coherent behavior in a dynamic environment is to maintain a rich representation of context. The ability to react to unexpected events and handle interruptions is crucial for task-oriented collaboration in virtual worlds, yet it threatens the coherence of the agent's behavior. A good representation of context allows the agent to be responsive while maintaining its overall focus. For task-oriented collaboration, an agent must maintain two separate but complementary types of context: the task context and the dialogue context.

6.2.1 Task Context

The task context represents the state of the task on which the student and Steve are collaborating. It must specify which task goals are already satisfied and how the remaining task goals can be achieved.

Steve is given knowledge of domain tasks represented as hierarchical plans. Each task consists of a set of steps, ordering constraints, and causal links. Each step is either a primitive action (e.g., press a button) or a composite action (i.e., itself a task). Composite actions give tasks a hierarchical structure. Each ordering constraint specifies that one step must be performed before another. These constraints define a partial order over the steps. Finally, the causal links specify the role of each step in the task. Each causal link specifies that one step achieves a goal that is a precondition for another step (or for completion of the task). This basic task representation has proven effective in a wide variety of research on task-oriented collaboration and generation of procedural instructions (Delin et al. 1994; Mellish and Evans 1989; Young 1997).

Steve represents the task context as an annotated task model for the current task on which he and the student are collaborating. When he and the student begin collaborating on a task, he creates the task model through simple top-down decomposition (Sacerdoti 1977). That is, starting with his representation for that task, he recursively expands each composite action with its task definition until he has a complete hierarchical model of the task. Then, on each subsequent decision cycle, he uses input from his perception module that specifies the current state of the virtual world to annotate the task model.

While the task model provides general information about how to perform the task, the annotations specify how to complete it from the current state of the virtual world. Using the input from the perception

module, Steve marks each goal in the task model as currently satisfied or not. This includes the end goals of the task as well as all intermediate goals.

Using this information about the state of the task goals, Steve uses a partial-order planning algorithm (Weld 1994) to determine which steps are relevant to completing the task (i.e., which subset of the task model constitutes his current plan), as described in Rickel and Johnson (1999a). This allows him to revise his plan dynamically when the state of the virtual world changes, perhaps unexpectedly. The planner runs as a set of background production rules (i.e., not requiring deliberate operator selection). Because the planning algorithm is linear in the size of the task model, it is predictably fast, preventing the planner from slowing down Steve's decision cycle and threatening his responsiveness.

6.2.2 Dialogue Context

The dialogue context represents the state of the interaction between the student and Steve. It includes the following information:

• Each decision cycle, the perception module tells the cognition module whether the student is currently speaking, based on messages from the speech recognition component.

• Steve keeps track of whether he is currently speaking. When he outputs a speak command, he records his current speech act. When his utterance is complete, the speech synthesis component sends a message to Steve's perception module, which informs his cognition module.

• Steve keeps track of which objects are currently in the student's field of view, using messages from the visual interface software.

• He records who currently has the task initiative. When he has the task initiative, he teaches the student how to complete the task. When the student has the task initiative, Steve watches as the student performs the task, evaluating the student's actions and answering her questions when she needs help. Students currently control the task initiative by asking Steve to demonstrate a task or asking to finish the task themselves. In the future, we plan to use the approach used in TOTS (Rickel 1988) to allow Steve to initiate shifts in task initiative based on a model of the student's knowledge.

• Steve keeps track of the steps he and the student have executed. For his own steps, he maintains an episodic memory that records the state of the world when each step was executed, so that he can explain later why he performed the step (Rickel and Johnson 1999a). Keeping track of prior steps also allows Steve to recognize when a step is being repeated, and it

lets him use cue phrases (Grosz and Sidner 1986) to indicate the relation between the previous step and the current one (Rickel and Johnson 1999a).

• When he answers a student question (currently just "What next?" or "Why?"), he records his answer in case the student asks a follow-up question (e.g., "Why?").

• To represent the attentional state of the collaboration, Steve uses a discourse focus stack (Grosz and Sidner 1986). Each element of the stack is a task step, either a primitive or composite action. Initially, the stack contains one element, the task on which Steve and the student are collaborating. When they begin collaborating on a new step, whether it is a step in the task definition of the current focus or an interruption (e.g., a step being executed out of its normal order), the step is pushed onto the focus stack. When the step currently in focus is completed, it is popped off the stack. This helps ensure a systematic, coherent execution of the task, and it allows Steve to recognize digressions and resume the prior demonstration when unexpected events require a temporary deviation from the usual order of task steps.

• For the step currently in focus, Steve records the status of his collaboration with the student on that step (e.g., whether he proposed the step, whether he explained it, whether he or the student performed it, whether they discussed the results).

• When the student makes a request, Steve records it until he responds. This is needed when Steve chooses to finish his current activity (e.g., utterance or demonstration) before responding.

6.3 Controlling Steve's Behavior

Given Steve's repertoire of behaviors and his representation of the current context, the cognition module must repeatedly choose his next action. His behavioral building blocks are implemented as Soar operators. Each decision cycle, the cognition module must choose to continue the current operator in the next decision cycle or select a new one. The representation of the current context gives it the information to make this decision.

Soar provides architectural support for action selection. To program an agent in Soar, one writes three types of production rules: operator proposal rules suggest an operator for selection in the next decision cycle, operator comparison rules assert preferences among the proposed operators, and operator application rules execute an operator after it has been selected. At the end of each decision cycle, Soar considers the

proposed operators and the preferences among them, and it chooses one operator for application during the next decision cycle.

Thus, the behavioral building blocks for Steve are implemented as operator application rules, and their sequencing comes from operator proposal and comparison rules. Since Steve typically executes five to ten decision cycles per second, he can be very responsive to the environment.

Steve can choose his next action to fill one of three roles. First, he can respond to the student. This includes responding to a student's request, giving them tutorial feedback on their action, or simply listening when they are talking. Second, Steve can choose for himself how to advance the collaborative dialogue. This includes things like suggesting the next task step, describing it, or demonstrating it in cases where the student did not explicitly request such help. Third, Steve can choose a turn-taking or grounding act (Traum and Hinkelman 1992) that helps regulate the dialogue between the student and himself without advancing the task. This includes offering the student the conversational turn or acknowledging understanding of an utterance with a head nod.

While operator proposal rules may propose several such actions, operator comparison rules give these three types of actions different priorities. The highest priority is to respond to the student. If no such operators are proposed, the next priority is to perform any relevant conversational regulation action. However, if an opportunity for a conversational regulation action is missed due to a higher priority operator for responding to the student, it will not be deferred and displayed later. Only when neither of these types of operators is proposed will Steve take the initiative to advance the task collaboration, and he only does that when he has the task initiative. Traum proposed a similar priority scheme in his model of spoken task-oriented dialogue (Traum 1994).

Because of their importance in conversation, grounding acts deserve special mention. Research has shown that listeners frequently provide some combination of verbal and nonverbal cues to indicate understanding or lack thereof (Clark and Schaefer 1989). The cue can be explicit, as with back-channel feedback (e.g., a brief "Mmm hmm" or head nod), or it can be implicit if the listener simply follows the speaker's utterance with an appropriate response (e.g., an answer to a question). Steve uses the implicit approach (i.e., no separate grounding act) when responding to student requests and task actions. When possible, his response is chosen to indicate what he understood, in case of faulty speech recognition. For example, when he thinks the student just asked to finish the task, he responds with "OK, you finish"; if he misunderstood the

student's utterance and simply responded with "OK," the student could be confused when he subsequently watches the student and waits for the student's next action.

Steve uses an explicit grounding act in only two cases. First, he uses a head nod to acknowledge his understanding of an utterance when someone informs him of some condition, which only occurs during team tasks. In this case, the teammate may not watch his subsequent activity, so a clear acknowledgment of understanding is helpful. Since speech recognizers cannot indicate to whom an utterance is directed, the acknowledgment also indicates that the agent has determined from the current context that the speech was directed to him. Second, Steve asks the student to repeat an utterance when the speech recognizer cannot understand it.

Some operator proposal productions trigger other operator proposals. When Steve proposes an operator whose execution would require a particular location for Steve (e.g., demonstrating a task step), that precondition appears on the structure for the proposed operator. If Steve is not currently at the location, another production rule will propose a higher priority operator for moving Steve to it. Similarly, an operator for an utterance whose focus is a particular object may, depending on context, be annotated with a deictic precondition. The deictic precondition will trigger a proposal rule for Steve to point at the object first. Our criteria for choosing when to accompany a verbal referring expression with locomotion or a deictic gesture are currently simple; before demonstrating an action on an object, Steve first moves to it, and then he points at it while describing what he is about to do. Work by Lester and his colleagues (Lester et al. 1999) offers more sophisticated criteria that could be incorporated easily into Steve. Finally, some operators (e.g., an action demonstration) have a precondition that requires the student to be looking at a particular object (e.g., to see the action). If the operator is proposed and the object is not in the student's field of view, Steve will propose an operator to get their attention (e.g., by saying "Look over here!").

Since task steps may be only partially ordered, there may be multiple steps that could be performed next. From the standpoint of completing the task, any of them could be chosen. However, research has shown that collaborators shift focus from one task step or subtask to another only with good reason (Grosz and Sidner 1986). To maximize the coherence of his actions, Steve uses the discourse focus stack. As long as the current task step or subtask in focus is still appropriate, all his proposed operators will relate to it. When Steve has the task initiative, and several task steps could be performed next, he chooses one and pushes it

onto the focus stack before performing any operators for one of the steps (e.g., move, point, explain, demonstrate). This does not prevent Steve from pushing a new step or subtask onto the stack to interrupt the current one (e.g., to handle an unexpected emergency) or from popping the current focus if it suddenly becomes irrelevant; he handles such cases by explaining the unexpected (but required) focus shift to the student. However, the focus stack prevents Steve from shifting focus in cases where it is not required.

The sequence of operators that Steve applies when teaching the student a new task step (e.g., describe step, perform step, explain result) depends on the type of action the step requires. Steve has a class hierarchy of action types (e.g., manipulate an object, move an object, check a condition), and each type of action is associated with a suite of communicative acts. Each suite is essentially a finite-state machine represented as Soar productions. Each node represents an operator, such as describing the step or performing it, and arcs represent the conditions for terminating one operator and beginning another. Each action type in the class hierarchy inherits from the action types above it, so the communicative suite for an action type can be represented compactly as the deviations from its parent's suite. Currently, all action types inherit their communicative suite from one of a few general action types. However, our approach allows us to extend Steve's behavior if these suites prove inadequate for new types of actions encountered in new domains. The communicative suites need only specify the next contribution to the dialogue needed to advance the task collaboration; operators for responding to the student, moving and pointing, and regulating the conversation (i.e., turn-taking and grounding acts) are proposed independently as described above.

Steve's communicative suites are similar to the schemata approach to explanation generation pioneered by McKeown (1985). In contrast, André and her colleagues employ a top-down discourse planning approach to generating the communicative behavior of their animated agent, and they compile the resulting plans into finite-state machines for efficient execution (André, Rist, and Müller 1999). The trade-offs between these two approaches are well known (Moore 1995).

Many of Steve's behavioral building blocks take several decision cycles to execute, and some (speech and locomotion) can take many decision cycles. During this period, Steve's perception and cognition modules remain active, monitoring the state of the virtual world and deciding whether to react. In principle, any of the building blocks could be aborted at any time. In practice, Steve does not abort most actions. A

notable exception is that Steve halts his demonstration of a task immediately, even in midsentence, when students interrupt and ask to abort the task or to finish it themselves. Steve's reluctance to abort his actions rarely seems unusual, since, for example, people often finish what they are saying before acknowledging an interruption. Nonetheless, our current work is focusing on cases where Steve should be more responsive to interruptions.

7 Conclusions and Future Work

We have tested Steve on a variety of naval operating procedures. He can perform tasks on several of the consoles that are used to control the gas turbine engines that propel naval ships, he can check and manipulate some of the valves that surround these engines, and he can perform a handful of procedures on the high-pressure air compressors that are part of these engines. A student in the USC School of Education recently completed an evaluation of users interacting with Steve in a variety of circumstances. The student is currently analyzing the data and formulating conclusions, which will appear in his dissertation (Kroetz forthcoming).

Our work has focused more on multimodal behavior generation than multimodal input. To model face-to-face communication, we must extend the range of nonverbal communicative acts that students can use. To handle multimodal input in virtual reality, the techniques of Billinghurst and Savage (1996) neatly complement Steve's current capabilities. Their agent, which is designed to train medical students to perform sinus surgery, combines natural language understanding and simple gesture recognition. The work of Thórisson and Cassell on the Gandalf agent (Cassell and Thórisson 1999; Thórisson 1996) is even more ambitious; people talking with Gandalf wear a suit that tracks their upper body movement, an eye tracker that tracks their gaze, and a microphone that allows Gandalf to hear their words and intonation. More recent work by Cassell and her colleagues on their Rea agent is exploring the use of a vision system for tracking the user's nonverbal communication (Cassell et al. 1999). Although Gandalf and Rea were not developed for conversation in virtual reality, many of the techniques used for multimodal input apply.

Our work on Steve complements the long line of research on verbal task-oriented dialogues in the computational linguistics community. Steve currently has no natural language understanding capabilities; he can only understand phrases that we add to the grammar for the speech recognition program. Steve's natural language generation capabilities are also simple; all of Steve's utterances are generated from text templates,

although more sophisticated methods could be added without affecting other aspects of Steve's behavior. We are particularly interested in integrating Steve with recent work on spoken dialogue systems. For example, the TRAINS system (Allen et al. 1996; Ferguson, Allen, and Miller 1996) supports a robust spoken dialogue between a computer agent and a person working together on a task. However, their agent has no animated form and does not cohabit a virtual world with users. Because TRAINS and Steve carry on similar types of dialogues with users, yet focus on different aspects of such conversations, a combination of the two systems seems promising.

Our work also complements research on sophisticated control of human figures (Badler, Phillips, and Webber 1993). That research targets more generality in human figure motion. Our human figure control is efficient and predictable, and it results in smooth animation. However, it does not provide sophisticated object manipulation (Douville, Levison, and Badler 1996), and it would not suffice for movements such as reaching around objects or through tight spaces. Our architecture is carefully designed so that a new body, along with its associated control code, can be integrated easily into Steve; a well-defined API separates Steve's control over his body from the detailed motion control code.

Steve illustrates the enormous potential for face-to-face, task-oriented collaboration between students and synthetic agents in virtual environments. Although verbal exchanges may be sufficient for some tasks, we expect that many domains will benefit from an agent that can additionally use gestures, gaze, facial expressions, and locomotion. Although Steve has only been tested on a virtual shipboard environment for naval training, he can be used for other domains when given only a description of the domain tasks and minimal knowledge of the spatial environment; none of the capabilities described in this chapter are specific to our naval domain. Moreover, Steve's architecture is designed to accommodate advances in related research areas, such as natural language processing and human figure control. This makes Steve an extensible foundation for further research on task-oriented collaboration in virtual worlds.

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 component (Vista Viewer). Allen Munro and his colleagues at the Behavioral Technologies Laboratory 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 current body and the code in the visual interface component that controls its animation.

References

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

André, E., ed. 1999. Special issue on Animated interface agents: Making them intelligent. *Applied Artificial Intelligence* 13(4-5).

André, Rist, and Müller. 1999. Employing AI methods to control the behavior of animated interface agents. *Applied Artificial Intelligence* 13:415-448.

Argyle, M., and M. Cook. 1976. *Gaze and mutual gaze*. Cambridge: Cambridge University Press.

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)*, 168-175. Los Alamitos, Ca.: IEEE Computer Society 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.

Cassell, J., T. Bickmore, L. Campbell, K. Chang, H. Vilhjálmsson, and H. Yan. 1999. Requirements for an architecture for embodied conversational characters. In *Proceedings of Computer Animation and Simulation '99*, 109-120. Berlin: Springer Verlag.

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 SIGGRAPH '94*, 413-420. Reading, Mass.: Addison-Wesley.

Clark, H. H., and E. F. Schaefer. 1989. Contributing to discourse. *Cognitive Science* 13:259-294.

Delin, J., A. Hartley, C. Paris, D. Scott, and K. Vander Linden. 1994. Expressing procedural relationships in multilingual instructions. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, 61-70, Kennebunkport, Maine.

Deutsch, B. G. 1974. The structure of task oriented dialogs. In *Proceedings of the IEEE Symposium on Speech Recognition*, Pittsburgh, Penn.: Carnegie-Mellon University. (Also available as Stanford Research Institute Technical Note 90, Menlo Park, Ca.)

Douville, B., L. Levison, and N. I. Badler. 1996. Task-level object grasping for simulated agents. *Presence: Teleoperators and Virtual Environments* 5(4):416-430.

Duncan Jr., S. 1974. Some signals and rules for taking speaking turns in conversations. In S. Weitz, ed., *Nonverbal communication*, 298-311. New York: Oxford University Press.

Durlach, N. I., and A. S. Mavor, eds. 1995. *Virtual reality: Scientific and technological challenges***.** Washington, D.C.: National Academy Press.

Ekman, P. 1979. About brows: Emotional and conversational signals. In M. von Cranach, M., K., Foppa, W. Lepenies, and D. Ploog, eds., *Human ethology*. Cambridge: Cambridge University Press.

Ferguson, G., J. Allen, and B. Miller. 1996. TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third Conference on AI Planning Systems*, 70-77. Menlo Park, Ca.: AAAI Press.

Graesser, A. C., K. Wiemer-Hastings, P. Wiemer-Hastings, R. Kreuz, and the Tutoring Research Group. In press. AutoTutor: A simulation of a human tutor. *Journal of Cognitive Systems Research.*

Grosz, B.J., and C. L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics* 12(3):175-204.

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*, in press.

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

Kendon, A. 1972. Some relationships between body motion and speech. In A. W. Siegman and B. Pope, eds., *Studies in Dyadic Communication*, 177-210. New York: Pergamon Press.

Kroetz, A. Forthcoming. The role of intelligent agency in synthetic instructor and human student dialogue. Ph.D. diss., University of Southern California, Los Angeles.

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., 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.

Lochbaum, K. E. 1994. Using collaborative plans to model the intentional structure of discourse. Ph.D. diss., Harvard University. Technical Report TR-25-94, Center for Research in Computing Technology.

McKeown, K. R. 1985. *Text generation.* Cambridge: Cambridge University Press.

Mellish, C., and R. Evans. 1989. Natural language generation from plans. *Computational Linguistics* 15(4):233-249**.**

Moore, J. D. 1995. *Participating in explanatory dialogues.* Cambridge, Mass.: The MIT Press.

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

Najjar, L. J. 1998. Principles of educational multimedia user interface design. *Human Factors* 40(2):311-323.

Newell, A. 1990. *Unified theories of cognition*. Cambridge, Mass.: Harvard University Press.

Pierrehumbert, J., and J. Hirschberg. 1990. The meaning of intonational contours in the interpretation of discourse. In P. Cohen, J. Morgan, and M. Pollack, eds., *Intentions in communication*, 271-311. Cambridge, Mass.: The MIT Press.

Rickel, J. 1988. An intelligent tutoring framework for task-oriented domains. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, 109-115. Montréal, Canada: Université de Montréal.

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

———. 1999b. Virtual humans for team training in virtual reality. In *Proceedings of the Ninth International Conference on Artificial Intelligence in Education*, 578-585. Amsterdam: IOS Press.

Sacerdoti, E. 1977. *A structure for plans and behavior*. New York: Elsevier/North-Holland.

Smith, R. W., and D. R. Hipp. 1994. *Spoken natural language dialog systems*. New York: Oxford University Press.

Stiles, R., L. McCarthy, and M. Pontecorvo. 1995. Training studio: A virtual environment for training.
In *Workshop on Simulation and Interaction in Virtual Environments (SIVE-95)*. New York: ACM Press.

Thórisson, K. R. 1996. Communicative humanoids: A computational model of psychosocial dialogue skills. Ph.D. diss., Massachusetts Institute of Technology, Cambridge, Mass.

Traum, D. R. 1994. A computational theory of grounding in natural language conversation. Ph.D. diss., Department of Computer Science, University of Rochester, Rochester, N.Y.

Traum, D. R., and E. A. Hinkelman. 1992. Conversation acts in task-oriented dialogue. *Computational Intelligence* 8(3):575-599.

Walker, M. A. 1996. The effect of resource limits and task complexity on collaborative planning in dialogue. *Artificial Intelligence* 85:181-243.

Weld, D. S. 1994. An introduction to least commitment planning. *AI Magazine* 15(4):27-61.

Young, R. M. 1997. Generating descriptions of complex activities. Ph.D. diss., University of Pittsburgh, Pittsburgh, Penn.