Pedagogical Agents in Virtual Learning Environments

W. Lewis Johnson
USC / Information Sciences Institute & Computer Science Dept.
4676 Admiralty Way, Marina del Rey, CA 90292-6695
WWW: http://www.isi.edu/isd/johnson.html

Abstract

Advances in visualization and networking technologies make it possible to construct *virtual learning environments*, i.e., virtual worlds in which learners can immerse themselves. This paper examines issues involved in the design of computer-generated virtual agents with which learners may interact in such environments. Functions common to intelligent tutoring systems, such as student modeling and coaching, are assigned to individual agents, rather than being disembodied functions of the tutoring system as a whole. This stance offers new perspectives on long-standing problems in intelligent tutoring system design, such as how to structure tutorial interactions. It also reveals issues that tend to be ignored in conventional systems, such as interpersonal relations between learners and tutoring systems. The paper describes progress achieved to date in creating pedagogical agents for virtual learning environments, and discusses planned future work in this area.

1 Introduction

Interactive learning environments (ILE's) typically incorporate "microworld" simulations, i.e., simplified simulations of real-world phenomena which students can manipulate and control. They may also incorporate intelligent tutors or collaborators that can interact with the student. However, the word "environment" in the phrase "interactive learning environment" refers to the educational computing system as a whole, not the microworld contained within it. The computer in the role of tutor or collaborator views and operates on the microworld from the outside, just as the learner does.

With the advent of virtual environment technology, it is now possible to make the simulation environment and the human-computer interaction context one and the same. Microworlds become three-dimensional spaces into which learners can enter. By networking simulators together, multiple learners can inhabit the same virtual space. It therefore makes sense for computer tutors and collaborators to inhabit the virtual space as well—in fact this is inevitable, since all interactions between learners and the computer take place via the immersive interface.

We are developing virtual agents which can interact with learners for pedagogical purposes within virtual environments. The agents are a part of the virtual scene, just as the learners are. They can interact with humans as collaborators, adversaries, or mentors. This paper describes the requirements that such agents must meet, and techniques, both current and envisaged, that help meet these requirements.

2 Interactions in Pedagogical Settings

Most studies of tutorial interactions, such as (Merrill et al 1992) focus on one-on-one interaction between students and tutors solving problems using pencil and paper. These tend to conform to the cognitive apprenticeship model of instruction (Collins et al 1989): teachers model the desired skill, coach students

as they practice the skill, and gradually withdraw their support as students gain proficiency. Once the training is complete, students should be able to perform the skill in isolation. Many intelligent tutoring systems seek to emulate such teacher-student interactions.

However, many skills are performed not in isolation, but within groups. Manufacturing, construction, and military operations all require teams of workers, and may involve interactions with other individuals outside the team. Even driving a car falls into this category, since drivers must learn to respond to the actions of other drivers. Models based on one-on-one interaction over pencil and paper are not appropriate for such skills.

In learning sessions for multi-person skills, the dividing line between teachers and other participants can blur. Apprentices and trainers work together to perform tasks; trainers model skills in the context of applying them to the task at hand. Similarly, military trainees often practice their skills against instructors playing the role of opponents; the instructors may then switch to a teaching role during after-action review, when trainees and instructors attempt to draw lessons from what happened during the engagement. Interactions Between teachers and learners are thus much more complex than what is typically seen in pencil-and-paper tutorial sessions.

2.1 Implications for Learning Environments

Multi-user distributed simulations are one way to provide effective group learning experiences. The military already makes extensive use of such simulations (Sterling 1993). . Multi-User Dungeons (MUDs) can provide similar experiences in educational settings (Soloway1995) . However, a distributed simulation is effective only if there are enough qualified people available to participate in the simulation, and enough simulation interfaces for all participants. Otherwise, it may be useful to create computer-generated agents that can operate within the simulation. Ideally such agents should be capable of autonomous behavior, freeing instructors from the burden of having to maintain constant supervision over the agents.

Once agents are inserted in a distributed simulation, they can be used for a variety of pedagogical purposes. In domains where teachers and learners get together after an exercise and review what took place, it is useful for agents to be able to participate in such dialogs. They can then answer questions posed by the learners, and enable the learners to understand what took place from multiple perspectives. In some domains it may be useful to have agents tightly interleave work and pedagogy. A learner might want to ask the agent a question, or watch the agent perform the task. The agent might be able to recognize when the learner is encountering difficulties, and offer assistance as appropriate. Such interactions come much closer to what is observed in real-world apprenticeship learning.

3 Application Projects

The author is involved in two projects aimed at developing agents for virtual learning environments. The first, Soar/IFOR, has been developing automated agents that can participate in training exercises. The second, VET, is extending the range of pedagogically relevant capabilities that can be incorporated into autonomous agents.

3.1 Soar/IFOR

The Soar/IFOR (Tambe et al. 1995) is developing human-like, intelligent agents that can interact with humans, and with each other, in battlefield simulations. The agents play a variety of roles such as fighter pilots, helicopter pilots, and airspace controllers. The fighter pilot agents in particular have been successfully deployed in large-scale simulation exercises, such as the Synthetic Theater of War (STOW) exercise in November, 1994, a four day battle scenario involving approximately 2000 military vehicles.

Soar/IFOR agents have the following capabilities relevant to virtual learning environments. They can be used in multi-agent exercises for pilot training, by networking flight simulators and computers running Soar/IFOR agents. The simulated agents behave in a fashion that appears realistic to the pilots

sitting in the flight simulators. In addition, Soar/IFOR agents have an automated debriefing facility, which enables trainees to ask an agent to explain the motivations for its actions during the engagement, and to give an assessment of what took place from the agent's perspective. This can help pilots to critique their own skills and compare their own performance against that of the agents.

3.2 Virtual Environments for Training

The Virtual Environments for Training project is developing virtual environment technology for a variety of training purposes. The main purpose of the project is to determine how most effectively to employ immersive and intelligent technologies to maximize training effectiveness. This project is being conducted in collaboration with Lockheed Martin, Inc., and the USC Behavioral Technologies Lab.

When these technologies are fully developed, trainees will enter a virtual environment by putting on head-mounted displays. They then will see a model of a work area, such as a control room. The work area may contain a number of human figures, some of which represent other trainees, and some of which represent computer-generated agents. The computer-generated agents may participate in the trainees' activities, may be perform unrelated activities, or act as observers. The agents will be able to explain and demonstrate to trainees how to perform particular tasks, and offer advice and assistance when the trainees encounter difficulties.

4 Architectures for Virtual Environments

The two systems described above share the same basic architecture, which is described below. Each system employs slightly different components, in order to meet somewhat different requirements.

At the core of each environment is a communication bus, on which entities in the simulation broadcast updates of their status. For example, when a student pushes a button on a virtual control panel, the button broadcasts a message indicating that it has been pressed. In the Soar/IFOR application the communication network uses the Distributed Interactive Simulation (DIS) protocol, which is standard among military simulators. Entities in the VET environments communicate using the Tooltalk^{TM} package, which is a more general-purpose communication mechanism.

Objects in the simulated world are modeled using a collection of object simulators. Each object simulator is responsible for controlling the state of a subset of the objects in the simulation, and monitoring the network traffic to track the state of objects controlled by other object simulators. Each object is modeled abstractly within the object simulators as a collection of attributes, just as in many other non-immersive simulations. The role of the object simulator is played in Soar/IFOR by the ModSAF package (Calder et al. 1993); in VET it is played by USC / BTL's RIDES simulator (Munro et al 1993).

Trainees and instructors interact with the simulated world via viewers which monitor the network traffic to determine the state of objects and then produce graphical renderings of those objects. Details of the 3D rendering process are kept local to the viewer, and hidden from the rest of the simulation system. With Soar/IFOR agents flight training simulators may be used, as well as workstation interfaces built on top of the ModSAF package. VET uses the Vista Viewer package developed by Lockheed Martin.

Each agent interacts with simulation via an object simulator, rather than with 3D graphical renderings of objects. The agent "sees" the simulated world as a set of objects, namely those objects that are in the agent's simulated field of view, are discernable by other senses such as hearing, or are accessible by sensors such as radar. Associated with each object is a limited number of attributes describing the properties of the objects that are relevant to the agent's tasks or the student's tasks. For example, for a rotary switch the only attributes that might be modeled are the switch setting and the switch's location on the virtual console; other attributes such as the switch's size and color may be omitted, since they have little bearing on how a trainee might use the switch. The agent operates on the simulation by sending command actions to the object simulator, which cause objects to change state. Since the agents are situated in the virtual world, they must also sense and control their own realization within the virtual world. In the Soar/IFOR case the only realization that other participants see is the aircraft

that the simulated is pilot is flying, so having an agent control its own realization simply involves issuing control commands to the aircraft to climb, turn, etc. In the VET case the agents appear as simulated humans, in which case it is necessary to model body movement. The $Jack^{TM}$ package is to be used for this purpose (Badler et al 1993) .

4.1 Agent Architecture

A key implication of the above system organization is that there is a clear architectural separation between the routines responsible for sensing and motor control and the cognitive processing of the agents. This contrasts with common approaches to agent modeling such as ModSAF, where there is no separate modeling of vehicles and the pilots that control them. The modular organization affords the following advantages. The separation of concerns makes it easier to develop complex behaviors. The virtual world appears to the cognitive agent as a generic object-oriented simulation. This implies that cognitive modeling tools that apply to other object-oriented simulations are also applicable to virtual environments. We therefore have been able to incorporate such tools into our systems.

We use Soar (Laird et al. 1987) to model agents in our virtual environments. Although intended originally as a general cognitive simulation mechanism, and not designed specifically for use within virtual environments, it has proven eminently suited for such environments. If has a number of advantageous features.

- It supports hierarchical task modeling. Processing takes place in a hierarchy of problem spaces which is constructed dynamically during problem solving. Within each problem space, the problem solver applies operators until the goal state of the problem space is achieved, or until the goal is no longer relevant to the current situation. Other task modeling approaches such as GOMS (Card et al 1983) are easily mapped onto this framework.
- It is well suited for modeling goal-directed and reactive behavior in dynamic environments. Soar systems execute via a repetitive decision cycle, in which they repeatedly determine which operator is most applicable at a given moment. If the environment situation suddenly changes, Soar can immediately switch operators if necessary. The precise sequencing of operators thus is not programmed in, but varies depending upon the environment state.
- It incorporates a general learning mechanism called *chunking* which has proven useful in enabling agents to improve their performance over time, in particular agent faculties such as episodic memory, and in modeling skill acquisition (Hill and & Johnson 1993). Chunks are production rules that apply in situations similar to the one in which the chunk was originally learned.

5 Plan Recognition

In order for agents to contribute most effectively to the pedagogical process, they require a number of capabilities beyond the ability to perform the task itself. Several of these capabilities are described in the following sections. The first capability to be discussed is plan recognition.

Plan recognition capabilities enable pedagogical agents to provide appropriate assistance to learners interacting with a complex simulation. If the simulation is in an state it may respond in unexpected ways to the trainee's actions, causing them to fail. This can lead to impasses, where the student is unable to proceed. A tutor called REACT (Hill & Johnson 1995) has been developed which monitors both the trainee's actions and the device states, in order to offer assistance when impasses occur. In a similar manner, students in virtual environments are expected to reach impasses, and require assistance.

The plan recognition technique used in REACT, and being applied in VET, is called *situated plan attribution* (SPA). SPA is a variant of the model tracing approach to student modeling [1], designed to reduce the computational cost of tracking and understanding student actions. During ordinary student activity, student behavior is matched against a library of procedural plans. Recorded in each plan are

the plan steps, their preconditions, and the goal conditions that are expected to be met when the plan is completed. A cognitive model of the task is a source of predictions of which goals and plans may be active. While the steps are being matched, a situation monitor tracks the simulation state and checks for indications that the plan is not having the desired effect or that the environment state necessitates deviation from the plan. When these occur, a more detailed cognitive model of the task is invoked to determine how best to overcome the impasse, or prevent its occurrence in the future.

The following example from the domain of fire control illustrates how this scheme applies to virtual environments. A firefighter trainee enters a virtual simulation of an industrial plant, in which a fire is spreading into a storeroom. The student sprays water on the fire. However, as soon as she does this, some canisters in the storeroom explode, releasing toxic fumes.

As it turns out, the storeroom contains, among other things, chlorine trifluoride, a chemical which is used in fuel propellants and which reacts violently with water, releasing chlorine gas and hydrofluoric acid. The student failed to notice that there were no sprinklers in the store room, a cue that the storeroom contents should not come into contact with water. A tutoring agent monitoring the student behavior recognizes the procedure that the student is following, i.e., fire suppression with water, and also notes the presence of a chemical that could case the plan to fail. By detecting the incompatibility between the plan and the situation the agent is able to explain what is wrong with the plan, or perhaps intervene before the damage is done.

In general impasse recognition requires detecting when the simulation is in an undesirable state. Some states, such as the explosion in the above example, are undesirable regardless of the student's plan. These are recognized by context-independent rules that monitor the states of simulation objects. Other impasses are implied by the failure of the student to make progress toward the goal state of the plan. In either case, a detailed model of the student's intentions is not required; at most the agent needs to know what the student's goal is, so that it tell whether or not the student is making progress. This contrasts with conventional model tracers which try to model every action the student makes.

6 Explanation and Debriefing Capabilities

Explanation and debriefing capabilities are essential in order to assist students as they work in virtual environments. Agents may need to explain to students what is wrong with their actions, or answer questions about the actions that they are performing. The explanation may occur in the midst of the activity, or afterwards during a debriefing session, depending upon the nature of the activity. Several agent capabilities have been developed that are necessary for explanation and debriefing: a decision analysis capability that determines what aspects of a decision should be explained, an episodic memory capability that enables agents to recall episodes and analyze and discuss them, and a multimedia generation capability. These capabilities are implemented in a domain-independent way, so that they can be adapted to a variety of different types of agents.

6.1 Decision analysis

Decision analysis enables agents to examine in detail the decisions that led to their own actions, in order to determine how best to explain them. It reorganizes the agent's knowledge into a form suitable for explanation (Johnson 1994) . Knowledge encoded in rules is ill-suited for explanation, because the rules may refer to data structures that are internal to the agent. What is of interest to a student is what goals and situational factors led to a decision, and what alternative choices were available, not what data structures were used. The decision analysis capability determines these by proposing hypothetical changes to the situation in which the decision was made. After each change, it replays the decision, to see whether the same outcome results. If it does, the attribute must not have been relevant from the decision. If a different decision results, the agent analyzes the alternative decision in a similar fashion In the process of this analysis, chunks are built that recognize the factors are relevant to similar decisions in the future. Once the agent's knowledge is reorganized into these chunks, further decision analysis

is unnecessary. The agent thus becomes increasingly proficient at explaining to people how to perform tasks.

6.2 Maintaining an Episodic memory

Episodic memory involves at least two abilities: to remember what events occurred during an activity, and to recall what the situation was at the time that the event occurred. Episodic memory is essential for pedagogical agents, because it allows agents to discuss the activity after it is over. It also proves useful in other ways; for example, it is being used to enable Soar/IFOR compare successful and unsuccessful missions, and thereby learn from experience (Johnson & Tambe 1995) .

Episodic memory in our agents is currently organized as follows. Events are recorded sequentially as they occur. At the same time, chunking is used to record changes to the agent's internal memory state. These chunks enable the agent to recall the state in which each event occurred.

The episodic memory mechanism relies on two sets of chunks. The first set consists of *recognition chunks*, which fire in response to some description that serves as a memory probe, indicating that an instance matching the probe has been seen before. In the virtual agent case, the memory probe consists of a description of an event, together with a possible state change. If the state change occurred while the event was observed, the recognition chunk will fire. The second set of chunks are *recall chunks*, which recall the complete state in which an event occurred, when presented with an event description as a memory probe. The first time Soar/IFOR attempts to recall the state associated with an event, it first recursively tries to recall the state immediately preceding the event. It then proposes possible state changes that might have occurred. The recognition chunks recording state changes then fire, identifying the state changes that in fact occurred. Once the recall process is complete, a recall chunk is created, so that the next time the event is used as a memory probe the state is immediately recalled. This type of chunking process is known as *data chunking*, and has been studied extensively in Soar systems (Rosenbloom et al. 1987)

6.3 Presentation Capabilities

Once an agent has determined what information needs to conveyed to a student, it must determine how to present the information. The after-action debriefing system in Soar/IFOR uss a combination of graphical depictions of episodes during the engagement, together with natural language text commentary describing that the agent was doing at that point and why. Question-answer dialogs can be obtained by selecting segments of natural language text and asking for explanations and justifications of the assertions made at that point in the text.

The presentation generator is designed to be applicable to a range of presentation media. For each presentation medium (e.g., snapshots of engagements or natural language), there is a description of what kinds of information that medium can represent. The generator then allocates the information to be presented among the available media. This approach can be extended to new kinds of graphical media such as those used in immersive displays.

6.4 Additional Issues

The above capabilities are likely to provide a basis for effective pedagogical use of agents in virtual environments. There are, of course, a range of additional issues raised by the use of pedagogical agents, which will be explored in the VET project.

One of the most important capabilities afforded by agents in virtual environments is the ability to *demonstrate* activities, rather than simply to explain how they are performed. This is especially important in the early phases of cognitive apprenticeship in order to model the skill being taught, and is bound to be useful in other situations as well. Demonstration-based help has been investigated in the context of 2D interfaces (Sukaviriya 1988) , but has not yet been integrated into immersive environments.

We are exploring techniques that may help instructional designers to author agent behavior. Richard Angros in our group is adapting programming-by-demonstration techniques for this purpose. Packages such as KidSim (Smith et al. 1994) have already demonstrated the effectiveness of such techniques in restricted contexts.

Another important problem is how to assign personalities to agents that are consistent with their role in the modeled activity. Reeves and his colleagues have argued that people interact with computers at a personal and emotional level [14], and that cooperation between computer and human is facilitated if the computer exhibits a personality that is compatible with that of the human teammate [6]. These factors are bound to be even more important with agents that assume concrete form within the environment. This may prove to be a way of getting a deeper understanding of the problem of designing effective pedagogical strategies. After all, pedagogical interactions are simply another type of interpersonal interaction. Therefore selecting an appropriate style of interpersonal interaction for agents is a necessary step toward identifying effective strategies for pedagogical interaction.

7 Conclusion

This paper has described the main capabilities required of computer-generated agents to make them pedagogically effective, and some techniques for implementing these capabilities. Because many of the component technologies are being applied to multiple application areas, generic implementations of these capabilities are beginning to emerge. Since the cost of graphics processors is dropping rapidly, virtual environments will soon be practical for a range of educational applications. The technologies described here can be an important contribution to such applications.

Much work remains to be done to evaluate the effectiveness of the pedagogical agent approach. Formative evaluations with subject matter experts have been conducted, and pilot studies have been performed evaluating component technologies with subjects, all with promising results. However, formal evaluations have not been conducted, and are not feasible until a broader range of capabilities are integrated into a single agent. Evaluations are currently being planned with James Fleming's group at Armstrong Laboratory.

If the agent-based approach proves successful, it will enable a range of new techniques for facilitating learning. For example, instructional designers could define the instructional goals of an exercise, and leave it up to the agents to recognize and exploit situations in which those goals can be achieved. A variety of new possibilities in learning environment design are thus waiting to be explored.

Acknowledgement

The author gratefully acknowledges the contribution of the other members of the team Soar/IFOR team, including John Laird, Paul Rosenbloom, Jill Lehman, Randolph Jones, Karl Schwamb, and Frank Koss, and of the members of the VET project, in particular Randy Stiles, Allen Munro, and Pedro Szekely, and John Schlossberg. This research was supported in part under contract N00014-92-K-2015 from the Advanced Systems Technology Office (ASTO) of the Advanced Research Projects Agency (ARPA) and the Naval Research Laboratory (NRL) to the University of Michigan, via a subcontract to USC; and under contract N66001-95-C-6013 from ARPA and the Naval Command and Ocean Surveillance Center, RDT&E division (NRAD). It was also supported in part by the Office of Naval Research.

References

[1] J.R. Anderson, C.F. Boyle, A.T. Corbett, and M.W. Lewis. Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, 42:7–49, 1990.

[2] N. Badler, C. Phillips, and B. Webber. *Simulated Agents and Virtual Humans*. Oxford University Press, 1993.

[3] R.B. Calder, J.E. Smith, A.J. Courtemanche, J.M.F. Mar, and A.Z. Ceranowicz. ModSAF behavior simulation and control. In *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, pages 347–359, Orlando, FL, March 1993. Institute for Simulation and Training, University of Central Florida.

[4] S.K. Card, T.P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.

[5] A. Collins, J.S. Brown, and S.E. Newman. Cognitive apprenticeship: teaching the crafts of reading, writing, and mathematics. In L.B. Resnick, editor, *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*, pages 453–494. Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.

[6] B.J. Fogg and Y. Moon. Computer as teammate: Effects on user attitude and behavior. In *Proceedings of Lifelike Computer Characters '94*, page 54, Snowbird, Utah, October 1994.

[7] R.W. Hill and W.L. Johnson. Designing an intelligent tutoring system based on a reactive model of skill acquisition. In *Proceedings of the World Conference of Artificial Intelligence in Education*, pages 273–281, Edinburgh, Scotland, 1993.

[8] R.W. Hill and W.L. Johnson. Situated plan attribution. Accepted for publication in *Journal of Artificial Intelligence in Education*, 1995.

[9] W.L. Johnson. Agents that learn to explain themselves. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1257–1263, Seattle, WA, August 1994. AAAI, AAAI Press.

[10] W.L. Johnson and M. Tambe. Using machine learning to extend autonomous agent capabilities. In *Proceedings of the Summer Computer Simulation Conference*, pages 275–280, Ottawa, ONT, 1995.

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

[12] D.C. Merrill, B.J. Reiser, M. Ranney, and J.G. Trafton. Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems. *The Journal of the Learning Sciences*, 2(3):277–305, 1992.

[13] A. Munro, M.C. Johnson, D.S. Surmon, and J.L. Wogulis. Attribute-centered simulation authoring for instruction. In *Proceedings of the AI-ED 93 World Conference of Artificial Intelligence in Education*, pages 82–89, Edinburgh, Scotland, 1993.

[14] B. Reeves. Emotional responses to media. In *Proceedings of Lifelike Computer Characters '94*, page 67, Snowbird, Utah, October 1994.

[15] P.S. Rosenbloom, J.E. Laird, and A. Newell. Knowledge level learning in Soar. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 618–623, Menlo Park, CA, July 1987. American Association for Artificial Intelligence.

[16] D.C. Smith, A. Cypher, and J. Spohrer. KidSim: Programming agents without a programming language. *CACM*, 37(7):55–67, July 1994.

[17] E. Soloway. Beware, techies bearing gifts. *Communications of the ACM*, 38(1):17–24, January 1995.

[18] B. Sterling. War is virtual hell. *Wired*, 1(1):46–51, 1993.

[19] N. Sukaviriya. Dynamic construction of animated help from application context. In *Proceedinggs of the ACM SIGGRAPH 1988 Symposium on User Interface Software and Technology (UIST'88)*, pages 190–202, New York, NY, 1988.

[20] M. Tambe, W.L. Johnson, R.M. Jones, F. Koss, J.E. Laird, P.S. Rosenbloom, and K. Schwamb. Intelligent agents for interactive simulation environments. *AI Magazine*, 16(1):15–39, Spring 1995.