Steve Meets Jack: the Integration of an Intelligent Tutor and a Virtual Environment with Planning Capabilities

Gonzalo Méndez[1], Jeff Rickel[2], and Angélica de Antonio[1]

[1] Computer Science School
Technical University of Madrid
Campus de Montegancedo, 28660 Boadilla del Monte (Madrid)
gonzalo@gordini.ls.fi.upm.es, angelica@fi.upm.es
[2] Information Sciences Institute
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292-6695
rickel@isi.edu

Abstract. In this paper, we describe how we have integrated Steve, an intelligent tutor based on Soar, and HeSPI, a human simulation tool for planning and simulating maintenance tasks in nuclear power plants. The objectives of this integration were to test Steve's flexibility to be used in different applications and environments and to extend HeSPI to use it as a virtual environment for training. We discuss the problems encountered and the solutions we have designed to solve them.

1 Introduction

Intelligent animated agents that interact with human users and other agents in virtual worlds have been applied to a wide variety of applications, such as education and training [1], therapy [2], and marketing [3, 4]. However, it is rare for such an agent to be reused across multiple virtual worlds, and rarer still for one to be applied to a new virtual world developed independently, rather than with the agent in mind. Most animated agents, especially those that interact closely with their virtual world, were designed for a particular virtual world and application. Because of the effort required for the development of such agents, it would be desirable to be able to reuse these agents in new environments easily. An unlimited exchange of agents and environments will only be possible if some standard is developed to facilitate this task. Unfortunately, this standard is still far from being available.

In this paper, we describe an experiment in reusing an intelligent animated agent, Steve [5, 6], in a new virtual world, HeSPI [7], which was developed independently. This experience has allowed us to reflect on the problems that may arise when doing such an integration as a first step towards the definition of a new starndard.

Steve was designed to be easy to apply to new domains and virtual worlds. It was originally applied to equipment operation and maintenance training on board a virtual ship. Subsequently, it was significantly extended and applied to leadership training in virtual Bosnia [8]. However, the leadership training application was designed with Steve in mind. In contrast, HeSPI was developed independently as a tool for equipment operation and maintenance training for Nuclear Power Plants (NPPs). Thus, HeSPI provides a good test for how portable Steve really is.

In addition to evaluating Steve's portability, our experiment was also motivated by a real need. HeSPI was designed as a tool for planning and simulating procedures in nuclear power plants. Via a user interface, an experienced operator can specify the required steps in a procedure, and they will be carried out by an animated human figure, Jack [9]. These procedures can subsequently be replayed by a less experienced operator learning to perform them, thus providing a training tool. However, the trainee is limited to watching the procedure, and cannot ask questions, get explanations, or practice the task himself, something that has already proven to be useful in previous systems applied to training in NPPs [10]. In contrast, Steve can demonstrate procedures, it can monitor students as they practice a task, giving them feedback on their actions, and it can answer simple questions. Thus, integrating Steve into HeSPI would greatly enhance its training capabilities.

In the remainder of the paper, we provide brief background on Steve and HeSPI and then discuss our experience integrating the two. As we will discuss, many aspects of the integration went smoothly, but some difficulties did arise. The results of our effort serve not only as an evaluation of Steve and HeSPI but also as important lessons in building portable agents and flexible virtual worlds.

2 Steve

Steve (Soar Training Expert for Virtual Environments) is an autonomous, animated agent for training in 3D virtual environments. Steve's role is to help students learn procedural tasks, and he has many pedagogical capabilities one would expect of an intelligent tutoring system. However, because he has an animated body, and cohabits the virtual world with students, he can provide more human-like assistance than previous disembodied tutors. For example, he can demonstrate actions, use gaze and gestures to direct a student's attention, guide students around in the virtual world, and play the role of missing teammates for team training. His architecture allows him to robustly handle a dynamic virtual world, potentially populated with people and other agents; he continually monitors the state of the virtual world, always maintaining a plan for completing his current task, and revising the plan to handle unexpected events. This novel combination of capabilities makes Steve a unique substitute for human instructors and teammates when they are unavailable.

To support these capabilities, Steve consists of three main modules: perception, cognition, and motor control [5]. The perception module monitors messages from other software components, identifies relevant events, and maintains a snapshot of the state of the world. It tracks the following information:

the simulation state (in terms of objects and their attributes), actions taken by students and other agents, the location of each student and agent, the objects within a student's field of view, and human and agent speech. The cognition module, implemented in Soar [11], interprets the input it receives from the perception module, chooses appropriate goals, constructs and executes plans to achieve those goals, and sends motor commands to the motor control module. The cognition module includes a wide variety of domain-independent capabilities, including planning, replanning, and plan execution; mixed-initiative dialogue; assessment of student actions; simple question answering; episodic memory; path planning; communication with teammates [6]; and control of the agent's body. The motor control module accepts the following types of commands: move to an object, point at an object, manipulate an object (about ten types of manipulation are currently supported), look at someone or something, change facial expression, nod or shake the head, and speak. The motor control module decomposes these motor commands into a sequence of lower-level messages that are sent to the other software components (simulator, graphics software, speech synthesizer, and other agents) to realize the desired effects.

Steve was designed to make it easy to connect him to new virtual worlds. First, the perception and motor control modules include two layers: an abstract layer, which deals with the types of information Steve needs to exchange with the other software components, and a virtual world interface layer, which maps the abstract layer to the messages used to communicate with a particular set of software components. Thus, one can connect Steve to a new virtual world implemented with new software components simply by rewriting the virtual world interface layer.

Second, to allow Steve to operate in a variety of domains, his architecture has a clean separation between domain-independent capabilities and domain-specific knowledge. The code in the perception, cognition, and motor control modules provides a set of general capabilities that are independent of any particular domain. To allow Steve to operate in a new domain, a course author simply specifies the appropriate domain knowledge in a declarative language. The domain knowledge that Steve requires falls in two categories:

Perceptual Knowledge This knowledge tells Steve about the objects in the virtual world, their relevant simulator attributes, and their spatial properties. It resides in the perception module.

Task Knowledge This knowledge tells Steve about the procedures for accomplishing domain tasks and provides text fragments so that he can talk about them. It resides in the cognition module and is organized around a relatively standard hierarchical plan representation [5].

Finally, Steve's cognition module has a library of actions he understands, including things like manipulating objects, moving objects, and checking the state of objects. Each action in the library is implemented by a set of Soar production rules that tell Steve what sorts of messages to send out to perform the action and what sorts of messages to expect from the simulator to know whether the action succeeded. The library is organized in a hierarchy with inheritance so that new actions can often be written by simply specializing existing (general) actions.

3 HeSPI

VRIMOR is a shared cost project funded by the European Union whose aim has been to combine environmental laser-scanning technologies with human modelling and radiological dose estimating tools and to deliver an intuitive and cost-effective system to be used by operators involved with human interventions in radiologically controlled areas [7].

HeSPI (Tool for Planning and Simulating Interventions) is a tool that has been developed in the VRIMOR project to serve as a means to plan and simulate maintenance tasks in a 3D environment reproducing a Nuclear Power Plant. This system has been built on top of Jack [9], a human simulation tool that provided us with the necessary mechanisms to import the scanned virtual environment (VE) and animate the virtual operators.

HeSPI provides users with two kinds of interfaces. One is the classic graphical interface which is based on windows and is controlled using a mouse and a keyboard. This has been complemented with a voice recognition system that allows the user to interact with the system in a much faster way.

Once an operation is planned, the user can generate as an output the trajectories of all the operators that have taken part in the intervention. These trajectories measure the position of the operators' head, hands and chest during the operation, so they can be used by an external application to determine the radiation dose received by each operator during the intervention.

There are some other actions the user can perform, such as including semantic information about the objects of the scene or creating new actions to complete the predefined library of activities that HeSPI offers for the operators.

HeSPI has a very simple architecture that makes it easy to communicate with other applications. Since one of the main objectives of the project was to test different user interfaces, we created an API to control all the actions that could be performed in the VE, so that different applications could make use of the API to communicate with the VE.

HeSPI has already been under evaluation at the Almaraz NPP (Spain) and the results have been quite satisfactory, especially in those aspects concerning the voice controlled tasks.

4 Integration

The integration of these two systems, Steve and HeSPI, has taken place during the second half of 2002. There are still some features that have to be fully tested, but in general it has been quite successful.

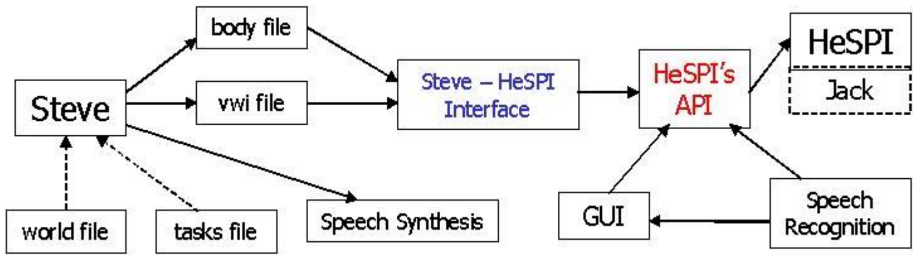

Fig. 1. Architecture for the integration

4.1 The Integration Process

As described earlier, there are several steps in connecting Steve to a new VE. The architecture for this integration can be seen in Fig.1.

First of all, it was necessary to let Steve connect to the VE and have its own physical representation. HeSPI already provided the necessary functions to create a virtual operator using the GUI, so it was just a matter of letting Steve use them. Steve's virtual world interface layer made this easy.

As a second step, it was necessary to provide Steve with all the domain specific knowledge he needed about the virtual world, i.e. the objects of the world and the tasks that had to be performed in the training process, so that he could have all the necessary information to act as a tutor. Steve's representation for task knowledge was sufficient to represent the tasks that had to be carried out inside the NPP, since procedures are so strict in NPPs that, in most cases, only one course of action is valid to complete a task.

The most logical way to continue the integration was to let Steve perform basic actions, where no interaction with the environment was required, so we chose to make him walk around the VE and look at different places. This involved having to modify the control flow in HeSPI, since it was designed to return the control to the user once the execution of a command had started, so that the user could keep on working while the command was being executed. However, Steve needs to know when a command (e.g., to walk to a new location) has finished executing. Thus, it was necessary to make HeSPI wait for the action to finish before returning the control to Steve, so that the system would work properly.

The interaction with objects required a bigger effort than the previous steps, since these interactions are, in many cases, domain dependent. In previous occasions, Steve had been used in static environments where he had to press buttons, move handles or turn valves. In this case, we were facing a more dynamic environment, where he had to move objects or use tools to mount and dismount some pipes.

To let Steve work in the NPP it was necessary to extend the existing action library in order to allow him to perform tasks such as picking up objects, which was defined in terms of an existing action, or dropping them in a specific location, which was defined from scratch.

At this point, Steve was already able to perform all the required actions, but he wasn't able to explain them to the student, yet. We used IBM's Via Voice as the text to speech tool to let Steve communicate with the student. It was easy to map Steve's required speech synthesis API, which is part of the virtual world interface layer, to the commands provided by ViaVoice.

Finally, Steve's path planning requires the virtual world to be described as a graph, where the nodes are different locations in the VE and the edges establish the possibility to go from one place to another [5]. This information was easy to provide for the nuclear power plant.

4.2 Difficulties

There were undesired behaviours due to the fact that both Steve and HeSPI performed redundant actions. When Steve has to work with an object, he first tries to approach it and then sends the command to work with the object. When HeSPI receives this command, it tries to make the mannequin approach the object and then commands him to work with it. Thus, the mannequin receives the command to walk towards the object twice and, due to Jack's management of positions and movements, this produces undesired results.

Another difficulty we found was that, for some actions, Steve needs information that external applications may not have. For example, when dropping an object, you need to tell Steve where you drop it. However, HeSPI doesn't need this information, since a mannequin drops his objects wherever he is in that moment. Thus, when a mannequin moved somewhere, it was necessary to store that destination in a temporary register so it could be used whenever Steve performed an action that required that location.

To grasp and manipulate objects, Jack requires more information about them than Steve's perceptual knowledge provides. Thus, Jack sometimes grasps things awkwardly. This is a problem in HeSPI too: users complained that it was too complicated to provide such information. Ultimately, some extension in Steve or HeSPI needs to compute such information automatically.

Steve has been used in environments where he couldn't change objects' location. However, in a NPP, many tasks involve moving objects from one place to another. As both Steve and the student have their own body, if Steve started explaining an action where he had to move an object and the student wanted to finish that action, he would have to take the object from Steve's hands. Then, if the student needed some help to finish the task, Steve would have to take the object again, which would complicate the process. A possible solution would be for Steve and the student to share the same body, but it would cause two new problems. First, the VE should allow different users to manipulate the same mannequin, and if that were possible, then Steve and the student might try to manipulate the mannequin at the same time. Thus, the types of actions in a domain may constrain the times at which Steve and a student can switch control.

Steve assumes that once he commands an action, it will be possible to perform it. If that is not the case, because, for example, there is an unexpected obstacle in the VE that doesn't allow Jack to keep on moving towards the object Steve wants to manipulate, there might be two courses of action. If we decide not to send feedback to Steve until the action has finished, he will be waiting for a callback that may never arrive. If we return a value saying that the action couldn't be performed, he would keep on trying to carry it out, because his plan says that it is the right action to perform. Thus, Steve needs extensions that support more general reasoning about failed actions.

5 Future Work

HeSPI has been tested with an intervention where a team of operators have to change a filter that is kept inside a pipe. We have used this operation to test the integration with Steve, so that the number of actions that Steve has had to perform is still a bit limited. Thus, one of the first things that must be done in the near future is to test both HeSPI and Steve with different operations in order to see if Steve's action library is good enough for these and other environments.

In addition, the integration has been completed for a single tutor teaching a single student, but further work is needed in order to test whether teamwork can also be supported, so that different Steve agents can perform the role of a tutor or a teammate in HeSPI. Steve is already able to support teamwork, and there shouldn't be much trouble in integrating this functionality with HeSPI, since tutors and teammates would be treated as different users.

One last thing that could be achieved is the generation of Steve's domain specific knowledge using HeSPI's planner and semantic information about the world. As far as we have been able to see, not all the necessary information could be generated, but a fairly complete skeleton could be created.

6 Conclusions

We have tested both Steve and HeSPI in order to see how easy it might be to integrate an intelligent tutor and a VE developed independently. We have proven that Steve is flexible enough to plug it into a different application for training in new domains. In addition, we have shown that HeSPI has been designed in such a way that is has been easy to convert it in a training tool.

However, some issues have arisen that require further consideration. For the most part, they have to do with the responsibilities each application must assume and with the information each system must provide to external systems and can expect to receive from other applications. For example, the responsibility of deciding whether to approach an object before manipulating it should be assumed by the agent, whereas the VE should only check if the action is physically feasible. Besides, the VE should be able to provide complete information about the state of the environment and the effects of any event that may occur. Actions should be parameterized and the agents should be able to set these parameters or let them take default values (i.e. walk expressing a certain mood or just walk normally).

As it can be seen, a big effort must still be devoted to this standardization process so that, in the near future, it may be possible to easily connect any agent to any VE.

Acknowledgements Steve was funded by the Office of Naval Research under grant N00014-95-C-0179 and AASERT grant N00014-97-1-0598. HeSPI has been funded by the EU through the Vrimor project under contract FIKS-CT-2000-00114. This research has been funded by the Information Sciences Institute - University of Southern California, the Spanish Ministry of Education under grant AP2000-1672 and the Spanish Ministry of Science and Technology through the MAEVIF project under contract TIC2000-1346.

References

1. Johnson, W.L., Rickel, J.W., Lester, J.C.: Animated pedagogical agents: Face-to-face interaction in interactive learning environments. International Journal of Artificial Intelligence in Education **11** (2000) 47–78
2. Marsella, S.C., Johnson, W.L., LaBore, C.: Interactive pedagogical drama. In: Proceedings of the Fourth International Conference on Autonomous Agents, New York, ACM Press (2000) 301–308
3. André, E., Rist, T., van Mulken, S., Klesen, M., Baldes, S.: The automated design of believable dialogues for animated presentation teams. In Cassell, J., Sullivan, J., Prevost, S., Churchill, E., eds.: Embodied Conversational Agents. MIT Press, Cambridge, MA (2000)
4. Cassell, J., Bickmore, T., Campbell, L., Vilhjálmsson, H., Yan, H.: Conversation as a system framework: Designing embodied conversational agents. In Cassell, J., Sullivan, J., Prevost, S., Churchill, E., eds.: Embodied Conversational Agents. MIT Press, Cambridge, MA (2000)
5. Rickel, J., Johnson, W.L.: Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. Applied Artificial Intelligence **13** (1999) 343–382
6. Rickel, J., Johnson, W.L.: Extending virtual humans to support team training in virtual reality. In Lakemayer, G., Nebel, B., eds.: Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann, San Francisco (2002) 217–238
7. de Antonio, A., Ferré, X., Ramírez, J.: Combining virtual reality with an easy to use and learn interface in a tool for planning and simulating interventions in radiologically controlled areas. In: 10th International Conference on Human - Computer Interaction, HCI 2003, Creta, Greece (2003)
8. Rickel, J., Marsella, S., Gratch, J., Hill, R., Traum, D., Swartout, W.: Toward a new generation of virtual humans for interactive experiences. IEEE Intelligent Systems **17** (2002) 32–38
9. Badler, N.I., Phillips, C.B., Webber, B.L.: Simulating Humans. Oxford University Press, New York (1993)
10. Méndez, G., de Antonio, A., Herrero, P.: Prvir: An integration between an intelligent tutoring system and a virtual environment. In: SCI2001. Volume VIII., Orlando, FL, IIIS, IEEE Computer Society (2001) 175–180
11. Laird, J.E., Newell, A., Rosenbloom, P.S.: Soar: An architecture for general intelligence. Artificial Intelligence **33** (1987) 1–64