Integrating an Intelligent Tutoring System for TAOs with Second Life

Jeremy Ludwig, Erik Sincoff,
Emilio Remolina, Richard Stottler

Stottler Henke Associates, Inc.

San Mateo, CA

ludwig, esincoff, remolina, stottler
@stottlerhenke.com

Anh Nguyen

Naval Undersea Warfare
Center, Division Newport

Newport, RI

anh.b.nguyen1@navy.mil

ABSTRACT

The Tactical Action Officer on board a U.S. Navy Cruiser, Destroyer, or Frigate is responsible for the operation of the entire watch team manning the ship's command center. Responsibilities include tactical decision-making, console operation, communications, and oversight of a variety of watchstander responsibilities in air, surface, and subsurface warfare areas. In previous work the PORTS TAO ITS, an Intelligent Tutoring System (ITS) for the instruction of Tactical Action Officers (TAOs) was developed to support training at the Surface Warfare Officers School. The system was built on the PC-based Open-architecture Reconfigurable Training System (PORTS). This paper describes a novel extension of the PORTS ITS, where it is integrated with the popular Second Life 3D virtual world. In this integration, the TAO logs on as an avatar in Second Life (SL) and interacts with a number of computer-controlled objects that take on the roles of the TAO's teammates. TAOs rely on the same mechanism to communicate with the simulated teammates in SL as they would to communicate with other human players. That is, the TAO speaks to these simulated teammates using the chat window built into SL and sees their replies and comments in the chat window as well. We provide both a high-level overview of the integration process as well as the details of integrating a deployed training system with the Second Life virtual world. Additionally, this paper presents "food for thought" on how recent advances in technology and social connectivity can be applied to military training domains and outlines proposed future work based on this integration.

ABOUT THE AUTHORS

Jeremy Ludwig, Ph.D. is a Research Scientist and Project Manager at Stottler Henke Associates. His research areas include intelligent tutoring systems, behavior modeling, and machine learning, focusing on a number of research projects that utilize both game and simulation technology for training. He joined Stottler Henke in the fall of 2000 and holds a Ph.D. in computer science from the University of Oregon.

Erik Sincoff is a Lead Software Engineer at Stottler Henke Associates. At Stottler Henke, he has focused on the design and implementation of intelligent tutoring systems in a variety of domains including emergency medical training, military equipment training, and homeland security training. Previously, he was a senior software engineer at Teknowledge Corporation, where he worked in numerous domains, including implementing tutors in multiuser worlds. He has been at Stottler Henke since 2005 and has a MS in computer science from Stanford University.

Emilio Remolina, Ph.D. is an Artificial Intelligence Research Scientist at Stottler Henke Associates. He received his Ph.D. in Computer Science from the University of Texas at Austin in 2001. Dr. Remolina has been the main designer and developer of different ITS systems: ICT (Intelligent Counter Intelligence in Combating Terrorism Tutor), CITTP (Computerized Individual Trainer for Team Performance), AIS-IFT (helicopter flying trainer) and PORTS TAO-ITS (tactical training of TAOs using the AEGIS system). Dr. Remolina's research interests include intelligent tutoring systems, planning, simulation and common sense reasoning.

Richard Stottler co-founded Stottler Henke Associates, Inc., an artificial intelligence consulting firm in San Mateo, California, in 1988 and has been the president of the company since then. He has been the principal investigator on a large number of tactical decision-making intelligent tutoring system projects conducted by Stottler Henke including projects for the Navy, Army, Air Force and Marine Corps including the PORTS TAO ITS for the US Navy, and a Littoral Combat Ship project. He has a Masters degree in Computer Science from Stanford University.

Anh Nguyen is a Computer Engineer at the Naval Undersea Warfare Center (NUWC) located in Newport, RI. He holds a B.S. in Computer Systems Engineering from the University of Massachusetts Amherst. He works in the Undersea Warfare Combat Systems Department, Tactical Control and Contact Management Branch. His current assignment entails prototyping, design, and development of future capabilities for the Advanced Processor Build (APB) - Tactical program, focusing on Target Motion Analysis (TMA) enhancements for the fleet. He is also the NUWC Virtual Training Lead for the NUWC Metaverse Exploration Project in which he focuses on leveraging the capabilities of the Virtual World Technologies (VWTs) in order to improve current conventional Undersea Warfare (USW) training.

Integrating an Intelligent Tutoring System for TAOs with Second Life

Jeremy Ludwig, Erik Sincoff, Emilio Remolina, Richard Stottler
Stottler Henke Associates, Inc.
San Mateo, CA
ludwig, esincoff, remolina, stottler @stottlerhenke.com

Anh Nguyen
Naval Undersea Warfare Center, Division Newport
Newport, RI
anh.b.nguyen1@navy.mil

INTRODUCTION

The Tactical Action Officer (TAO) on board a U.S. Navy Cruiser, Destroyer, or Frigate is responsible for the operation of the entire watch team manning the ship's command center. Responsibilities include tactical decision-making, console operation, communications, and oversight of a variety of watchstander responsibilities in air, surface, and subsurface warfare areas. The PORTS TAO ITS has been previously implemented and deployed to provide instruction for Tactical Action Officers (TAOs) in training at the Surface Warfare Officers School (Stottler et al., 2007). This Intelligent Tutoring System (ITS) builds on the PC-based Open-architecture Reconfigurable Training System (PORTS).

This paper describes a novel extension of the PORTS ITS, where it is integrated with the popular Second Life 3D virtual world. In this integration, the TAO logs on as an avatar in Second Life (SL; www.secondlife.com) and interacts with a number of computer-controlled objects that take on the roles of the TAO's teammates. The TAO relies on the same communication mechanism to interact with the simulated teammates in SL as they would with other human avatars in SL. That is, the TAO "speaks" to these simulated teammates using the chat window built into SL and sees their replies and comments in the chat window as well. Communication with others is a vital part of the TAO's responsibilities, which creates the opportunity to investigate extending this training to social virtual worlds. The communication modality differs in the extension (voice input/audio output in PORTS ITS; chat input/output in SL) but the communication acts remain the same.

The motivation behind this extension is to provide a way to investigate the possible benefits of such an integration. That is, this integration can be used in the future to experiment with how to take advantage of the significant capabilities provided by SL (support for online, multi-player interaction in a 3D environment) to enhance training for the Navy. For example, one possibility lies in providing support for simulated role players to be optionally replaced by human avatars. With this type of functionality, a human could step in as the instructor for more ambiguous scenarios even if the TAO and instructor were at different locations. Taken further, it could also support multiple configurations of team training, with simulated role players used to fill in the gaps when humans are not available.

The remaining introduction provides background information on the training problem, the PORTS system, and the ITS. Following this, the methods section contains both a high-level overview of the integration process as well as the details of integrating a deployed training system with the Second Life virtual world. The outcome of this integration is briefly described in the results section. Finally, the conclusion section includes lessons learned while performing this integration and directions for future work. While this paper does not contain experimental results on the evaluation of the efficacy of SL integration, it does provide the technical details, and lessons learned to assist others with similar integration efforts. Additionally, this paper presents "food for thought" on how recent advances in technology and social connectivity can be applied to military training domains.

Background

The mission of the Surface Warfare Officers School (SWOS) in Newport, Rhode Island is to provide professional education and training to prepare officers of the U.S. Surface Navy to serve at sea. As part of his training at SWOS, each Surface Warfare Officer learns how to "fight" their ship as a Tactical Action Officer. The TAO training consists of three months of classroom and simulator time wherein students are exposed to all elements of surface warfare; air, surface, subsurface, and amphibious operations as well as electronic and other support mechanisms. One major responsibility a TAO has is to be able to exercise command over the major systems of their ship (weapons, support platforms, radar and sonar, and navigation) during potentially hostile situations. The tactical decisions he makes during such situations can easily affect the outcome of the ship's mission, as well as having life or death consequences. In summary, to paraphrase the SWOS instructors, the TAO gathers information, analyzes it, and ensures, by issuing verbal orders and queries that the correct decisions are made and actions taken based on the tactical situation. Additionally, TAOs "command by negation" in that much of what the Combat Information Center (CIC) team needs to accomplish in tactical situations is decided upon autonomously by individual watchstanders who also state their intentions before they execute their tasks. When these decisions are correct, the TAO must merely acknowledge the watchstander's decision. However, if these decisions are incorrect or omitted, the TAO must negate the incorrect decision or proactively initiate the omitted actions.

PORTS

Previously, in order to address the need for watchstanders to practice tactical scenarios without requiring the use of expensive special purpose hardware, the Navy commissioned the development of the Generic Reconfigurable Training System (GRTS) now renamed the PC-based Open-architecture Reconfigurable Training System (PORTS), which replicates watchstation functionality with high fidelity, on low-cost generic PC hardware. One of the PORTS watchstations is the TAO's. It also includes a simulation of the naval tactical environment. The system was already used at SWOS to train TAO students in console operation, though an instructor was needed for every two students, to play the role of other CIC team members and provide tutoring. SWOS had already set up an electronic classroom that included 42 student PCs for viewing electronic materials, networked to a single instructor console.

Figure 1 shows the PORTS simulated TAO console. It includes panels for Variable Action Buttons (VABs), display selection (map control keys), radio control, tactical situation map (a scaled version of the large screen display), and Automatic Status Boards that, among other things, display information on the hooked track. The mouse is used to push buttons and select tracks. A tactical simulation of the ownship's sensors and weapons, external platforms, and the environment drives these displays. PORTS simulations are initialized from a PORTS scenario file created using a graphical scenario editor.

PORTS TAO ITS

PORTS TAO-ITS (Stottler et al., 2007) uses a learn-by-doing strategy whereby the TAO is presented with a computer-simulated tactical situation, in which he should act as if aboard an actual ship. The ITS uses a high-fidelity simulation of the Aegis system consoles based on PORTS. Artificial intelligence techniques are employed to model the behavior of automated crewmembers that autonomously react to the tactical situation and interact among them and with the TAO. As described previously, the majority of the TAO's decisions are manifested by verbal commands. The ITS enables TAO students to interact naturally, using spoken language to command and query simulated entities corresponding to other crewmembers and off-ship personnel through a natural language interface.

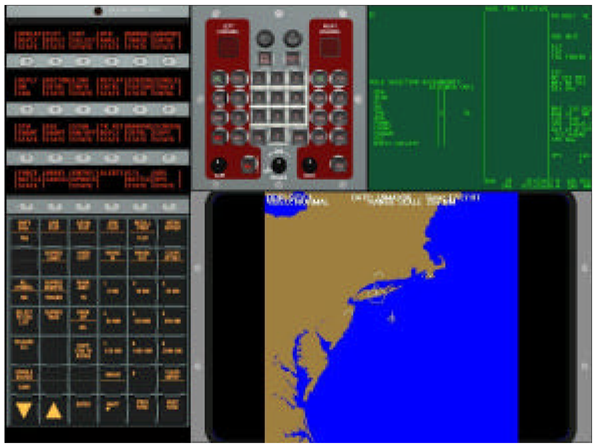


Figure 1. PORTS Simulated TAO Console

During an exercise, the TAO is responsible for making correct decisions with respect to verbal orders and queries. These decisions are evaluated for correctness, based on the current tactical situation and performance of other, automated, team members. The TAO's mastery of relevant tactical decision-making principles and ability to apply them in tactical situations is modeled along dozens of dimensions based on the entire history across several scenarios. This student model and the student's immediate performance are used by the ITS to automatically make real-time coaching decisions, assemble debriefings, choose the next scenario to give the student more practice on their weaknesses, and to make other instructional decisions. The simulated crewmembers also respond proactively to the tactical situation, generating the appropriate communications message traffic. For more insight into the performance assessment and modeling in PORTS TAO-ITS see Stottler and Panichas (2006).

Before the use of PORTS TAO ITS, an instructor was needed for every two students. The instructor played the role of other teammates and provided coaching and after action review. The logistics of this training setup provided limited training opportunities to the students. With the advent of the ITS, only 1 instructor is needed in a classroom of 42 students (Remolina et al., 2009). The ITS teaches material that has the least ambiguity and fewest controversies. Ambiguous / controversial situations are discussed with instructors and other students and rehearsed using a traditional fully manned simulation.

METHODS

The objective of the work described in this paper is to support the same training that is currently being carried out in PORTS ITS in the Second Life virtual world environment. That is, the TAO would log on as an avatar in SL and interact with simulated role players (SL objects) that perform the actions of other crewmembers and the instructor. In this section we present an overview of the integration process followed by the technical details of the integration. Note that throughout this section we use the word *avatar* to refer to the human student in SL and *object* to refer to the simulated role players in SL that are controlled by the PORTS ITS.

Integration Overview

An overview of the integration is shown in Figure 2, where the main components are PORTS, PORTS ITS, SL Communciation Layer (COMM), and SL. PORTS was not modified at all for this integration effort.

The second component, PORTS ITS, was modified in two minor ways. First, the Text to Speech manager that normally converted a text phrase to be spoken in the voice of a particular simulated role player was modified to instead notify COMM that a particular object in SL should speak the given phrase. Second, the speech recognition manager that usually converts the spoken words of the TAO to text was modified to instead take as input the chat text of the TAO from SL.

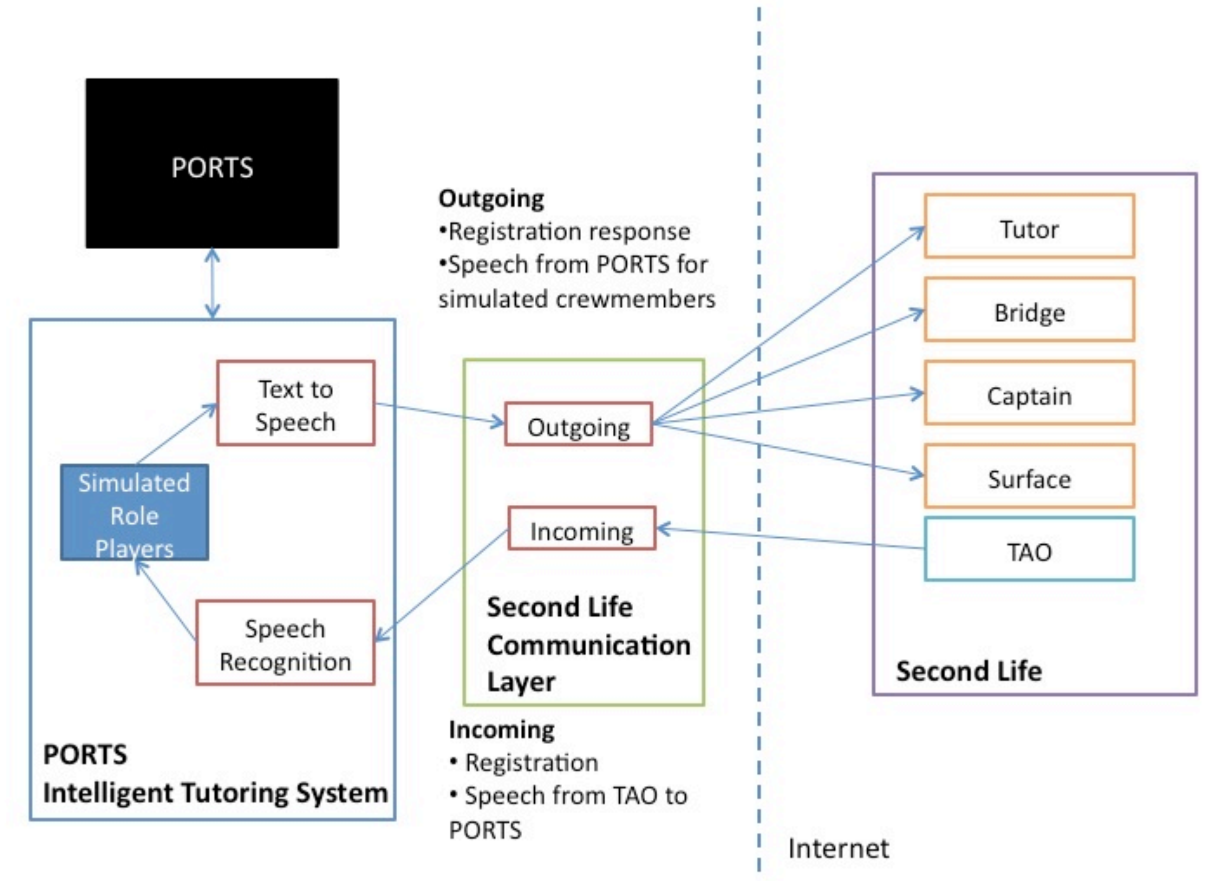


Figure 2. Component level integration overview among PORTS, PORTS ITS, the SL Communication Layer, and Second Life

Additionally, when running in Second Life mode, the PORTS ITS is also responsible for launching the COMM component. PORTS ITS and COMM communicate with each other through standard Java method calls. In similar integration efforts, other training systems would need to be similarly modified to support input from, and output to, SL avatars and objects.

The COMM component is a Java program that was developed specifically to link the PORTS ITS with the SL application program interface (API). As part of the incoming sub-component, COMM sets up a TCP/IP port to listen for messages that are broadcast from objects (e.g. Captain) in Second Life via HTTP. These messages include registering a SL object with COMM and in some cases, any chat messages that a nearby human enters to be broadcast, appropriately. If COMM receives a message from an object that indicates that the TAO avatar has spoken (by entering text in the local chat) COMM will pass this information along to PORTS. The outgoing sub-component of COMM sends messages to SL using the SL XML-RPC (remote procedure call) API. When the Text to Speech manager in PORTS ITS notifies COMM that a particular object (e.g. Tutor) should "speak" a phrase, COMM sends an XML message that identifies the object that should receive the message.

Finally, in Second Life, each object that represents a simulated role player has a script connected to it that listens for these XML-RPC messages. In this particular integration, the commands received include the text that the object should appear to say. The script parses the message and outputs the appropriate text to the local chat. These local chat messages look the same as messages typed in by other nearby humans. Additionally, information spoken by PORTS ITS simulated entities is also displayed over the object's head as shown in Figure 3.

Technical Details

In this section, low-level technical details are provided. These include select Java code excerpts for COMM, select script excerpts for Second Life that are used to transfer information from SL to PORTS ITS and vice versa, and how the particular integration demonstration is set up in SL. Readers not interested in the gritty implementation details should skip to the results.

COMM

In listening for message from Second Life, COMM uses standard java socket methods (*ServerSocket)* to listen for HTTP requests. A new *Socket* is created when the server socket accepts a message from SL. The socket reads in the message, after which it is checked for correct formatting. The required formatting includes standard HTTP information (such as that it begins with "GET" and ends with "HTTP/1.0") along with the COMM-specific string "/SLTranslate?" on the HTTP path. The information following this designation is parsed into one of two specific commands for the given name. The first is registering of a SL object that mirrors a PORTS ITS simulated entity (e.g. Captain) and has the same name, and the second indicates that the human TAO has entered chat text that should be relayed to PORTS ITS using standard Java method calls.

The COMM program makes use of Apache XML-RPC (http://ws.apache.org/xmlrpc/) to send messages back to SL. The XML-RPC client setup is quite simple:

```
XmlRpcClientConfigImpl config =
   new XmlRpcClientConfigImpl();
config.setServerURL(new
   URL("http://xmlrpc.secondlife.com/
   cgi-bin/xmlrpc.cgi"));
XmlRpcClient client = new
   XmlRpcClient();
client.setConfig(config);
```

Once the client is established, messages are sent to SL by placing several values into a hashmap: *Channel* - the unique object identifier acquired during registration, *IntValue* - an integer not used in our implementation, and *StringValue* - what the object should say). This map is then sent to the designated Second Life object by executing a remote procedure call as shown in line 7. For more information on how to use this functionality, see http://wiki.secondlife.com/wiki/Category:LSL_XML-RPC.

```
1. HashMap<String, Object> map = new
   HashMap<String, Object>();
2. map.put("Channel", objectId);
3. map.put("IntValue", intval);
4. map.put("StringValue", stringVal);
5. Vector<HashMap<String,Object>>
   params = new
   Vector<HashMap<String,Object>>();
6. params.add(map);
7. client.execute("llRemoteData",
   params);
```

SL Scripts

Object in Second Life can be configured to run user-defined scripts that control their behavior. The scripts are written in the Linden Scripting Language (LSL). For this integration, scripts were created that sent messages to COMM for registration and forwarding of what the human TAO typed in, and scripts that listened for remote data calls.

For example, the microphone object worn by the TAO listens on the local chat and forwards whatever the TAO enters on to the COMM object in the following LSL excerpt. The *state receiving* (1.) and *listen* (2.) keywords are built-in aspects of LSL, which are used to listen for a message on local chat. The message itself is sent as an HTTP request to the pre-defined ip and port numbers used by the COMM server (3.). The *llKey2Name(kOwner)* converts the unique id of the human speaker wearing the microphone to the human-readable name (e.g. TAO).

```
1. state receiving {…
2. listen(integer channel, string
   name, key id, string message) {…
3. llHTTPRequest("http://" + ip + ":"
   + port + "/SLTranslate?name=" +
   llKey2Name(kOwner) +
   "&action=taoSpoke" + "&text=" +
   message, [], ""); …}…}
```

The LSL script for the headset also includes code to display the text that would be heard by the TAO through the general communication network. This code is also included in the *state receiving* (1.) script. However, the *remote data* (2.) keyword indicates that the script is listening for *llRemoteData* rpc calls. Once the data is received, the object makes two calls into the built-in scripting language. The first is to say the string passed into the remote procedure call (*sval)* on the local chat stream (3.). The second displays the same text above the object in a color distinct from the other simulated entities. This script also includes some extensions not described here, such as handling the display of long text messages and ensuring that the "bubble text" above the object fades away at the appropriate time.

```
1. state receiving {…
2. remote_data(integer type, key
   channel, key message_id, string
   sender, integer ival, string
   sval){…
3. llSay(0,sval);
4. llSetText(sval, mycolor, 1.0); …}
   …}
```

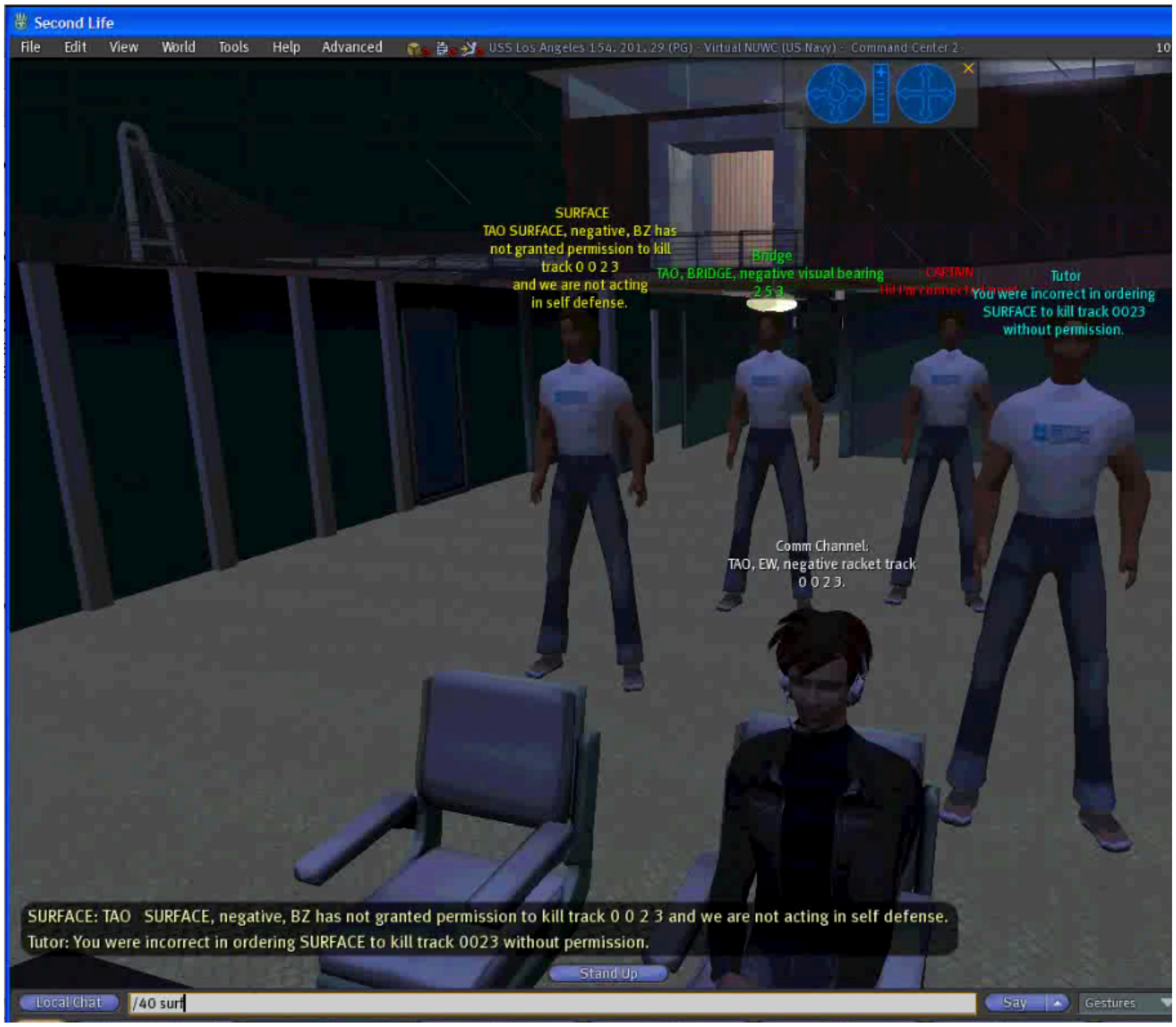
For more information on working with Second Life, see the official development website at: http://develop.secondlife.com/ in addition to the cited wiki address.

Setup

With these components and scripts in place, the actual setup is fairly simple. First, PORT ITS is run in an integration mode that also launches the COMM server. Second, the object instances are created and placed in Second Life based on pre-defined models. For this integration we used a headset model (worn by the TAO) and an avatar-like model (replicated for each of the simulated entities in PORTS ITS). Third, once the objects are defined, the associated script for each entity is updated. In this integration we had one script for the headset and one that was re-used for each of the avatar-like models. The re-used script was customized for each associated object to use a unique color when displaying text spoken by it. Fourth, the avatar of the human TAO needs to pick up and wear the headset. Finally, an activation message is sent by the TAO and the scenario begins.

RESULTS

The end result of this integration is that the TAO can log on as an avatar in Second Life (SL) and interact with a number of computer-controlled objects that take on the roles of the TAO's teammates, as shown in Figure 3. The seated figure is the avatar for the human TAO, who is typing text into the Second Life chat bar (bottom). Each of the standing figures, and the headset, represent a simulated role player in PORTS ITS (Surface, Bridge, Captain, Tutor, and Communication Channel). The TAO relies on the same communication mechanism to communicate with the simulated teammates in SL as they would to communicate with other human users. That is, the TAO speaks to these simulated teammates using the chat window built into SL and sees their replies and comments in the chat window as well. For clarity, the TAO also sees the text that the simulated entity speaks above the object's head in a "text bubble". In the end, this integration allows the TAO to carry out their communication decisions with the other simulated entities in Second Life using any of the existing PORTS ITS scenarios.


Second Life
File
Edit
View
World
Tools
Help
Advanced
USS Los Angeles 154, 201, 29 (PG) - Virtual NUWC (US Navy) - Command Center 2
Stand Up
Local Chat
Say
Gestures

SURFACE
TAO SURFACE, negative, BZ has not granted permission to kill track 0 0 2 3 and we are not acting in self defense.

Bridge
TAO, BRIDGE, negative visual bearing 2 5 3

CAPTAIN
Hi! I'm connected now!

Tutor
You were incorrect in ordering SURFACE to kill track 0023 without permission.

Comm Channel:
TAO, EW, negative racket track 0 0 2 3.

SURFACE: TAO SURFACE, negative, BZ has not granted permission to kill track 0 0 2 3 and we are not acting in self defense.
Tutor: You were incorrect in ordering SURFACE to kill track 0023 without permission.

/40 surf

Figure 3. Example of PORTS ITS and Second Life integration in action

CONCLUSION

This paper describes a novel extension of the deployed training system PORTS ITS, where it is integrated with the popular Second Life 3D virtual world. The paper also discusses the motivation behind this extension and the technical details that were used to carry it out. While performing this experiment, we learned a number of lessons and developed some specific directions for future work utilizing simulation-based intelligent tutoring systems and the Second Life virtual world.

Lessons Learned

- *Modularity* – The fact that the ITS system was already developed in a modular fashion made the integration process relatively straightforward. If the dialog management was implemented in a less modular fashion then significant changes would have been required.
- *Learning Curve* – We encountered two distinct learning curves in this integration. The first was from the developer perspective. Even though, as shown in this paper, the integration was relatively small in size, it could still take a significant amount of time given that most software engineers have no prior experience working with SL. Second, and more importantly, there is also a learning curve for users of the SL virtual world. That is, the human sitting down for TAO training will need to spend some time getting used to interacting with Second Life before they are ready to start training. The built-in SL tutorial for new users takes about 15 minutes to complete and would give them enough understanding to participate in an exercise. While using the training system the student would be expected to improve their familiarity with certain features (e.g. how to

show/hide the chat window) beyond this introduction. However, there is a lot of non-required functionality that could keep students busy for quite awhile (e.g. customizing avatars) if the students choose to spend their time on it.

- *Limitations* – A number of lessons were also learned about the limitations imposed by working with the existing SL API. For example, at the time of this writing, the TAO console itself (Figure 1) could not be displayed within SL. From a training perspective, this is an important piece of the puzzle that is missing. Another example is that an SL object cannot receive messages faster than once every 3-5 seconds. This had a negligible effect in this integration but could obviously cause problem for some ITS / simulators. A third limitation is the use of chat as the communication method. The integration would be greatly improved by being able to get voice input from SL and feed that into the PORTS ITS voice recognition and similarly to make use of text-to-speech to provide audio output for the simulated characters in SL. However, SL is a growing system so it updates APIs from time to time and some of these features are slated to be better supported in the future.
- *Changing API* – This paper represents a static snaphot of what can be done to integrate an existing training system with SL. The specific technical details behind the integration will need to be updated as the SL API evolves, but the general premise described in this paper is easily adapted to handle these changes. Design of similar integration efforts will want to take this into account as well.

Future Work

Based on the integration of PORTS ITS into SL, an example of a virtual submarine Fire Control Technician tutor was developed at the virtual Naval Undersea Warfare Center campus in SL. This example demonstrates the potential of an intelligent agent capable of augmenting traditional training in a virtual environment. An extension to this integration work is being considered by the U.S. Navy to create a platform that can rapidly elicit expert knowledge and develop intelligent agents based off this knowledge. These intelligent agents could be used as tutors in training scenarios or as automated bots in fleet team experimentations where they can model individual roles to reduce manning and cost.

While intelligent agents provide many benefits in the form of increased training availability and lower manpower cost, these agents are unable to perfectly emulate humans in teaching, learning, or behaving as a human teammate. Along with the development of the platform, work could also be extended to include an investigation to understand how best to integrate intelligent agents into submarine school curriculums in the virtual world.

REFERENCES

Remolina, E., Ramachandran S., Stottler R., Davis A. (2009) Rehearsing Naval Tactical Situations Using Simulated Teammates and an Automated Tutor,. *IEEE Transactions on Learning Technologies Vol 2 no 2 pp 148-156, Apr-June 2009.*

Stottler, R, A. Davis, S. Panichas, M. Treadwell (2007) Designing and Implementing Intelligent Tutoring Instruction for Tactical Action Officers. Proceedings of the Industry/Interservice, Training, Simulation & Education Conference (I/ITSEC 2007).

Stottler, R., S. Panichas (2006) A New Generation of Tactical Action Officer Intelligent Tutoring System (ITS). Proceedings of the Industry/Interservice, Training, Simulation & Education Conference (I/ITSEC 2006).