Easy Authoring of Intelligent Tutoring Systems for Synthetic Environments

Stephen B. Gilbert
Shrenik Devasani
Sateesh Kodavali
Virtual Reality Applications Center
Iowa State University
1620 Howe Hall, Ames, IA 50011.
515-294-3092
gilbert@iastate.edu, shrenik@iastate.edu, skodaval@iastate.edu

Stephen B. Blessing
University of Tampa
401 W. Kennedy Blvd.
Tampa, FL 33606
813-257-3461
sblessing@ut.edu

Keywords:
Intelligent tutoring system, Game engine, LVC training, physiology

ABSTRACT: *We describe how the Extensible Problem Specific Tutor (xPST), an open source engine for intelligent tutoring systems, has been adapted to provide training within game-engine based synthetic environments. We have designed a web-based tutor-authoring tool and have conducted a study that shows that xPST can be used by authors with minimal programming experience to create tutors for 3D game environments. As a proof of concept, we also describe how xPST has been extended to provide support to tutor based on real-time physiological data. We suggest that xPST could be a key component of the future of real-time personalized and adaptive live, virtual, and constructive training.*

1. Introduction

The military has used simulated environments and computer assisted instruction since the 1950s. Most recently, warfighters participate in live, virtual, and constructive training missions, which means the some fighters are in a field or urban practice site with BB guns or laser rifles ("live"), some are in simulators of Humvees or aircraft cockpits ("virtual") and some are playing serious games with virtual environments against computer-generated enemies ("constructive") (Gorman, 1991).

There has been much study of how much simulation fidelity is required for good training transfer (Andrews, Carroll, & Bell, 1995; Castner et al., 2007), and whether the simulation can induce a sense of presence, or immersion (Dede, 2009; Lessiter, Freeman, Koegh, & Davidoff, 2001; Stanney, 2002). Clark Aldrich continues to be enthusiastic about their potential for training (Aldrich, 2009). In some domains, simulation games may be the only possible means of simulating and practicing real world problems. Simulations are being used extensively in the military for teaching pilots to fly as well as for training on combat scenarios that would otherwise be extremely dangerous and expensive to train in the field (Stottler, 2000).

However, we suggest that the future of effective training lies not in the fidelity of the synthetic environment and virtual entities, but in the relevance of the feedback received by the learner. The ideal training environment (see Figure 1) will offer real-time adaptive training that offers personalized feedback and scenario customization based not only on trainees' behavior in the scenario but also on their skill sets upon entering the training and on their personal profiles, e.g., information about their personalities and their physiological responsiveness to stress, both of which affect performance (Beilock, 2010). Adapting training based on both performance and the trainee's stress response is critical to accurate personalized feedback.

1.1 Personalized Adaptive Training

The idea of personalized adaptive training embodies two concepts from the learning sciences. The first is adaptive testing or tailored testing, used, for example, by the Educational Testing Service on standardized tests such as the GRE to offer students harder questions when they answer correctly and easier questions when they choose incorrectly (Thissen & Mislevy, 2000).

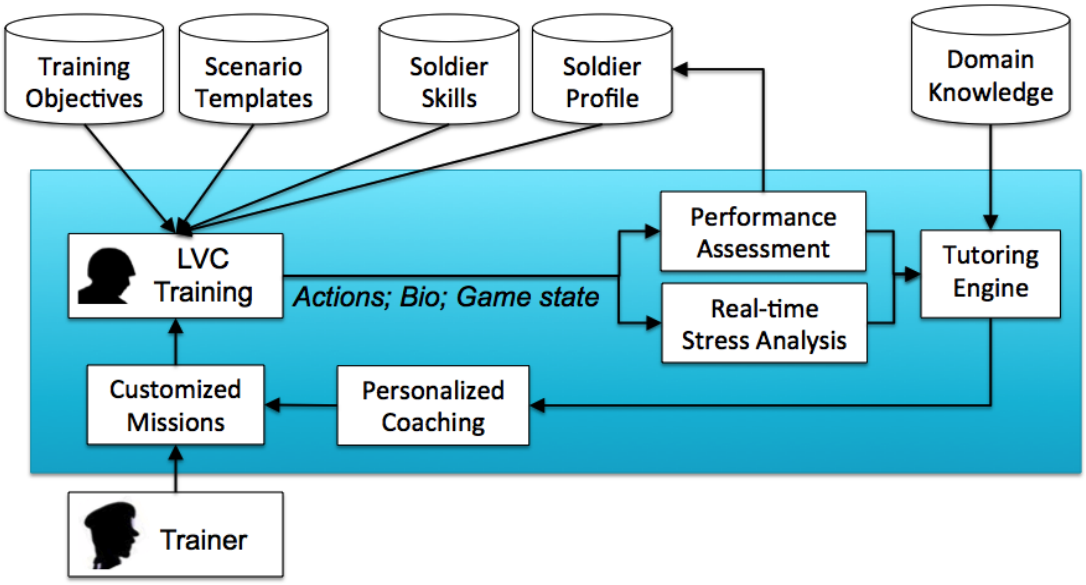

Figure 1: The vision for the future of personalized adaptive training. The live, virtual and constructive training experience is determined by the training objectives, the soldier's previous skills, and the soldier's personality and stress resilience profile. In real-time, soldier information is updated based on performance, and the training experience is updated and personalized for the soldier.

The key principle is that the system adapts itself based on the learner's performance.

A system that not only tracks a learner's performance but also attempts to offer the learner useful feedback based on his or her mistakes is called an intelligent tutoring system (ITS). This is the second characteristic of the personalized adaptive training approach. ITSs have a cognitive model of an expert's domain knowledge that is used to identify patterns of learner behavior (model tracing) and give appropriate hints or corrective feedback. A cognitive model of algebra, for example, would know that students frequently forget the negative sign when solving equations, and would be able to say appropriately, "You might check to see if you've forgotten a negative sign somewhere…" Similarly, the Nintendo Wii Fit game system could be considered a simplistic ITS, since it tracks your skills and offers feedback such as "You're leaning too far to the left."

ITSs have been demonstrated to have effective in a variety of school knowledge domains, such as algebra, geometry, and economics (Anderson, Conrad, & Corbett, 1989; Koedinger, Anderson, Hadley, & Mark, 1997; Ritter, Kulikowich, Lei, McGuire, & Morgan, 2007; VanLehn et al., 2005), resulting in up to a 30% improvement in standardized test scores (Franklin & Graesser, 1996) and learning time reductions (Corbett, 2001).

1.2 Challenge: Easily Creating an ITS for a Synthetic Environment

ITSs have also been created and customized for a variety of military synthetic environments (SEs) (W. R. Murray, 2006; Remolina, Ramachandran, Stottler, & Howse, 2004; Stottler, 2000; Stottler, Fu, Ramachandran, & Jackson, 2001) and Livak, et al created a more generalized tutoring approach using Unreal Tournament (2004). But what is still missing is 1) a more **generic ITS authoring tool** that could easily create ITSs for multiple SEs using modular abstraction from the SEs themselves, 2) an **ITS protocol that leverages physiological data**, and 3) an **easy-to-use authoring tool** for ITSs that could be used by military trainers with no programming experience. We suggest that xPST, an open source intelligent tutoring system engine, has the potential to address these gaps (http://code.google.com/p/xpst/).

2. Extensible Problem-Specific Tutor API

The Extensible Problem-Specific Tutor (xPST) application programming interface (API) has been developed to enable teachers or trainers without a technical background to build ITSs on existing third-party interfaces such as websites or software clients, or networked applications such as game-engine based synthetic environments. Research studies on xPST have demonstrated that it is a relatively easy tool to use for non-programmers with some training (Gilbert, Blessing, & Kodavali, 2009).

A variety of ITS authoring tools have been developed (T. Murray, Blessing, & Ainsworth, 2003), and there is always a tradeoff between the expressive power of the

tool and the ease of use. The choice with xPST is to limit the power in favor of usability, which means that the trainer creates a new tutor for each specific problem or training scenario. The Carnegie Mellon Cognitive Tutor Authoring Tools (CTAT) suite for authoring ITSs shares some of the features and goals of xPST while also focusing more providing a graphical authoring interface. Both CTAT and xPST can create example-based tutors (Aleven, McLaren, Sewall, & Koedinger, 2009).

The xPST architecture is shown in Figure 2. Using drivers that are developed for each third-party software client, xPST can eavesdrop on learner events and state variables (Listener) and give feedback messages within the client (Presenter). These drivers are usually simple to develop if the third-party client has a scripting API that will allow event logging and messaging functions. xPST uses a cognitive modeling language that can provide tutoring on an arbitrary number of interfaces. This language has been kept as simple as possible, to promote authoring by non-programmers. It requires the author to list the sequence of steps to be performed by the learner, linked with the operators "then," "and", and "or," e.g., FindTarget then (WalkToTarget or JumpToTarget) then CaptureTarget. For each step, the author then adds hints and just-in-time error messages associated with each step.

This simpler approach to modeling feedback means that xPST does not represent a true "cognitive model" with the cognitive complexity and fidelity of ACT-R, for example. However, the expressive power of the xPST modeling language, while not yet a context-free grammar, is strong, e.g. able to use variables within feedback strings and condition feedback on learner input. We suggest that this approach supports tutoring behavior similar to that of the model-tracing ACT Cognitive Tutors.

It is also worth noting that currently, xPST does not maintain a learner knowledge model (tracking the skills across multiple tasks), but that that feature will be added shortly.

The xPST communications occur over TCP/IP sockets, so that the tutoring engine can reside on a separate machine from the client. So far, in using xPST with over 60 learners, no known network lags have been noted. For more detailed information about xPST, see (Blessing, Gilbert, Blankenship, & Sanghvi, 2009; Gilbert et al., 2009).

The remainder of this paper describes in further detail how xPST addresses the gaps described above: using xPST to tutor on a synthetic environment, working with physiological input, and using a web-based authoring tool to create tutors.

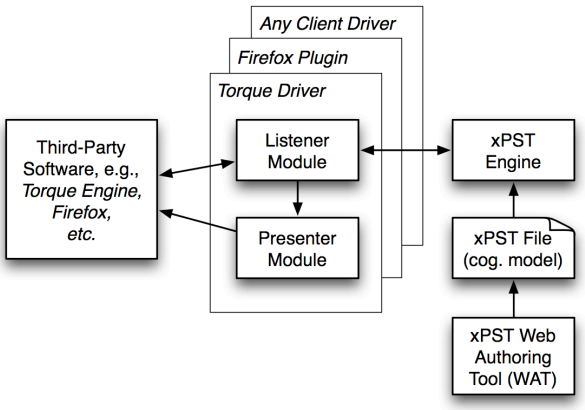

Figure 2: xPST architecture

3. Developing ITSs in Synthetic Environments

Most of the ITS authoring tools described in (T. Murray et al., 2003) focus on creating ITSs for relatively slow-moving interfaces, e.g., for a spreadsheet or for a web-form-like interfaces with radio buttons and text boxes. Few, if any, ITS authoring tools exist for developing tutors for game engines like VBS2, Second Life, or other synthetic environments. It is worth noting the conceptual differences between tutoring a typical graphic learner interface (GUI) vs. in a multiplayer fast-moving game.

3.1. Synthetic Environment vs. Traditional GUI

In a traditional GUI application or website, there are usually a set of controls (e.g. buttons, menus) that correspond one-to-one with a set of features. Some of these controls typically remain on screen while the learner works. The system typically features a two-part architecture consisting of an application (e.g., Microsoft Word, Adobe Photoshop, Amazon.com, or Google Docs) that has a particular state (e.g. "current color choice is red") while learner-created content (e.g., a document, an image, or a query) is shaped by the learner.

In a GUI, a learner's actions will typically evoke similar responses if done repetitively. In a game or SE, on the other hand, a learner's actions are frequently dependent on the context of other entities and the time course within the SE. Rather than the learner changing a file or a query within an application that maintains a state, the gamer is focused on changing the application's state within the game state space. The states can be discretely defined, and they then act as subgoals within the tutor: the learner's goal is not to complete a textbox with a certain correct answer but rather to reach a specific state. The granularity at which

these states need to be defined depends on the author and the complexity of the task. The traditional GUI software and websites are typically a static system where all the events are triggered by the learner (player). But SEs are more like a dynamic system where interactions can happen between various entities in the game apart from the player and the events can be triggered by different entities in the game (e.g. by constructive forces). It is thus useful to categorize the events as player events and non-player events.

Unlike the traditional GUI software or websites, 3D games require the student to navigate through a simulated environment, the map, and sometimes communicate with the other entities in the game. This calls for the authoring system to provide tools to support tutoring on communication-based and location-based subtasks, e.g., "Identify yourself to the guard" or "Return to mission headquarters to give a report."

Synthetic environments and traditional GUI software also differ in the kinds of feedback required for tutoring. In GUI-based tutors, many researchers have concluded that two broad types of feedback are sufficient: Hints (information requested by the learner if needed) and Just-In-Time error messages, or JITs, that appear when the learner makes mistakes. In SEs, because the subgoals within a task can represent a much broader range of activities than simply typing an answer in a box or choosing a drop down menu option, and because it is sometimes not visually apparent whether the subgoal has been completed ("Did I reach the location?"), it is sometimes important for the learner to have feedback from the tutor that is neither requested, like a Hint, or based on an incorrect event (like a JIT). We call these "prompts." For example, "You are now in a high risk zone, potentially surrounded by mines."

The xPST platform has been recently extended to accommodate these requirements for SEs in the tutor authoring system. More details can be found in (Kodavali, Gilbert, & Blessing, 2010).

3.2. Simulation Engine: Torque

To create an example of tutoring on a synthetic environment, we have used Torque Game Engine Advanced (TGEA) as our simulation engine. It is a commercial off-the-shelf game engine from Garage Games. It provides various core functionalities required for game development like the rendering engine, physics engine, 3D graphs, collision detection etc. Instead of starting from scratch, using an off-the-shelf game engine drastically reduces the game development time and helps the author concentrate more on the tutoring task.

TGEA supports scripting using TorqueScript. TorqueScript is similar in syntax to JavaScript and allows the developer to create modifications (mods) of existing games. We have used TorqueScript to create the xPST Torque driver which contains the two major modules shown in Figure 2 above, the Listener module and the Presenter module. The Listener module listens to the various events happening in the game and sends them to the xPST engine over the network. The xPST engine then sends the appropriate tutoring feedback to the Listener module. This feedback is then presented to the learner through the Presenter module.

The framework of the xPST driver can be leveraged to various other game engines by making syntactical changes to the script, as required, in order to be compatible with a particular game engine. Our research team has done significant work with scripting the VBS2 game engine, and plans to create a VBS2 driver for xPST next.

3.3 Torque Example: Evacuate the Buildings

We developed several demonstration tasks called *DemoTutor* and *Evacuate* to show that xPST can be used to create ITSs in a game environment. *Evacuate* teaches the learner (player) how to evacuate the civilians from all the buildings in the scenario. The scenario has three buildings, each with one civilian inside. The player enters each building, checks for civilians present in it, issues the Evacuate command, and waits for the civilian to come out. When the learner does this for all the buildings in the scenario, then the task is complete. This is a simple task, but illustrates the ease of creating subgoals and feedback for them within Torque. Figure 3 shows a scene in the Torque *DemoTutor* scenario with a hint of what to do next. See below for a study evaluating non-programmers' abilities to develop tutors using Torque.

4. Tutoring with Physiological Data

xPST has recently been extended to provide support for tutoring based on physiological signals. The xPST author can offer customized just-in-time feedback based on the values of physiological signals related to stress and arousal. We used a physiological monitoring device called the FlexComp that returns the electrocardiogram signal (EKG), the heart rate signal (the heart rate in beats per minute) and the heart rate variability signal (measures how the heart rate varies over time). When combined with blood pressure, these data can be useful for identifying trainee stress, e.g., when approaching of the end of the completion time of a time-based test. When a trainee is attempting to complete a scenario, tasks that cause high stress could be identified, and the trainee could be asked to practice

Figure 3: A screen from a Torque scenario showing a user-requested hint from the xPST tutor.

such tasks further. Also, trainees who are noted as high-responders to stress might be assigned to less stressful duties longer term.

While this extension of xPST for physiology is still in its early stages, we have demonstrated the use of heart rate while tutoring on a timed web-based college statistics assignment, giving different feedback depending on whether time is running out, and depending on whether stress levels were higher when learners made mistakes (Figure 5). This extension work will continue for an ongoing project at Iowa State led by Nir Keren and Warren Franke analyzing the effect of stress on firefighters during immersive CAVE (Cave Automatic Virtual Environment)-based training.

5. The Web-based Authoring Tool (WAT)

A web-based authoring tool is essential for any ITS authoring tool that is meant to be used by non-programmers, since the management and deployment of tutors can be a laborious task. With game-based scenarios, this task can get even more complex due to the wide range of resources, both hardware and software are required. An efficient web-based authoring tool can take care of these issues by managing all the resources and dependent files at a single location, thereby allowing the learner to concentrate on the task of building tutors.

To allow easy creation of ITSs and their deployment on the web, we have created a Web-based Authoring Tool (WAT). The WAT (see Figure 6) supports both learner management and tutor management on a single platform.

Learners can create their own accounts to use the Web-based Authoring Tool and, once registered, develop tutors there (http://xpst.vrac.iastate.edu/home.html). Authors may edit tutors using the built-in xPST editor, which checks syntax. The Authoring Tool supports version control, so that previous versions of the tutor files are not lost. The tutor files, including the xPST file and any scenario or external files associated with each tutor can be downloaded, so that the learner can deploy the tutor on his local machine for testing purposes if needed. This authoring tool has been used successfully by five people with no programming experience to develop tutors for webpages with college statistics homework problems (Maass & Blessing, Submitted).

The Authoring Tool also supports logging of events for the purposes of data mining and research studies such as described below. It records tutor creation time, tutor opened time, tutor saved time (both manual save and auto-save which happens every 20 seconds), and tutor edit time. The details of these events allow researchers to identify how long it takes to create tutors for a particular problem domain.

The following process would be used to create a tutor with a new game engine or synthetic environment (SE) that xPST did not work with yet. Note that Steps 1 and 2 require programming skills, but must be done only once. Thereafter, non-programming trainers can create tutors on their own for the SE (Steps 3 and 4).

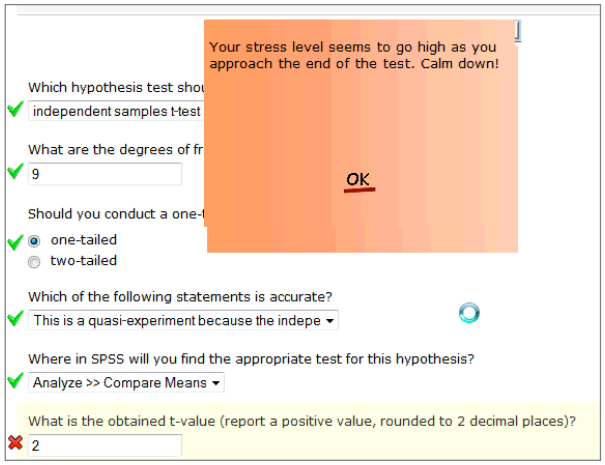

Figure 5: Screenshot of xPST prototype giving feedback based on heart rate during a statistics problem. "Stress" in this prototype is used loosely.

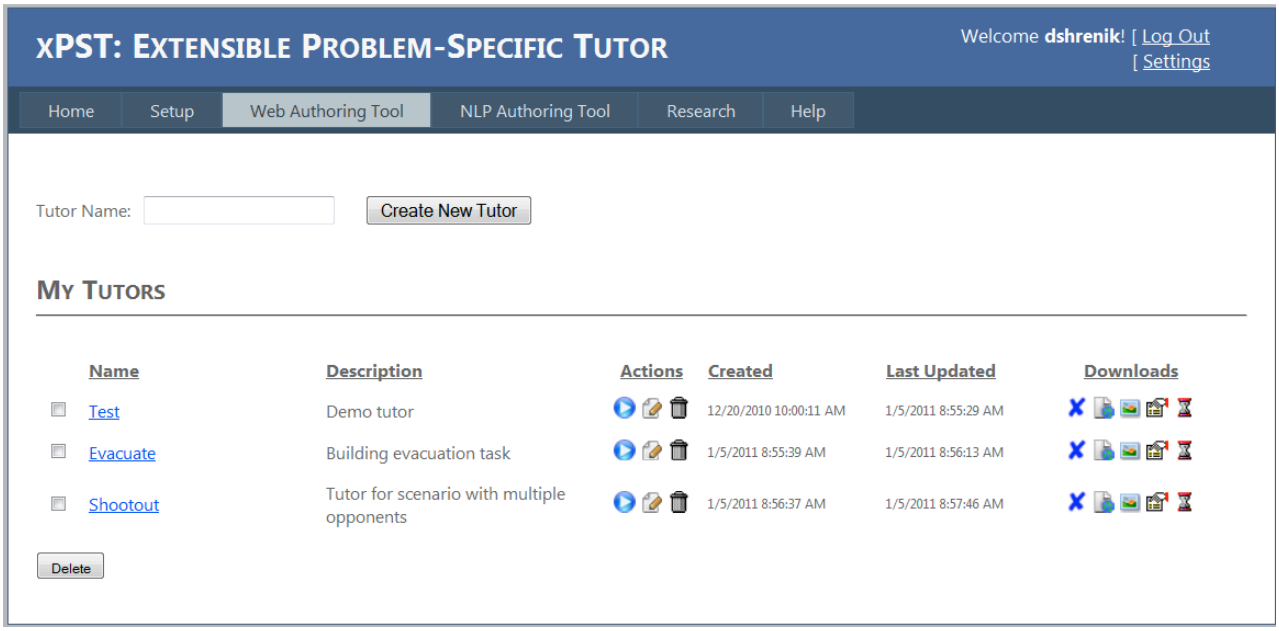

Figure 6: A screenshot of the Web-based Authoring Tool (WAT) that can be used by anyone to create an xPST tutor.

1. Write xPST driver for a new SE
2. Create list of SE events likely to be used within the tutor for the xpst authors
3. Create the scenario within the SE
4. Design the tutor using xPST.
    a. Login to WAT
    b. Draft xPST file using built-in editor
    c. WAT creates the necessary files, links and deploys them for testing
    d. Test the tutor by clicking "Run"

6. Research Study on Using xPST to Build Tutors for Torque

We conducted a study to ensure that xPST can be used by tutor authors with minimal programming experience to create tutors for 3D game environments with little training. We also wanted to examine the learning curve of novice xPST users.

We had 10 participants who were selected based on a pre-survey that included them only if they had a minimal amount of programming experience, e.g. editing HTML or using SPSS, a statistics application with an interpreted language. They were asked to build tutors for two tasks within Torque using xPST. The participants were shown a 15-minute video tutorial that demonstrated how to create a tutor for a demo task, and were given access to other online resources that included five documentation-style wiki pages, a commented sample xPST file, and an optional 45-minute video on detailed use of xPST. In sum, we estimate the typical training time to be 1-2 hours. Participants completed the models at their own pace over a 2-week time period.

The first task, Task A, was titled "Target Acquisition" and was meant to teach soldiers how to locate an enemy target. The "Target Acquisition" task required 3 subgoals to be described by the participant in the xPST file using a highly simplified coding language, e.g., the sequence of subgoals might be described as "Step1 then Step2 then (Step3a or Step3b)." Task B was titled "Evacuate" and was aimed at teaching soldiers how to evacuate cottages in a threatened village. The "Evacuate" task required 7 subgoals to be coded by the participant.

One of us scored each of the models using an evaluation rubric to quantitatively evaluate the data. A total of 20 models were created by the 10 participants. The models were classified into one of five categories. Table 1 shows how they were scored.

Score	Description	No. of models
5	A very good model that is beyond being just sufficient	8
4	A sufficient model where a trainee can complete the task	8
3	Model provides hints but does not provide enough guidance to a novice	2
2	Model runs but provides nonsensical help	0
1	Model has the required feedback but does not run due to syntax errors	2

Table 1: Scoring of the cognitive models

The average time to complete Task A was 19.74 minutes with a standard deviation of 9.16 minutes, and the average time to complete Task B was 13.81 minutes with a standard deviation of 6.24 minutes (The completion-time data refers to only the time spent in writing the xPST file, and not testing the tutor).

Figure 7 shows a bar graph of the completion-time data for Task A and Task B for the eight participants who built models without any syntax errors.

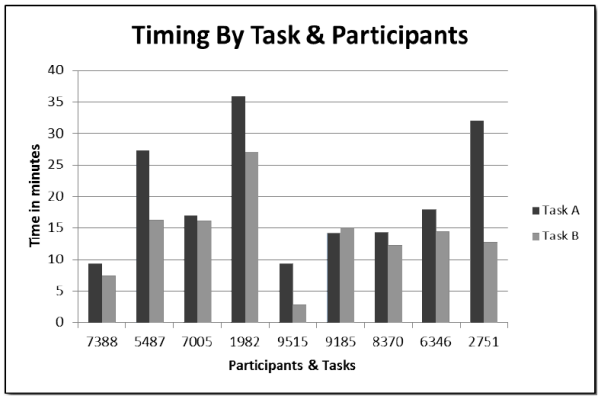

Figure 7: Bar graph of the completion-time data

All participants first completed Task A and then moved on to Task B. It is worth noting that that Task B required less time to complete, on average, when compared to Task A, though Task B was more complicated and required a higher number of subgoals. The bar graph shows that all participants except one required less time to complete Task B, when compared to Task A. The results demonstrate that users with minimal programming experience with less than 2 hours of training can use xPST to create tutors for 3D game environments.

7. Conclusion and Future Work

We have described a vision for a personalized adaptive training using synthetic environments. A key component of this approach is an intelligent tutoring engine that can communicate with game engines, physiological systems, and be easy to use for authors. xPST is a good candidate for such an engine. We described the xPST Web-based Authoring Tool that enables easy development of tutors for 3D games based on the game engine Torque, and potentially for other game engines such as VBS2, Unity, and BigWorld. Our research study shows that non-programmers can use xPST with minimal training.

xPST's ability to do physiological tutoring on web interfaces can be extended to 3D games as well, where stress is a major factor that affects a player's performance. The goal will then be to provide military trainers, as well as others who author scenarios in 3D environments, the ability to create in-scenario tutoring and personalized After Action Review in an appropriate and easy-to-author manner.

Future work for xPST includes completing a natural language processing extension that's underway so that we can tutor on typed or spoken words, and implementation of a networked learner model, so that the system can track trainee skills across multiple platforms and multiple training scenarios. We also plan on a learning management system (LMS) that will offer scenarios according to trainees' performance and personalized needs and skill deficits.

7. References

Aldrich, C. (2009). *The Complete Guide to Simulations and Serious Games*. San Franscisco: Pfeiffer.

Aleven, V., McLaren, B., Sewall, J., & Koedinger, K. (2009). A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education, 19*, 105-154.

Anderson, J. R., Conrad, F. G., & Corbett, A. T. (1989). Skill acquisition and the LISP tutor. *Cognitive Science, 13*, 467–505.

Andrews, D. H., Carroll, L. A., & Bell, H. H. (1995). The Future of Selective Fidelity in Training Devices. *Educational Technology, 35*(6), 32-36.

Beilock, S. (2010). *Choke: What the Secrets of the Brain Reveal about Getting It Right When You Have To*. NY, NY: Free Press.

Blessing, S., Gilbert, S., Blankenship, L., & Sanghvi, B. (2009). *From SDK to XPST: A New Way To Overlay a Tutor on Existing Software.* Paper presented at the Twenty-Second International FLAIRS Conference.

Castner, A. K., Chukhman, I. A., Colbert, E. J., Dale, M. E., Lewis, B. Y., & Zaret, D. R. (2007). *An agent-supported simulation framework for metric-aware dynamic fidelity modeling*. Paper presented at the Proceedings of the 2007 spring simulation multiconference - Volume 2.

Corbett, A. T. (2001). *Cognitive computer tutors: Solving the two-sigma problem.* Paper presented at the Conference of User Modeling, Sonthofen, Germany.

Dede, C. (2009). Immersive Interfaces for Engagement and Learning. *Science, 323*(5910), 66-69.

Franklin, S., & Graesser, A. (1996). *Is it an agent, or just a program? A taxonomy for au-tonomous agents* Paper presented at the Third International Workshop on Agent Theories, Architectures, and Languages.

Gilbert, S. B., Blessing, S. B., & Kodavali, S. (2009). *The Extensible Problem-Specific Tutor*

(xPST): Evaluation of an API for Tutoring on Existing Interfaces. Paper presented at the 14th International Conference on Artificial Intelligence in Education.

Gorman, P. F. (1991). The Future of Tactical Engagement Simulation. In D. Pace (Ed.), *Proceedings of the 1991 Summer Computer Simulation Conference* (pp. 1181-1186). Baltimore, MD: Society for Computer Simulation.

Kodavali, S., Gilbert, S. B., & Blessing, S. (2010). Expansion of the xPST framework to enable non-programmers to create intelligent tutoring systems in 3D game environments. *In the Proceedings of the 10th International Conference on Intelligent Tutoring Systems*

Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent Tutoring Goes to School in the Big City. 30-43.

Lessiter, J., Freeman, J., Koegh, E., & Davidoff, J. (2001). A Cross-Media Presence Questionnaire: The ITC-Sense of Presence Inventory. *Presence: Teleoperators and Virtual Environments, 10*(3), 282-297.

Livak, T., Heffernan, N. T., & Mover, D. (2004). *Using Cognitive Models for Computer Generated Forces and Human Tutoring.* Paper presented at the 13th Annual Conference on (BRIMS) Behavior Representation in Modeling and SImulation, Arlington, VA.

Maass, J., & Blessing, S. (Submitted). *XSTAT: An intelligent homework helper for students.* Paper presented at the 2011 Convention of the Southeast Psychological Association, Jacksonville, FL.

Murray, T., Blessing, S., & Ainsworth, S. (Eds.). (2003). *Authoring Tools for Advanced Technology Educational Software*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Murray, W. R. (2006). *Intelligent Tutoring Systems for Commercial Games: The Virtual Combat Training Center Tutor and Simulation.* Paper presented at the The Second Artificial Intelligence for Interactive Digital Entertainment Conference (AIIDE), Marina del Rey, California.

Remolina, E., Ramachandran, S., Stottler, R. H., & Howse, W. R. (2004). *Intelligent Simulation-Based Tutor for Flight Training.* Paper presented at the Industry/Interservice, Training, Simulation & Education Conference (I/ITSEC), Orlando, FL.

Ritter, S., Kulikowich, J., Lei, P., McGuire, C. L., & Morgan, P. (2007). What Evidence Matters? A Randomized Field Trial of Cognitive Tutor Algebra I*,* pp. 13-20.

Stanney, K. M. (2002). *Handbook of Virtual Environments*. Mahwah, NJ: Erlbaum.

Stottler, R. H. (2000). *Tactical Action Officer Intelligent Tutoring System.* Paper presented at the Industry/Interservice, Training, Simulation & Education Conference (I/ITSEC), Orlando, FL.

Stottler, R. H., Fu, D., Ramachandran, S., & Jackson, T. (2001). *Applying a Generic Intelligent Tutoring System Authoring Tool to Specific Military Domains.* Paper presented at the Industry/Interservice, Training, Simulation & Education Conference (I/ITSEC), Orlando, FL.

Thissen, D., & Mislevy, R. J. (2000). Testing algorithms. In H. Wainer (Ed.), *Computerized adaptive testing: A primer*. Mahway, NJ: Lawrence Erlbaum Associates.

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., et al. (2005). The Andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education, 15*(3), 147-204.

Author Biographies

STEPHEN GILBERT is Associate Director of the Virtual Reality Applications Center at Iowa State University. His background includes cognitive science and engineering, and he supervises Iowa State's graduate program in Human Computer Interaction. He is PI on a Live, Virtual and Constructive training contract for the U.S. Army RDECOM STTC.

SHRENIK DEVASANI is a Graduate Research Assistant at Iowa State University's Virtual Reality Applications Center. He is pursuing his Masters in Human Computer Interaction. His research interests are in the fields of intelligent tutoring systems and technology in education.

SATEESH KODAVALI, recently graduated with his MS in Human Computer Interaction and Computer Science and is now a Software Development Engineer (SDE) at Microsoft in the Microsoft Business Division (MBD). His work focuses on designing, implementing and/or testing Microsoft Dynamics ERP software applications.

STEPHEN BLESSING is an Associate Professor of Psychology at the University of Tampa. His main research interest is in authoring tools for intelligent tutoring systems, and he has co-edited a book on the topic.