2013 International Conference on Signal-Image Technology & Internet-Based Systems

MAS Controlled NPCs in 3D Virtual Learning Environment

Grant McClure, Maiga Chang, Fuhua Lin
School of Computing and Information Systems
Athabasca University
Edmonton, Canada
maiga@ms2.hinet.net, oscarl@athabascau.ca

Abstract—**Incorporating intelligence and social behaviours into virtual worlds for learning is becoming more desirable in making them smart, adaptive, personalized, and therefore, more effective and engaging. Realistic non-player controlled characters (NPCs) are essential of a game world and are making the virtual world more real for players. This is true in video games where more interactive NPCs support the story narrative of a game, making the game more immersive, more convincing, but it is also true in other areas where virtual worlds are used such as business and education, increasing the effectiveness of those environments. Research that has been done with virtual agents and multi-agent systems can be leveraged to create more realistic NPCs through purposeful communication channels, inter-agent interactions and environment-agent interactions for game-based learning applications. This research proposes an approach to controlling avatars with intelligent agents through the creation of an interface between a multi-agent system to a virtual world engine. Basic NPC behaviours controlled by agents using Jason AgentSpeak are used to test the feasibility of the approach. A QuizMASter is designed and illustrated as a proof of concept of the use of such agent controlled avatars in educational context.**

***Keywords-** virtual learning environments; multi-agent systems, virtual world, non-player characters, social behavior, intelligent agent, social bots.*

I. INTRODUCTION

Research supports the effectiveness of game-based learning in 3D virtual environments that could make us quickly see and understand the connection between the learning experience and our real-life work. Within a virtual world, students work toward a goal, choosing actions and experiencing the consequences of those actions along the way. This keeps students get engaged and be motivated.

There have been relatively mature technologies for modeling virtual humans in virtual worlds in their appearance, gestures, kinematics, and physical properties. The challenges faced in virtual worlds research and development are to emulate or simulate the way human beings act in their environment, interact with one another, cooperatively solve problems or act on behalf of others, solve more and higher-level, more complex problems by distributing tasks or enhance their problem solving performances by competition. Finding ways to solve the challenges is even more important for researchers and developers of game worlds. If a Non-Player Character (NPC) does not act in a believable way it detracts from the realism of the player's game experience. In the gaming industry, Artificial Intelligence (AI) techniques have been used to drive NPC behaviour [2]. However, this is usually done in a prescriptive manner that limits the freedom of an NPC. This leads to predictable NPC behaviours, which decreases their credibility.

Multi-agent systems (MAS) have been adopted by researchers in virtual worlds community. A multi-agent system is well suited to application domains where virtual entities, called agents, are self-directed and can actively pursue their goals within an environment that they can interact with, including interactions with other agents that are also in pursuit of their own goals. Agents are ideally suited for modeling real people – they are active and social, similar to the way people are. Agents can be used for modelling the NPCs, keeping track of individual interests, motivations, and goals in game worlds. Although researchers have developed vary human like NPCs in the existing virtual worlds and game engines, such as Open Wonderland [11], Second Life [17], Open Simulator [15], Unreal [19] and gamebot [10], no gaming engine currently has multi-agent system framework built in to manage and to control NPCs.

The main idea of this research is using an autonomous agent to control an NPC just like a human user controlling his or her avatar in the virtual world. That is, behind an NPC is an agent. Or we can say an NPC is an agent's avatar. All avatars, including player's avatars and agent's NPCs, live in the virtual worlds, while the agents live in the multi-agent system. The integration of multi-agent systems and virtual worlds provides a possible solution to the challenge and opens a number of extremely interesting and potentially useful research avenues concerning game-based learning through inter-agent negotiation, persuasion, and competition.

To realize the benefits of using a multi-agent system framework such as JADE and Jason in a gaming engine, there are two approaches: 1) developing a completely new gaming engine in which a multi-agent system framework could be natively built-in, or 2) designing an integrative framework which could connect a multi-agent system and an existing gaming engine, so that the gaming engine could benefit from NPC management handled by the multi-agent

978-1-4799-3211-5/13 $31.00 © 2013 IEEE
DOI 10.1109/SITIS.2013.166

system. The first approach does not allow for the direct leveraging of existing open source technologies and the amount of work would be prohibitive when it comes to a research project. The second approach is more viable, but offers challenges when it comes to state synchronization across what are effectively two separate systems.

This paper describes the design and development of a platform which is capable of hosting agent-controlled avatars through constructing an interface between a multi-agent system and a 3D virtual world.

II. Related Work

Research has been done over the last decade when it comes to creating agent control mechanisms for 3D virtual environments. The first project of note is Gamebots [10], which was a client-side approach to controlling Unreal Tournament [19], an engine used for first-person shooter games. Gamebots was developed by the University of Southern California's Information Sciences Institute, and jointly developed by Carnegie Mellon University; their goal was to foster a domain for AI research in multi-agent systems. They picked the Unreal engine because it supported multiple simultaneous tasks, had a scripting language that could be used to extend scenarios, supported multiple dynamic environments, and provided an environment where player character and non-player characters could interact [10].

All of these things together meant that the environment could be used in different scenarios, giving a great deal of freedom to AI researchers. Unreal is a commercial engine, but its licensing is open enough that it allowed for it to be used in this manner. Since Unreal is a commercial engine, the source code for it is not available, and the creation of scripts using Unreal's scripting language and configuration of virtual worlds are the only things under a programmer's control. The fact that it is a commercial engine was not entirely a negative though as the engine was and is a popular one; a great deal of money had been invested in it to ensure its quality, and it had been used in the field, "stress tested by thousands of people everyday" [10].

Gamebots has also been used as a platform for training NPCs to be more realistic by interacting with real players and building up a model of user behavior. Dinerstein and Egbert (2005) developed a custom multi-agent system with beliefs, desires, and intentions underpinnings, using a layered cognitive model [3]. They noted that their system was fast but also that a notable amount of CPU was required when using knowledge to make decisions on NPC actions. This is always a consideration when working with a multi-agent system in a real-time scenario. Agents must be able to work through their evaluation routines quickly to come up with a plan of action. Since the server was not open-source and they could not get the level of detail they needed in terms of how the server was responding to player interactions with the environment, they had to use a simple environment for running their simulations.

Pogamut [7] is another agent implementation, written in Java, which works with Gamebots. It is a toolkit that included things as emotional modeling, adding support for avatar gestures, and a framework geared towards further AI research as well as working with educational scenarios. Gemrot and colleagues (2009) spent some time discussing the merits of different gaming engines; they went with the Unreal engine because of the strength of the existing community, among other factors. Controlling more than ten agents at the same time proved difficult; they do not say what hardware they were using, but the hardware can be expected to be reasonably current. They noted that multi-agent system frameworks like JADE are not used as NPC controllers in games, and speculated it was because a lot of work would need to be redone that already exists in Gamebots. They also thought there might be a downside in communication rates when using a multi-agent system due to increased overhead in messaging with FIPA [6]. Gamebots and Unreal Tournament have a messaging system that is built for fast communications. It could be argued that it is possible to create a messaging system using a multi-agent system that is equally succinct.

Virtual Singapura is a fantasy world, takes the learners to nineteen-century Singapore in the throes of disease epidemic [14]. The project is one of the first to integrate an intelligent agent architecture with a virtual world to explore ways in which adaptive synthetic characters might enhance the learner's experience in a virtual world and perhaps to enhance learning as well.

Lim and colleagues (2012) integrated a FAtiMA [5] architecture and a drives-based PSI model in an attempt to create more believable NPCs [13]. FAtiMA is based on a belief, desire, and intentions (BDI) model with an emotional element added to it. PSI is a psychological model that attempts to model the human mind looking at from the perspective of fundamental needs and motivations; those things that regulate human action, including "perception, motivation, cognition, memory, learning and emotions" [13]. They created an educational role playing game called ORIENT with the resulting framework. The game did have a visual component, but it was not the focus of the work. This work shows that it is possible to use enhanced BDI agents to create credible characters in a game.

Silverman and colleagues' explored human-behavior models with a focus on the social aspect of human interactions with a goal of creating rich scenarios with multiple characters, resulting in a game world called NonKin Village [18]. The focus of the work was on the models that could be used to drive agent behavior; agents could explain their dilemmas and motivations in terms of psychology, sociology, politics, economics, as well as along family, ethnic, personal, commercial, or religious lines. As a secondary goal they did work on 3D viewers for the game. This project is not open-source so the code is not available to work with, but again, it shows that a multi-agent system can be used to create credible characters and scenarios in a game.

Second Life [17] is a very popular commercial (closed-system) 3D virtual world so there could be value in developing technologies that work with it. In Guerra's work [8], OpenSimulator (more commonly known as OpenSim) [15] was used in a framework that controls agents in a virtual world. OpenSim is an open-source .Net-based technology, which means it is limited to the Windows OS (or Mono implementations which can introduce its own challenges). It supports the core of the Second Life protocol [15]; Since OpenSim uses C# as a programming language, it would be more difficult to integrate with Java-based multi-agent systems like JADE and Jason.

Blair and Lin (2011) create an interface between virtual world – Open Wonderland – and a JADE multi-agent system [1]. They use Open Wonderland's extensible modules to create the hub for the communications happened in-between the two systems. In this research, a similar architecture which is even more tightly integrated to make better use of features within Open Wonderland is designed. Leung and colleagues (2011) use CArtAgO multi-agent system to control Open Wonderland and manipulate avatars in the environment [12]. In their system, avatars are controlled through the Open Wonderland client GUI, meaning that it is restricted to running in operating systems with a graphical layer like Windows. In the approach proposed in this paper the virtual world and the multi-agent system are integrated through the headless (without a GUI) client session and to run without any dependency on a graphical layer.

III. The Architecture

This research is about the creation of a robust and viable framework that interfaces a multi-agent system to a 3D virtual world. The following technologies have been selected to meet the functional and non-functional requirements. They are all open-source, and Java-based, which should make an integration effort more straightforward. For a virtual world engine, Open Wonderland [11] has been used. For a multi-agent system the combination of JADE and Jason has been used, leveraging the strengths of both where appropriate.

We use Open Wonderland as the virtual world. In selecting Open Wonderland over something like Gamebots which has a great deal of research done with it, it is important to look at the pros and cons. Gamebots is established and has a well-realized game engine that is graphically rich. Frameworks like Pogamut have been built on top of it, giving researchers the ability to focus on agents rather than concerning themselves with lower-level concerns. Gamebots works with the Unreal game engine, which has a rich scripting language. Therefore game worlds are highly configurable. Unreal is focused on a first-person shooter style of game play, but there are ways to use the environment for other purposes.

Open Wonderland offers an environment that is also graphically rich but which has not been used to the same degree as a commercial engine like Unreal. The environment is not focused on a particular style of game play and will not have the same level of built in components that the Unreal engine provides when it comes to building a commercial game. Open Wonderland is modular and the potential is there to add any modules that might be required. There are already modules related to business-use cases such as whiteboards, playing videos, sharing desktops, none of which exist in Unreal. Open Wonderland might not offer as much as commercial product like Unreal, but it is open source, written in Java, and is fully supported by a development community.

Considering how much has been done with Gamebots, the research team thinks there is just as much potential, if not more, for a fully open-source solution incorporating a 3D virtual environment and a multi-agent system. If it is done correctly, an integrative framework could be embraced by the research community and spark further work in the area. The goal is to create something functional that can be used for an educational game, but also something that is well put together and well documented so it can be shared with the Open Wonderland community as an Open Wonderland module.

Jason [9] is a platform for the development of Java-based open-source multi-agent systems, is used for creating agents that are based on the model of belief, desires, intentions (BDI), and is a Java-based interpreter for an extended version of AgentSpeak. The core code is easily extendible making it easy to add customizations at the agent and environment level. AgentSpeak, a Prolog derivative that is well suited to programming in AI, is a language used to control agents. AgentSpeak is the language that this research uses as agent scripts run to control NPCs in an Open Wonderland virtual world. The research team chose AgentSpeak because it is in more declarative style and offers a different way to script NPC behaviors than the functional methods like JavaScript which is already available in Open Wonderland.

Java Agent DEvelopment Framework (JADE) [16] provides a communications framework for FIPA-compliant agents, building a recognized standard for inter-agent communications into the proposed integrated framework. JADE is capable of running Java-based agents from other multi-agent systems such as Jason and JADEX. It provides a way of organizing agents into containers (main or secondary), and includes an agent management system and directory facilitator to facilitate the management of agents. JADE has been chosen because it can run Jason/AgentSpeak agents and offers an easy way to manage agents through its agent management system.

Fig. 1 shows the system overview of the proposed virtual world-multi-agent system interface. This diagram shows the modular nature of the integration and how the project code is localized in the MAS-JADE module. The Open Wonderland Server Connection Agent (run by JADE) has access to and is running the client-side aspects of the MAS-JADE module, along with all the other Open Wonderland code that typically runs on a client.

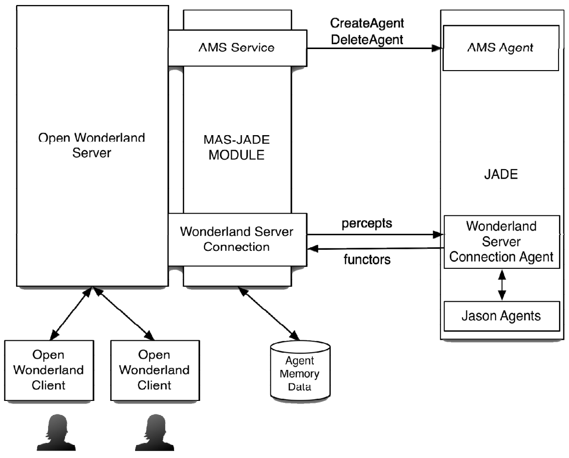


Figure 1. The proposed architecture.

IV. Implementation

An agent-controlled avatar can be added to the virtual world by its administrator. The administrator can insert an NPC object into Open Wonderland as Fig. 2 shows. The agent then controls the avatar inserted via the JADE-Controller that is usually invisible for users but now is set to see-able for readers on purpose. The administrator can also load any AgentSpeak script for the agent so the agent can manipulate its avatar to act and to respond appropriately. We have built a list of AgentSpeak scripts. These scripts are categorized into five libraries: NPCs, proximity, text chat, memory, and presence. More AgentSpeak scripts and libraries can be created, modified, and stored into the system easily. When the administrator right clicks on an avatar in the virtual world and selects "AgentSpeak Script", a Script Editor Panel pops up. The panel has buttons to "Open" or "Save" a script. The administrator can then select a file to open as Fig. 3 shows and load it into the editor panel. The administrator can edit the script and even activate the agent manually from this panel. Before an agent is able to come to "live" in Wonderland, the script has to be saved back to the content repository on the virtual world – Open Wonderland server so that the latest version is accessible to the multi-agent system – JADE server.

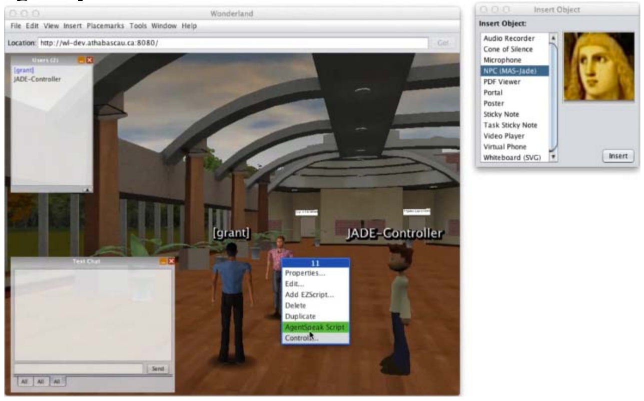
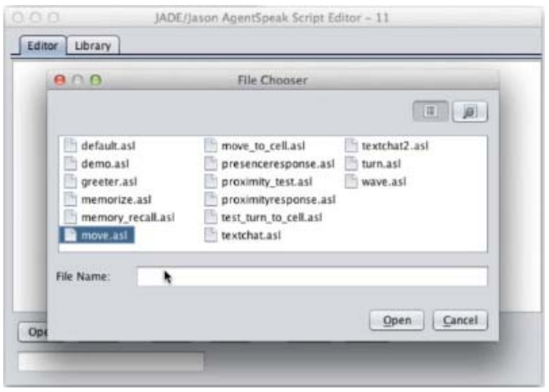


Figure 2. Add an agent controlled avatar into a virtual world and attach an AgentScript script to an agent behind the avatar.



default.asl
demo.asl
greeter.asl
memorize.asl
memory_recall.asl
move.asl
move_to_cell.asl
presenceresponse.asl
proximity_test.asl
proximityresponse.asl
test_turn_to_cell.asl
textchat.asl
textchat2.asl
turn.asl
wave.asl

Figure 3. Some pre-built AgentSpeak scripts.

In order for an agent to know what is happening within the virtual world, it needs to be able to perceive any in-world object, including regular objects, user avatars, and other agent-controlled avatars. This information could be absolute positions or relative positions in the virtual world. Either can be used to determine proximity.

The agent model is based on inputs (percepts) and outputs (actions). The primary action an agent is capable of doing in the virtual world is moving its avatar around in the environment. The agent should be able to direct its avatar to: (1) move towards an object; (2) move towards a place mark; (3) move in a direction (Left, Right, Forwards, Backwards); (4) turn to face an object; (5) turn to face a place mark; and, (6) turn to a certain orientation.

Percepts are needed in the area of communication. Wonderland has a text chat module, which is installed by default. It would be useful if that module were accessible to the agent, so that it could monitor what users are "saying" in the environment. For this to be useful, it will need the support of Natural Language Processing (NLP). In terms of actions, the agent should be able write to the chat module (both public and private chats). It would also be useful to provide a secondary interface for communication input, one that does not necessitate the use of NLP. With this in mind, it was envisioned that a control would be made available whereby the agent could offer a list of choices and a user could select one from the list; that choice would be made available to the agent in the form of a percept. There is an existing Open Wonderland module that presents "speech" in a speech bubble over an avatar's head. That could be utilized as another means of communicating to users.

An agent also needs to be able to remember things, short-term and long-term. Some of that can be stored in memory in the agent's beliefs (from the BDI model), but there should be a means of saving and retrieving memories so that not everything needs to be stored in the belief base, and so that if there server needs to be restarted for any reason (planned or otherwise) that information is all still available.

V. The Prototype and An Application Example

The proposed architecture and prototype system can be the foundation of the educational game, QuizMASter [4].

The QuizMASter is a kind of assessment game and it can be applied in any course context at any level from elementary school to post-secondary education, e.g., English and Physics. The QuizMASter can be controlled by a BDI agent whose belief is the answers of quizzes and its desire may include the expectation of seeing students focus on the questions as well as answer a question within a preset time limit. In order to assess students fairly, the agent may remind the students to focus on the game and explain the answers to the student at the end of each round through the QuizMASter. This research will be key, as it will be used by agents controlling avatars as well as agents controlling other aspects of the game, such as score keeping and learner modeling. To ascertain what is technically feasible when it comes to creating agent controlled avatars in a 3D virtual world like Open Wonderland. The following works for the proof of concepts have been done by the research team:

As an initial step, we integrated Open Wonderland and JADE through the starting of a JADE server separate from Open Wonderland [1]. The interface is a modified Open Wonderland module to which code has been added to start a JADE agent. The JADE agent is started via a runtime call and the group of agents that make up the MAS can be on different computers. The agents started by the Open Wonderland module will communicate with the rest of the JADE agents using the FIPA specified protocol.

The results of our previous work showed the possibilities and challenges that arise when the systems aren't coupled enough [1]. These challenges informed our decision to keep things simple and integrate everything right into Open Wonderland. The second step is adding a Jason module to Open Wonderland server that takes an avatar and controls its movement in the environment. Jason[1] is an interpreter for an extended version of AgentSpeak and a platform for the development of multi-agent systems.

The agent runs on the Open Wonderland server and pulls an AgentSpeak script from the content repository in the Open Wonderland server, sets up its belief system and goals, and then starts to control its avatar. As yet it doesn't get any perceptual information from the environment, but is able to guide its direction. It will be easy to extend this to have the agent makes any available move that an avatar can do within Open Wonderland. The JADE and Open Wonderland are more tightly coupled than what had originally been planned so that development, deployment, and management of QuizMASter are easier.

The third step is to use CArtAgO (Common ARTifact infrastructure for AGents Open environments) framework. CArtAgO is a general purpose framework that makes it possible to program and execute virtual environments for multi-agent systems [12]. CArtAgO is based on the Agents & Artifacts (A&A) meta-model for modeling and designing multi-agent systems. Since these artifacts can be modeled in Java, the core components of the Wonderland such as avatars have been wrapped as artifacts in order for the Jason agents to manipulate, manage, and communicate with them. A MaryTTS server is also added to the system to realize speech synthesis function. One of the main advantages of leveraging the capabilities of Jason-CArtAgO multi-agent system is that it enables customization of the rendering of visible artifacts for different clients [12]. Another advantage is that it bridges the gap between multi-agent system design and implementation.

Here is an example of how the proposed architecture and solution being used in educational context. When a learner signs into the virtual world – Open Wonderland, he or she can see a greeter as the greeter can "sense" the learner's presence with the Proximity Detection AgentSpeak script as Fig. 4 shows. When the messages are received on the Proximity channel, *proximity* beliefs are automatically added to the agent's belief base by the framework, including the name of the user that triggered the proximity message (FromUser), and the message type (Enter) indicating whether the user is entering or exiting the proximity space, and the distance (Distance). In the sample code shown in Fig. 4, the @msgHandler processes those beliefs as they are added, printing a message to the system log and then making a *text_chat_send_message* call to echo the message back from the "Proximity Agent".

The greeter, therefore, decides to say hello to the learner by approaching him or her. There are three avatars showing up in the world. The top right screenshot is given for giving readers the view that another player sees. The greeter also asks which classroom the learner may want to go so it can show him or her the way. When the learner tells it that he or she wants to go to Physics classroom as Fig. 5 shows, the greeter asks the learner to follow it. While walking the learner to the designated place, the greeter also keeps its "eyes" on the learner. When it is aware of the learner doesn't follow it, it will come back to the learner and ask him or her to follow it as Fig. 6 shows.

When the learner arrives at the Physics classroom, another agent – QuizMASter – is waiting for him or her. QuizMASter is a competitive educational game [4][20]. Several learners can play together and the QuizMASter will ask questions for them. At the end of the game, the learner who receives highest score wins as Fig. 7 shows. It can be told that the learner Lin receives highest score so far. The QuizMASter agent is also capable of detecting the distraction that a learner is by "seeing" his or her avatar not pays attention on itself but is looking around while it asks questions.

VI. Conclusions

Many of the project goals were achieved in this implementation and a number of them were not. The integration effort proved to be more difficult than initially thought. Open Wonderland is a complex system. Event processes are all transactional so it is a challenge to closely couple a multi-agent system with Open Wonderland. Three different approaches were tried with limited success before a workable architecture was arrived at (see Section 5 for details).

Open Wonderland is alpha software and the existing default avatar module has issues. The avatar module was used as a foundation for the project, with the assumption that

[1]http://jason.sourceforge.net/Jason/Jason.html

it would work without alteration. The alpha version of Open Wonderland does not just cause stabilization concerns with the avatar module, but it also causes stability issues in the core code. This added time to testing and troubleshooting issues, as it was often difficult to know if a problem was due to the Open Wonderland core code or the new module that was under development. The fact that it is not mature or well documented definitely added a great deal of time to the development effort.

There were challenges with both JADE and CArtAgO when it came to working with them, especially the later, when it came to lack of documentation and good examples of how to create and run Jason agents on a remote server instance/node. In neither case did it turn out to be complex, but it did require some effort and the testing of different approaches to get it to work. The research team had to create modified versions of the Agent Architecture class that is part of Jason's source code so that it would work properly in this framework.

To make environments more effective for educational applications, we need to provide a personalized and customized environment where they feel encouraged. We are working on user modeling for such environments. More work about the benchmarking of performance for the proposed architectures is also needed to be done.

At current stage, we have NPC controlled by BDI agent running in the multi-agent system based virtual world. Our next step is to extend the existing system to particular learning subject (e.g., English language learning) to verify the effectiveness of the proposed virtual learning environment and the benefit that students perceive from interacting with the proposed NPCs.

References

[1] J. Blair and F. Lin, "An Approach for Integrating 3D Virtual Worlds with Multiagent Systems," Proceedings of IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA 2011), IEEE Press, March 2011, pp. 580-585.

[2] M. Buckland, Programming Game AI by Example. Plano, TX: Jones & Bartlett Publishers, 2004.

[3] J. Dinerstein and P. K. Egbert, "Fast multi-level adaptation for interactive autonomous characters," Transactions on Graphics, vol. 24, no. 2, 2005, pp. 266-288.

[4] M. Dutchuk, K. A. Muhammadi, and F. Lin, "QuizMASter - A Multi-Agent Game-Style Learning Activity," Proceedings of International Conference on E-learning and Games (Edutainment 2009), Springer, August 2009, pp. 263-272.

[5] FatiMA: http://sourceforge.net/projects/fatima-modular/files/

[6] FIPA : http://fipa.org

[7] J. Gemrot, R. Kadlec, M. Bída, O. Burkert, R. Píbil, J. Havlíček, L. Zemčák, J. Šimlovič, R. Vansa, M. Štolba, T. Plch, and C. Brom, "Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents," Proceedings of International Workshop on Agents for Games and Simulations (AGS 2009), Springer, May 2009, pp. 1-15.

[8] A. Guerra, A Framework for Building Intelligent Software Assistants for Virtual Worlds. ETD Collection for Pace University. Available: http://digitalcommons.pace.edu/dissertations/AAI3462987/

[9] C.-H. Jo, G. Chen, and J. Choi, "A new approach to the BDI agent-based modeling," Proceedings of ACM symposium on Applied Computing (SAC 2004), ACM, March 2004, pp. 1541-1545.

[10] G. A. Kaminka, M. M. Veloso, S. Schaffer, C. Sollitto, R. Adobbati, A. N. Marshall, A. Scholer, and S. Tejada, "GameBots: a flexible test bed for multiagent team research," Communications of the ACM, vol. 45, no. 1, 2002, pp. 43-45.

[11] J. Kaplan and N. Yankelovich, "Open Wonderland: An Extensible Virtual World Architecture," IEEE Internet Computing, vol. 15, no. 5, 2011, pp. 38-45.

[12] S. Leung, S. Virwaney, F. Lin, A. J. Armstrong, and A. Dubbelboer, "TSI-enhanced Pedagogical Agents to Engage Learners in Virtual Worlds," International Journal of Distance Education Technologies, vol. 11, no. 1, 2013, pp. 570-575.

[13] M. Y. Lim, J. Dias, R. Aylett, and A. Paiva, "Creating adaptive affective autonomous NPCs," Autonomous Agents and Multi-Agent Systems, vol. 24, no. 2, 2012, pp. 287-311.

[14] M. J. Jacobson, B. Kim, C. Miao, Z. Shen, and M. Chavez, "Design Perspectives for Learning in Virtual Worlds," in Designs for learning environments of the future: International perspectives from the learning sciences, M. J. Jacobson, and P. Reimann, Eds. Springer, 2009, pp. 111-141.

[15] OpenSimulator: http://opensimulator.org/wiki/Main_Page

[16] A. S. Rao and M. P. Georgeff, "BDI agents: From theory to practice," Proceedings of International Conference on Multiagent Systems (ICMAS 1995), AAAI, June 1995, pp. 312-319.

[17] Second Life: http://secondlife.com

[18] B. G. Silverman, D. Pietrocola, N. Weyer, R. Weaver, N. Esomar, R. Might, and D. Chandrasekaran, "NonKin Village: An Embeddable Training Game Generator for Learning Cultural Terrain and Sustainable Counter-Insurgent Operations," Proceedings of International Workshop on Agents for Games and Simulations (AGS 2009), Springer, May 2009, pp. 135-154.

[19] Unreal Technology: http://www.unrealtechnology.com/technology.php

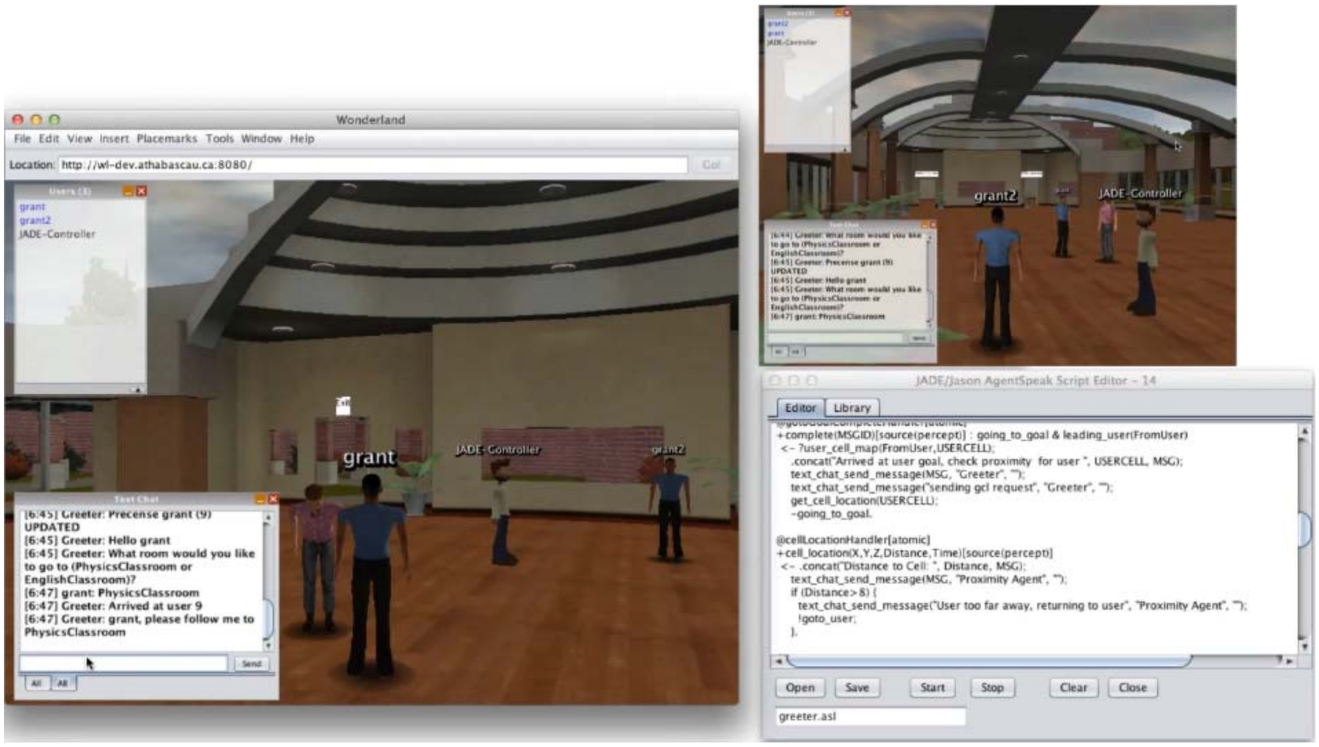
[20] M. Weng, F. Lin, T. K. Shih, M. Chang, and I. Fakinlede, "A Conceptual Design of Multi-Agent based Personalized Quiz Game," Proceedings of IEEE International Conference on Advanced Learning Technologies (ICALT 2011), IEEE Press, July 2011, pp. 19-21.

```
/* Initial goals */
!listen.

@msgHandler[atomic]
+proximity(FromUser,Enter,Distance) : proximity(_,_,_)[source(percept)]
 <- .print(FromUser);
 text_chat_send_message(FromUser, "Proximity Agent", "").

@listenHandler[atomic]
+!listen : true
 <- proximity_listen;
 text_chat_send_message("listening to proximity messages", "JADE
Controller", "");
 cheer.
```

Figure 4. Sample AgentSpeak Code for Proximity Detection.



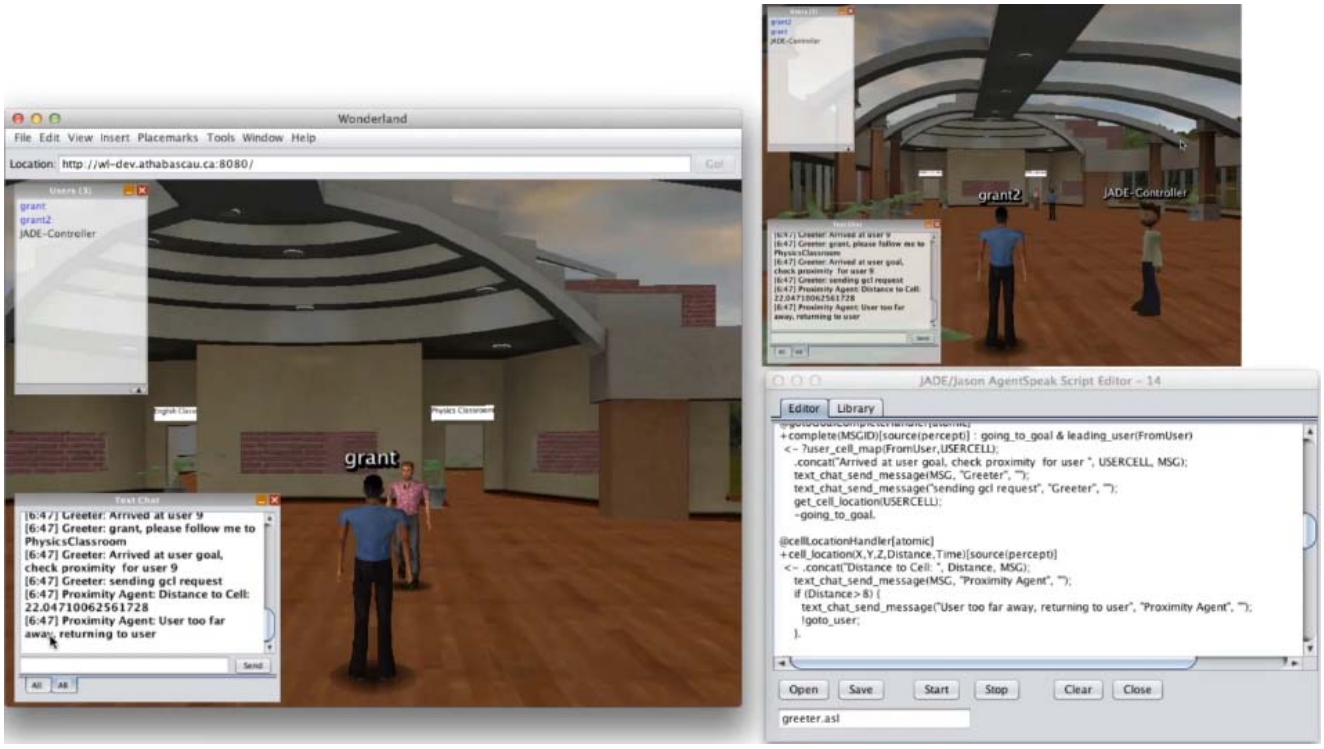
```
[6:45] Greeter: Precense grant (9)
UPDATED
[6:45] Greeter: Hello grant
[6:45] Greeter: What room would you like
to go to (PhysicsClassroom or
EnglishClassroom)?
[6:47] grant: PhysicsClassroom
[6:47] Greeter: Arrived at user 9
[6:47] Greeter: grant, please follow me to
PhysicsClassroom
```

```
[6:44] Greeter: What room would you like
to go to (PhysicsClassroom or
EnglishClassroom)?
[6:45] Greeter: Precense grant (9)
UPDATED
[6:45] Greeter: Hello grant
[6:45] Greeter: What room would you like
to go to (PhysicsClassroom or
EnglishClassroom)?
[6:47] grant: PhysicsClassroom
```

```
+complete(MSGID)[source(percept)] : going_to_goal & leading_user(FromUser)
 <- ?user_cell_map(FromUser,USERCELL);
   .concat("Arrived at user goal, check proximity  for user ", USERCELL, MSG);
   text_chat_send_message(MSG, "Greeter", "");
   text_chat_send_message("sending gcl request", "Greeter", "");
   get_cell_location(USERCELL);
   -going_to_goal.

@cellLocationHandler[atomic]
+cell_location(X,Y,Z,Distance,Time)[source(percept)]
 <- .concat("Distance to Cell: ", Distance, MSG);
   text_chat_send_message(MSG, "Proximity Agent", "");
   if (Distance>8) {
     text_chat_send_message("User too far away, returning to user", "Proximity Agent", "");
     !goto_user;
   }.
```

Figure 5. Greeter leads the user to Physics classroom as per user request.



```
[6:47] Greeter: Arrived at user 9
[6:47] Greeter: grant, please follow me to
PhysicsClassroom
[6:47] Greeter: Arrived at user goal,
check proximity  for user 9
[6:47] Greeter: sending gcl request
[6:47] Proximity Agent: Distance to Cell:
22.04710062561728
[6:47] Proximity Agent: User too far
away, returning to user
```

```
@gotoGoalCompleteHandler[atomic]
+complete(MSGID)[source(percept)] : going_to_goal & leading_user(FromUser)
 <- ?user_cell_map(FromUser,USERCELL);
   .concat("Arrived at user goal, check proximity  for user ", USERCELL, MSG);
   text_chat_send_message(MSG, "Greeter", "");
   text_chat_send_message("sending gcl request", "Greeter", "");
   get_cell_location(USERCELL);
   -going_to_goal.

@cellLocationHandler[atomic]
+cell_location(X,Y,Z,Distance,Time)[source(percept)]
 <- .concat("Distance to Cell: ", Distance, MSG);
   text_chat_send_message(MSG, "Proximity Agent", "");
   if (Distance>8) {
     text_chat_send_message("User too far away, returning to user", "Proximity Agent", "");
     !goto_user;
   }.
```

Figure 6. Greeter is aware of the user doesn't follow its lead.

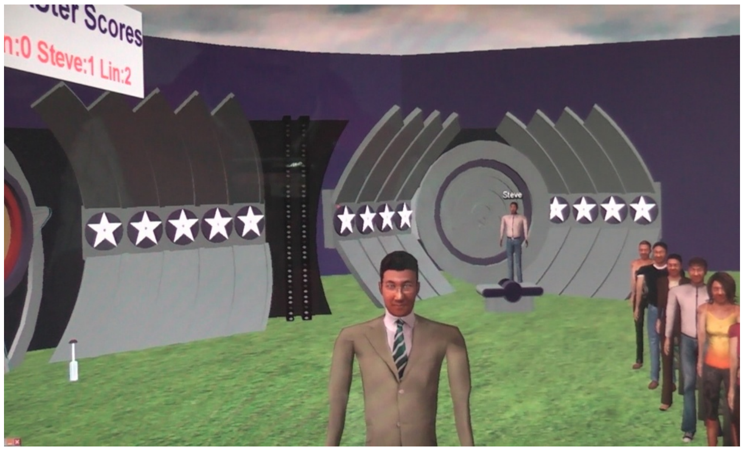


Figure 7. QuizMASter is asking questions for three learners.