Integrating Multiagent Systems into Virtual Worlds

Grant McClure Sandeep Virwaney and Fuhua Lin[1]

Abstract. Incorporating autonomy and intelligence into virtual worlds to build an engaging virtual environment for applications such as serious games is becoming more desirable. There are challenges in integrating these systems including concerns with synchronization, communication, monitoring, efficiency, and control. This paper presents an approach to integrating a virtual world engine with a multiagent system platform through the creation of an interface between them. We show the feasibility and effectiveness of the approach through developing agents controlled non-player controlled characters (NPCs) in Open Wonderland and purposeful communication channels among agents using Jason AgentSpeak.

Keywords: Virtual Worlds • Multiagent Systems • Integration • Serious Games

1 Introduction

Virtual Worlds (VW) are 3D graphical environments that provide multimedia, engaging, and immersive user experience. They can provide more dimensions than physical environments, more social nuanced ways for people. Multiagent systems (MAS) encompass distributed problem-solving applications; such as network management has a focus on inter-agent communication, coordination, and negotiation. Recently, researchers in VW community have adopted MAS as they are 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. Also, agents can be used for modelling non-player character (NPCs), keeping track of individual interests, motivations, and goals in game worlds. However, there is not a platform that can help programmers to develop and test agent behaviors in virtual worlds that need autonomy and intelligence and interaction with other agents and the environments.

[1] Fuhua Lin (✉)
School of Computing and Information Systems, Athabasca University, Canada
e-mail: oscarl@athabascau.ca

AI techniques have been used to drive agents, but this is usually done in a prescriptive manner that doesn’t give a lot of freedom to agents' actions. These agents, such as NPCs, require simple scripting as the tasks they are following are simple procedures and need almost no interaction with the environment.

In this paper, we propose an approach that combines agent-oriented programming and environment-oriented programming to building a multiagent virtual world platform. Such a platform allows programmers to specify the coordination, scheduling and management without worrying about communication, protocols, load balancing, and other vagaries of game programming.

To show the feasibility of the proposed approach, we develop a set of AgentSpeak scripts of autonomous agents to control NPCs 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. We believe that the integration of multi-agent systems and virtual worlds opens a number of extremely interesting and potentially useful research avenues concerning agent coordination for applications such as serious games and simulation.

This rest of the paper is organized as follows. First, we will summarize some related work and identify research problems. In Section 3, we describe the proposed architecture. Section 4 explains briefly the developed agent-controlled NPCs. In Section 5 shows an application example QuizMASter. Finally, we will draw some conclusions and discuss the future work.

2 Related Work

There have been relatively mature technologies for modeling virtual humans in virtual worlds in their appearance, gestures, kinematics, and physical properties. We are facing the challenge in VW research and development to emulate or simulate the way human 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.

Research has been done over the last decade when it comes to creating agent control mechanisms for VW. The first project of note is Gamebots (Kaminka et al., 2002), which was a client-side approach to controlling Unreal Tournament[2], an engine used for first-person shooter games. 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 (Dinerstein & Egbert, 2005). They noted that their

[2] http://www.unrealtechnology.com/technology.php

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.

Pogamut is another agent implementation which works with Gamebots (Gemrot et al., 2009). 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 et al. (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.

Virtual Singapura is a fantasy world, takes the learners to nineteen century Singapore in the throes of disease epidemic (Jacobson et al., 2009). 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[3] architecture and a drives-based PSI model in an attempt to create more believe NPCs (Lim et al., 2012). 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 (Lim et al., 2012). They created an educational role playing game called ORIENT with the resulting framework. The work showed that it is possible to use enhanced BDI agents to create credible characters in a game.

Second Life[4] is a very popular commercial 3D virtual world so there could be value in developing technologies that work with it. In Guerra’s work (Guerra, 2011), OpenSimulator[5] (i.e., OpenSim) 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; Since OpenSim uses C# as a programming language, it would be more difficult to integrate with Java-based multi-agent systems like JADE (Bellifemine et al., 2004) and Jason (Bordini et al., 2007).

In Dignum et al. (2009), an excellent overview of the state of the art is provided when it comes to incorporating MAS concepts with gaming systems or other 3D virtual environments. As pointed out by Dignum et al. (2009), the main difficulties in integrating any game engine with a multiagent system are related to synchronization, information representation, and communication (Dignum et al., 2009).

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

[4] http://secondlife.com

[5] http://opensimulator.org/wiki/Main_Page

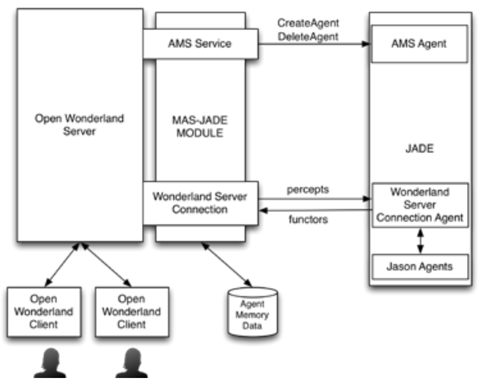
3 The Proposed Architecture

The goal of this research is to propose an approach to building an interface between a virtual world and a multiagent system. The interface would have capabilities of MAS communication and coordination and further sensory feedback to actually enable interaction between human avatars and software agents.

Such an integrated platform will allow programmers to program and test a complex application of virtual worlds such as serious games in an earlier way without worrying about communication, protocols, load balancing, and other vagaries of game programming. 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.

We use Open Wonderland (Kaplan and Yankelovich, 2011) 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. 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 in Java, and is fully supported by a development community.

Jason (Bordini, et al., 2007) 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) (Rao and Georgeff, 1995), 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. Java Agent DEvelopment Framework (JADE) (Bellifemine et al., 2004) 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. Fig. 1 shows the system overview of the proposed multi-agent virtual world system interface. This diagram shows the modular nature of the integration.



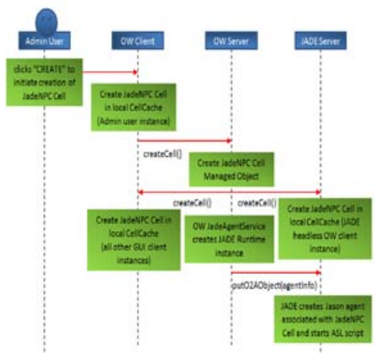

Admin User
OW Client
OW Server
JADE Server
clicks "CREATE" to initiate creation of JadeNPC Cell
Create JadeNPC Cell in local CellCache (Admin user instance)
createCell()
Create JadeNPC Cell Managed Object
createCell()
createCell()
Create JadeNPC Cell in local CellCache (all other GUI client instances)
OW JadeAgentService creates JADE Runtime instance
Create JadeNPC Cell in local CellCache (JADE headless OW client instance)
putO2AObject(agentInfo)
JADE creates Jason agent associated with JadeNPC Cell and starts ASL script

Fig. 1. The proposed architecture.

Fig. 2: NPC Agent Creation Sequence Diagram.

3.1 The Interface

The interface is localized in the MAS-JADE module which can be broken out into the following components:

Jason Controller Component: This component handles core Jason functionality (server start-up, logging, etc.). It can be attached to any object, but it makes the most sense to attach it to the root world object. When this component is attached to the world, an additional administrative area will show up on the properties panel for the world. There will be settings to enable/disable the MAS, enable/disable logging, and list the active agents in the MAS.

Jade Controller Component This component is similar to the Jason Controller Component, only it runs a JADE server instance.

Jason NPC Module This is a modified version of the NPC module available in Open Wonderland, extended to read in an AgentSpeak file, connect to a Jason server instance. It associates a single NPC in the Open Wonderland environment to a single agent in the Jason MAS. There will be an additional settings tab for this type of object allowing for the configuration of that NPC. The type of MAS can be selected (Generic, Host, Guide, etc.) allowing for the select of canned AgentSpeak code that will be made available to the module. There will also be a place to add specific AgentSpeak code that can be tailored to the particular instance of the NPC, allowing for the creating of unique behaviors when configuring the environment, rather than having to define everything at a development stage.

Jade NPC Module Similar to the Jason NPC Module, only it will allow for the running of JADE Java code.

The interface includes the Open Wonderland Server Connection Agent (run by JADE) which 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.

4 Agent-Controlled NPCs

To show the feasibility of the proposed architecture, we develop a set of AgentSpeak scripts to control avatars. An agent-controlled avatar can be added to the world by the administrator of the virtual world. Fig.2 shows a sequence diagram describing the communications involved in creating an agent to control an NPC. The administrator can insert an NPC object into Open Wonderland as Fig. 3 shows. The agent then controls the avatar inserted via the JADE-Controller which is usually invisible for users but now is set to see-able for readers on purpose.

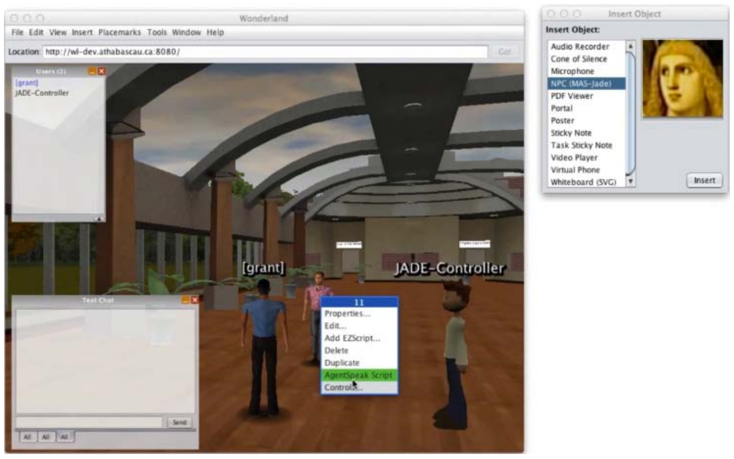


Fig. 3. Add an agent controlled avatar into a virtual world and attach an AgentScript script to an agent behind the avatar.

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.

NPC perception Percepts are an important part of the agent model. In order for an agent to be able to evaluate what it should do next to best pursue its goals, it needs to know the current state of its environment. That state information needs to be fed to the agent as an input known as a percept. In order to for an agent to know what is happening within the Open Wonderland environment, it needs to be able to perceive any in-world object, including regular objects, user avatars, and other NPCs. This information could be absolute positions in world, or a relative position from the NPC. Either can be used to determine proximity.

NPC Movement The agent model is based on inputs (percepts) and outputs (actions). The primary action an NPC is capable of doing in Open Wonderland is move around in the environment. 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. Open Wonderland's collision system is used to ensure an NPC cannot do things like walk through walls. It is allowed in the environment for avatars (both user and NPC) to walk through each other so that is not a concern. Fig. 4 shows a basic AgentSpeak script that has an NPC move to a particular cell in world, in this case the Physics Classroom. The agent is given an initial goal to *move*. The plan for *move* has the agent tell the NPC to bow, then move to the cell in question, and then wave. When the NPC moves it makes use of the underlying code in the avatarbase of Open Wonderland for NPC movement (its GoTo logic). It does have some collision avoidance logic built into it, but that only applies to certain objects. Otherwise the NPC should take the most direct root, which includes walking through walls. When the NPC arrives at its goal, it should place itself in front of the cell in question, and then turn to face it.

```
!move. /* Initial goals */
@waveHandler[atomic]
+!move : true  <- bow;
 move_to_cell("PhysicsClassroom");  wave.
-!move: true  <- .print("got stuck");  !!move.
```

Fig. 4: Sample AgentSpeak Code for Move to Cell.

Communication Percepts are needed in the area of communication. Open Wonderland has a text chat module, which is installed by default. It would be useful if that module were accessible to the NPC 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.

NPC Memory An agent 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.

5 Conclusions

We have presented an approach to integrating a multiagent system with a virtual world through the creation of an interface between them. The powerfulness of the integrated system platform would be considerable as it combines the advantages of virtual worlds and multiagent systems. We show the feasibility and effectiveness of the approach through developing agents controlled non-player controlled characters in Open Wonderland and purposeful communication channels among agents using Jason AgentSpeak. We are working the memory function and char function and agent coordination models for real-world applications such as serious games in virtual worlds.

References

1. Bellifemine, F., G. Caire, and D. Greenwood, (2004), Developing multi-agent systems with JADE, Wiley.
2. Bordini, R. H., J. F. Hubner, and M. Wooldridge (2007), Programming multi-agent systems in AgentSpeak using Jason, Wiley.
3. Dignum, F., J. Westra, W. A. van Doesburg, and M. Harbers, games and agents designing intelligent gameplay, nternational Journal of Computer Games Technology, Volume 2009 (2009), Article ID 837095, 18 pages. doi:10.1155/2009/837095
4. Dinerstein, J., Egbert, P. K.: Fast multi-level adaptation for interactive autonomous characters. Transactions on Graphics 24(2), 266--288. (2005)
5. Gemrot, J., Kadlec, R., Bída, M., Burkert, O., Píbil, R., Havlíček, J., Zemčák, L., Šimlovič, J., Vansa, R., Štolba, M., Plch, T., Brom, C.: Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents. In: First International Workshop on Agents for Games and Simulations, pp. 1--15. Springer. (2009)
6. Guerra, A.: A Framework for Building Intelligent Software Assistants for Virtual Worlds. ETD Collection for Pace University. Paper AAI3462987. (2011) Available: http://digitalcommons.pace.edu/dissertations/AAI3462987/
7. Jo, C.-H., Chen, G., Choi, J.: A new approach to the BDI agent-based modeling. In: 2004 ACM symposium on Applied computing. pp. 1541--1545. ACM. (2004)
8. Kaminka, G. A., Veloso, M. M., Schaffer, S., Sollitto, C., Adobbati, R., Marshall, A. N., Scholer, A., Tejada, S.: GameBots: a flexible test bed for multiagent team research. Communications of the ACM 45(1), 43--45. (2002)
9. Kaplan, J. & Yankelovich, N.: Open Wonderland: An Extensible Virtual World Architecture. IEEE Internet Computing 15(5), 38--45. (2011)
10. Lim, M. Y., Dias, J., Aylett, R., Paiva, A.: Creating adaptive affective autonomous NPCs. Autonomous Agents and Multi-Agent Systems 24(2), 287--311. (2012)
11. Jacobson, M. J., Kim, B., Miao, C., Shen, Z., Chavez, M.: Design Perspectives for Learning in Virtual Worlds, in Jacobson, M. J., & Reimann, P. (Eds.). Designs for learning environments of the future: International perspectives from the learning sciences. Springer, 111--141. (2009)
12. Rao, A. S., Georgeff, M. P.: BDI agents: From theory to practice. In: First International Conference on Multiagent Systems, pp. 312--319. AAAI. (1995)