ATHABASCA UNIVERSITY

An Approach for Integrating 3D Virtual Worlds and Multi Agent Systems

BY

Jeanne Blair

A project submitted in partial fulfillment

Of the requirements for the degree of

MASTER OF SCIENCE in INFORMATION SYSTEMS

Athabasca, Alberta

March 2011

© Jeanne Blair, 2011
DEDICATION

To my wonderful husband David, thank you for your patience, support and understanding while I undertook this work. Also, to the many co-workers who helped me with their humor and empathy, and to my sponsors, Andrea Wilson and Darren Gour, who took the time to write wonderful letters of reference for my introduction to Athabasca University. And especially, to the excellent and inspiring professors of Athabasca University, to all of you I am grateful. Finally, to my son Douglas, who is a true “digital native”, perhaps because my involvement in technology exposed him to all things digital from toddler onwards. Douglas introduced me to the massively multiplayer online gaming 3D worlds and challenged me to understand the benefits of spending time playing games in 3D virtual worlds.
The research objective of this essay is integrating multi agent systems (MAS) into networked 3D virtual learning environments (VLE). Integrating a multi agent system with a 3D virtual learning environment allows abstracting from the peculiarities of interacting with the virtual environment, so designers may focus on the pedagogical problem to solve. The focus of this research is specifically on the “how to” integrate a multi agent system into a 3D virtual learning environment.

This approach for integrating MAS with 3D VLE was developed by devising, implementing and testing an approach using open source technologies, namely, Open Wonderland and JADE. This approach results in a multi agent system being started by the 3D virtual learning environment when an action is done to a 3D module. Once the MAS is started it can then perform its functions and report back to the 3D virtual learning environment the result of those functions.

Implementation was accomplished by adding code to the Open Wonderland sample module to start one agent of a multi agent system. A proof-of-concept using the JADE Book Trading example encouraged further implementation of a case study using a MAS named QuizMAStEr as an example of a educational application. Success was tested using Windows and Mac clients on the Internet connecting to the Open Wonderland virtual world and playing the QuizMAStEr game. This work provides a framework for
building upon. Future research work would be to improve the pedagogical functionality of the QuizMASter system.
ACKNOWLEDGMENTS

This work was made possible by Athabasca University, Canadian Foundation for Innovation and by Dr. Fuhua (Oscar) Lin.
# TABLE OF CONTENTS

CHAPTER I .......................................................................................................................... 8

INTRODUCTION .................................................................................................................. 8

3D Virtual Worlds as Learning Environments ................................................................. 8

Multi Agent Systems that Support the Learning Environment .................................... 10

CHAPTER II ....................................................................................................................... 12

REVIEW OF RELATED LITERATURE ............................................................................. 12

CHAPTER III ..................................................................................................................... 16

METHODOLOGY ............................................................................................................... 16

CHAPTER IV ..................................................................................................................... 19

ISSUES, CHALLENGES, AND TRENDS ......................................................................... 19

Architecture ....................................................................................................................... 19

Implementation .................................................................................................................. 22

Proof-of-concept: Integration of Book Trading with Open Wonderland ...................... 27

Case Study: Integration of QuizMASter with Open Wonderland .................................... 31

CHAPTER V ....................................................................................................................... 34

CONCLUSIONS AND RECOMMENDATIONS ................................................................. 34

Future Research ............................................................................................................... 37

REFERENCES ..................................................................................................................... 39

APPENDIX A ....................................................................................................................... 44
LIST OF FIGURES

Figure 1 Architecture ................................................................. 22
Figure 2 Poster module listens to TCP port........................................... 29
Figure 3 Book Buyer agent communicates with other agents to find and buy book .... 30
Figure 4 Book Buyer agent updates Poster............................................... 31
Figure 5 QuizMASter integration with Open Wonderland.......................... 33
CHAPTER I

INTRODUCTION

The research objective of this essay is integrating multi agent systems (MAS) into networked 3D virtual learning environments (VLE). Integrating a multi agent system with a 3D virtual learning environment allows abstracting from the peculiarities of interacting with the virtual world software, so designers may focus on the pedagogical problem to solve. The focus of this research is specifically on the “how to” integrate a multi agent system into a 3D virtual learning environment The approach developed and presented in this paper results in a multi agent system being started by the 3D virtual learning environment when an action is done to a 3D module in the virtual world. Once the MAS is started it can then perform its functions and report back to the 3D virtual learning environment the results of those functions. Implementation was accomplished by adding code to the Open Wonderland sample module to start one agent, thereby adding it to an already operational multi agent system. The intent of this research is to support the use of 3D virtual worlds as learning environments with the added functionality provided by multi agent systems that support the learning environment.

3D Virtual Worlds as Learning Environments

Learning environments that provide an immersion into the content is a natural step towards creating learning environments for the new “digital native” group of learners [1]. The current generation of learners are used to multimedia applications that are immersive
and highly interactive because many of them have been video-gaming since they were kids [2]. Multi user virtual learning environments have been created by many of the world’s universities to provide such a learning environment in a 3D virtual world. A multi user virtual learning environment is best used by a synchronous but geographically separated group of learners [3]. Multi user virtual learning environments are said to be perfect for “experiential” learning by providing concrete experience, reflective observation, abstract conceptualization, and active experimentation [4].

Two different approaches toward multi agent virtual learning environments are surfacing; the virtual classroom and the fantasy world. The virtual classroom has the look and feel of a regular classroom and is often made to look like a replica of the sponsoring university. Second Life (SL) is often the choice for building the 3D virtual classroom [3-5]. Results are a mixed bag, with some reporting that SL has too many difficulties and is not game like [4]. Some learners reported that SL is not an environment where they would want to spend time [6]. SL did not appeal to those who enjoy social software because of difficulties engaging with others, nor did it appeal to gamers because of the lack of any gaming scenarios.

The fantasy world teleports the learner to a different place or time. Virtual Singapura [7], an example of a fantasy world, takes the learner to nineteenth century Singapore in the throes of a disease epidemic. Here, faced with the daunting task of being limited to only the technologies of the time, the students have to solve the problems of the time. Their guided activities are based on the Virtual Singapore Lab Book, leading the student in the
use of scientific inquiry skills and science content about communicable diseases. Another example is Mare-Monstrum where the tutor takes the form of a dragon and learners face challenges and communicate at the cliff [2]. Fantasy worlds offer the ability to construct learning environments that use Problem Based Learning (PBL) as the underlying pedagogical strategy [2]. Fantasy multi user virtual learning environments offer socio-constructive pedagogy via PBL collaborations where the tutor role changes from being the “sage on the stage” to being a guide in the process of finding a solution to a problem [8]. Using such a learning environment, the Electrical Engineering School at the University of Madrid reported a decline in drop-out rates in engineering education due to the incorporation of virtual world games [2]. A study of World-of-Warcraft, arguably the most popular 3D virtual world, led researchers to define a reward model of four reward groups, glory, sustenance, access and facility that would be transferable to an educational 3D virtual world [9].

Multi Agent Systems that Support the Learning Environment

Agents are encapsulated computer systems that are designed to behave flexibly and somewhat autonomously to achieve some goal(s) [11]. Agents are situated in some environment and have some autonomy and capabilities to observe that environment. They can communicate those observations to other agents. This makes them particularly suited to distributed environments. Within the distributed environment, the Multi-Agent System (MAS) can be designed using one or more architectures. The autonomous nature of
agents implies that the architecture can even develop dynamically at run-time. A group of peer-to-peer agents can appoint one agent to be a central agent to whom the others report.

The application of agent-based technology to education can be identified by two groups of technologies, 1) Intelligent Tutoring Systems (ITS) and 2) Interactive Learning Environments (ILE) [12]. Some criticism of the current state of tutoring systems is the lack of sufficient intelligence in the tutoring system with a possible solution being the incorporation of multi agent systems, such as a teacher agent and a student agent into the virtual classroom [12-14]. Multi agent systems can also provide agents that act as guides, information retrieval assistants, and help systems in 3D virtual learning environments [15-18].

The pedagogical benefits provided by the interactive MAS in learning environments are shown to be [10]:

- Interactive learning experience motivates students and helps to retain knowledge
- Instant feedback helps identify weakness in areas
- Interaction with other students introduces the competitive element which in turn makes the experience more engaging, interesting and fun
- Students collaborate with each other
- Students progress at their own pace
CHAPTER II

REVIEW OF RELATED LITERATURE

According to [19] computer-based training systems with the greatest potential are based on the concept of virtual reality or virtual environments (VE). For training purposes, simulated situations within the virtual environment tolerates human made errors without costs in terms of human injury or equipment repair. Incorporating artificial intelligence (AI) technology into the virtual reality based training systems has improved usability and effectiveness [19]. The combination of AI techniques and virtual reality has given birth to the field of intelligent virtual environments (IVE) [20]. The usual approach is to integrate an agent as a bot player, using agent techniques such as JACK to provide a Belief-Desire-Intentions based agent. In this approach, the AI is embedded in the graphic engine and is thus not very extensible or scalable. Adopting a multi agent system (MAS) approach for the AI component addresses concerns of robustness, scalability, re-usability and maintainability, is autonomous, pro-active, re-configurable and agile [20]. Integrating AI using multi agent systems and virtual environments creates a framework that is flexible and versatile. Implementing AI as a multi agent system (MAS) provides scalability of both the number of participating devices and the number of agents interacting in the environment. The AI and VE operate independently and there is a clear separation of the intelligence part and the visualization part. This approach supports the use of specific graphic workstations exclusively for the rendering, and a separate computer for each of the different agents in the multi agent system, allowing for scalability of applications built using this framework. Barella, Carrascosa and Botti [20]
developed a framework that uses JADE as a multi agent platform. The JADE agents
don’t need a control module, as it is JADE itself that provides control functions. Their
MAS architecture consists of three separate subsystems.

- The Multi Agent Platform, which is JADE.
- The Multi Agent System, which operates to achieve a goal.
- Visualization Module, which is a graphic application developed to display the 3D
  objects and scenarios.

With their architecture, the main feature is that the AI and VE operate independently such
that it is even possible to run the MAS even if there are no graphical users. Their
approach to integrate AI in VE allows abstracting from the peculiarities of interacting
with the VE, so the designers may focus on the problem to solve.

An early example of integrating 3D virtual worlds with MAS and perhaps the best known
is GameBots developed at the University of Southern California in collaboration with
Carnegie Mellon University, which integrated agents into a 3D virtual game environment
called Unreal Tournament [21]. GameBots is based on a commercially developed game
engine, Unreal Tournament (UT) that was extended and enhanced to provide the
capabilities for integration of a multi agent system. Unreal Tournament is primarily a
first-person shooter game in which human players can select from three different game
scenarios; Deathmatch, Domination or Capture the Flag. GameBots consists of a
customized UT module that communicates with the multi agent system via network
sockets connected to a bot client. The multi agent system has advanced AI capabilities to
plan the actions that are then sent to the customized UT module. GameBots tries to
appeal to the artificial intelligence community as a general test-bed for AI research. GameBots is important because it proved that multi agent systems can be integrated with 3D virtual worlds, but its suitability for educational environments is limited because it is limited to the scenarios provided in a first-person shooter game.

More recently, the University of Lisbon developed e-Game, which is an integration of a JADE multi agent platform with a video game environment called Counter-Strike [22]. Counter-Strike also falls in the first-person shooter genre, but is a team-based first-person shooter game with two teams, terrorists and counter-terrorists. Extending Counter-Strike proved to be challenging because it is poorly documented and the code is poorly organized. Providing integration of JADE agents with Counter-Strike required the development of a new platform, with several new modules for navigation, vision, combat, finances and communication. The platform supports up to 32 players, either human players or agents. Agents interact with other agents and with the 3D world using the platform modules. The main goal of e-Game is to use a 3D virtual world and game playing to teach a software course for developing JADE agents, thus not lending itself to the larger educational community.

More aligned to the larger educational community, The University of Le Havre developed an Intelligent Tutoring Systems (ITS), built as a JADE multi agent system, which is integrated with a 3D virtual world based on SCOL [23]. The result is named GE3D: A Virtual Campus for Technology-Enhanced Distance Learning. The 3D virtual campus contains tools specifically designed for the learner: such as a mail box for
sending email, a board for time tables, a board for assessment marks, a board for system help and an amphitheatre for synchronous presentations. The ITS tracks students work, tailoring feedback and hints. There are three main parts to the ITS, the Human Computer Interfaces, knowledge parts and a four layer multi agent systems (MAS) developed in JADE. The GE3D architecture can combine lectures, learning through projects (i.e. collaborative learning), learning with and from the peers (i.e. cooperative learning) and learning by problem solving.
CHAPTER III

METHODOLOGY

A survey of the research presented above led to a realization that multi agent systems could alleviate the perceived lack of intelligence in MUVE based tutoring systems. The challenge is to find an approach that is not overly difficult and does not require drastic changes to either the 3D virtual world or the multi agent software. Our approach to this challenge was to find a way to integrate a simple multi agent system developed on the Java Agent Development framework (JADE) (jade.tilab.com) with Open Wonderland (OWL), an open source 3D virtual world developed by SUN (www.openwonderland.org).

The sheer complexity of implementing an educational game in a 3D virtual world presents the most significant barrier to widespread educational use. For example, jMonkeyEngine (JME), the engine on which Open Wonderland is based, is a powerful game-engine that provides the low-level infrastructure needed to build 3D graphics systems and games. The JME provides functionality for rendering, collision detection, and other core functionalities required to create a 3D virtual world. It could be therefore argued that the JME provides all the necessary tools to construct an educational game. However, the JME game engine is all but inaccessible to most educators. To implement an educational activity using the JME as a starting point would be impossible for anyone lacking advanced computer programming skills, and a significant amount of time to devote to its implementation. Open Wonderland takes an important step towards making
3D virtual world technology more accessible to non-programmers. The Open Wonderland toolkit builds upon the JME infrastructure, adding features that enable the content developer to more easily add virtual world elements such as buildings, furniture, scenery, and other elements needed to provide an immersive 3D experience. For example, educators will benefit greatly from Open Wonderland’s ability to import graphics elements from Google’s 3D Warehouse, which contains thousands of 3D elements in a wide range of categories, and are available free of charge. The toolkit also provides support for adding items such as whiteboards, PDF viewers, and other components useful for educational applications. Perhaps most importantly, the Open Wonderland toolkit provides the user with an avatar through which s/he interacts with the 3D virtual world.

JADE is the best-known and most widely used platform that supports FIPA (the Foundation for Intelligent Physical Agents) messaging [24]. JADE agents are implemented as Java threads. One important principle in JADE is that communication is transparent, in the sense that the programmer need not be aware of the mechanism used to actually deliver messages. All communication among agents is handled by the JADE runtime environment.

Open Wonderland is a 100% Java open source toolkit for creating collaborative 3D virtual worlds. JADE is also a 100% Java open source toolkit. Since both JADE and Open Wonderland are Java toolkits, integrating these two technologies would not suffer from problems of cross-platform issues, thus they were selected for this research.
During a comprehensive review of both of these technologies we discovered that JADE provides a method to start a JADE agent from a third party application and we felt that it should be possible for Open Wonderland to be that third party application. A careful study of the JADE InProcess example provided the “aha” moment as it became clear that it would be possible to start a JADE agent inside the Open Wonderland world by simply using the JADE provided methods in the `jade.core.Runtime` package. Thus the idea formed for modifying the SAMPLE module to start a JADE agent and have the agent send pertinent information back to Open Wonderland in the form of messages to a TCP port. Open Wonderland scripting provides a powerful method for animating modules without changing any source code and this feature was identified as the means of adding TCP port listening capabilities.

The methodology was conducted in two phases. During phase 1, a proof-of-concept was developed and tested in a local area network (LAN). Phase 2 expanded on the proof-of-concept to integrate a simple game and testing was expanded to include players located in different geographies and connected to the server via the Internet.
CHAPTER IV

ISSUES, CHALLENGES, AND TRENDS

Architecture

Open Wonderland is a client/server architecture [25]. The client components communicate with the server components using network ports allowing for clients to connect over a TCP/IP network, such as the Internet.

Open Wonderland depends heavily on other open source efforts:

- Project Darkstar is a platform for scalable communications and persistence in games. The Wonderland server includes a Darkstar service that manages all client and world state while interacting with Wonderland.

- Glassfish v3 is a scalable, open source web server. The Wonderland server is based on an embedded instance of the Glassfish server. Wonderland web applications include web-based management of the server and worlds, a content repository for hosting all world data, and an integrated single-sign on system used to maintain identity across Wonderland services.

- jMonkeyEngine (JME) is a 3D game engine written entirely in Java. JME provides core graphics APIs, including graphics primitive and shader support. The Wonderland graphics system is based on these core APIs, with some extensions from MTGame to support multithreading.

Developers extend the functionality of the Project Wonderland platform with modules and much of the functionality shipped with the core software are also stored in modules.
and downloaded by the Project Wonderland client on-demand: this modular architecture allows parts of the platform to be updated at any time. Project Wonderland modules consists of a collection of code, artwork and other resources. Each module implements a number of Java classes in three categories:

- **Client Classes** consisting of Cell, CellComponent, CellComponentFactory, CellFactory, CellProperties, CellSubComponent, ComponentProperties, Renderer
- **Server Classes** consisting of CellMO, CellComponentMO, CellSubComponentMO
- **Common Classes** consisting of CellClientState, CellComponentClientState, CellServerState, CellSubComponentClientState, CellSubComponentServerState

The Open Wonderland organization provides a number of tutorials on building modules and includes all code for a sample module.

JADE is a middleware that facilitates the development of multi-agent systems [26]. It includes

- A runtime environment where JADE agents can “live” and that must be active on a given host before one or more agents can be executed on that host.
- A library of classes that programmers have to/can use (directly or by specializing them) to develop their agents.
- A suite of graphical tools that allows administrating and monitoring the activity of running agents.
Each running instance of the JADE runtime environment is called a Container as it can contain several agents. The set of active containers is called a Platform. A single special Main container must always be active in a platform and all other containers register with it as soon as they start. It follows that the first container to start in a platform must be a main container while all other containers must be “normal” (i.e. non-main) containers and must “be told” where to find (host and port) their main container (i.e. the main container to register with).

As shown in Figure 1, the link between these two separate systems lies in the interface. 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, therefore the agent will be started on the local computer. However, the rest of the agents that make up the multi agent system (MAS) can be on other computers in the network. The agent started by the Open Wonderland module will communicate with the rest of the JADE agents using the FIPA specified protocol. One of the agents in the MAS communicates back to the Open Wonderland world by sending messages to a TCP port that has been activated on another, or the same, Open Wonderland module.
Implementation

Implementation was supported by the loan of a SUN Ultra workstation with 8 GB of RAM. Open Solaris proved to be the only operating system that was installable on the SUN Ultra and supported both Open Wonderland and JADE. Open Wonderland v0.5 source code was downloaded from the SVN site http://openwonderland.googlecode.com/svn/trunk. Open Wonderland compiled without issue.

Similarly the JADE framework version 4.0.1 was downloaded from the jade.tilab.com site and installed without issue. Learning the features of both Open Wonderland and JADE consumed a large part of this research work.
With this approach Open Wonderland is virtually untouched with only the customization of a single module and the addition of a script. For the proof-of-concept, the SAMPLE module was selected as it provides a complete 3D module that is easily customizable.

The customized SAMPLE module imports the needed JADE libraries and code is added to start an agent. The scriptable Poster module listens to a TCP port and displays any received message. Only one of the Java classes required modification, that being the Client class Cell. The code snippets below shows the code additions (a complete listing of the SampleCell.java module can be found in Appendix A).

Code snippet showing the import statements to include the jade libraries:

```java
import jade.core.Runtime;
import jade.core.Profile;
import jade.core.ProfileImpl;
import jade.wrapper.*;
import java.net.*;
```

Code snippet to start a JADE agent:

```java
public void setClientState(CellClientState clientState) {
    super.setClientState(clientState);
    shapeType = ((SampleCellClientState)
    clientState).getShapeType();
    if (cellRenderer != null) {
        cellRenderer.updateShape();
        /** jeanne adding some code here */
    }
}
```
System.out.println( "Hello World!" );

System.out.println("version="+System.getProperty("java.version")
});

System.out.println("class
path="+System.getProperty("java.class.path"));
String hostname = "noHost";
try{
 InetAddress addr = InetAddress.getLocalHost();
 hostname = addr.getHostName();
 System.out.println (" The hostname is "+hostname);
} catch (Exception e){
 System.out.println("unknown host ...");}

try
{
 // Get a hold on JADE runtime
 Runtime rt = Runtime.instance();
 // Exit the JVM when there are no more containers
 around
 rt.setCloseVM(true);
 Profile p = new ProfileImpl("aul", 1099,
 "aul",false);

 // Create a new non-main container, connecting to the
default
 // main container (i.e. on host aul, port 1099)
 System.out.println("Launching the agent
 container ..."+p);
AgentContainer ac = rt.createAgentContainer(p);

System.out.println("just after create Container...");
// Wait for 10 seconds
Thread.sleep(10000);
System.out.println("just after thread.sleep...");

Object[] Agentargs = new Object[1];

Agentargs[0] = "MatrixMeetsSkyNet";

System.out.println("Launching the agent PlayerAgent...");

AgentController dummy = ac.createNewAgent(hostname, "PlayerAgent", Agentargs);

//Fire up the agent
System.out.println("Starting up the PlayerAgent...");
dummy.start();

}

try
{
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
Likewise, the JADE framework is also virtually untouched with all modifications limited to a single agent module to include code to send messages to a TCP port. The following code snippet was added to the agent:

```java
// send the message to the open wonderland poster

String Message = "QuizMASter asks <br> " + PlayerList + "<br>" +
TargetQuestionAnswer.substring(0,TargetQuestionAnswer.indexOf("?")+1);

String sendMessage = String.format("%05d",Message.length()+5) + Message;

Socket echoSocket = null;

PrintWriter out = null;

try {
    echoSocket = new Socket("192.168.1.74", 2048);
    out = new PrintWriter(echoSocket.getOutputStream(), true);
} catch (UnknownHostException e) {
    System.err.println("Don't know about host: 192.168.1.74.");
}

try {
    echoSocket = new Socket("192.168.1.74", 2048);
    out = new PrintWriter(echoSocket.getOutputStream(), true);
} catch (IOException e) {
    System.err.println("Couldn't get I/O for " + "the connection to: 192.168.1.74.");
}

System.out.println("Hello World. + sendMessage" + sendMessage);
System.out.println("My name is "+ getLocalName());
```
Proof-of-concept: Integration of Book Trading with Open Wonderland

For the proof-of-concept, we selected a multi agent system from the JADE demonstration package, namely the Book Trading system. The Book Trading system is ideal for this proof-of-concept because it is a well known example of a multi agent system and is fully tested, allowing concentration on the integration of the multi agent system and the 3D virtual world and not on developing and debugging a new multi agent system. In addition, the Book Trading example lends itself to future modification to turn into a general information retrieval and filtering system as discussed in [15-18]. The general idea is that the agent started by the Open Wonderland system would be a “seeker” agent that communicates with a group of “provider” agents to find the real world information that is most relevant to the current situation in the virtual world. For this example, the Book Buyer agent plays the role of the information seeker agent and the Book Seller agents play the role of the information provider agents. Finding that an Open Wonderland module could start a Jade agent is only half the challenge; the other being that the Jade agent then needs to be able to communicate back the information that it has “found”. Our approach is to utilize the Open Wonderland scripting capabilities to have a second module listen on a TCP port, and the JADE agent sends messages to this port. The mechanism for starting the MAS system is simple. Every logged in user gets the open wonderland modules downloaded to their computers. Everyone portrayed as an
Avatar within Open Wonderland will see the modified sample module and will also have the JADE libraries and agent software downloaded to their computer.

The modified sample module is designed such that when its shape is changed, either from a box to sphere or from a sphere to box, the code to start the player agent is executed and the JADE player agent is started with a connection to the main platform, located on a remote host. The result is impressive, as a player agent is started on each and every logged-in users computer and they all connect to the main-platform. The scalability is guaranteed by the JADE architecture.

The results prove the concept that integration of JADE multi agent systems with the Open Wonderland 3D virtual worlds is possible and not overly difficult. The following screen shots illustrate. Figure 2 shows the Open Wonderland Poster module being activated to listen to a TCP port. The SAMPLE module is invoked to start the Book Buyer agent.
Figure 2 Poster module listens to TCP port

Figure 3 shows the Book Buyer agent communicating with the Book Seller agents. For this demo the Book Buyer agent is seeking a book named “MatrixMeetsSkyNet”. A number of Book Seller agents have this book in its catalogue, each with a different price. Book Buyer agent negotiates with each of the Book Seller agents and selects to buy the book at the lowest price. Once the transaction is complete, the Book Buyer agent sends a message to a TCP port to communicate this information back to Open Wonderland where it is displayed by a Poster module.
Figure 3 Book Buyer agent communicates with other agents to find and buy book

Figure 4 shows the result of the completed action.
The main contribution of this proof-of-concept is to show that integrating multi agent systems into a 3D virtual world can be accomplished without the need to make drastic changes to either the 3D virtual world or to the MAS. The fact that both Open Wonderland and JADE are written in Java and that JADE provides runtime creation of agents facilitates this integration.

Case Study: Integration of QuizMASter with Open Wonderland
As a case study we selected to integrate a JADE MAS system previously developed by students of Athabasca University, QuizMASter, which is a JADE Multi agent system that models a TV Quiz show [14]. QuizMASter is composed of two main agents, the QuizMASter agent and the Player agent. The QuizMaster agent controls the game by sending a question to all of the Player agents. Human players control the player agents and provide the answers to the questions. Human players might or might not answer the question in the allotted time, but of those that do answer the question, QuizMASter will declare the Player agent with the best time to be the winner.

The goal for this case study is to use the same integration approach as was used in the proof-of-concept described above, namely to start a JADE agent from an Open Wonderland module and have the MAS system complete some activity and then forward a message back to Open Wonderland. In this case the agent started by Open Wonderland is the Player agent. The QuizMASter agent looks for Player agents and if there are one or more Player agents, QuizMASter sends out a question to all of the Player agents. QuizMASter declares the Player agent with the correct answer and the fastest time to be the winner and sends a message to Open Wonderland announcing the winner. QuizMASter cycles through a short list of questions, each time determining and declaring the winner.

Integrating the QuizMASter MAS did require some changes, in this case the GUI was removed from the QuizMASter agent, since this agent needed to update the poster module inside Open Wonderland. However, no changes were required to the Player agents. Figure 5 shows a number of participating players.
Integrating QuizMASter into Open Wonderland demonstrates that this approach is versatile and could possibly be used to integrate other JADE MAS systems with Open Wonderland. Future work will be based on the knowledge that integrating JADE multi agent systems into Open Wonderland will not be overly difficult and opens the door to developing other types of MAS systems to support the learning environment.
CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

This research developed an approach to integrate multi agent systems with 3D virtual learning environment and developed a working prototype.

Higher Education institutions are incorporating the capabilities allowed by the Internet by providing virtual classrooms and the increasing development of multi agent systems that support the learner. A literature review led to a realization that multi agent systems could alleviate a perceived lack of intelligence in multi user virtual learning environments.

Integrating the disparate technologies of virtual worlds and multi agent systems is desireable. This research explored the challenges of integrating virtual worlds with multi agent systems by devising, implementing and testing an approach using readily available technologies, Open Wonderland and Java Agent Development Environment (JADE). The integration challenge is to find an approach that is not overly difficult and does not require drastic changes to either the 3D virtual world or the multi agent software. This approach to integrate AI in a virtual learning environment allows abstracting from the peculiarities of interacting with the virtual world software, so that the designers may focus on the problem to solve.

Our approach to this challenge was to find a way to integrate a simple multi agent system developed on the Java Agent Development framework (JADE) (jade.tilab.com) with Open Wonderland (OWL), an open source 3D virtual world developed by Sun and now available as Open Source. As an applied research project, this work has resulted in
the development of a customized Open Wonderland module and a working prototype system.

JADE is the best-known and most widely used platform that supports FIPA messaging. One important principle in JADE is that communication is transparent, in the sense that the programmer needs not be aware of the mechanism used to actually deliver messages. Communication is handled by the JADE run-time environment. Open Wonderland is a 100% Java open source toolkit for creating collaborative 3D virtual worlds. JADE is also a 100% Java open source toolkit. Since both JADE and Open Wonderland are Java toolkits, integrating these two technologies would not suffer from problems of cross-platform issues, thus they were selected for this research.

Developers extend the functionality of the Open Wonderland platform with modules. The link between Open Wonderland and JADE developed in this research project 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 therefore the agent is started on the local computer and represents the players avatar.

As a proof of concept, the customized Open Wonderland module was used to integrate a well know JADE example, BookTrading. The Book Trading system is ideal for this proof-of-concept because it is a well known example of a multi agent system and is fully tested, allowing concentration on the integration of the multi agent system and the 3D virtual world and not on developing and debugging a new multi agent system.
Based upon the proof-of-concept, a case study, integrating QuizMASter into Open Wonderland followed. QuizMASter is a pedagogically based multi player game developed in JADE. QuizMASter is a JADE multi agent system that models a TV Quiz show. QuizMASter is composed of two main agents, the QuizMASter agent and the Player agent.

With our integration approach, Open Wonderland is virtually untouched with only the customization of a single module and the addition of a script. Rather than developing an Open Wonderland module from scratch, the SAMPLE module was modified as it provides a complete 3D module that is easily customizable.

This applied research project was conducted in two phases. During phase 1, a proof-of-concept was developed and tested in a local area network (LAN). Phase 2 expanded on the proof-of-concept to integrate QuizMASter and testing was expanded to include players located in different geographies and connected to the server via the Internet.

Integrating QuizMASter with Open Wonderland demonstrates the viability of using Multi Agent Systems to provide the intelligence needed for adoption of 3D virtual world games as an educational tool. Through the implementation of this working prototype we have demonstrated that our approach can bring essential educational components to 3D virtual learning environment to produce effective and engaging immersive educational learning activities. Layering problem based learning, rich reward models, collaboration and
fantasy 3D worlds offer pedagogic values by immersing the learner into a world for an interactive, engaging and fun encounter that encourages active learning and motivates participation.

Integration of Open Wonderland with JADE multi agent systems is both feasible and not overly difficult. This opens the door for devising JADE multi agent systems specifically developed to enhance the learning environment and incorporating these systems into a 3D virtual world in a fairly straightforward manner. This approach can be generalized to allow for the integration of other multi agent systems by simply “starting” a JADE agent. One of the advantages of this approach is that it is a simple interface that is easily extended to support other JADE based multi agent systems. In addition, this approach provides an abstraction from the peculiarities of interacting with a virtual world software, so the designers may focus on the problem to solve.

Future Research

Future research opportunities lie in expanding the functionality of QuizMAster. The Player agent could be improved by linking the person/avatar to a database to keep a permanent record of the player’s achievements and could use this information to provide hints that are appropriate to the players level. The QuizMAster agent would benefit from an expanded repertoire of questions, perhaps even polling questions from a database linked to a particular course. Further enhancements to QuizMAster could allow the questions to be determined based on the achievement level of the currently logged in
players/avatars. In addition, the appearance of the QuizMASter agent could be dynamic and linked to the achievements of the players to show encouragement and add interest. Improving QuizMASter and integrating it with Open Wonderland would, hopefully, lead to an engaging 3D virtual learning environment.
REFERENCES


[9] Molka-Danielsen, Judith; Panichi, Luisa; Deutschmann, Mats;"Reward models for active language learning in 3D virtual worlds “; Information Sciences and Interaction Sciences (ICIS); Digital Object Identifier: 10.1109/ICICIS.2010.5534711 Publication Year: 2010 , Page(s): 40 - 45

- 40


[15] Yin Bing; Lin Zhuying; Du Li; "The applications of Multi-Agent technology in distance education system"; Networking and Digital Society (ICNDS 2010, Digital Object Identifier: 10.1109/ICNDS.2010.5479221 Publication Year: 2010 , Page(s): 429 – 432

[16] Xiao Laisheng; Wang Zhengxia; "Teaching Resource Grid System Model Based on Multi-Agent Distributed Expert System"; Computational Intelligence and Software


[22] de Melo, C.; Prada, R.; Raimundo, G.; Pardal, J.P.; Pinto, H.S.; Paiva, A.;
"Mainstream Games in the Multi-agent Classroom“; Intelligent Agent Technology,
2006. IAT ’06. IEEE/WIC/ACM Digital Object Identifier: 10.1109/IAT.2006.86
Publication Year: 2006 , Page(s): 757 – 761

Technology-Enhanced Distance Learning” doi:10.3991/ijet.v5i3.1388; 2010, iJet
Volume5, Issue 3, Page(s): 12-17


package org.jdesktop.wonderland.modules.sample.client;

import java.util.ResourceBundle;
import org.jdesktop.wonderland.client.cell.Cell;
import org.jdesktop.wonderland.client.cell.Cell.RendererType;
import org.jdesktop.wonderland.client.cell.CellCache;
import org.jdesktop.wonderland.client.cell.CellRenderer;
import org.jdesktop.wonderland.client.cell.annotation.UsesCellComponent;
public class SampleCell extends Cell {

/**
 * Client-side cell for rendering JME content
 *
 * @author jkaplan
 * @author Ronny Standtke <ronny.standtke@fhnw.ch>
 */

private static final ResourceBundle BUNDLE =
ResourceBundle.getBundle("org/jdesktop/wonderland/modules/sample/client/resources/Bundle");

/* The type of shape: BOX or SPHERE */
private String shapeType = null;
private SampleRenderer cellRenderer = null;
@UsesCellComponent
ContextMenuComponent menuComponent;

public SampleCell(CellID cellID, CellCache cellCache) {
    super(cellID, cellCache);
}

/**
* Called when the cell is initially created and any time there is a major
* configuration change. The cell will already be attached to it's parent
* before the initial call of this method
*
* @param clientState
*/
@Override
public void setClientState(CellClientState clientState) {
    super.setClientState(clientState);
shapeType = ((SampleCellClientState)
clientState).getShapeType();

if (cellRenderer != null) {
    cellRenderer.updateShape();
    
    
    /* jeanne adding some code here
    */
    System.out.println( "Hello World!" );

    System.out.println("version="+System.getProperty("java.version"
    ));

    System.out.println("class
path="+System.getProperty("java.class.path"));
    String hostname = "noHost";

    try{
        InetAddress addr = InetAddress.getLocalHost();
        hostname = addr.getHostName();
        System.out.println ( " The hostname is " + hostname);
    } catch (Exception e){
        System.out.println("unknown host ...");
    }

    try
    {
        // Get a hold on JADE runtime
        Runtime rt = Runtime.instance();
        // Exit the JVM when there are no more containers
        rt.setCloseVM(true);
Profile p = new ProfileImpl("au1", 1099,
"au1",false);

// Create a new non-main container, connecting to the
default
// main container (i.e. on host au1, port 1099)
System.out.println("Launching the agent container ..."+p);
AgentContainer ac = rt.createAgentContainer(p);

System.out.println("just after create Container...");
// Wait for 10 seconds
Thread.sleep(10000);
System.out.println("just after thread.sleep...");

Object[] Agentargs = new Object[1];

Agentargs[0] = "MatrixMeetsSkyNet";

System.out.println("Launching the agent PlayerAgent ...");
AgentController dummy = ac.createNewAgent(hostname,
"PlayerAgent", Agentargs);

//Fire up the agent
System.out.println("Starting up the PlayerAgent...");
dummy.start();

}
catch(Exception e)
{
    e.printStackTrace();
}

public String getShapeType() {
    return shapeType;
}

@Override
protected void setStatus(CellStatus status, boolean increasing) {
    super.setStatus(status, increasing);
}
switch (status) {
    case ACTIVE:
        if (increasing) {
            //                    menuComponent.setShowStandardMenuItems(false);
            menuComponent.addContextMenuFactory(
                new SampleContextMenuFactory());
        }
        break;
    case DISK:
        // TODO cleanup
        break;
}

/**
  * Context menu factory for the Sample menu item
  */
class SampleContextMenuFactory implements ContextMenuFactorySPI {

    public ContextMenuItem[] getContextMenuItems(ContextEvent event) {
        return new ContextMenuItem[] { new SimpleContextMenuItem(
            BUNDLE.getString("Sample"), null,
            new SampleContextMenuListener())
        };
    }
}
/**
 * Listener for event when the Sample context menu item is selected
 */

class SampleContextMenuListener implements ContextMenuActionListener {

    public void actionPerformed(ContextMenuItemEvent event) {
        logger.warning("Sample context menu action performed!");
    }
}
