Extending Game Technologies with Virtual Reality and Multi-Agent Systems

A. Barella and C. Carrascosa and V. Botti

Dept.de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Campus de Vera, 46022 - Valencia
{tbarella, carrasco, vbotti}@dsic.upv.es

Abstract
Nowadays, videogames, computer simulations and training are very complex applications because users require higher and higher levels of quality. Artificial Intelligence plays nowadays a fundamental role in videogames and virtual reality applications. Agents interacting with humans must have a credible visual appearance and must maintain user immersion. The most important issues of user’s interest are both realistic visualization and convincing intelligence. This paper presents a system that allows a full integration of two areas which are becoming increasingly important recently: virtual reality (visualization) and intelligent multi-agent systems (artificial intelligence), in a flexible and easily scalable way. This system extends the technological possibilities of the videogames.

Introduction
Computer game development covers many different areas, such as graphics, artificial intelligence and network communications. Nowadays, players demand more sophisticated and credible computer-controlled opponents, but results are sometimes unsatisfactory. This is due to the need of real-time processing constraints to reach an acceptable feeling of immersion for the user. Sometimes, these high computational costs are solved by using dedicated hardware, such as graphic cards. However, in the case of artificial intelligence, there is no specific hardware developed to help programmers and users, and outcomes are sometimes poor. The development of systems, tools and development environments can play an important role in the progress of this field. Thus, the combination of artificial intelligence techniques and virtual reality (or virtual environments) has given birth to the field of intelligent virtual environments (IVEs). An IVE is a virtual environment simulating a physical (or real) world, inhabited by autonomous intelligent entities (Luck 2000). These entities have to interact in / with the virtual environment as if they were real entities in the real world. In addition, entities and the virtual environment have to be shown to users in an appropriate way. There are some other typical problems of an IVE that we have to solve, as explained in (Bierbaum 2001). Independence of the underlying technologies is one of the principal requirements to support cross-platform application development (both hardware and operating system). A virtual reality system has to allow user interactions: not only displaying images, but using specific devices such as trackers, gloves, etc. That is, a device abstraction layer is necessary for user interaction. Furthermore, synchronization is a critical issue, specifically both data and image synchronization. Every visualization module must have the same data to create a scene graph, in order to maintain coherence. In case of multiple displays, every display must show the same order of image generated.

The main purpose of this work is to integrate Artificial Intelligence and Virtual Environments. One of the most interesting AI technique for this purpose, due to its features (scalable, autonomous, proactive, reconfigurable, agile, etc…), is the Multi-Agent Systems (MAS) approach.

Background and Motivation
Using 3D videogame engines to create intelligent virtual environments has been a choice used widely, because of the commercial success and the high level of photorealism that is achieved by current technology (Andreoli 2005). The usual approach for applications to commercial game engines is to integrate an agent as a bot player in the game. Quake and Unreal Tournament are the most prefered because of this facility to create and modify bots. These bots can be implemented using any AI technique; for example, M. van Lent et al. (Laird 2000) use SOAR as an inference engine and B. Gorman et al. (Gorman 2005), use MATLAB to control the bot. Considering a bot as an agent, agent techniques can also be applied; for example E. Norling (Norling 2003) uses JACK for a BDI agent bot.

Other choice is to use simulation packages. There are some powerful packages for the simulation of decentralized systems. One of the most popular ones is Swarm(Minar 1996), but it doesn’t provide a framework for 3D simulations or visualizations. Another package is Breve(Klein 2002), which is an integrated simulation environment for the implementation of de-
centralized systems and artificial life simulations in 3D worlds. In Breve, simulations are written in an interpreted language (“steve”) and, therefore, they are integrated in this application for execution.

A more laborious choice is to create a framework from the scratch. A wide range of approaches may be found in literature following this choice. For instance, DIVA (Vosinakis 1999) (and its evolution, VITAL (Anastasakis 2001)) is developed to use a Prolog-based engine for deliberation. Another approach is to develop a virtual environment with only one agent acting as a wizard agent for training and educational purposes such as STEVE (Rickel 1997). It can be found some research in the field of crowd simulations in virtual environments (Ulicny 2001)(Musse 2001). Or even there are approaches that use AI to give human-like expressiveness or movement to their virtual characters (Thalmann 1999).

In these approaches, artificial intelligence and graphics use to be embedded. Thus, they are ad-hoc applications that are not very extensible nor scalable.

This paper presents a case study of a multi-agent system applied to a virtual environment, using JGOMAS (Barella 2006) (as acronym for Game Oriented Multi-Agent System based on JADE) as framework for IVEs. Specifically, agents playing a capture the flag game in a 3D virtual environment while are visualized on different systems. Thus, the purpose of this case study is to show the flexibility of this approach, which allows both distributed intelligence calculation and visualization of the IVE on different devices, such as Head Mounted Displays, CAVE™, and Powerwall™.

**Description of the System**

According to this approach, both systems, virtual reality and multi-agent, have to work separately, in an independent way in order to create a flexible and versatile framework. Thus, there is a clear separation (figure 1) of the intelligence part (where reasoning is computed) from the visualization part (where the results of reasoning are displayed), as proposed by (Ulicny 2001).

On the one hand, the artificial intelligence problem has been put as a distributed system, breaking the problem down into subproblems. Agents are used as subproblem-solvers. Thus, an agent has only part-information knowledge, that is, it cannot obtain direct access to the complete virtual environment state. Using its reduced knowledge, each agent can figure out what it needs to do in order to reach its designed objectives. This means that nobody has to tell the agent explicitly what to do at a given moment. In this way, a multi-agent system is a system that consists of a number of agents, that communicate messages through a computer network. These agents will cooperate, coordinate, and negotiate with each other, trying to achieve their own goals and common objectives, as a result of emergent behaviour(Wooldridge 1995).

On the other hand, a visualization module is necessary in order to show to users what is happening in the virtual world. This module should cover from stand-alone users at home to complex virtual reality systems, such as a CAVE™.

The framework is, at present time, at its first stage. Thus, a capture the flag game is proposed as the kind of social interaction to simulate, where agents are grouped into two teams (allies and axis). Allies must go to the axis base, capture the flag, and bring it to their base, in order to win the game. Axis agents must defend their flag against the allies and, if the flag is captured, they must return it to their base. There is a time limit for the allies to bring the flag to their base. If the time limit expires, the axis team wins the game.

This framework can be used as a simulator for validating the coordination, communication, and learning algorithms in the field of multi-agent systems (or artificial intelligence in general). To do this, JGOMAS provides an API that allows designers to add their own code modifications (mods), to improve the intelligence of the player agents.

**JGOMAS Architecture**

The framework is composed of three subsystems, as shown in figure 2:

- a multi-agent platform,
- a set of agents (conforming a multi-agent system),
- a visualization module,

**Multi-Agent Platform**

JGOMAS uses JADE (Bellifemine 2001) as multi-agent platform. JADE (Java Agent DEvelopment Framework) is a software framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA (FIPA 2002) specifications1.

1FIPA is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies.
The reason of JGOMAS using JADE as a multi-agent platform is because JADE has become a de facto standard in the multi-agent community. Thus, JGOMAS takes advantage of all the resources that JADE offers: behaviour mechanisms, message passing, FIPA interaction protocols, etc., in compliance with the FIPA specifications (a collection of IEEE standards which are intended to promote the inter-operation of heterogeneous agents and the services that they can represent).

Therefore, JGOMAS user has only to implement the intelligence of his agents, avoiding wasting time in low-level technical issues as, for example, inter-agent communication.

Multi-Agent System

JGOMAS’ multi-agent system can be viewed as an abstraction upper layer over JADE (Barella 2007). Two main classes of agents are defined: simulation controller and inhabitant (player) agents.

There is only one simulation controller running during a game. It can be viewed as a wrapper of the virtual environment, because it translates all player agents’ actions to the virtual world. This is the way in which this agent synchronizes the virtual environment’s data, maintaining the consistency throughout the game at any time. In addition, simulation controller is the interface for visualization module, because it sends the game state to graphic viewer clients, acting as a server.

The other agents are inhabitant agents simulating players in the virtual scenario. These agents can carry out basic actions, such as look, move, etc. The way an inhabitant agents achieves a goal is carrying out tasks that changes the virtual world state. Then, these actions are encapsulated into messages to be sent to simulation controller using the communication mechanisms of JADE platform.

Next, simulation controller performs those actions in the virtual world and, finally, this agent informs inhabitant agents affected by the consequences of the actions (simulation controller generates events for inhabitant agents involved in those changes).

Visualization Module

As mentioned above, one of the main goals is that artificial intelligence and virtual reality systems have to work independently (it is possible to throw the JGOMAS multi-agent system even if there are no graphic viewers connected). In order to make it easy to users, a basic graphic viewer has been implemented. Render Engine is the graphic viewer application developed ad hoc to display the 3D agents, objects, and the scenario in JGOMAS, and it has been designed as an external module (not as an agent) and written in C++, using the graphic library OpenSceneGraph (OSG). The OpenSceneGraph is an open source high performance 3D graphics toolkit, used by application developers in fields such as visual simulation, games, virtual reality, scientific visualization and modelling.

Render Engine is an important part of the framework, but JGOMAS is not forced to use it, because other graphic engines (both commercial and open source) could be used, for example, to utilize an existing one or to get better image rendering. This is possible because the visualization module in JGOMAS framework is independent of the artificial intelligence module (multi-agent system).

Although Render Engine can be used both by stand-alone users and head mounted display users (stereoscopic view), it is difficult to synchronize multiple displays to perform a virtual reality system, such as a CAVE. The CAVE is a projection-based VR system that provides real-time head-tracked perspective with a large field of view, interactive control, and stereo display. It use to be a "cube" with images projected onto three walls and the floor. To avoid that effort, Render Engine has been extended to use VRJuggler (VRJuggler) as middle-ware to use a complete virtual reality system.

Virtual Reality System VR Juggler is a collection of technologies which provide the tools necessary for virtual reality applications development. VR Juggler allows a user to run an application on almost any virtual reality system. It acts as "glue" between all the other Juggler components. From this, VR Juggler provides a virtual platform for virtual reality application development.

The flexibility of VR Juggler allows applications to execute in many virtual reality system configurations including desktop virtual reality, Head Mounted Displays, CAVE-like devices, and Powerwall-like devices.

Figure 3 shows the way Render Engine uses VRJuggler. The kernel interface provides hardware abstraction, and the draw manager (specifically we have used...
the OSG draw manager) provides the abstraction for the graphics API. This way we get device abstraction and operating system independence. In addition, VR-Juggler provides the ability to run multiple applications simultaneously (image synchronization). As mentioned above, the simulation controller is in charge of data synchronization. In this way, synchronization is solved.

Example for using JGOMAS

In this section, an example for using JGOMAS is shown, where it is noticed how the virtual reality system and the multi-agent system work independently from each other. In this example, several flexible and versatile configurations has been designed. These configurations cover from stand-alone users at home to complex virtual reality systems, as shown in figure 4.

Since JGOMAS is a distributed system, it is not necessary to use a high-performance computer to support the execution of several agents. In this example, JGOMAS’ multi-agent system was executed in a cluster of computers. This way, cheap computers connected through a high-speed network were used in order to get the necessary performance.

Once JGOMAS’ multi-agent system is being executed in the cluster, many visualization modules can be used at the same time:

- **stand-alone system**: users can be playing or supervising the game at anywhere.
- **head mounted display system**: users are watching the game in stereoscopic mode.
- **Powerwall™ system**: a group of users viewing the game (perhaps in stereoscopic mode) in a grid of displays which are composing a macro-display (that is, a Powerwall™).
- **CAVE™ system**: users are each one immersed in a CAVE™ to get an astounding experience in virtual reality systems.

Figure 5 shows an execution of JGOMAS. A hand made Powerwall™ with four monitors was composed, and images are rendered by two computers forming a cluster. VR-Juggler is used to synchronize the images displayed. At the same time, the CAVE™ system was connected to the current JGOMAS game. Figure 6 shows a picture of a person using the CAVE™.

Conclusions

This paper presents a multi-agent framework based on JADE for virtual environments that has been designed to study a full integration of artificial intelligence (specifically multi-agent systems) and real-time graphic applications (specifically virtual reality systems) that can be applied to videogames.

In this way, JGOMAS is designed to have intelligent computer-controlled elements in 3D virtual environments, such as games, training and simulation applications. This means that JGOMAS is not either an artificial intelligence engine or a graphic engine. JGOMAS is simply a framework that allows to have agents in a virtual environment.

Due to the easy-to-use JGOMAS API, and the potentially attractive working scenario, it seems appropriate to use this system as a toolkit to introduce multi-agent systems and artificial intelligence techniques to senior computer engineering students.

JGOMAS package for download is available at http://jgomas.gti-ia.dsic.upv.es

Acknowledgements

This work is partially supported by the PAID-06-07/3191, TIN2006-14630-C03-01 projects and CONSOLIDER-INGENIO 2010 under grant CSD2007-00022. Furthermore, we thank Área de Sistemas de Información y Comunicaciones (ASIC) at Universidad Politécnica de Valencia for providing the CAVE and its infrastructures.

References


