JGOMAS: A new approach to AI teaching

Antonio Barella, Soledad Valero, and Carlos Carrascosa

Abstract— **This paper presents a new environment for teaching (in a practical way) AI subjects. The main purpose of such environment is to make more appealing AI techniques to the students along with facilitating the approach of these students to toolkits that are currently and widely used in research and in development. This new environment is composed of a toolkit for developing and executing agents, called JGOMAS, and a web-server dedicated to it where students may access to different documentation and information and interact with teachers. Lastly, it is presented a real case of application of this environment to the practical work of an advanced AI subject.**

Index Terms— **MAS, Agents, Virtual Environments, ...**

I. Introduction

THE advanced Artificial Intelligence (AI) courses cover core concepts and techniques needed to develop intelligent systems, such as planing, case base reasoning, decision theory, machine learning, agents and multi agent systems. Moreover, not only it is expected than students know and master basic concepts and techniques, but also they use new available tools to solve practical problems.

Furthermore, the employed tools must not be difficult to use, since they can cause rejection in the students, making the learning of the desired techniques more difficult. Otherwise, attractive tools are needed in order to motivate students to work hard in their practical problems, so they can reach the demanded requirements and provide additional features too.

In this way, a new toolkit to be used for educational purposes on AI subjects is presented in this paper. This toolkit has two components, the first one is JGOMAS, a Game-Oriented Multi-Agent System based on JADE [1] . Specifically, JGOMAS is an environment to develop and run intelligent agents over simulated 3D worlds. Therefore, JGOMAS allows to run agents in different teams, which compete to achieve their own and team objectives. Those two teams compete in a capture-the-flag-like game, where one team defends its flag against the other team, which tries to capture it. The agents can play different roles: soldier, medic and field operations. Moreover, agents need to cooperate with their team-mates in order to achieve their objectives. For example, a medic can deliver medic packs to a soldier mate with a low level of health who sent a ”call for medic” request.

The second component of this educational framework is formed by a Web, where is possible to get last versions and updates of JGOMAS, documentation, user manuals, usage examples, news, etc. So, this framework allows students to carry out their projects from everywhere, and not only in practical rooms.

Finally, besides JGOMAS was created for educational purposes, it can be used in different scopes, for example as a testbed for AI techniques: cooperation, coordination, learning, etc. Another possibility could be the study of the complete integration between Multi-agent Systems and Virtual Reality.

In the following sections, this new environment for teaching (in a practical way) AI subjects is detailed in its two components, JGOMAS toolkit and web-server. Lastly, it is presented a real case of application of this environment to the practical work of an advanced AI subject. But before these descriptions, some basic concepts and definitions regarding multi-agent systems are provided in the next section.

II. Multi-agent systems' concepts

Traditionally, complex systems are solved by huge monolithic applications, which have the required information for a valid solution. To handle all this information implies high computational costs, and therefore, increased time consumption. Because of time constraints, the results obtained could be a poor solution.

However, if the original problem can be decomposed into subproblems, then the initial complexity of the problem can be reduced. This is the basis of the distributed system approach, which is a powerful technique for solving complex systems.

Artificial intelligence in computer games could be tackled as a distributed system: nodes dividing and sharing information to reach a solution. These nodes will have different tasks, and they will work together according to their partial information, finally achieving a desired goal. This is the way a multi-agent system works, as Wooldridge's definition of agent and multi-agent system states: “Multi-agent systems are systems composed of multiple interacting computing elements, known as *agents*. Agents are computer systems with two important capabilities. First, they are at least to some extent capable of *autonomous action* - of deciding *for themselves* what they need to satisfy their design objectives. Second, they are capable of interacting with other agents - not simply by exchanging data, but by engaging in analogues of the kind of social activity that we all engage in every day of our lives: cooperation, coordination, negotiation, and the like.” [2]. Each agent can figure out what it needs to do in order to reach its design objectives. This means that nobody has to tell the agent explicitly what to do at a given moment. On the other hand, a MAS is a system that consists of a number of agents, communicating messages through a computer network. These agents will cooperate, coordinate, and negotiate with each other, trying to achieve their own goals and collective objectives, as a result of emergent behaviour[3].

A MAS platform can be viewed as an abstraction of an Operative System for agents. This platform provides mechanisms for: executing agents, communication between agents, controlling the life cycle of each agent, and so on. Agents and their system would be in an upper layer of abstraction,

and developers could apply those appropriate technologies for each problem (for example, a Neural Network, or a Finite State Machine, etc...). The complete system can be viewed as a multi-layer architecture as shown in Fig. 1.

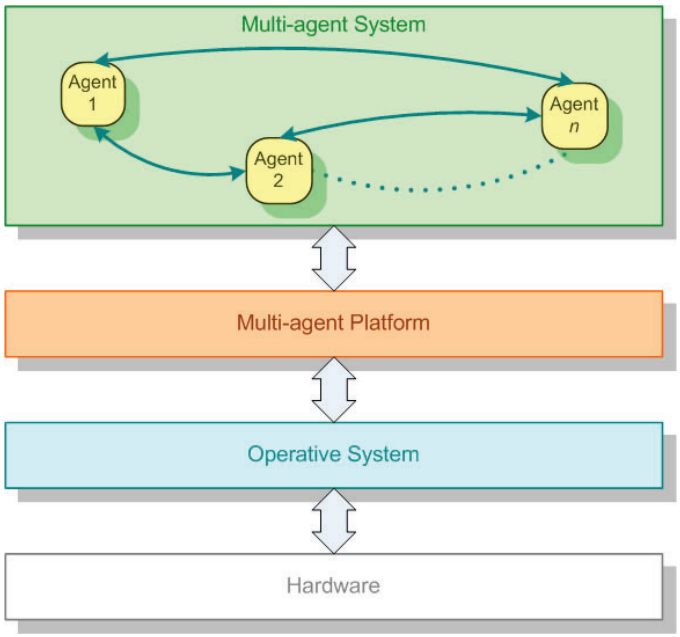

Fig. 1. Multi-layer architecture to develop a MAS

In the literature we can find several platforms for the implementation of multi-agent systems [4]. Some of them follow the standard of the *Foundation for Intelligent Physical Agents*[1] (FIPA) abstract architecture [5], while others are based on independent developments. FIPA tries to support both agent-level and platform-level interoperability through a comprehensive set of specifications. At the platform level, FIPA specifies an abstract architecture for agent platform and services, a message transport service to enable agent communication over several network environments, and agent management to control agents' evolution. At the agent level, FIPA mainly deals with *Agent Communication Language* (ACL), interaction protocols, message content and message ontology issues.

The FIPA Abstract Architecture [5] (Fig. 2) defines, at an abstract level, how two agents can locate and communicate with each other by registering themselves and exchanging messages. Concretely, FIPA proposes that an agent platform must contain at least the following mandatory roles:

- Agent Management System (AMS): controls agents access and use of the platform. It keeps information of all the agents within the platform including their identifiers and transport addresses. It gives a white pages service to the agents connected to the platform.
- Directory Facilitator (DF): provides yellow page services to the agent platform. Agents within the platform can register their services to the directory facilitator and can query it in order to know the services offered by other agents.
- Agent Communication channel (ACC): is the default communication method which offers a reliable, orderly and accurate message transport service. The ACC provides agent communication inside and outside the platform.

[1]FIPA is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies

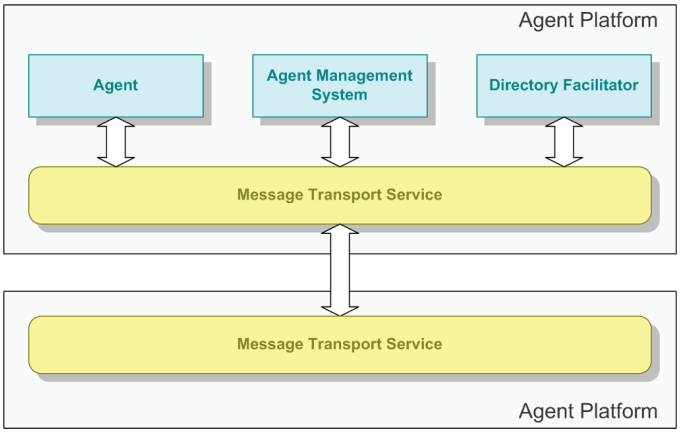

Fig. 2. FIPA abstract architecture

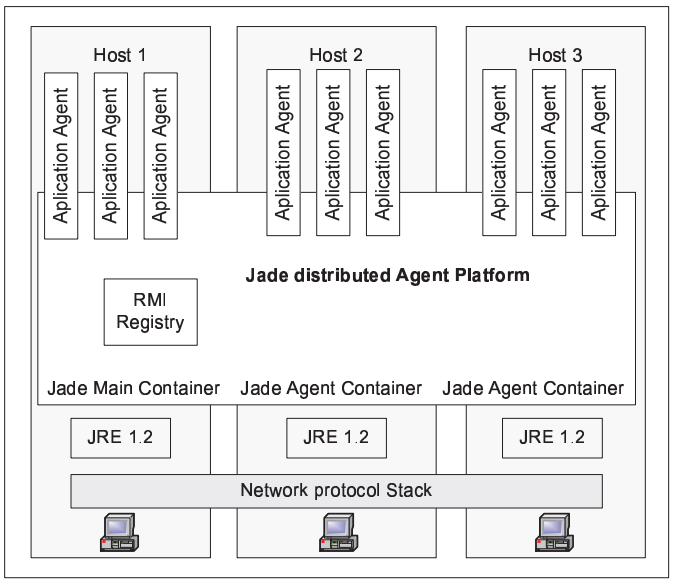

Fig. 3. JADE architecture

The JADE platform [6][1] is a software development framework which contains a FIPA compliant agent platform developed in JAVA [7] and a package to develop JAVA agents. The platform (figure 3) is perceived from outside as a single entity but it can be divided in different agent containers, each one executing a Java Virtual Machine and it can be distributed in different hosts. When the platform is launched the FIPA AMS, the DF and the ACC are started. Each agent in JADE is executed within an agent container and it must have a global unique identifier within the platform. One of the agent container is called the main container, and it contains the AMS and the DF.

III. Related Work

A. Competitions

1) RoboCup: It is an international research and education initiative [8][9][10][11][12]. Its goal is to foster artificial intelligence and robotics research by providing a standard problem where a wide range of technologies can be examined and integrated. In July 1997, the first official conference and games were held in Nagoya, Japan. The following annual events attracted many participants and spectators (Paris, Stockholm, Melbourne, Seattle,Fukuoka-Busan, Padua, Lisbon, Osaka, Bremen). Nearly 300 teams from all continents competed in the Robocup 2007, hosted at the Georgia Institute of Technology, Atlanta. Approximately 1700 students and faculties from leading universities, high schools, middle schools and elementary schools competed in the different events.

RoboCup chose to use soccer game as a primary domain, aiming at innovations to be applied for socially significant problems and industries. Moreover, in the last competitions was introduce another socially significant domain: disaster rescue in large scale disasters. Both domains are represented in two different cups: RoboCupSoccer and RoboCupRescue.

1) RoboCupSoccer. The game of soccer was the original motivation for RoboCup. Besides being a popular worldwide sport, therefore an appropriate medium to attract people to an event, it contains a significant set of challenges for researchers. In order for a robot team to actually perform a soccer game, various technologies must be incorporated including: design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning, robotics, and sensor-fusion. RoboCup is a task for a team of multiple fast-moving robots under a dynamic environment. RoboCup also offers a software platform for research on the software aspects of RoboCup. RoboCupSoccer is divided into the following leagues:
   - *Simulation league.* Independently moving software agents play soccer on a virtual field inside a computer. Matches have 5-minute halves. This is one of the oldest fleet in RoboCupSoccer. There are no actual robots in this league but spectators can watch the action on a large screen, which looks like a giant computer game. Many computers are networked together in order for this competition to take place.
   - *Small-size robot league (f-180).* Small robots of no more than 18 cm in diameter play soccer with an orange golf ball in teams of up to 5 robots on a field with the size of bigger than a ping-pong table. Matches have 10-minute halves. This league focuses on the issues of multi-agent cooperation with a hybrid centralized/distributed system. Relevant objects are marked by color and colored coded markers which are on the top of the robots. Commands are transmitted to the team robots by wireless communication.Some robots play with on-board vision and therefore require no overhead camera. No external intervention of humans is allowed, with the exception of the insert or removal of robots in/from the field.
   - *Middle-size robot league (f-2000).* Middle-sized robots of no more than 50 cm diameter play soccer in teams of up to 4 robots with an orange soccer ball on a field the size of 12x8 meters. Matches are divided in 15-minute halves. All sensors are on-board. Communication among robots is supported on wireless communications. No external intervention by humans is allowed, except to insert or remove robots in/from the field.
   - *Four-legged robot league.* Teams of 4 four-legged entertainment robots (SONY's AIBO) with all sensors on-board, play soccer on a 3 x 5 meters field. Matches have 10-minute halves. Relevant objects are marked by colors. The robots use wireless networking to communicate with each other and with the game referee. Challenges include vision, self-localization, planning, and multi-agent coordination. No external intervention by humans is allowed, except to insert or remove robots in/from the field.
   - *Humanoid league.* This league was introduced in 2002 and the robots will have their third appearance, most are constructed by the participating teams. Some commercially available robots also participate. Biped autonomous humanoid robots play in "penalty kick" and " 2 vs. 2" matches and "Technical Challenges". This league has two subcategories: Kid-size (<60cm) and Teen-size. The humanoid soccer robots are fully autonomous. Help from outside the file is not permitted while the ball is in play. One particular challenge in the Humanoid League is maintaining the balance while the robots are walking and kicking the ball. If the robots go to the ground, they must get up by themselves again.
2) RoboCupRescue. Disaster rescue is one of the most serious issues involving very large numbers of heterogeneous agents in a hostile environment. The intention of the RoboCupRescue project is to promote research and development in this significant domain by involving multi-agent team work coordination, physical robotic agents for search and rescue, information infrastructures, personal digital assistants, standard simulator and decision support systems, evaluation benchmarks for rescue strategies and robotic systems that are all integrated into a comprehensive system in future. RoboCupRescue is divided into two leagues:
   - *Simulation League.* The main purpose of the RoboCupRescue Simulation Project is to provide emergency decision support through the integration of disaster information, prediction, planning, and human interface. Heterogeneous intelligent agents conduct search and rescue activities in this virtual disaster world. This problem introduces researchers to advanced and interdisciplinary research themes. The league is composed of three competitions: the virtual robot competition, the agent competition, and

the infrastructure competition.
In a virtual robot competition run, a team of simulated robots has to explore, map and clear a block-sized disaster area, featuring both carefully modeled indoor / outdoor environments. Robots and sensors used in this competition closely mirror platform and devices currently used in physical robots.
The Agent competition involves scoring competing agent coordination algorithms on different maps of the RobocupRescue simulation platform. The challenge in this case involves developing coordination algorithms that will enable teams of Ambulances, Police forces, and Fire Brigades to save as many civilians as possible and extinguish fires in a city where an earthquake has just happened.
The Infrastructure competition involves evaluating tools and simulators developed for the simulation platform and for simulating disaster management problems in general. Here, the intent is to build up realistic simulators and tools that could be used to enhance the basic RobocupRescue simulator and expand upon it.

- *Robot League.* Robots explore a specially const ructed disaster site, including mannequins with various signs of life, such as waving hands, shouting noises and heat, hidden amongst stairs, platforms and building rubble. The robots, some under human control, must find and approach the victims, identify their signs of life and produce a map of the site showing where the victims are located. The aim is to provide human rescuers with enough information to safely perform a rescue. Each team is scored based on the quality of its maps, the accuracy of the victim information and the number of victims found.

2) Agent Reputation and Trust Testbed (ART): It is a multi agent game designed for trust issues [13] [14] [15]. The testbed simulates an open multi agent system where ART appraisal agents communicate with the purpose of increasing the accuracy of their appraisals.

The testbed operates in two modes: competition and experimentation [13]. In competition mode, the testbed compares different researchers' strategies as they act in combination. Each participant controls a single agent, which works in competition against every other agent in the system. The competition consist of several game sessions; the winner is selected by averaging results over all sessions, to even out possibly unfair game settings. The duration of each session is randomly determined by the simulation and is unknown to each agent to prevent agents from exploiting end-game strategies. It is possible to include 'dummy' agents in the competition, whose strategies are unknown to the other competitors, in order to increase the number of players. Dummy agents compete in the game throughout the duration of the competition.

In experimentation mode, users my choose to allow agents (including dummy agents) to enter or leave the game as desired. The users also has the flexibility of complete control over all experiment parameters. Result may be compared for benchmarking purposes, since the testbed provides a well-established environment for easily repeatable experimentation.

In the ART appraisal domain, agents function as painting appraisers with varying levels of expertise in different artistic eras. Clients request appraisals for paintings from different eras; if an appraising agent does not have the expertise to complete the appraisal, it can request opinions from other appraiser agents. Appraisers receive more clients, and thus more profit, for producing more accurate appraisals.

A successful agent in the ART testbed must take both integrity and competence into account. Integrity comes from the knowledge of how well other agents self-report their capabilities when they have been queried for an opinion. Competence is important since each agent has its own level of knowledge for various types ad artwork. Moreover, an agent in the ART framework is capable of using intuition, experience and hearsay. Intuition could be used to consider how agents' behaviours might change over the course of the game. Experience and hearsay are the main mechanisms supported by the testbed's design.

3) The Trading Agent Competition (TAC): The Trading Agent Competition (TAC) is an international forum [16] designed to promote and encourage high quality research into the trading agent problem. Michael Wellman led the team that organized the first years competitions, based on the "travel agent scenario" (called TAC Classic). From TAC 2002, Swedish Institute of Computer Science (SICS) organized the competition together with the TAC community. Since 2003 the competition also has a supply chain scenario, based on a PC manufacturer scenario, created by Carnegie Mellon University and SICS (called TAC SCM). Moreover, many universities are using the TAC infrastructure for education of students in e-commerce and artificial intelligence.

- *TAC Classic*. In the TAC shopping game [16][17], each "agent" (an entrant to the competition) is a travel agent, with the goal of assembling travel packages (from TAC-town to Tampa, during a notional 5-day period). Each agent is acting on behalf of eight clients, who express their preferences for various aspects of the trip. The objective of the travel agent is to maximize the total satisfaction of its clients (the sum of the client utilities). Travel packages consist of the following: A round-trip flight; a hotel reservation; and Tickets to some entertainment events (alligator wrestling, amusement park, museum).
There are obvious interdependencies, as the traveler needs a hotel for every night between arrival and departure of the flight, and can attend entertainment events only during that interval. In addition, the clients have individual preferences over which days they are in Tampa, the type of hotel, and which entertainment they want. All three types of goods (flights, hotels, entertainment) are traded in separate markets with different rules.
A run of the game is called an instance. Several instances of the game are played during each round of the competition in order to evaluate each agent's average performance and to smooth the variations in client preferences. The game occurs during nine minutes.
At the end of the game, the travel agent holds several

plane tickets, hotel rooms and event tickets. If it ends the game holding negative balances of any entertainment tickets (because it sold tickets it did not have), it is assessed a penalty of 200 for each ticket owed. The TAC scorer allocates the agent's travel goods to its individual clients in order to construct feasible trips. Value for a particular allocation is the sum of the individual client utilities. The agent's final score is the value of the allocation of the goods to clients, minus the travel agent's expenses, minus a penalty for negative entertainment balances (if applicable). The scorer attempts to construct an optimal allocation, and usually succeeds or comes very close.

- *TAC SCM*. It was designed to capture many of the challenges involved in supporting dynamic supply chain practices, while keeping the rules of the game simple enough to entice a large number of competitors to submit entries [18][19]. A TAC SCM game consists of a number of days or rounds where six personal computer (PC) assembly agents compete for customer orders and for procurement of a variety of components. Each day, customers issue requests for quotes and select from quotes submitted by the agents, based on delivery dates and prices. The agents are limited by the capacity of their assembly lines and have to procure components from a set of eight suppliers. Four types of components are required to build a PC: CPUs, Motherboards, Memory, and Disk drives. Each component type is available in multiple versions. Customer demand comes in the form of requests for quotes for different types of PCs, each requiring a different combination of components.
  A game begins when one or more agents connect to a game server. The server simulates the suppliers and customers, and provides banking, production, and warehousing services to the individual agents. The game continues for a fixed number of simulated days. At the end of a game, the agent with the highest sum of money in the bank is declared the winner.
  The game is representative of a broad range of supply chain situations. It is challenging in that it requires agents to concurrently compete in multiple markets (markets for different components on the supply side and markets for different products on the customer side) with interdependencies and incomplete information. It allows agents to specialize in particular types of products, stocking up components that are in low supply, etc. To succeed, agents will have to demonstrate their ability to react to variations in customer demand and availability of supplies, as well as adapt to the strategies adopted by other competing agents.

B. Intelligent Virtual Environments (IVEs)

The combination of artificial intelligence techniques and virtual reality (or virtual environments) has given birth to the field of *intelligent virtual environments* (IVEs) [20].

An IVE is a virtual environment simulating a physical (or real) world, inhabited by autonomous intelligent entities. 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.

1) Commercial Game Engines: 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 [21]. 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. [22] use SOAR [23][24][25] as an inference engine and B. Gorman et al. [26], use MATLAB®[27] to control the bot. Considering a *bot* as an agent, agent techniques can also be applied; for example E. Norling [28] uses JACK™[29][30] for a BDI agent bot.

2) Simulators: 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[31][32], but it does not provide a framework for 3D simulations or visualizations. Another package is Breve[33], which is an integrated simulation environment for the implementation of decentralized 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.

3) From the scratch: 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 [34] (and its evolution, VITAL [35]) 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* [36]. It can be found some research in the field of crowd simulations in virtual environments, as for example [37][38]. Moreover, there are approaches that use AI to give human-like expressiveness or movement to their virtual characters [39][40].

C. Discussion

In the above IVEs approaches, artificial intelligence and graphics use to be embedded. In this way, they are *ad-hoc* applications that are not very extensible nor scalable. This work pretends to establish a framework that integrates a MAS and a Virtual Environment for developing Intelligent Virtual Environments, so that a designer will not to be worry about the low-level management and interaction with the virtual world. This will allow him/her to focus in the implementation of the Artificial Intelligence peculiarities of his agents, that is, in their deliberation process whatever technique he/she will use for them (neural networks, FSM, rules, etc…).

JGOMAS (Game-Oriented Multi-Agent System, based on JADE) has emerged as a test platform to study a full integration of multi-agent system and real-time graphic applications. Therefore, JGOMAS is a multi-agent framework designed to

have intelligent computer-controlled elements in 3D virtual environments, such as games, virtual reality systems, etc.

Unlike some packages mentioned above, JGOMAS cannot be considered as a (3D graphic) simulator where intelligent elements are integrated into the package for execution. In JGOMAS the main difference is that there is a module for artificial intelligence (specifically MAS) and another one for visualization, and they work independently of each other, that is, Intelligence and Visualization parts are separated. Moreover, this AI module is a distributed system by itself, where each one of the elements of the system (each agent) has some independence in a fashionable way as a player in a multi-player game.

From a teaching point of view, it is possible to launch two different lines of teaching, which are both *artificial intelligence* and *graphics*, in different subjects.

IV. JGOMAS Toolkit

As it has been stated before, there are different reasons to create a new environment for teaching artificial intelligence in a practical way. One of this reasons is to make the artificial intelligence techniques more attractive to students, increasing the interaction between teachers and students. Moreover, it is intended that students will be able to make or to extend further their practical work from anywhere, not only from the educational laboratories. These reasons lead to both the creation of a new toolkit for students to develop their practical work, and the launch of a web server dedicated to this toolkit that allows to spread documentation, new versions, examples and any general news interesting to the student about it.

A. Game Description: Capture the Flag (CTF)

JGOMAS has a framework allowing to develop and execute agents over 3D simulated worlds. In fact, these agents will belong to one of two different teams that are competing in a “Capture the Flag”-like game. In this kind of games, two teams (red and blue, allies and axis) must compete to capture the opponent’s flag. This game modality has become an standard included in almost all multiplayer games appeared since Quake [41].

It is very easy and intuitive to apply multi-agent system to this type of games, because each soldier may be seen as an agent. Moreover, agents in a team must cooperate among them to get the team’s objective. In this way, they compete with the other team.

In fact, it is not odd to find applications of agent technology to game field, in general (i.e. the board game developed by S. Offerman et al. [42]) and to the Capture the Flag, in particular. In this last case, it can be found ad-hoc applications such as the CTF Project [43], or applications to commercial games [44] such as the above mentioned Quake.

So, a CTF game is proposed as the kind of social interaction to simulate, where the agents group in two teams (allies and axis). On one hand, allies agents must go to axis base, capture the flag and take it to their base, in which case allied team win the game. On the other hand, axis agents defend their flag against the other team and, if the flag is captured, they must return it to their base. There is a limit time for allies to bring the flag to their base. If time expires, axis team win the game.

Fig. 4. JGOMAS’ web main page

Of course, it is necessary an additional module which will display the 3D virtual environment: agents, objects and scenario.

B. Web

A web for JGOMAS environment was created in order to provide an accesible way from anywhere, where students were able to get the last source code version, contact with teacher or look up reference manuals.

Nowadays, a short description about the JGOMAS toolkit is accesible from this Web. Moreover, how JGOMAS works and over which components it is based on are explained. From the ”DOCUMENTATION” section, users can access to on-line manuals, as well as it is possible download its pdf version. From this section, access to all didactic material is also possible, as the slides used in class, for example. Similarly, a ”DOWNLOAD” section is available where users can obtain different versions of the JGOMAS toolkit (such as linux or windows platforms, etc.), as well as they can read a short description that explains how to run the environment.

On the other hand, relevant news about the JGOMAS framework are posted at the ”NEWS” section of the main web page (Figura 4), so that to advise students about new versions or changes in exercises requirements is very easy. Furthermore, students can contact with their teacher and JGOMAS developers through the Web. In this way, students are encouraged not only to make questions about their doubts, but also they can propose improvements to JGOMAS, such as additional features that make more easy their changes on the basic agents behaviors provided by JGOMAS.

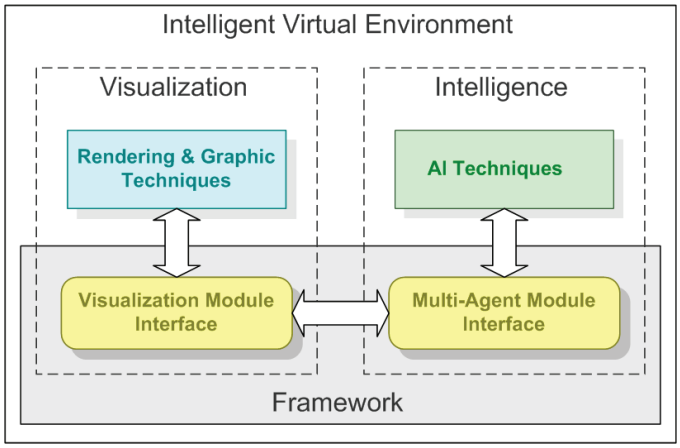

Fig. 5. Application.

C. JGOMAS Framework

Along with its teaching objectives, JGOMAS' framework has been developed with the goal of allowing the developer of a multi-agent system situated in a virtual environment to abstract himself from the peculiarities of such environments.

Thus, a requirement in the design of JGOMAS is that both systems, visualization (virtual reality) and intelligence (multi-agent system), work separately in an independent way, in order to create a flexible and versatile framework. Thus, there is a clear separation of the intelligence part (where reasoning is computed) from the visualization part (where the results of reasoning are displayed) as proposed by [37]. Figure 5 shows the abstraction level provided by the framework, allowing to distribute in an independent way both visualization and intelligence parts. This distribution facilitates the scalability of the intelligent virtual environment application.

In this way, the framework permits not only the distributed execution of all these components, but also facilitates the incoporation of intelligence to the system. Next section presents a detailed description of the framework.

V. Description of JGOMAS' Framework

JGOMAS' framework has been developed to work over an specific multi-agent system platform, JADE, using the facilities provided by this platform, that, nowadays is the most used one in the world. The framework allows designers to incorporate intelligence in agents interacting in a virtual environment, being able to follow the evolution of such agents in the virtual environment through a non-determined number of visualization modules in a distributed fashion. Among all the possible kind of agents in the framework, there is one deserving special attention, the Agent Manager, because is in charge of controlling the simulated environment. The rest of this section details the framework, beginning with the description of the architecture, after that, it shows the agent taxonomy developed, and ending with a detailed view of the Agent Manager.

A. Architecture

The framework is composed of three subsystems, as shown in Fig. 6:

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

as shown in Fig. 6.

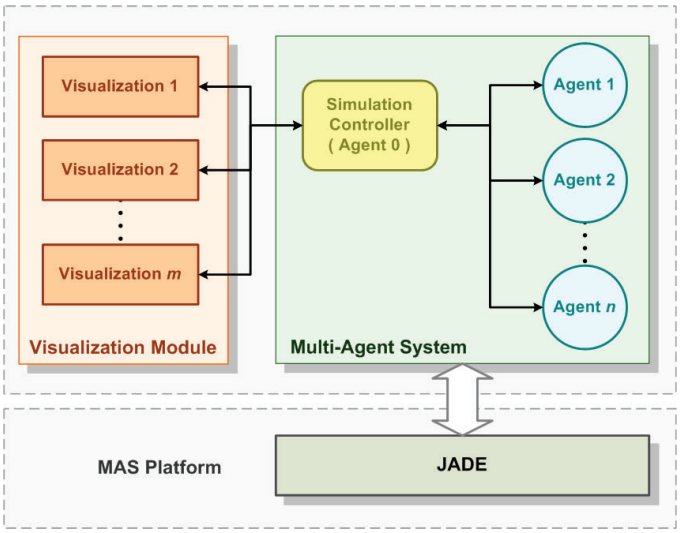

Fig. 6. JGOMAS' architecture overview.

1) Multi-Agent Platform: JGOMAS uses a FIPA-compliant [5] multi-agent platform. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications[2]. Thus, JGOMAS can take advantage of all the resources such platforms offer: behaviour mechanisms, message passing, FIPA interaction protocols, etc..., in compliance with the FIPA specifications (a collection of standards which are intended to promote the interoperation of heterogeneous agents and the services that they can represent).

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

2) Multi-Agent System: JGOMAS' multi-agent system can be viewed as an abstraction upper layer over a multi-agent platform. ¿From this abstract point of view, there is a classification of agents in accordance with the related architecture (Fig. 6). This classification is based on the type of relationship of an agent and the virtual environment. Two main classes of agents are defined: a *simulation controller* and *inhabitant* agents (*player* agents).

Simulation controller is in charge of keeping the virtual environment's data, maintaining the consistency at any time. On the other hand, the other agents are *inhabitants agents* (or player agents) simulating humans, animals, etc., situated in the virtual world. These agents are moving, looking, hearing, etc... in the virtual scenario. Furthermore, they can comunicate each other in order to achieve their goals. The way an *inhabitant agent* achieves a goal is carrying out tasks, as mentioned above. Thus, an *inhabitant agent* interacts with other *inhabitant agents* and with the scenario. As result, the

[2]FIPA is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies.

virtual world can be changed. And the *simulation controller* generate events for those *inhabitant agents* involved in the changes.

3) Visualization Module: As mentioned above, one of the main goals is that artificial intelligence and virtual reality systems have to work independently. In fact, 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 IVE desegners, there has been implemented a basic graphic viewer. *Render Engine* is the graphic viewer application developed *ad hoc* to display the 3D agents, objects, and the scenario in JGOMAS. According to the requirements of graphic applications (high computational cost for short periods), *Render Engine* has been designed as an external module (and not as an agent). It has been written in C++, using the graphic library OpenSceneGraph [45].

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. Written entirely in Standard C++ and OpenGL, it runs on all Windows platforms, OSX, GNU/Linux, IRIX, Solaris, HP-Ux, AIX and FreeBSD operating systems.

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).

On the other hand, *Render Engine* covers from stand-alone users at home to complex virtual reality systems, such as CAVEs^{TM}. The *CAVE*^{TM} 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 a great amount of implementation effort, *Render Engine* has been extended to use VRJuggler[46] as middleware to use a complete virtual reality system, as shown in Fig. 7.

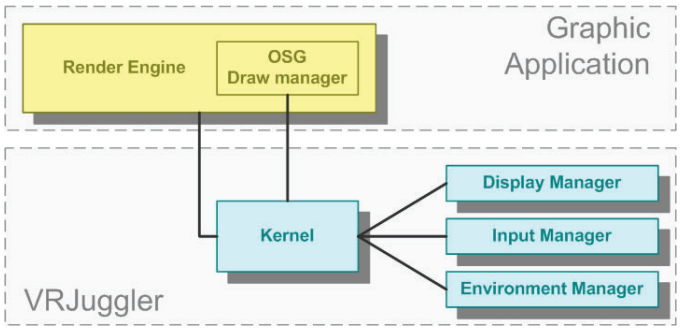

Fig. 7. Integration of Render engine with VRJuggler

B. Agent's Taxonomy

This section covers a different classification of JGOMAS' agents. In this case, this classification is based on an specific implementation, that is, from the point of view of the IVE designer. Thus, there has been defined an agent's class inheritance hierarchy. At the top of the agent hierarchy, there is the base class *JGomasAgent*. It inherits directly from JADE's *Agent* class, and it provides a set of basic features/services to all JGOMAS agents (e.g. services registry). So, JGOMAS agents must derive from *JGomasAgent* class. Moreover, a JGOMAS agent can be, according to its modifiability, an internal or external one.

- Internal agents: are staff in the JGOMAS' MAS subsystem. Their behaviors are predefined, and designer cannot change them. An agent must specialize in:
  - Manager: this is an special agent, and it corresponds with the *simulation controller* named in section V-A.2. Its main goal is to coordinate the current game. Besides, it must answer to requests of the rest of agents. Another task it does is to provide an interface for Render Engine. Thus, any instance of Render Engine can connect to the current game to display the 3D virtual environment. Due to the *Manager Agent* importance, the next section does a more detailed explanation of this agent.
  - Pack: those are *medic packs* (used to heal agents), *ammo packs* (used to give ammunition to the agents) and the *objective pack*, that is, the flag to capture. They are created and destroyed dynamically during the current game, with the exception of *objective pack* (there is only one flag, and it exists during all the game and can not be destroyed).
- External agents: they are really the players of the current game (*inhabitant agents*). They have a set of basic predefined behaviors. However, user can both modify those behaviors and even add new ones.
  - BasicTroop /Troop: basic classes containing all common services and structures that agents need to play a game. These classes are specialized in three other subclasses (but IVE designer can define new ones), where each one is performing a role. Each role has different features, services and behaviors. Furthermore, an agent can play a unique role during the current game.
    Agents are specialized in:
    - *Soldier*: provides a *CallForBackup* service (agent goes to help teammates).
    - *Medic*: provides a *CallForMedic* service (agent goes to give medic packs).
    - *FieldOps*: provides a *CallForAmmo* service (agent goes to give ammo packs).

Fig. 8 shows the JGOMAS taxonomy tree (based on a *capture-the-flag* game), where the bottom level is the most specialized. Having in mind that the number of agents is limited, user's election of roles is a decisive factor to win the game.

C. Agent Manager

This is an special agent in the JGOMAS' MAS subsystem. In fact, there is only one running during the current game. It

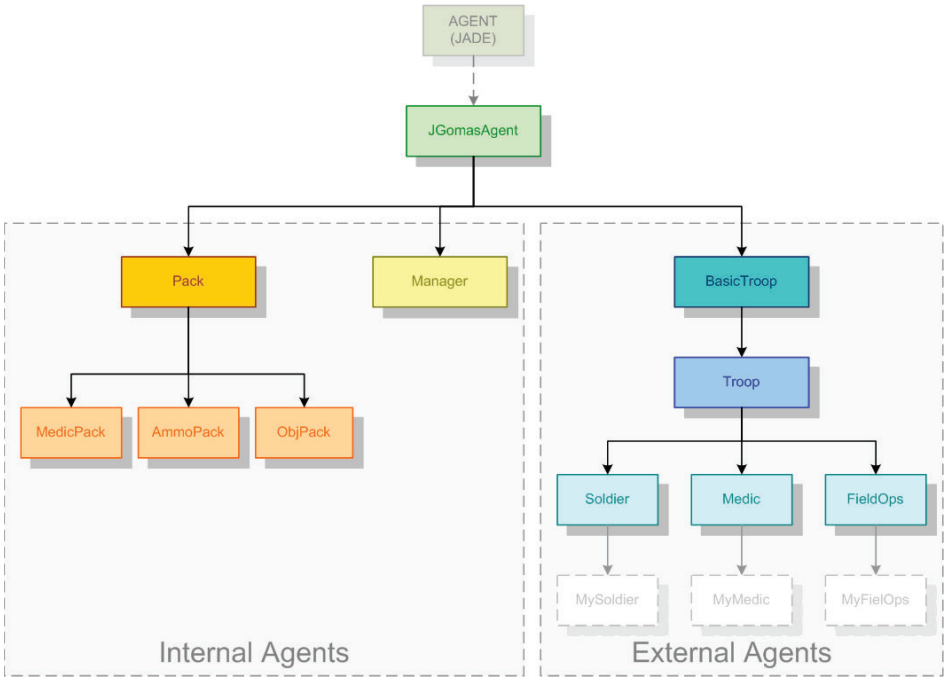

Fig. 8. JGOMAS agent's taxonomy. The bottom level is the most specialized.

has to do two very different tasks:

- Interface for Graphic Viewer.
- Game Logic Management.

1) Interface for Graphic Viewer: *Agent Manager* is in charge of functioning as a server for any Graphic Viewer client interested in connecting to the current game. At the beginning of JGOMAS, *Agent Manager* executes a thread. This thread is simply a server for Graphic Viewer clients. First, the server creates a listening socket in a specific port, and waits for connection attempts from clients. The client creates a socket on its side, and attempts to connect with the server. The server then accepts the connection, and communication can begin. For each accepted connection, a new thread is executed. So, we can have several viewers running at the same time (perhaps in different machines), all connected to the current game.

Moreover, *Agent Manager* holds the game state: *troop* agents, static and dynamic objects and their main attributes (position, direction, velocity, etc.). So, it sends all this information to each graphic viewer client connected, once for frame. Thus, graphic viewers can render their images continuously, keeping the desired framerate. Figure 9 shows JGOMAS architecture, and how agents, JADE platform and graphic viewers are integrated.

2) Game Logic Management: This subject is really a level of abstraction over the JGOMAS MAS. Game Logic involves many aspects, but all oriented to manage the course of the current game. For example, When does the match start? Which is the map to play? Where are agents and what are they seeing at a certain point? and so on…

The control of game logic is centralized just in one place: the *Manager* agent. It is the agent in charge of some tasks, as:

- Management of the life-cycle of the current game: it is in charge of synchronization of all agents for the beginning of the game, and their destruction at the end of the game, besides other more specific details of the game control like informing about the match's map, the objective, etc.
- Coordination and management of services registration: an agent cannot register a service if *Agent Manager* does not allow him to. Moreover, it can manipulate the service's name to prevent cheating (agents from one team subscribing to services of the other team).
- Holding the game state of the current game: Each agent calculates its new position and action to do. Then, they send all that information to *Agent Manager*. So, it controls all information regarding the current game state, agents and their main attributes (position, direction, velocity, etc.).
- Attention of some agent's requests regarding interactions with the environment: *Manager* listen to requests, process them, and returns the results to agents requesting information regarding actions such as look or shot.
- Statistics about agents efficiency: the *Agent Manager* is also in charge of calculating a report about the development of the current game. The purpose of this report is to have a quantitative measure to complement the qualitative data offered by the Graphic Viewer.

VI. Designing Agents in JGOMAS

JGOMAS uses JADE as MAS platform to take advantage of all resources it offers: behaviour mechanisms, message passing (where FIPA ACL is the language to represent messages), naming service and yellow-page service, FIPA interaction protocols, etc., in compliance with the FIPA specifications.

JGOMAS agents are written in JAVA to make the most of JADE's features. Thus, they are FIPA compliant, besides platform-independent.

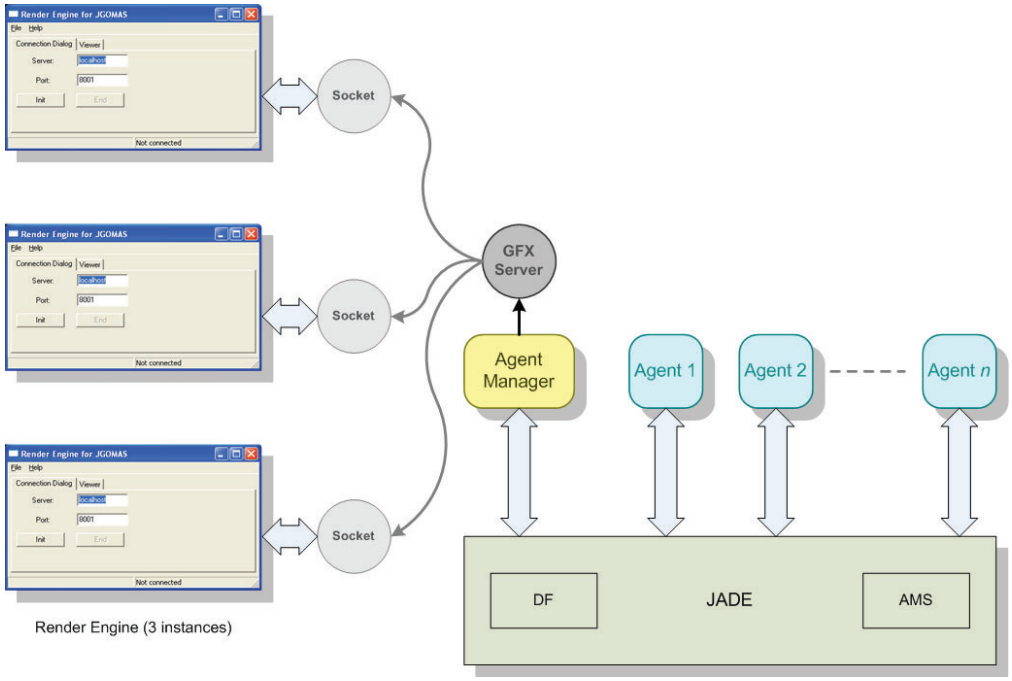

Fig. 9. JGOMAS architecture and integration of its subsystems

A. Specification

- Tasks, Task List
- *Field Of View* objects list
- FSM, GoToTarget Cycle
- Using Kernel functions
- Kernel modifications through Entry Points
- Working Cycle of an agent
- Support for Pathfinding
- Using Kernel functions
- Kernel modifications through Entry Points -¿ Search algorithms and planners
- Decision functions: RNN, Fuzzy Logic, CBR...
- API

1) Parameters:

2) Finite State Machine: To obtain a customizable architecture that may accept user code, agents have to have at least a generic working mechanism. This mechanism have to be able to solve automatically different kind of tasks. The chosen mechanism is implemented as a FSM, formed by three states (Fig. 10):

*a) **STANDING**:* is the initial state. When an agent comes to this state, it extracts the most priority task from the list of pending tasks. Next, agent goes to state *GOTO TARGET*.

*b) **GOTO TARGET**:* once an agent knows which one is the current task, it keeps in this state till it arrives to the place where it has to carry out the task. Then, agent goes to state *TARGET REACHED*.

*c) **TARGET REACHED**:* in this state, an agent carries out the current task. When it has finished it, agent erases it from the list of pending tasks, and agent goes to state *STANDING*, ready to get other task.

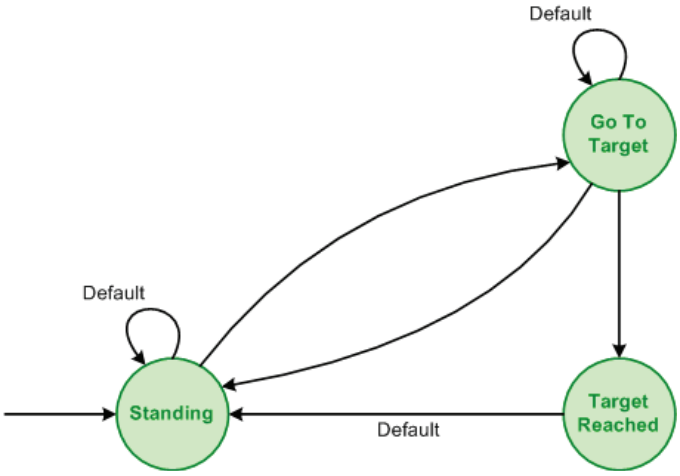

Fig. 10. Finite State Machine executed by each agent.

B. Code Modifications

A user may configure JGOMAS MAS to his needs, and improve the JGOMAS agents intelligence through an API, a set of basic services, behaviors and methods, that JGOMAS kernel offers.

User can add new source code to develop his new agents. This new source code (mods) will be integrated into the JGOMAS' kernel at run-time. This allows the user:

- To create new roles (specialized roles) derived from *Troop* class, or any of its inherited roles (i. e., *Soldier*, *Medic* and *FieldOps*). Thus, JGOMAS taxonomy is extended.
- To provide new services, or to modify existing ones.
- To add new behaviors to launch a new strategy to get the objective.
- To add new features and functionality to take complex

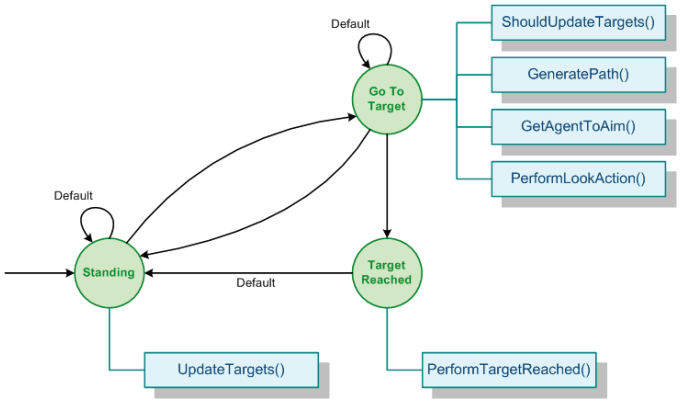

Fig. 11. fsm functions

decisions which will influence in both team and individual emergent behavior. To use the JGOMAS API to do this, it has to be taken into account the agent working cycle (implemented as a Finite State Machine –FSM–).
- To improve path generation.
- Etc.

User can create new classes of agents, derived from existing ones. This way, user can either overload some methods (predefined in the kernel) and add new ones. It is very useful to download the file(s) of example of source code. Concretely, there is a file which is a basic skeleton for a new class (MyMedic) derived from CMedic class. Some interesting methods to overload are:

- protected void UpdateTargets(); ⇒ It may be used to update priority of all 'prepared (to execute)' (or pending) tasks.
- protected boolean ShouldUpdateTargets(); ⇒ When an agent is in the state *GOTO TARGET*, it can go to the state *STANDING* to recalculate the priority of his pending tasks.
- protected boolean GetAgentToAim(); ⇒ It calculates if there is an enemy at sight.
- protected void PerformLookAction(); ⇒ Action to perform when the agent is looking at, according to the objects or agents there are in the *Field of View* (FOV).
- protected boolean checkMedicAction(); ⇒ It decides, when the agent receives a Call For Medic request, if it accepts the proposal.

These methods are executed normally in each working cicle of the agent.

In this way, and as it has been mentioned before, JGOMAS can be used as a testbed for proofs and validation of AI algorithms.

JGOMAS is an environment to develop and to run intelligent agents over simulated 3D worlds. Concretely, it can be used in different scopes, for example:

- Study the complete integration between Multi-agent Systems and Virtual Reality,
- Educational purposes on AI,
- Testbed for AI techniques: cooperation, coordination, learning...

Specifically, JGOMAS allows to run agents in different teams, which compete to achieve their own and team objectives. Those two teams compete in a capture-the-flag-like game, where one team defend its flag against the other team, which tries to capture it.

The agents can play different roles: soldier, medic and field operations. Moreover, agents need to cooperate with their team-mates in order to achieve their objectives. For example, a medic can deliver medic packs to a soldier mate with a low level of health who sent a "*call for medic*" request.

VII. Using of JGOMAS' Framework

Using JGOMAS' framework means both to extend (enhancing) agents' behaviours, mainly in a team-work oriented way, and to test those behaviours under different conditions and scenarios. Once the designer has modelled the IVE, user can launch JGOMAS (all agents) from JADE GUI, one by one. This can be hard work, and due to *Agent Manager* needs all agents be connected to begin the current game, some scripts are given. In this way, although user executes Render Engine, it will not display agents if the current game has not begun.

The last version of the JGOMAS framework is available for download at `http://jgomas.gti-ia.dsic.upv.es`

It includes the multi-agent platform, Render Engine, maps, documentation and a sample ready to use. Fig. 13 shows an execution example of this package, where JADE GUI, text console, and some instances of the Render Engine can be seen.

Following there is a more detailed explanation of executing either JGOMAS MAS and Render Engine.

1) Executing a configuration of JGOMAS visualization : 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 **??**.

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^{TM} 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^{TM}).
- *CAVE^{TM} system*: users are each one immersed in a CAVE^{TM} to get an astounding experience in virtual reality systems.

Figure 15 shows an execution of JGOMAS. A *hand made* Powerwall^{TM} with four monitors was composed, and images

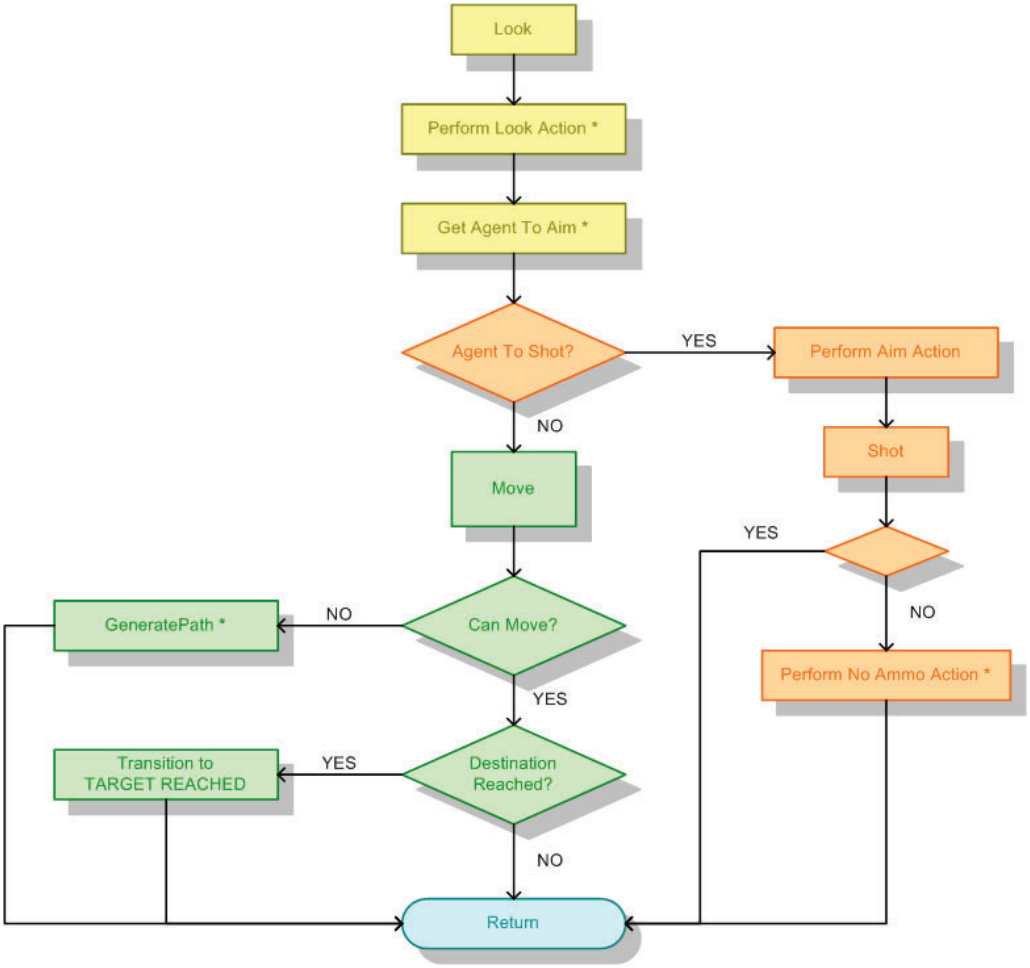

Fig. 12. A cycle of execution

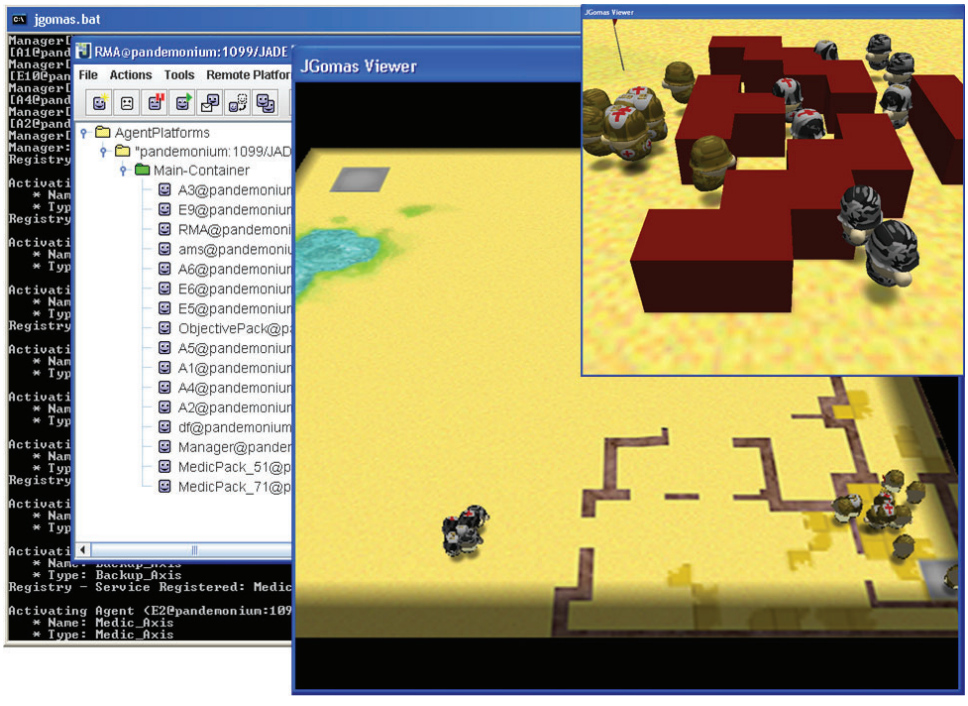

Fig. 13. Ejemplo de ejecucin de JGOMAS

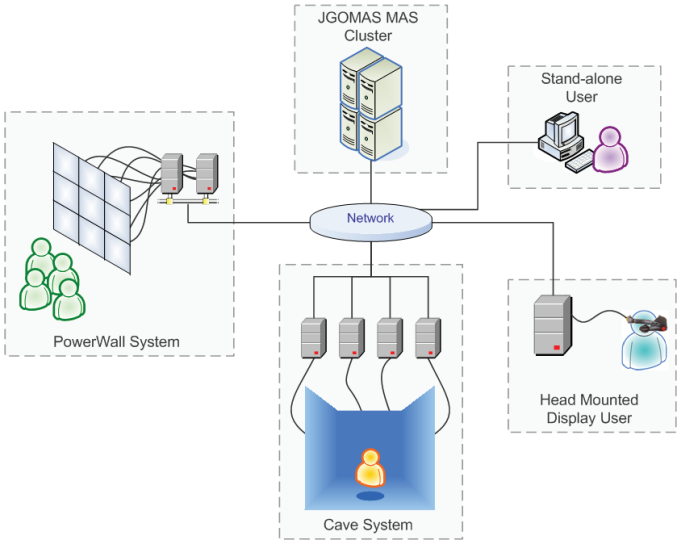

Fig. 14. A JGOMAS scenario with some visualization configurations.

are rendered by two computers forming a cluster. VRJuggler is used to synchronize the images displayed. At the same time, the CAVE^{TM} system was connected to the current JGOMAS game. Figure 16 shows a picture of a person using the CAVE^{TM}.

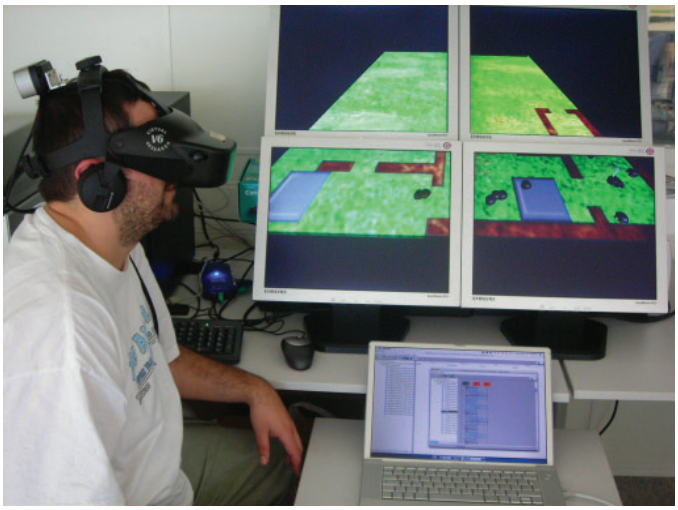

Fig. 15. A practical Powerwall^{TM} done with four monitors and a *head mounted display* user.

2) Executing JGOMAS MAS: One of the advantages of JGOMAS is its flexibility for configuring its start-up. This is possible because user can choose the number and type of agents, along with the parameters for each agent.

These configuration parameters expected for the agents depends on its type:

- The parameters addressed to the *Agent Manager* are the following:
  1) Number of *Troop* agents to start the match.
  2) Map (scenario) where the match is going to be played.
  3) Graphic viewer refresh frequency (in milliseconds).
  4) Match duration (in minutes).
- *Troop* agents accept just one parameter, the team, that can be either `ALLIED` or `AXIS`.
  User can consider necessary to increment the number of parameters accepted by his agents. For that, he must create his own agents, as it was explained previously, to handle these new parameters.

An example of execution of JGOMAS is:

```
java
-classpath lib\jade.jar;.JGOMASomas.jar;.
jade.Boot -gui
Manager:Classes.CManager(4 map_04 125 10)
A1:Classes.CMedic(ALLIED)
A2:Classes.CMedic(ALLIED)
E1:Classes.CMedic(AXIS)
E2:Classes.CMedic(AXIS)
```

According to this example, the *Agent Manager* begins a match with 4 players, which play in the map `map_04` for 10 minutes, at 8 frames per second (125 ms.).

Fig. 16. JGOMAS being used in a CAVE^{TM}.

A. User evaluation

Finally, another important subject is the game's evaluation. User can evaluate what happened during the game, both in a quantitative and in a qualitative ways.

On one hand, user can make a qualitative evaluation because there is a graphical component which allows user to see the game's evolution. For example, user can check if an agent moves correctly with his new path generation algorithm, or if the strategic distribution of agents in the map is as he designed.

On the other hand, quantitative evaluation refers to statistics generated at the end of the match. Thus, user can check agents' efficiency. For example, the number of *medic packs* delivered versus which ones were picked up by team-mates, or by enemies. This allows user to have a numerical result to compare it against other matches played.

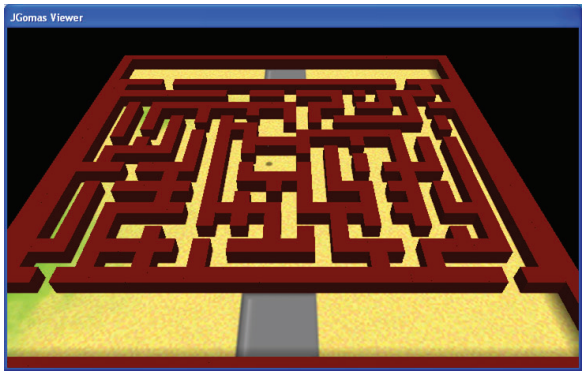

Fig. 17. Scenario with a laberynth for the pathfinding exercise.

VIII. A PRACTICAL EXPERIENCE

This new approach here presented to teaching AI subjects has been carried out in a subject on the last course of the AI speciality in the computing engineering degree of the Politechnical University of Valencia (Spain). The main goal of such subject is to let students know multi-agent systems.

A. Schedule of the sessions

The laboratory work of this subject consists of one weekly session of two hours, during 13 weeks. These sessions can be classified as follows:

1) Introduction to JADE platform: During 2 sessions this platform is presented, both at organizing level and at API level (describing the JAVA functions library included to develop agents working over this platform). For this reason, examples and exercises, to be solved by the students during these sessions, are used.
2) Introduction to JGOMAS framework: Following the previous part, and as it has been developed over JADE, JGOMAS toolkit is presented during 2 sessions. This presentation also includes the description of the functions library complementing the JADE one and allowing to develop JGOMAS agents. Along these sessions, some examples and exercises are also presented and solved by the students.
3) Management of an agent: The purpose of the following 2 sessions is to let the student to test JGOMAS API, beginning to improve the behavior of an agent. In this way, the following exercise is proposed: to improve the pathfinding abilities of an agent to make it able to get the flag and come back to its departure point before past ten minutes, taking into account that the flag is in the middle of an stage such as the one in figure 17.
4) Focusing in the “Multi” aspect in Multi-Agent Systems: In the next two sessions, the communication and coordination of JADE and JGOMAS agents is presented by means of examples and exercises. It is presented not only the API functions and implicit communications between these agents, but also the possibilities of using such an important mechanism (as for instance, coordination, emergent behavior, . . .).
5) Developing a competitive JGOMAS team: The purpose of the next four sessions, is that the students make use of the JGOMAS functions library to put into practice the obtained knowledge. So, they must develop a team of ten agents to be as efficient as possible, having as a minimum purpose to be better than the teams provided in the toolkit distribution. That is, when facing the basic team provided, they must accomplish to capture the flag and to carry it to their base (when they are attacking), and to avoid the enemy team to make it so (when they are defending).
6) Evaluation / Contest: The last session of the subject is dedicated to evaluate the teams developed by the students. This evaluation is made at two levels. At one level, each student developed team is checked against the basic distributed team in different stages. On the other level, a contest is made between the students’ teams. The final qualification obtained by the students depends, partially, on the contest classification, but always taking into account that they must win to the basic team to pass the subject. It has to be underlined that the students do not have the stages where their teams must compete before the evaluation session (though they have some stages to test their teams). Moreover, each team is checked in the stages as attacker and defender.

The results obtained by using JGOMAS framework in the laboratory work of this subject must be considered as satisfactory, having 70% of the students positively completed the development.

IX. CONCLUSIONS

Se han desarrollado y puesto en marcha una nueva herramienta as como una Web ad-hoc para su uso dentro del marco de las prcticas docentes de Inteligencia Artificial. En concreto, la herramienta desarrollada, JGOMAS, es multiplataforma y sigue los estndares fijados dentro de los SMAs, como por ejemplo FIPA. As, esta aplicacin puede tambin ser usada como simulador para la coordinacin, comunicacin y algoritmos de aprendizaje dentro del campo de los MAS, o de la IA en general. Para conseguir esto, JGOMAS permite al usuario aadir sus propias modificaciones de cdigo. Adems, JGOMAS puede ser usado para estudiar la integracin de los MAS y de los sistemas de realidad virtual.

Por otro lado, la puesta en marcha de las nuevas sesiones de prcticas ha resultado muy satisfactoria, tanto por la acogida de los alumnos como por los resultados obtenidos en la evaluacin de las mismas.

As, el atractivo del tema de las prcticas junto con la competicin establecida en la sesin de evaluacin ha motivado a los alumnos a llevar a la prctica muchas de las tcnicas explicadas en teora, as como desarrollar estrategias innovadoras para sus equipos, con el objetivo de obtener el equipo ms ”inteligente”.

En concreto, a esta sesin de evaluacin se presentaron el 70

- Campen: Utilizaban una estrategia similar tanto en defensa como en ataque. sta se basaba en un algoritmo A* y una buena coordinacin de sus tropas, para ir al encuentro de las tropas contrarias a fin de eliminarlos.

- Subcampen: En defensa, dividan a sus soldados en dos grupos que hacan guardia alrededor de la bandera en trayectoria circular y en sentidos contrarios. As minimizaban el tiempo que tardaban en detectar un enemigo. Por el contrario, en ataque, todos los agentes del equipo son mdicos *pacifistas*. Su estrategia bsica era la velocidad (no perdiendo el tiempo en apuntar ni disparar). Adems, eran capaces de suministrarse paquetes de salud.

-

JGOMAS is an environment to develop and to run intelligent agents over simulated 3D worlds. Concretely, it can be used in different scopes, for example:

* Study the complete integration between Multi-agent Systems and Virtual Reality * Educational purposes on AI * Testbed for AI techniques: cooperation, coordination, learning

Specifically, JGOMAS allows to run agents in different teams, which compete to achieve their own and team objectives. Those two teams compete in a capture-the-flag-like game, where one team defend its flag against the other team, which tries to capture it.

The agents can play different roles: soldier, medic and field operations. Moreover, agents need to cooperate with their team-mates in order to achieve their objectives. For example, a medic can deliver medic packs to a soldier mate with a low level of health who sent a "call for medic" request.

ACKNOWLEDGMENT

This work was partially supported by CONSOLIDER-INGENIO 2010 under grant CSD2007-00022. The financial support received from the Spanish government and FEDER funds under TIN2005-03395 and TIN2006-14630-C03-01 projects are also gratefully acknowledged.

REFERENCES

[1] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa, "JADE - A FIPA-compliant Agent Framework," in *Proc. of the PAAM'99*, Apr. 1999, pp. 97–108.

[2] M. Wooldridge, *An introduction to multiagent systems*. John Wiley and Sons, February 2002.

[3] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995. [Online]. Available: citeseer.ist.psu.edu/article/wooldridge95intelligent.html

[4] E. Argente, A. Giret, S. Valero, V. Julian, and V. Botti, *Survey of MAS Methods and Platforms focusing on organizational concepts*. IOS Press, 2004, vol. 113, pp. 309–316.

[5] FIPA, "FIPA abstract architecture.technical report sc00001l," Foundation for Intelligent Physical Agents, Tech. Rep., 2002, checked on February 8th, 2007. [Online]. Available: http:///www.fipa.org/

[6] F. Bellifemine, A. Poggi, and G. Rimassa, "Developing multi-agent systems with jade," *In Intelligent Agents VII. Ed. Castelfranchi, C. and Lesperance, Y.*, no. 1571, pp. 89–103, 2001.

[7] I. Sun Microsystems, "Java technology." checked on February 8th, 2007. [Online]. Available: http://java.sun.com/

[8] "RoboCup," 2007, checked on July 24, 2007. [Online]. Available: http://www.robocup.org/

[9] A. Bredenfeld, A. Jacoff, I. Noda, and Y. E. Takahashi, *RoboCup 2005: Robot Soccer World Cup IX*, ser. LNCS. LNAI. Springer-Verlang, 2006, vol. 4020.

[10] M. Asada, P. Stone, H. Kitano, A. Drogoul, D. Duhaut, M. M. Veloso, H. Asama, and S. Suzuki, "The robocup physical agent challenge: Goals and protocols for phase 1," in *RoboCup-97: Robot Soccer World Cup I*. London, UK: Springer-Verlag, 1998, pp. 42–61.

[11] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "RoboCup: The Robot World Cup Initiative," in *AGENTS '97: First International Conference on Autonomous Agents*. New York, NY, USA: ACM Press, 1997, pp. 340–347.

[12] H. Kitano, Y. Kuniyoshi, I. Noda, M. Asada, H. Matsubara, and E.-I. Osawa, "Robocup: A challenge AI problem," *AI Magazine*, vol. 18, no. 1, pp. 73–85, Spring 1997.

[13] K. Fullam, T. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. Rosenschein, L. Vercouter, and M. Voss, "A specification of the agent reputation and trust (ART) testbed: Experimentation and competition for trust in agent societies," in *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)*, Utrecht, July 25-29 2005, pp. 512–518.

[14] K. Fullam, T. Klos, G. Muller, J. Sabater, Z. Topol, K. S. Barber, J. Rosenschein, and L. Vercouter, "A demonstration of the agent reputation and trust (ART) testbed: Experimentation and competition for trust in agent societies," in *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005) Demonstration Track*, Utrecht, July 25-29 2005, pp. 151–152.

[15] G. Becerra, J. Heard, R. Kremer, and J. Denzinger, "Trust attributes, methods, and uses," in *The Workshop on Trust in Agent Societies at The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2007)*, Honolulu, Hawaii, USA, May 15 2007, pp. 1–6.

[16] "TAC:trading agent competition," 2007, checked on July 25, 2007. [Online]. Available: http://www.sics.se/tac

[17] M. P. Wellman, A. Greenwald, P. Stone, and P. R. Wurman, "The 2001 trading agent competition," in *Eighteenth national conference on Artificial intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 2002, pp. 935–941.

[18] J. Collins, R. Arunachalam, N. Sadeh, J. Eriksson, N. Finne, and S. Janson, "The supply chain management game for the 2007 trading agent competition," School of Computer Science,Carnegie Mellon University, Pittsburgh, Tech. Rep., December 2006.

[19] J. Eriksson, N. Finne, and S. Janson, "Evolution of a supply chain management game for the trading agent competition," *AI Communications*, vol. 19, no. 1, pp. 1–12, 2006.

[20] M. Luck and R. Aylett, "Applying artificial intelligence to virtual reality: Intelligent virtual environments," *Applied Artificial Intelligence*, vol. 14, no. 1, pp. 3–32, 2000. [Online]. Available: citeseer.ist.psu.edu/article/aylett00applying.html

[21] R. Andreoli, R. D. Chiara, U. Erra, and V. Scarano, "Interactive 3d environments by using videogame engines," in *IV '05: Proceedings of the Ninth International Conference on Information Visualisation (IV'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 515–520.

[22] J. Laird and J. Duchi, "Creating human-like synthetic characters with multiple skill levels: A case study using the SOAR quakebot," Menlo Park, Calif., 2000, m. Freed, ed: Papers from the AAAI Fall Symposium,AAAI Press. [Online]. Available: citeseer.ist.psu.edu/laird00creating.html

[23] "SOAR: a general cognitive architecture for developing systems that exhibit intelligent behavior." checked on July 27, 2007. [Online]. Available: http://sitemaker.umich.edu/soar/home

[24] R. Lewis, "Cognitive theory, SOAR," *In International Encylopedia of the Social and Behavioral Sciences.Ed.:N. J. Smelser and P. B. Baltes. Amsterdam: Pergamon (Elsevier Science).*, 2001.

[25] J. F. Lehmann, J. Laird, and P. Rosenbloom, "A Gentle Introduction to SOAR, an Architecture for Human Cognition: 2006 update," 2006. [Online]. Available: http://ai.eecs.umich.edu/soar/sitemaker/docs/misc/GentleIntroduction-2006.pdf

[26] B. Gorman, M. Fredriksson, and M. Humphrys, "QASE - An integrated API for imitation and general AI research in commercial computer games," in *In Proceedings of 7th International conference on Computer games: Artificial intelligence, animation, mobile, educational, and serious games (CGAMES), Angoulme, France.*, 2005.

[27] "MATLAB®7.4," checked on July 27, 2007. [Online]. Available: http://www.mathworks.com/products/matlab/

[28] E. Norling, "Capturing the quake player: using a BDI agent to model human behaviour," in *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM Press, 2003, pp. 1080–1081.

[29] "JACK Intelligent Agents™," checked on Juy 27, 2007. [Online]. Available: http://www.agent-software.com/shared/products/index.html

[30] N. Howden, R. Rönnquist, A. Hodgson, and A. Lucas, "JACK Summary of an Agent Infrastructure," in *Proc. Workshop on Infrastructure for Agents, MAS, and Scalable MAS at the Fifth International Conference on Autonomous Agents*, Montreal, Canada, 2001.

[31] N. Minar, R. Burkhart, C. Langton, and M. Askenazi, "The swarm simulation system, a toolkit for building multi-agent simulations," 1996. [Online]. Available: citeseer.ist.psu.edu/minar96swarm.html
[32] "Swarm development group," 2006, checked on September 21, 2006. [Online]. Available: http://www.swarm.org/
[33] J. Klein, "BREVE: a 3D Environment for the Simulation of Decentralized Systems and Artificial Life," in *Proceedings of Artificial Life VIII, the 8th International Conference on the Simulation and Synthesis of Living Systems*. The MIT Press, 2002.
[34] S. Vosinakis, G. Anastassakis, and T. Panayiotopoulos, "DIVA: Distributed Intelligent Virtual Agents," in *Extended abstract, presented at the Virtual Agents 99 workshop on Intelligent Virtual Agents*, University of Salford,UK, 1999.
[35] G. Anastassakis, T. Ritchings, and T. Panayiotopoulos, "Multi-agent systems as intelligent virtual environments," in *Proceedings of Advances in Artificial Intelligence, Joint German/Austrian Conference on AI - KI*, 2001.
[36] J. Rickel and W. Johnson, "Steve: An Animated Pedagogical Agent for Procedural Training in Virtual Environments," in *Proceedings of Animated Interface Agents: Making Them Intelligent*, 1997, pp. 71–76.
[37] B. Ulicny and D. Thalmann, "Crowd simulation for interactive virtual environments and VR training systems," in *Computer Animation and Simulation '01*, 2001, pp. 163–170.
[38] S. R. Musse and D. Thalmann, "Hierarchical model for real time simulation of virtual human crowds," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 2, pp. 152–164, 2001.
[39] D. Thalmann and H. Noser, "Towards autonomous, perceptive, and intelligent virtual actors," *Artificial Intelligence Today. Lecture Notes in Artificial Intelligence. Springer*, no. 1600, pp. 457–472, 1999.
[40] M. Kallmann, J.-S. Monzani, A. Caicedo, and D. Thalmann, "ACE: A platform for the real time simulation of virtual human agents," in *EGCAS'00 - 11th Eurographics Workshop on Animation and Simulation*, 2000.
[41] "id Software, Inc." 2006, checked on July 24, 2006. [Online]. Available: http://www.idsoftware.com/games/quake/quake4/
[42] S. Offermann, J. Ortmann, and C. Reese, "Agent based settler game," 2005, part of NETDEMO, demonstration at the international conference on Autonomous Agents and Multi Agent Systems, AAMAS-2005. [Online]. Available: http://x-opennet.org/netdemo/Demos2005/aamas2005/_netdemo/_settler.pdf
[43] "EKSL: The Experimental Knowledge Systems Laboratory," Department of Computer Science at the University of Massachusetts – Amherst, checked on July 27, 2007. [Online]. Available: http://www-eksl.cs.umass.edu/
[44] M. van Lent, J. E. Laird, J. Buckman, J. Hartford, S. Houchard, K. Steinkraus, and R. Tedrake, "Intelligent agents in computer games," in *AAAI/IAAI*, 1999, pp. 929–930.
[45] "OSG: OpenSceneGraph, an Open Source high performance 3D Graphics toolkit." checked on February 8th, 2007. [Online]. Available: http://www.openscenegraph.org/
[46] "V RJuggler: a platform for virtual reality application development." checked on February 8th, 2007. [Online]. Available: http://www.vrjuggler.org/

Toni Barella Biography text here.

Soledad Valero is originally from Valencia (Spain). She received her BS and MS degrees in Computing Engineering from the Technical University of Valencia in 2000 and 2003, respectively. She is a PhD student in the Computer Science Department of the Technical University of Valencia. Her research interests are multi-agent systems, e-commerce and soft-computing techniques.

Carlos Carrascosa was born in Valencia (Spain) and received the MS degree in Computer Science from the Polytechnic University of Valencia in 1995. Currently, he is a lecturer and obtained his PhD. at the Computer Science Department at Polytechnic University of Valencia. His research interests include multi-agent systems, learning, information retrieval and real-time systems.