5

Adaptive Virtual Environments: The Role of Intelligent Agents

Marcus S. de Aquino[1] and Fernando da F. de Souza[2]
[1]Federal University of Campina Grande,
[2]Federal University of Pernambuco,
Brazil

1. Introduction

The aim of this chapter is to provide an understanding of how intelligent agents can improve user interaction to three-dimensional (3D) Virtual Environments by turning the later into an adaptive system.

A virtual environment (VE) can simulate a real environment or provide imaginary or even physically impossible scenarios. It allows one or more users to interact through visualization and manipulation of extremely complex representations (Frery, 2002). According to Chittaro & Ranon (2007), a VE can help people through direct experience by performing tasks that are suited to the learning or simulation tasks. Moreover, employing interactive 3D objects allows for more useful representations of subjects or sensations, offering the possibility of analyzing the same subject from different points of view. VE is defined here as a computer-generated, graphically-rich 3D world which the user can explore and perform actions on real time. The environment is based on 3D computer models and the user can navigate on the environment and interact with the 3D objects and other avatars existing there. The user's view of the environment is rendered dynamically and can be updated as the user moves.

More recently, three-dimensional Virtual Environments developed with Virtual Reality (VR) technologies and Graphical Computation techniques are emerging on the Web (Aquino et al., 2005, 2007, 2008). This makes it possible to build VEs more aware of their users. Moreover, it favors interactions with users bearing different profiles, cultures and knowledge levels. Under this paradigm, dynamic and interactive elements are introduced to VEs in order to increase their users' engagement. Such users' participations are directly related to both the environment and the feedback the system supplies him/her with. Accompaniment procedures to users' actions and updating VR environments accordingly have been used to accomplish this.

The personalization of VEs can enhance user's interaction in several applications such as distance-learning systems, to which students with different knowledge levels are remotely connected to interact with media, content and even to each other; electronic games for both amusement and learning; building construction providing different perspectives for architects, civil engineers, interior decorators and buyers; and to enable different kinds of access to specific systems e.g. a university information system that can be accessed by lecturers, students and other personnel. In this context, an adaptive VE shall be capable of managing the alterations that are occurring in the environment, to survey the user's behavior, to identify his/her eventual difficulties and, therefore, to suggest modifications in order to improve the user's interaction with the environment. Thus, VEs may take care of the user's necessities to guide his/her navigation in the environment. For instance, for those users that have previously interacted with the VE, the system may identify new ways of navigation or even reduce the number of steps needed for the interaction.

The need of user-aware 3D environments has contributed to the development of techniques to personalize VEs. One of such techniques has to do with using intelligent agents to identify user's actions and to adapt the environment according to the user's needs and his/her knowledge level. The use of intelligent entities (agents) and different forms of interactions makes it possible to build such environments, promoting bigger dynamism, realism and usability to them (Santos & Osório, 2004).

According to Russell & Norvig (2010), an agent has knowledge on the environment that it acts and on the actions that it can execute. The agent's behavior can be based on its own experience, as well as in the knowledge acquired from the particular environment it acts.

On the other hand, according to Celentano et al. (2004), an adaptive Virtual Environment must have intelligence to bring up to date the 3D environment with the objective to reduce the cognitive necessity of interaction. In such a way, the system can assist the user during the interaction by modifying the complexity of the environment or facilitating the navigation process.

Agents in decision-making processes allow constructing a more dynamic system in two main ways. Firstly, the choice of specific languages facilitates the capacity of communication between agents. Secondly, the determination of objectives, plans and actions makes it possible to model inference systems with enhanced precision.

The advantages of using agents are threefold: (i) to interact with the environment (and the user) through sensors and actuators; (ii) to evaluate the current state of the environment based on a knowledge base; and (iii) to make intelligent decisions in accordance with objectives to be reached (Russel & Norvig, 2010).

To achieve on-line VE updating, it must be guaranteed a communication protocol that keeps the connection of the active user and identifies his/her actions (plus underlying intentions) thoughtfully. Having this kind of information, an intelligent decision system must be activated to interpret such actions and to send a reply to the user accordingly. In adaptive VEs, the reply is translated into modifications to the environment, adapting the system to user's necessities (reduction/increasing the environment complexity level and access authorization, amongst others).

According to Woldrigde (2009), "there is obvious potential for marrying agent technology with that of computer games and virtual reality… In this case, the agents need to show emotions; to act and react in a way that resonates in tune with our empathy and understanding of human behavior".

2. Adaptive virtual environments

Adaptive VEs are addressed in this section, which highlights the techniques that may be employed to build such systems. These techniques are used for dynamic organization of objects on the environment (e.g., products more likely to interest the customer are shown in first plan in a virtual store) to facilitate the navigation and the exploration, in order to provide a more effective interaction. The adaptation can also modify the level of complexity of 3D objects according to the user´s knowledge degree on the environment. This aids VEs to deal more properly with their users' characteristics and necessities.

Some related works will be addressed below with the objective to illustrate several applications of intelligent agents in VE and to identify, in each case, the used techniques and the limitations on the generation of adaptive VEs.

The use of avatars as interactive guides in three-dimensional environments is proposed by Frery et al. (2002). The user's profile, obtained from forms (explicit collect) is identified and stored into the user model of the system's knowledge base. Having this information, the avatar (an intelligent agent) is capable to generate the strategy of navigation for the environment, as well as its graphical representation that will represent the user.

Even though only an explicit information collecting is carried out at the beginning of a session, there is already a concern with the identification of the user's characteristics and its use for adaptive environment construction.

However, although the environment is adaptable (the user's characteristics are used to generate the route for contents presentation), it is considered static because it does not make environment modification to adapt itself to the user's interests and necessities. In this case, the intelligent agent does not use its sensors to identify the user's interaction.

Chittaro & Ranon (2002a) developed the AWE3D (Adaptive WEb 3D), an adaptive VE that represents a virtual store in a Web environment, in which the users can browse and get information on objects spread over the environment. Information on the interests of the user, intended for the personalization of the environment, is obtained though explicit collect by monitoring user's actions in the environment (such as visualized products and effected purchases).

The environment is dynamic (possessing better interactivity with the user) and adaptable as well. Therefore, objects' presentation varies due to the customer's profile of purchase, which considers his/her necessities, preferences, visualization of products and options of purchase.

In this case, agents use their sensors to bring up to date the user's profile but they do not make the VE adaptation on real time, that is, they do not use its actuators to modify the VE with the objective, for example, to offer new promotion products (e.g., "loss leader").

In a posterior work, Chittaro et al. (2003) present a virtual agent assigned to assist the user to navigate on a virtual museum. From the description of the places or objects of interest to be visited, supplied at the beginning of a session, the agent creates an appropriate trajectory. Such an agent can also stop during its passage to present each object or place of interest previously determined by the user.

The trajectory is calculated by the Trajectory Manager whose task is to derive an appropriate sequence from the navigation points to pass by the objects that must be visualized by the user. The Trajectory Manager is inserted into the AWE3D environment to provide an additional resource to the adaptive environment.

The users' help system, carried out by an agent, is one of the strongest points of the work. It determines the route for presentation according to the users' interests. The environment is adaptable since it uses the user's profile to generate the trajectory. It is static because it does not modify the trajectory during a user's interaction with the virtual world.

The agent's sensors and actuators are used only to present the information asked by the user. The specific trajectory for the user's profile is defined at the beginning of a session and it does not provide adaptation on real time. Such an adaptation could be useful should the user be interested to see other objects of the museum that were not defined in the predefined route.

Santos & Osório (2004) has proposed the AdapTIVE environment (Adaptive Three-dimensional Intelligent and Virtual Environment) that shows adapted contents in accordance with both the interests of the users and the manipulation (insertion, removal or update) of contents on the environment.

A process of automatic categorization is applied to the creation of models of contents that are used in the spatial organization of the environment. The adaptation process uses models of both users and contents. Moreover, an intelligent virtual agent acts as assistant to the users with navigation and localization of relevant information.

The architecture of AdapTIVE presents an agent located in the customer's machine, which follows all the requests of the user in the virtual environment and guides him/her on searching for the desired information.

The agent in the AdapTIVE environment plays an important role on assisting navigation and to inform user's actions to the system, such as visited environments and accessed contents. Having such information, the system updates both the user model and the contents model in order to be able to reorganize the environment for a posterior session (defined by the system).

AdapTIVE is a static environment because it does not support object movements. However, it is an adaptable environment because it is concerned with adaptation of contents according to the user's profile.

In this case, the agent's sensors and actuators are used to guide users and to assist them on the VE. Its updating is carried out on posterior session rather than on real time. This kind of adaptation could contribute to present new contents to the user when he/she accesses a specific content.

An example of using multiagent system in VEs is ICSpace (Internet Culture Space) (Tavares et al., 2001). It has been proposed in order to support the interaction between different users visiting a same place. The agent based mechanisms will be responsible for taking the user into an interactive tour through a museum. They are able to show a same exposition personalized for each visitor. According to the places visited by the user, they can suggest either to see a particular work or to interact with other user with a similar profile (Tavares et al., 2005).

The agents' architecture structure is composed of three layers. The reactive layer has as its goal to perceive user actions on the environment. The intermediate layer is responsible for maintaining the knowledge database using a Database Agent to do this work. Finally, the service layer is capable of processing information and provides services to users.

The functionalities offered by service agents are: to build the user profile; to organize the exhibition according to the user's profile; to recommend service in accordance with the audience of works and rooms; to guide visitors in the environment rooms; and to encourage interpersonal communication between users through the use of communication tools.

Different agents' categories have been defined in the work. Each one of them identifies a specific role in the multiagent system architecture with a special function in a cultural environment context.

Thus, it can be seen that the layered division of the agents' tasks allows better accompaniment of the users' actions and a greater capacity in the decision making process. In this case, objects do not suffer modification on the VE (it is a static one), but the route suggested to the user becomes adaptable according with his/her profile. However, agents do not update the VE on real time. This could be useful should the user acquire new knowledge from the objects visited in the museum.

Costa et al. (2010) present a proposal to control the user's navigation on 3D games using intelligent agents. An application of their work was developed aimed at training the memory concentration capacity for patients with attention deficit.

They explore the integration of multiagent model methodologies to control the user performance, adapting the interface and automatically changing the difficulty level of tasks.

To support every phase of modeling and implementation of agents, a lexical catalog was specified to record and document the requirements. They defined four agents that are responsible for planning patient care; controlling the interactions to the game; analyzing the performance of the patient; and setting up the environment. It was defined twenty agents' behaviors that were implemented in the JADE framework (Bellifemine, 2003).

A system application proposes a scene which depicts bookshelves with different objects on them. Objects are randomly shown on a table and the patient must walk to one shelf and choose a similar one. Agents will monitor the time and the user interactions with the environment. They will control the rights and wrongs answers and combine these data with the information about the patient's impairments. The therapist informs the patient personal data when registering him/her in the system.

The advantage of their work is that the accompaniment of the patient's activities is managed by the agents. Moreover, they are capable to consider new activities with different levels of difficulties, depending on the correctness/incorrectness of the patient's attempt.

So, the multiagent system works to submit a problem to the user, to evaluate his/her actions and, after that, to generate new situations with different levels of difficulty. Agents work in group to reach an objective and, therefore, they increase the capacity of the system to interact with the user and to generate a suitable environment to the user model.

Cai et al. (2011) propose to simulate a coalmine risk accidents using virtual reality and multiagent technology. Agents are grouped into either Entity Agents or Service Agents.

Entity Agents endow the geometric entities rendered in bottom layer with specific semantics and behavior rules. Service Agents are responsible to support the network communication; to provide semantic information service related to the virtual environment, and primarily offer the necessary knowledge for intelligent behaviors of a coalmine virtual environment; and to provide services related to specific applications.

Risk accidents in coalmine are simulated by agents. On the course of moving though the panned paths, the virtual miner may encounter some obstacles or hazards such as unsupported roof, uncovered holes, uneven surfaces, or electrical cables. Virtual miner's tiredness value also will increase as fulfillment of safety inspection tasks. In this case, driven by its desire of going out of the underground mine, the virtual miner neglected an approaching locomotive and hurried to go through a crosscut. As a result, the virtual miner was seriously injured. Therefore, this approach can reconstruct risk accidents to enable better understanding risk factors in emergency situation and to assist interactive risk analysis in underground coalmine as well.

It is noticed that the division of tasks in a Multiagent System (MAS) can be very useful in a simulation environment. It is possible to generate several dangerous situations and to verify the behavior that would be carried through by a user (in this case, the behavior of the user is controlled by an avatar/intelligent agent). Although in this application it is not necessary the adaptation on real time, the MAS can generate several suitable scenes to the possible risk situations that can be found in a coalmine.

As it can be seen in the above examples, agents are capable to generate applications in accordance with the preferences and interests of the system's users. However, this requires the inclusion of new techniques to construct VEs nearest to the reality.

User's modeling techniques, for example, have been incorporated to these applications with the objective to identify the user's characteristics, necessities and preferences. According to Papatheodorou (2001), one of the key benefits of building systems capable of modeling the user is the possibility of adapting the system behavior to particular needs of their users.

A user model (UM) is a representation of the user properties, which allows the system to adapt aspects of its performance and its functionalities to individual necessities. To reach this objective, modeling techniques have been developed from the user's profile identification. Besides the conventional characteristics of the user, others can effectively be added to the model in order to describe the carried out actions (selected objects and contents accesses) as well as how the interaction is made (preferences for the environment organization, forms of navigation, type of preferred interface and knowledge acquired during the interaction with the environment).

The works of Chittaro & Ranon (2000a, 2000b, 2002a, 2002b) propose the adoption of a user template for customizing a 3D environment. Their proposal is aimed at deploying a user template to adjust the layout structure of a Web store. Besides, the adaptive VE AWE3D (Chittaro & Ranon, 2000a) is a pioneer in the literature on using UM to provide adaptation for a three-dimensional environment. The use of a form and the association of a user to a pre-defined profile contribute to the customization of a 3D environment.

Santos & Osório (2004) also deploys user template to generate a virtual adaptive environment. Their proposal obtains user profile information through a data collection form for composition of the initial model. As the user interacts with the environment, evidences of navigation, and request access to contents are collected and used in the process of updating the user template. Even though, the UM is updated during the session, the use of this information for the environment transformation can only be done offline. Tavares et al. (2005) use the UM: i) to suggest to the current user to see a particular work; and ii) to interact with another user bearing a similar profile. Costa et al. (2010) also use UM to control users' navigation on 3D games taking into consideration their profiles.

Table 1 summarizes the analysis carried out on the focused environments. It was based on i) the type of environment; ii) whether it uses agents or not; iii) the kind of agent-user interaction; iv) how the user's profile is dealt with by the system; and v) whether it is adaptive or not. The characteristics of agents and environments considered in the analysis are based on those presented by Russel & Norvig (2010).

It is also verified, in Table 1, that the systems that possess the module User Profile, and carry out explicit and/or implicit collection, use such information either to modify the environment or just to provide a guided navigation to the user. Thus, the works of Chittaro & Ranon (2002a), Santos & Osório (2004), Costa et al. (2010) and Cai et al. (2011) are those that perform scenes alterations. The works of Frery et al. (206.02), Chittaro et al. (2003) and Tavares et al. (2005) modify user´s routes with no alterations to the environment. However, none of them allows alterations due to modifications on the user profile to occur on real time.

The use of Artificial Intelligence in virtual environments made it possible the construction of virtual worlds closer to the user's reality. The deployment of intelligent agents in such environments allows, for example, the presence of assistants to help with the navigation or to act as a communication interface whenever the user needs to get some information or explanation on determined subject.

The inclusion of the user's profile in VEs enables the generation of personalized environments. The latter are capable of adapting their contents according to the user's preferences, his/her navigation style and cognitive capacity. Thus, a personalized environment may increase the satisfaction and productivity of its users.

One of the limitations found on current adaptive VE is the lack of real time (on-line) evolution, which would allow updating a virtual world during the user's interaction. Such a property may enhance to a large extent the range of applications that can be supported by such environments, mainly those that cannot wait for the next user's interaction to benefit from the system updating.

Objects manipulation on real time occurs when the system detects some user's action that implies scene changing. This change may need modifying one or more scene objects, but not necessarily the reloading of the whole environment. VEs focused above perform their modifications off-line and introduce them only in the next user's interaction.

An adaptive VE must identify each user's profile at the beginning of a session to be able to determine the most adequate environment to be generated. Thus, the representation of the virtual world must allow 3D objects to be inserted to or removed from the environment in order to provide such tailored updates. Furthermore, such updating should be carried out on real time.

System	Type of Environment	Agent / User Interaction	User Profile	Adaptive environ-ment	Real Time Adaptivity
Frery et al. (2002)	3D/2D, entertainment, static	Aid to navigation and suggestions making	Explicit collection (form), individual, static	No	No
Chittaro & Ranon (2002a)	3D, *e-commerce*, dynamic	Consultation, aid to navigation	Explicit and implicit collections, individual and group, dynamic	Yes	No
Chittaro et al. (2003)	3D, entertainment, static	Aid to navigation	Explicit collection (forms), individual, static	No	No
Santos & Osório (2004)	3D/2D, contents presentation, static	Consultation, aid to navigation	Explicit and implicit collections, individual, static	Yes	No
Tavares et al. (2005)	3D/2D, entertainment, static	Aid to navigation and suggestions making, aid to communication	Implicit and explicit collections, individual and group, static	No	No
Costa et al. (2010)	3D, serious game, dynamic	User's actions evaluation	Implicit and explicit collections, individual, static	Yes	No
Cai et al. (2011)	3D, simulation, static	It does not have (the agent simulates the user)	Implicit and explicit collections, individual, static	Yes	No

Table 1. A Comparative Analysis between Adaptive VEs

Bearing in mind the above considerations, we present VEPersonal (Aquino et al., 2010) as a solution to the problem of VE content real time adaptation by using a Multiagent System (MAS) capable of handling virtual objects with different complexity levels. VEPersonal is responsible for the creation and maintenance of adaptive three-dimensional environments in the Web. It is also capable of modifying the world according to the user's cognitive evolution. The system chooses the proper level of detail to display according to the user’s needs, knowledge and characteristics detected from his/her profile. Moreover, VEPersonal makes it possible a real time adaptation through the insertion and removal of objects into/from the scene according to the users’ actions.

3. Agents and adaptive systems

Multiagent Systems are characterized for the existence of more than one agent which may interact between them independently as well as they may work in group to execute tasks, to solve problems or to meet a particular objective. Such objectives can be common to all agents or not (Lesser, 1999). Each agent can communicate or cooperate with other agents when necessary, with the objective to add the local results to reach the solution of the general problem (Jennings, 2000). To meet an objective, it is necessary to define good mechanisms of communication (exchange of messages), cooperation (assistance) and coordination (organization) in the multiagent system. Different agents may have different "spheres of influence". These spheres of influence may coincide in some cases. Then, it may give rise to dependence relationships between agents (Woldridge, 2009). Therefore, an agent can cooperate with other agents that have influence on other environments.

According to Ferber (1999), a MAS is applied to a system comprising the following elements: an environment; a set of objects that can be perceived, created, destroyed and modified by agents; an assembly of agents which represent the active entities of the system; an assembly of relations, which link objects to each other; an assembly of operations to make it possible for the agents to perceive, produce, consume, transform and manipulate objects; and operators with the task of representing the application of such operations and the reaction of the world to this modification attempt.

3.1 The society of agents in VEPersonal

In the Virtual Environment we will be able to have an organization of responsible agents for a system of roles in different dimensions (interface and monitoring user’s actions, decision-making, storage and management of information and modifications in the VE for example).

To satisfy these necessities, we developed the VEPersonal architecture, shown in Figure 1. The characteristics of the users plus their behaviors are stored into the User Model (UM) as well as the history of the environment changes into the Environment Model (EM). Such information is obtained from the interface through sensors that detect the actions undertaken by the user. Thus, the system can monitor progress during a user’s interaction and can create or modify objects on the underlying virtual world accordingly.

Data structures in VEPersonal are stored into a Data Base Management System (DBMS) in order to guarantee a higher data integrity and integration. User’s personal data records and the environment information are stored using a DBMS relational structure. 3D objects use a

XML structure because they are constructed using the X3D language (X3D, 2007). Data recovery and in-memory management are performed by the agents society. Each agent is responsible for manipulating just one type of information. Moreover, information exchanging must comply with a specific ontology, and it is coordinated by a master agent. Therefore, both communication and information analysis are more consistent.

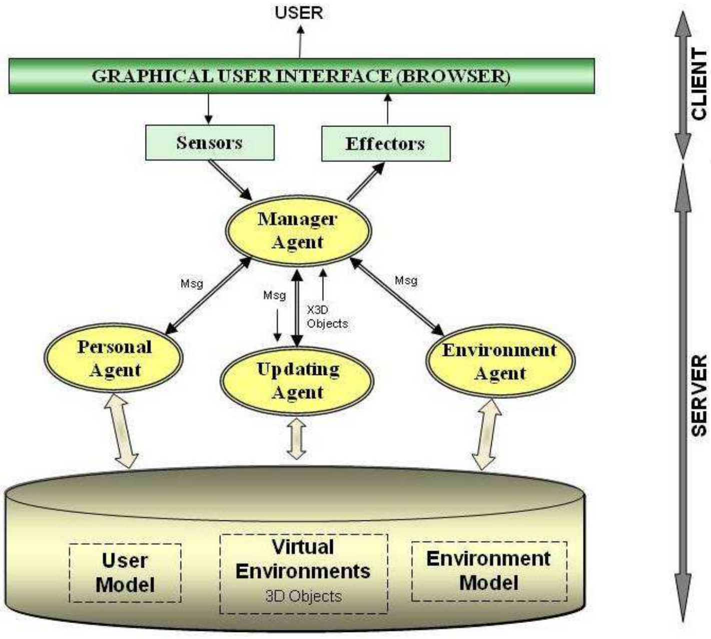


Fig. 1. The VEPersonal architecture

An important concept to be considered is levels of organization. It allows embedding one level into another. Therefore, it is appropriate to consider an organization as an aggregation of elements from a lower level as well as a component of the organization for a higher level (Ferber, 1999). Thus, in VEPersonal, the tasks were distributed according to the necessities to manage the environment and to generate the adaptations. It was defined which agents to do what, which resources should be used, and chosen goals and skills.

Four agents compose the society of agents (Aquino et al., 2007). The Manager Agent is responsible for monitoring the user’s actions as well as the current state of the virtual environment. It also determines the tasks to be carried out by other agents and coordinates communication between them. The Personal Agent determines the user’s profile and updates the UM whenever any change to the profile is detected. It has an inference system in its knowledge base that evaluates user's knowledge. The Environment Agent verifies the current state of the virtual environment and stores the changes into the EM. The Updating Agent is responsible for updating the virtual world based on the information provided by other agents.

During the decision-making process, the Manager Agent consults the Environment Agent about the current structure of the VE (stored in the EM). It also consults the Personal Agent to identify the user’s profile (stored in UM). This information is sent to the Updating Agent, which generates queries to a database to recover objects to be used for updating the virtual world. Next, the recovered objects are sent to the Manager Agent that generates the 3D structure of the virtual environment and sends it to the user interface. Following, the architecture of each agent with its respective functions is described.

3.1.1 Manager Agent

The Manager Agent is responsible for coordinating the other agents' actions: it gets the user's actions from the interface, it identifies the type of action to be performed, and it dispatches the action to the responsible agent. It also has methods to initiate the behaviors of others agents. Each behavior generates a message that includes the agent sender, the agent receiver, the message's content and the action to be carried through. The internal architecture of the Manager Agent is shown in Figure 2.

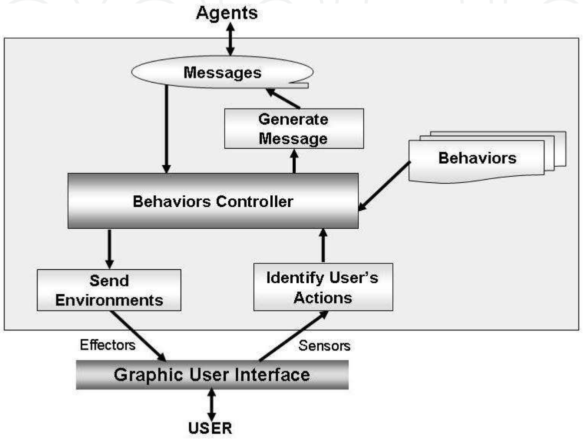


Fig. 2. Manager Agent architecture

The main component of the architecture is the Behavior Controller. It is used to supervise the requests sent to agents and verifies if the solicitations have been attended. The requests are carried out by message exchanging between agents. Its behavior is defined by a set of rules stored on its behaviors base.

3.1.2 Personal Agent

When the user accesses VEPersonal for the first time, an initial form is presented to him/her. Then, the user supplies information about points of interests, knowledge's level and needs. After the fulfillment of the form, the Personal Agent determines the initial user's profile. The form and the profile are stored in the UM. The determination of the profile and its updating is made by inference rules stored on the Profile Controller (Figure 3).

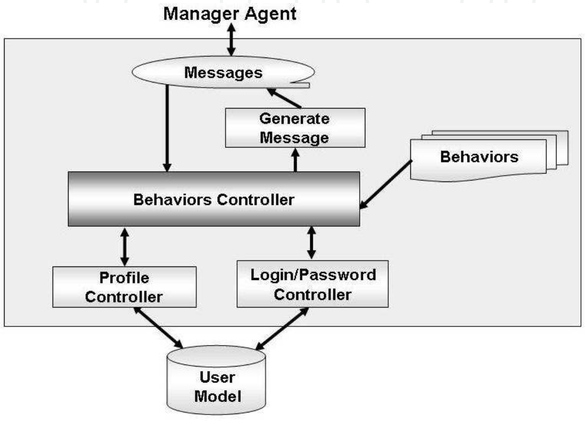


Fig. 3. Personal Agent architecture

The Personal Agent is a reactive agent. Therefore, it is only activated under a demand by the Manager Agent. The Personal Agent uses a logic-based system of inference that takes into account the user information that is held by the system. This technique is appropriate for reactive agents because the knowledge base of rules is activated due to the perception of the environment, and reacts through an action (for example, to store a specific information on the UM). The Personal Agent is responsible for supplying the user profile whenever the Manager Agent requests so.

3.1.3 Environment Agent

The Environment Agent is responsible for verifying the occurred modifications in the environment and for updating the Environment Model. This agent is also reactive. It is activated only when the Manager Agent demands it.

The Environment Model is updated whenever the Updating Agent generates a new virtual world or it defines new objects to be inserted into the current environment. This information is sent to the Manager Agent and dispatched to the Environment Agent that stores the modifications on the EM (Figure 4). The Environment Agent provides current information about objects of the environment (stored on the EM) whenever the Manager Agent requests it.

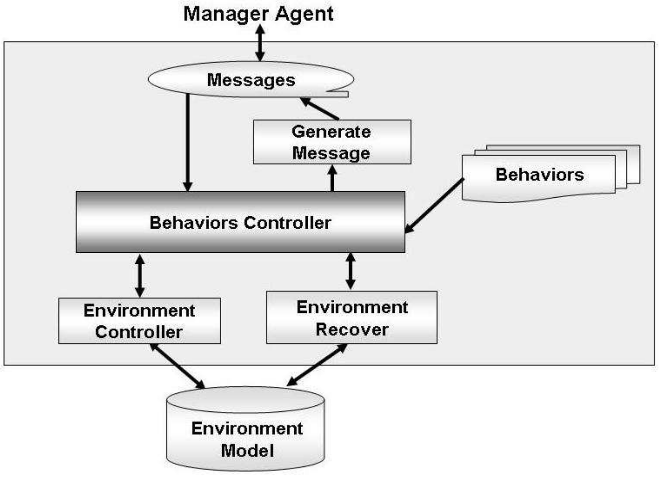


Fig. 4. Environment Agent architecture

3.1.4 Updating Agent

The Updating Agent is responsible for determining the new objects to be requested from the DBMS and added to the VE for updating the world (see Figure 5). It is a cognitive agent because it has objectives to reach, plans to elaborate and actions to perform.

Amongst its objectives is the updating of the VEs according to the user needs. Plans must be elaborated to request objects from the DBMS and to deliver new objects to the Manager Agent. A sequence of actions, such as identifying the user profile, generating queries to the DBMS and communicating with other agents must be determined.

The Updating Agent uses its knowledge base to evaluate the current state of the environment and, according to the objectives to be reached, it determines the necessary modifications to adapt the system to the user's profile.

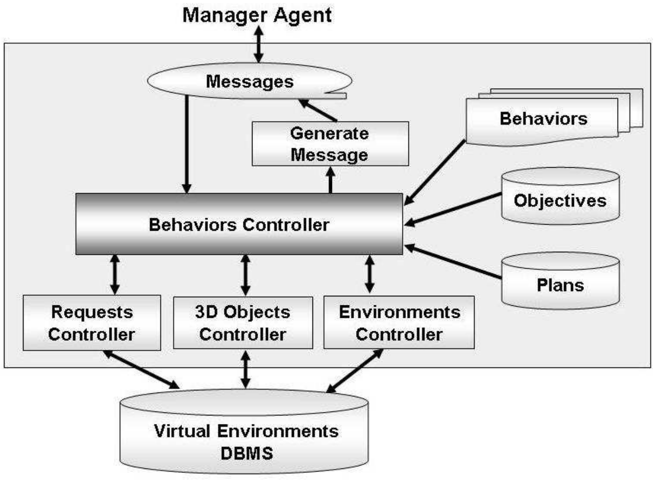


Fig. 5. Updating Agent architecture

3.2 The Agents' communication

Communication between agents is accomplished by using JADE (Java Agent Development framework, 2011), an application based on agents according to the Foundation for Intelligent Physical Agents specifications (FIPA, 2011). Message exchanging between agents is governed by a set of communication terms indicating the intent of the sender (the agent) to send a message. This communication promotes the necessary information retrieval that will allow updating the environment. Examples of communication terms defined by FIPA are: REQUEST - the issuer is requesting for some action to be performed; INFORM - the issuer is informing the result of some action; QUERY - the issuer is asking for recovering the reference to some object.

Agents' behavior is obtained considering the following features: user registration; a new environment request; an already visited environment request; a profile change assessment; and to determine possible changes to the user's profile. Each of these features is triggered by the Manager Agent, since it controls the actions of other agents on the system (Aquino & Souza, 2010). Due to the above, knowledge can be seen as a result of interactions between cognitive agents. Concepts, theories and laws, thus, can come out from a process of confrontation, objections, proof and refutations (Ferber, 1999).

Both confrontations with the world and communications with other agents are equally present under the concept of "interaction". Thus, specific learning that comes from confrontations between individuals and the world allied to the know-how that comes from interactions one individual can have with other individuals is another way to define knowledge (Ferber, 1999). As an example, Figure 6 describes the case where the user requests an environment that he/she has not yet visited (the system store all users' visited environments in the EM).

The interface sends the request to the Manager Agent. The later, initially, asks the Personal Agent for the user profile via a message (Msg 1. QUERY-REF). This agent queries a database to retrieve the user's knowledge level on the environment (userLevel) assigned to the current user, e.g. "beginner". When receiving the user profile from the Personal Agent (Msg 2. INFORM_REF), the Manager Agent asks the Environment Agent whether the user has already visited the requested environment.

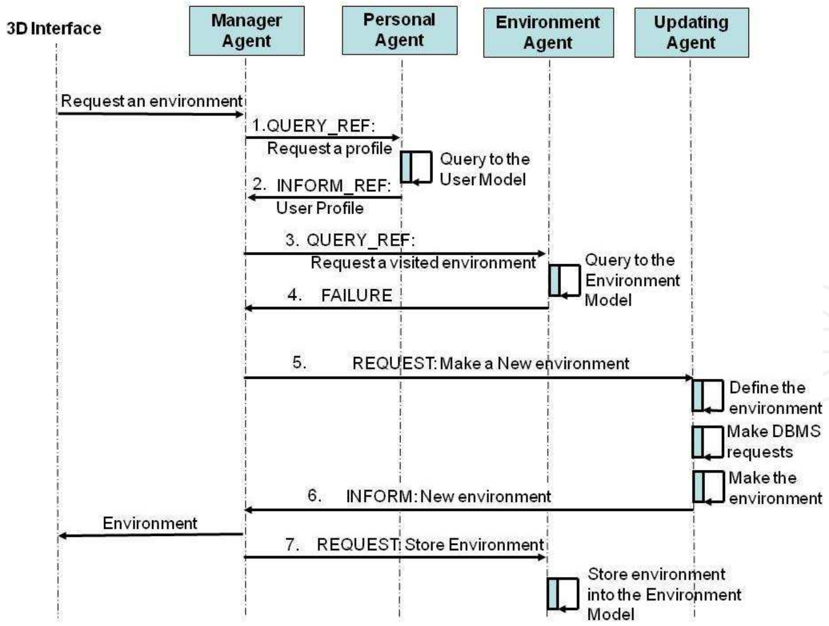

3D Interface
Manager Agent
Personal Agent
Environment Agent
Updating Agent
Request an environment
1.QUERY_REF: Request a profile
Query to the User Model
2. INFORM_REF: User Profile
3. QUERY_REF: Request a visited environment
Query to the Environment Model
4. FAILURE
5. REQUEST: Make a New environment
Define the environment
Make DBMS requests
Make the environment
6. INFORM: New environment
Environment
7. REQUEST: Store Environment
Store environment into the Environment Model

Fig. 6. Request for a new environment

The Environment Agent queries the Environment Model to learn if there is an association of any environment to the user (Msg. 3. QUERY_REF). Since it is a request for a new environment, the answer is empty, indicating that the user has not yet visited that environment (Msg 4. FAILURE). Each environment deals with a particular knowledge domain e.g. experiments on Physics.

Having the answer, the Manager Agent sends a message to the Updating Agent requesting a new environment (Msg 5. REQUEST), for example, a Free Fall experiment from the Physics domain. This agent performs the following steps: Environment definition – queries the database to find out the identity of the environment to be retrieved and objects that belong to such an environment; Database queries generation – retrieves the X3D code of objects belonging to the environment; and Environment Generation – to build the environment with the X3D codes of objects, inserting the X3D file header.

Next, a message from the Updating Agent is sent to the Manager Agent to inform that the operation has succeeded (Msg 6.INFORM_REF), and the X3D code for the created environment is appended. The Manager Agent sends the environment to the Web interface that, for its turn, sends it to the user's browser. The former sends also a message (Msg 7.REQUEST) having the objects that compose the environment to the Environment Agent, which stores such data into the Environment Model.

Once the message with the user profile changing received, the Manager Agent sends through the interface, a message to the user stating that there was a change from his/her profile and that the environment will be updated. It is given the option to accept the change or not. This procedure is performed in order that the user may be aware that changes will occur in the environment.

If such a user accepts that (see Figure 7), the Manager Agent asks the Environment Agent for the current environment that the user is interacting with (Msg 1. QUERY_REF). The Environment Agent queries the Environment Model; it retrieves the list of objects belonging to the current environment; and sends it to the Manager Agent (Msg 2. INFORM_REF). This agent writes a message to the Updating Agent to request the generation of the environment with the new profile (Msg 3. REQUEST).

The role of the Updating Agent now is to search into a database the list of objects that have the new profile defined for the user e.g. "intermediary". The objects found for this profile are retrieved. A message to return the list of names of objects with their X3D codes is then sent to the Manager Agent (Msg 4. INFORM_REF). Next, the Manager Agent sends that list to the interface.

The purpose of this list of objects (names plus X3D code) is to allow the interface to perform only the replacement for objects that need to be modified in the environment. Thus, it is not necessary to reload the entire VE due to the profile changing, which reduces the overhead in the process of data transmission while avoiding the scene rebooting as well.

The updating of the environment is also informed to the Environment Agent (Msg 5. REQUEST). A list with the names of the objects is sent to this agent that will update the Environment Model accordingly.

VEPersonal uses a Web environment for communicating with the user who is connected remotely. An interface for communication between the user and the system has been developed to achieve the synchronization of information in order to be a dynamic environment. Due to the amount of information that must be stored, retrieved and processed in a VE, a client-server application becomes more appropriate, decreasing the burden on the client machine. All the communication process will be detailed on the following subsection.

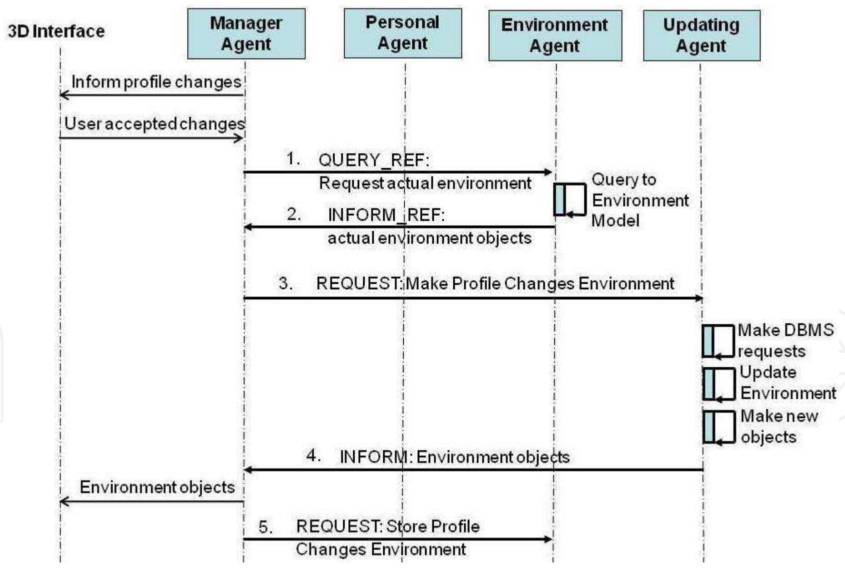

3D Interface
Manager Agent
Personal Agent
Environment Agent
Updating Agent
Inform profile changes
User accepted changes
1. QUERY_REF: Request actual environment
Query to Environment Model
2. INFORM_REF: actual environment objects
3. REQUEST: Make Profile Changes Environment
Make DBMS requests
Update Environment
Make new objects
4. INFORM: Environment objects
Environment objects
5. REQUEST: Store Profile Changes Environment

Fig. 7. Generation of a new VE depending on Profile Changing

3.3 Client-server communication

The interface task is to communicate the actions performed by the user to the Manager Agent, in addition to receiving objects to update the virtual world. This updating can be by sending a complete X3D environment, insertion of new objects (tags X3D) or removal of objects (by name) that exist in the VE. The interface should dispose means to aid the Manager Agent to carry out such operations.

The Xj3D browser (Xj3D, 2007) performs the visualization of virtual worlds. This browser is an open source project of the Web3D Consortium (Web3D, 2007), developed in Java, which allows viewing X3D and VRML environments. Applications using Xj3D have been developed for simulation, training and education as well as for medical visualization, games and projects involving viewing sensors (Matsuba, 2007). Besides that browser, the API Scene Access Interface (SAI, 2007) was used to establish communication between X3D environments and Java. This API allows for the insertion, deletion and modification of the objects in the Xj3D browser in real time.

The integration of these features in the Web environment used the Java Web Start technology (JWS, 2007) in order to enable the execution of Web applications, managing services and communications protocols. JWS is initiated automatically when it is done the first download of the VEPersonal application. JWS stores the application locally, thus, all the subsequent initiations are nearly instantaneous, since all of the needed resources are already available.

On the client side, the user accesses the system via the browser and can perform actions such as registration and authentication. When he/she requests the generation of the 3D world, a Java application is initiated to monitor the user actions in the environment and to inform to the server about the execution of an action (e.g. click of the mouse or proximity to objects). This application is also responsible for receiving the virtual world to be loaded or objects to be included and/or excluded in/from such a world (Figure 8).

On the server side, when an action is received from the user, the Manager Agent is responsible for determining what should be the actions to be performed. These actions are transformed into messages from the protocol and sent to the other agents. The society of agents takes the necessary decisions and returns to the interface the corresponding updating of the virtual world, if necessary.

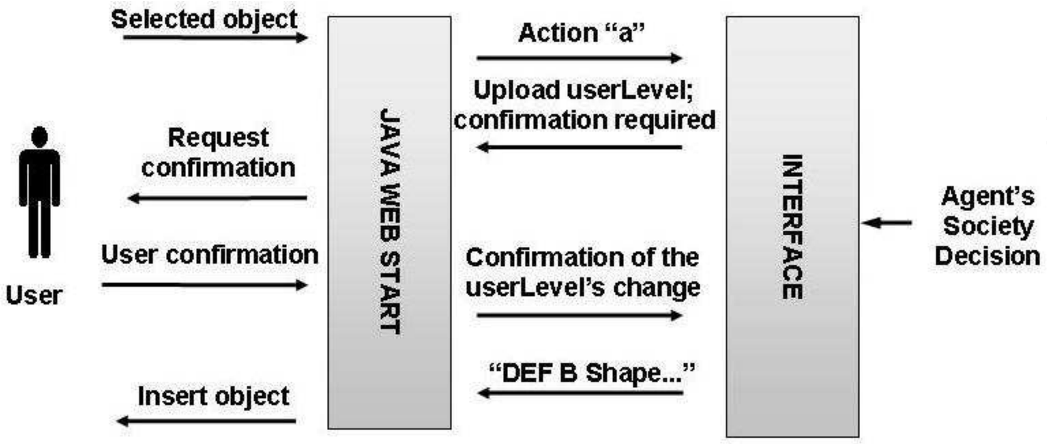


Fig. 8. Client-server interface of VEPersonal using Java Web Start

The communication interface of VEPersonal was structured in a way that the client-server connection could be performed decoupled. Figure 9 presents the structure of the interface and the project's design patterns used in the communication between the client and the server.

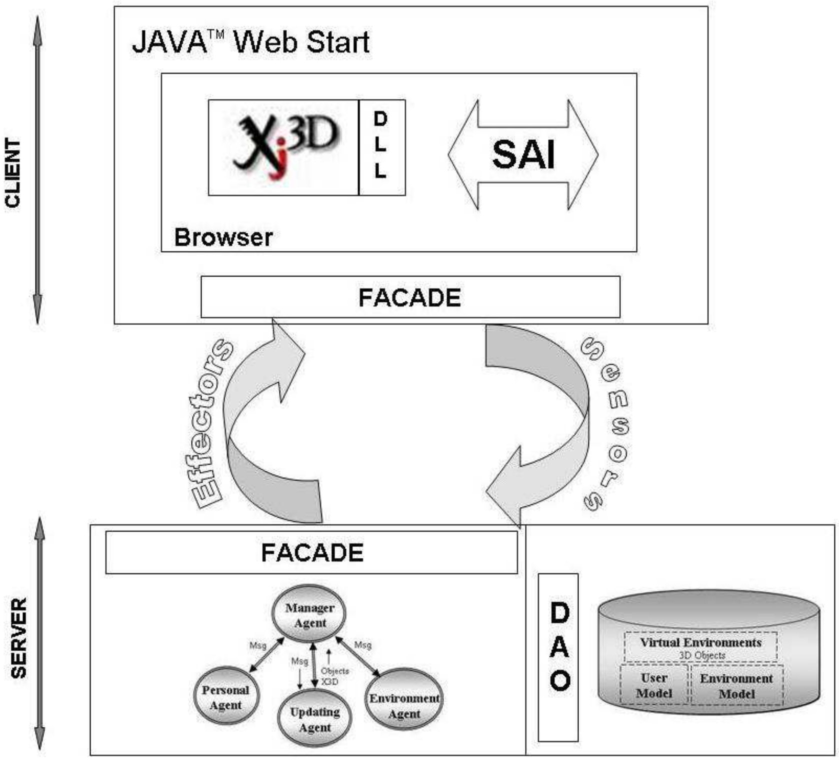


Fig. 9. The VEPersonal's interface structure and the used design patterns

On the client machine, the execution of the Xj3D browser needs various dynamic-link libraries (DLL, 2007). To make the application decoupled, that is, independent of the location of the library, it was built a Load function that locates and loads automatically to the client's machine the required libraries to the application. In this case, the application can run on any machine because it does not depend on external libraries.

The Interface in VEPersonal is responsible for communicating the application (which is running in the browser) with the server. This communication occurs through sensors and effectors. The sensors notify the changes made on the browser, identifying the object and the type of action performed by the user. For the implementation of the sensors it has been defined a method (Listener), which uses the design pattern Observer. This pattern defines a dependence on one-to-many between objects, so that when an object changes its state, all their dependents are automatically notified and updated. In this case, the Listener is used to propagate to the server the changes occurring in the browser.

Effectors are responsible for sending objects and virtual environments from the server to the browser. Both effectors and sensors use the framework LipeRMI (LipeRMI, 2007) for data transfer between the client and server via Web.

LipeRMI re-implements a Remote Method Invocation (RMI, 2007) so that the calls are made over the Internet, minimizing the use of bandwidth for communication (reducing the number of active connections). Once the client is already connected, this connection should remain open throughout the session for data exchanging between client and server (or until the application allows the connection to be closed). Hence the server will never need to open a connection from server to client, since it may use the existing connection. Therefore, the client is not concerned with firewall control - solving the biggest RMI problem.

By using LipeRMI, communication is defined by events that activate the connections of the client and the server, checking when a connection is initiated and terminated. The main events performed by the client during communication with the interface are: to create a connection when the user logs on; to send messages to the server to inform users actions; and to get messages sent by the server. The server on its turn shall wait for a client message and, therefore, remains on waiting state performing the listening event. Once established a connection with the client, the server performs a sending or receiving event.

In order to abstract features of the browser and to decrease the coupling with the server, making the communication independent of the employed technology, the *Facade* design pattern was adapted. *Facade* is a structural pattern that aims at hiding the details of a process by creating a *Facade* class. The client, then, just calls a method of *Facade* class and this class is responsible for executing this process. A *Facade* was also developed for the server, specifying the patterns of communication with the interface.

The methods used by the interface to the *Facade* standard are: openEnvironment, setEnvironment, get ObjectList, addObject, removeObject, getX3dEnvironment, setUser-Profile, addVEPersonalEventListener, removeVEpersonalEventListener, exitEnvironment.

The Interface of VEPersonal performs client-server communication with high cohesion and low coupling due to the design patterns used. Such a police clearly allows the alteration or substitution of the employed technologies, whenever occur the launching of new technologies or evolution of the already existing.

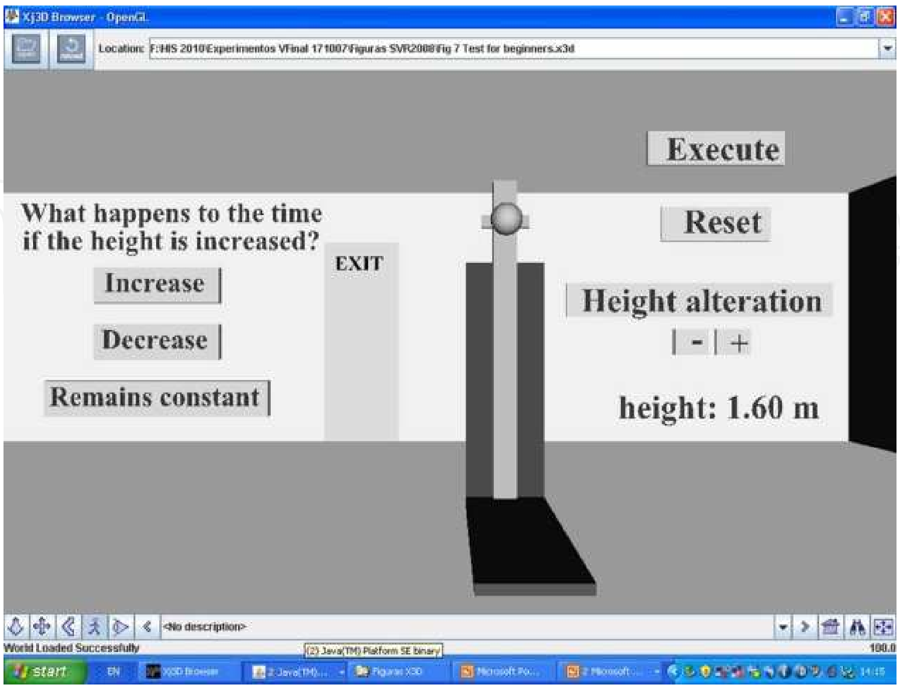
4. Adaptivity in virtual environments – An example

We present in this section an example to illustrate the benefits that adaptivity can bring to VEs. Particularly, our solution can be used in environments with different levels of complexity for any given situation, which adds more flexibility to such systems. Thus, let´s consider a VE designed to teach Physics. It can present an experiment with different levels of questions regarding the student's knowledge. On the other hand, the system can allow the user to access a new environment (a room with other experiments) or new details if his/her profile is suitable for that.

Figure 10 shows the student interaction with a Free Fall experiment. The objective of the experiment is to study the acceleration of gravity. For each experiment the student can consult the underlying theory and perform further interactions. At this time, new objects are recovered from a database to be inserted into the virtual environment. The objects are defined in X3D language that is appropriate to construct VEs in the Web, as mentioned before.

For example, the object "evaluation test" is presented according to the interactions carried out by the user. This test is specific to the knowledge level of the student (according to his/her profile). The test for a beginner student, presented in Figure 10, is composed of a question and three answer options.

The test insertion into the environment is determined by VEPersonal's agents that receive information on student's actions, e.g. the number of executions of the experiment and the consulted theory. Rules to determine what to modify and when this must occur are inserted into the knowledge base of the Update Agent. This agent recovers X3D objects from the DBMS using information supplied by both the Personal and the Environment agents.

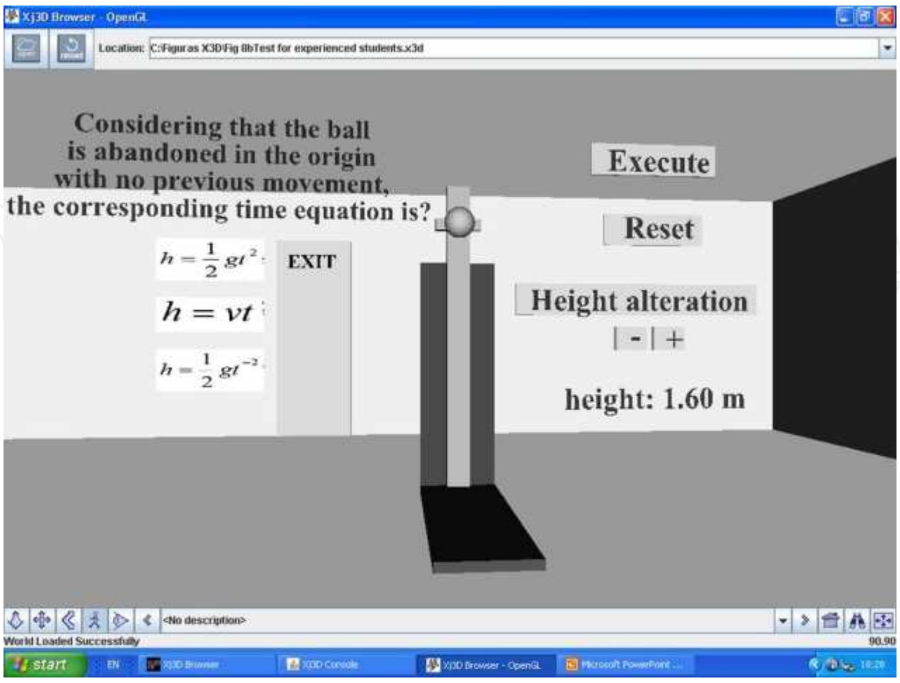

Location: F:\HIS 2010\Experimentos VFinal 171007\Figuras SVR2008\Fig 7 Test for beginners.x3d

What happens to the time if the height is increased?

height: 1.60 m

Fig. 10. Test for beginners

In the case of an experienced student (this information is provided by the Personal Agent through consultation to the user model of the student), a more elaborated question (with bigger rank of difficulty) is presented (Figure 11). Such a question is another object that replaces the beginners' question one. This object is also determined by the Update Agent.


Location: C:\Figuras X3D\Fig 8bTest for experienced students.x3d

Considering that the ball is abandoned in the origin with no previous movement, the corresponding time equation is?

height: 1.60 m

Fig. 11. Test for experienced students

Another advantage for constructing VEs with objects that are retrieved from a database is the possibility to determine the user's access priority. For example, the access to a room that has experiments on a given concept will only be granted to students that reached an adequate level of knowledge.

The Personal Agent is always bringing up to date the student's profile according with the number of right answers, number of experiments' executions, the kind of consultations carried out, amongst others. Thus, this agent is capable of determining whether the user has specific access priority to a given environment. Should the student possess appropriate knowledge for Newton Laws, a door to the right would be inserted into the virtual world. This permits the student to enter next room and to access information about such laws.

In this case, the Manager Agent, responsible for insertion/removal of objects, requests new objects to update the user's VE. After that, the Update Agent receives information from both the user's User Model, envoy by the Personal Agent, and the current state of the environment, envoy by the Environment Agent. Using such information, the Update Agent identifies the user's access priority and recovers the object "door" from the database. This object is envoy to the Manager Agent that inserts it into the user's VE. This modification in the environment is made immediately, without the necessity to reload the entire virtual environment. This example, shown in Figure 12, is the result of a real time adaptation according with modifications on the user's profile characteristics.

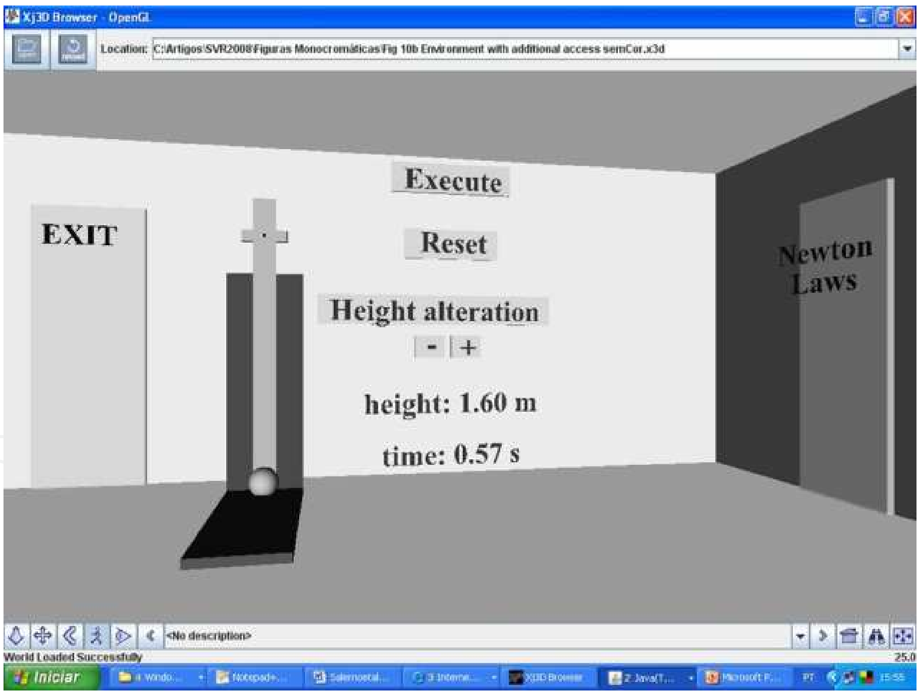

Xj3D Browser - OpenGL
Location: C:\Artigos\SVR2008\Figuras Monocromáticas\Fig 10b Environment with additional access semCor.x3d
EXIT
Execute
Reset
Height alteration
- +
height: 1.60 m
time: 0.57 s
Newton Laws
<No description>
World Loaded Successfully
25.0
Iniciar

Fig. 12. Environment seen by user with additional access

5. Conclusions

This chapter discussed the benefits brought by adaptivity to VEs. Particularly, it has focused on the role played by agents on this context. Thus, we presented a way to build adaptive VE systems by using a society of agents. Two important points must be considered in order to increase the capacity of adaptive VEs: (i) to record properly the current state of the virtual environment; and (ii) to keep track of the modifications carried out by their users. These allow keeping the virtual environment up to date on both situations: during the current user interaction as well as on the next ones. All needed data are modeled into both a User Model (UM) responsible for all user characteristics, and an Environment Model that has to do with all properties of the virtual worlds.

In order to carry out the tasks described above it is necessary a quick and efficient decision making process allied to a precise implementation of the reached decisions. By automating the whole process, agents play the most important role to construct VEs that can adapt to their users' needs on real time.

A large number of applications exploit the User Model in order to adjust themselves to the necessities of the user or to propose changes that are of his/her interest. In 3D environments, particularly in adaptive VEs, the UM is a very important element, since it also stores the behavior of the user; his/her style of navigation; forms of interaction with the environment; and the evolution of the user's learning in the system. Thus, agents will be able to carry out the decision-making process for generating virtual worlds and eventual adaptations using the information contained in the UM.

The main concern with the representation of the UM in a virtual environment must be guided by the identification of the objects that the user can manipulate (to visualize, to approach and to select; existing different levels of perception for each case). It is also necessary to consider his/her interest for determined contents; his/her capacity of interaction; and, mainly, his/her evolution during the learning process. To achieve this, it is needed to construct a framework that may store specific user information (level of knowledge and manipulated objects, for instance) and of a model based on logic that interprets the necessary information to be represented in the UM taking into consideration the objectives and beliefs of the user.

The use of intelligent agents in VEs has the objective to increase the capacity of analysis of the environment by the distribution of tasks and by the incorporation of user modeling techniques on its knowledge base. In this context, intelligent agents contribute to increase the capacity of monitoring the user's actions and the identification of his/her profile. Consequently, the construction of an adaptive VE more precise and more dynamic can be carried out with better accuracy, due to the capacity of agents in interpreting the "intentions" of the user.

6. References

Aquino, M. S. & Souza, F. F. (2010). An intelligent web interface to generate and update adaptive virtual environments. *Proceedings of 10th International Conference on Hybrid Intelligent Systems*, v. 01. pp. 51-54, Atlanta, Georgia, USA, August 2010.

Aquino, M. S.; Souza, F. F.; Frery, A. C.; Souza, D. A. C. M.; Fujioka, R. C. & Vieira, M. M. S. (2008). An Infrastructure to Generate Adaptive Virtual Environments with the Management of Client-Server Communication in Real Time. *Proceedings of X*

Symposium on Virtual and Augmented Reality, Porto Alegre: Sociedade Brasileira de Computação, pp. 61-69, João Pessoa, Paraíba, Brazil, Mai 2008.

Aquino, M. S.; Souza, F. F.; Frery, A. C; Souza, D. A. C. M. & Fujioka, R. C. (2007). Supporting Adaptive Virtual Environments with Intelligent Agents. *Proceedings of 7th International Conference on Intelligent Systems Design and Applications - ISDA'07*, pp. 217-222, Rio de Janeiro-RJ, Brazil, October 2007.

Aquino, M. S.; Souza, F. F. & Frery, A. C. (2005). VEPersonal - An Infrastructure of Virtual Reality Components to Generate Web Adaptive Environments. In: Simpósio Brasileiro de Sistemas Multimídia e Web, WebMedia 2005, Poços de Caldas-MG, Brazil, *ACM International Conference Proceeding Series*, pp. 1-8, New York, USA: ACM Press, v. 125.

Bellifemine, F.; Caire, G.; Poggi, A. & Rimassa, G. (2003). *Java Agent DEvelopment Framework - JADE - A White Paper*. In: Telecom Italia EXP magazine, Volume 3, n. 3, September 2003. Available: http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf

Cai , L.; Zheng, X.; Qu, H. & Luo, Z. (2011). Risk Accident Simulation Using Virtual Reality and Multi-agent Technology. *JDCTA: International Journal of Digital Content Technology and its Applications*. Vol. 5, No. 2, (February, 2011) pp. 181-190, ISSN: 2233-9310 (Online), ISSN: 1975-9339 (Print).

Celentano, A.; Nodari, M. & Pittarello, F. (2004). Adaptive Interaction in Web3D Virtual Worlds. *Proceedings of Web3D 2004 - 9th International Conference on 3DWeb Technology*: ACM Press, ISBN:1-58113-845-8, pp. 41–50, Monterey, California, USA.

Chittaro L. & Ranon R. (2007). Adaptive Hypermedia Techniques for 3D Educational Virtual Environments. IEEE Intelligent Systems, Vol. 22, No. 4, (July-August 2004), pp. 31-37, ISSN: 1541-1672.

Chittaro, R.; Ranon, R. & Ieronutti, L. (2003). Guiding Visitors of Web3D Worlds through Automatically Generated Tours. Proceedings of Web3D 2003: 8th International Conference on 3D Web Technology, ACM Press, New York, pp. 27-38.

Chittaro, L. & Ranon, R. (2002a). New Directions for the Design of Virtual Reality Interfaces to E-Commerce Sites. *Proceedings of AVI 2002: 5th International Conference on Advanced Visual Interfaces*, ACM Press, New York, pp. 308-315.

Chittaro, L. & Ranon, R. (2002b) Dynamic generation of personalized VRML content: a General Approach and its Application to 3D E-Commerce. *Proceedings of Web3D 2002: 7th International Conference on 3D Web Technology*, ACM Press, New York, pp.145–154.

Chittaro L. & Ranon R. (2000a). Adding Adaptive Features to Virtual Reality Interfaces for E-Commerce. *Proceedings of AH-2000: International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, Lecture Notes in Computer Science 1892, Springer-Verlag, Berlin, pp. 86-97.

Chittaro, L. & Ranon, R. (2000b). Virtual Reality stores for 1-to-1 E-commerce. *Proceedings of CHI 2000 Workshop on Designing Interactive Systems for 1-to-1 E-Commerce*, The Hague, The Netherlands.

Costa, R. M. E. M.; Souza, D. S. & Mendonça, I. (2010). Exploring Intelligent agents for controlling user navigation in 3D games for cognitive stimulation. *Proceedings of 8th*

International Conference on Disability, Virtual Reality and Associated Technologies – ICDVRAT, Chile, ISBN 978 07049 15022.

Dynamic-link library – DLL. Last access in September 2007. Available: http://msdn2.microsoft.com/en-us/library/ms682589.aspx

Ferber, J. (1999). *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley: ISBN-10: 0201360489, ISBN-13: 978-0201360486, London, UK.

Frery, A. C.; Kelner, J.; Moreira, J. & Teichrieb, V. (2002). Satisfaction Through Empathy and Orientation in Three-Dimensional Worlds. *CyberPsychology & Behavior*, Mary Ann Liebert (ed.), Vol. 5, pp. 451-459.

FIPA – Foundation for Intelligent Physical Agents. Last access in April 2011. Available: http://www.fipa.org

Java Agent DEvelopment framework – JADE. Last access in October 2007. Available: http://jade.tilab.com

Jennings, N. R. (2000). On Agent-based Software Engineering, In: *Artificial Intelligence*, Publisher: Elsevier, Volume 117, Issue 2, pp. 277-296.

Java Web Start Technology- JWS. Sun Developer Network. Last access in April 2007. http://java.sun.com/products/javawebstart

Lesser, V. R. (1999). Cooperative Multiagent Systems: a Personal View of the State of the Art. *Knowledge and Data Engineering, IEEE Transactions*, 11(1), pp. 133-142.

LipeRMI – A Light Weight Internet Approach for Remote Method Invocation. Last access in September 2007. Available: http://lipermi.sourceforge.net

Matsuba, S. N. (2007). *Interview Yumetech – Xj3D*. 3d-test – 3D temps réel et interactive. Last access in April 2007. Available: http://www.3d-test.com/interviews/xj3d_1.htm

Papatheodorou, C. (2001). Machine Learning in User Modelling. *Machine Learning and Its Applications*. Lecture Notes in Artificial Intelligence. Springer-Verlag, ISBN: 3-540-42490-3, London, UK.

Remote Method Invocation – RMI. Last access in September 2007. Available: http://java.sun.com/javase/technologies/core/basic/ rmi/index.jsp

Russell, S. J. & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Norfolk: Prentice Hall, New Jersey, Third edition.

Santos, C. T. & Osório, F. S. (2004). An intelligent and adaptive virtual environment and its application in distance learning. *Proceedings of the Working Conference on Advanced Visual Interfaces*. ACM Press, Gallipoli, Italy, pp. 362 – 365.

SAI - Scene Access Interface Tutorial. Last access in April 2007. Available: http://www.xj3d.org/tutorials/general_sai

Tavares, T. A.; Oliveira, S. A.; Canute, A.; Gonçalves, L. M. & Filho, G. S. (2005) A multi agent system for 3D media spaces assistance. Proceedings of *Third Latin American Web Congress*. LA-WEB 2005, ISBN: 0-7695-2471-0.

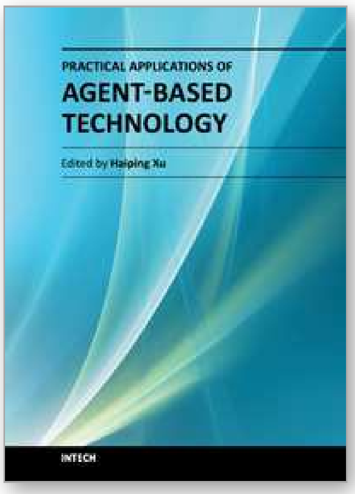
Tavares, T. A.; Araújo, A. & Souza Filho, G. (2001). *ICSpace – An Internet Cultural Space*. In: Active Media Technology. 6th International Computer Science AMT2001. Honk Kong, China, pp. 389-402.

Web3D Consortium. Open Standards for Real-Time 3D Communication. Last access in April 2007. Available: http://www.web3d.org

Wooldridge, M. (2009). *An introduction to multiagent systems* (2nd ed.), John Willey & Sons Ltd., ISBN 978-0-470-51946-2, United Kingdon.

X3D International Specification Standards. Last access in April 2011. Available: http://www.web3d.org/x3d

Xj3D - Java based X3D Toolkit and X3D Browser. Last access in April 2007. Available: http://www.web3d.org/x3d/xj3d



Practical Applications of Agent-Based Technology
Edited by Dr. Haiping Xu

ISBN 978-953-51-0276-2
Hard cover, 136 pages
Publisher InTech
Published online 18, March, 2012
Published in print edition March, 2012

Agent-based technology provides a new computing paradigm, where intelligent agents can be used to perform tasks such as sensing, planning, scheduling, reasoning and decision-making. In an agent-based system, software agents with sufficient intelligence and autonomy can either work independently or coordinately with other agents to accomplish tasks and missions. In this book, we provide up-to-date practical applications of agent-based technology in various fields, such as electronic commerce, grid computing, and adaptive virtual environment. The selected applications are invaluable for researchers and practitioners to understand the practical usage of agent-based technology, and also to apply agent-based technology innovatively in different areas.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821