A Flexible Agent-Based Framework to Control Virtual Characters

Luís Moniz, Graça Gaspar, Ricardo Abreu, Ana Paula Cláudio, and Maria Beatriz Carmo

Department of Informatics, University of Lisbon, Portugal
{hal,gg}@di.fc.ul.pt ricardolafabreu@gmail.com {apc,bc}@di.fc.ul.pt

Abstract. We present a generic platform, called IViHumans, for materializing a system of multiple intelligent agents into virtual humans. It is intended to be flexible and applicable to diverse realistic simulation environments. The platform is composed of two layers: the Graphical Layer, and the Artificial Intelligence Layer, that can run autonomously. The architecture of our platform was conceived in order to allow the two layers to control the virtual agents at different detail levels.

1 Introduction

The simulation of virtual humans (VHs) is an immense challenge that requires the resolution of many problems in various areas. In order to simulate virtual humans in virtual worlds we have to combine realistic representations of humans, including the capability of expressing emotions, along with mechanisms to simulate the behaviour [1].

In 1987, Craig Reynolds was responsible for bridging the gap between artificial life and computer animation, introducing behavioural animation techniques [2]. Since then, intensive research in this area has been performed giving rise to several techniques widely used with many different purposes. Entertainment industry has been exploring and using these techniques in games like "The Sims" and films like "The Lord of the Rings". A different context of application, covered by an important number of approaches, allows the exploration of stressful social situations in the safety of the virtual world [3–5] and, fully or partially, covers a number of technical areas in Artificial Intelligence.

Research in VHs needs to use the results from Artificial Intelligence in order to achieve believable characters, while research in Multi-Agent intelligent systems can profit from virtual humans frameworks for evaluation and increasing the usability of the applications. Believable characters require that agents' behaviour should not be scripted, but should rather emerge as a result of autonomous agents' interaction, in a shared environment.

We have been developing a graphical visualization platform, called IViHumans, for multi-agent system execution with the intention of applying it to the development of realistic and compelling simulation environments for some real-world situations, such as training, education and entertainment. However it is

intended to be a generic platform, not specialized in a particular problem or domain.

To pursuit this goal, the IViHumans platform is composed of two separate layers, one for Graphical Processing (GP) and one for the Artificial Intelligence (AI) computation. A special concern then is the interconnection between these two layers. They must be able to express the control of the embodied virtual agents at different detail levels. Support for relevant aspects such as sensory honesty, as described in [6], or the attention focusing needed for effective planning must be distributed among the two layers, maintaining a clear separation of concerns and responsibilities. Also, the interconnection between the two layers must accommodate their potentially different response speeds and account for delays of communication between them, without compromising the believability of the virtual human characters.

In the last decade some other authors have presented work integrating multi-agent systems with graphics components. In [7] a game-oriented multi-agent framework, built over the JADE multi-Agent System platform is presented. That approach is similar to ours, in that the MAS module and the visualization module work independently of each other. However in their system the simulation is centralized in the MAS, having the visualization module with the sole duty of exhibiting the world without allowing interaction.

Several other authors have used BDI agent platforms together with game engines or specific visualization platforms. Some of those works concentrate on particular types of applications and agents. For instance, in [8] an environment based on the Unreal Tournament game engine and the Soar AI engine is described, concentrating in the development of complex AI agents for interactive drama applications. In [9] BDI agents that model expert players of Quake 2 are developed.

In [10, 11] the BDI model is also used to control animated characters in virtual worlds. In their work, 3D articulated characters are controlled in real time by cognitive agents that are clients of the environment which, in its turn, acts as a server and is responsible for managing the information that can be perceived by the agents. However their architecture is limited in that everything an agent can perceive must be listed a priori in a list of boolean statements.

In [12] the JACK agent language is extended with a specific perceptual/motor system that allows the agents to interact with a graphical interface. Our approach is instead to make the GP layer responsible for the perception and basic motion of the agents, thus decoupling the perceptual/motor system from the cognitive part of the agents and making it more independent of the agents implementation. In [13] a military simulation tool, ROE3, is presented where the agents are also implemented using JACK. In ROE3, an Integration Layer translates messages between the agent and the synthetic environment, being able to aggregate raw percepts into higher level percepts. In this respect, our platform incorporates a similar translation mechanism, as part of the AI Layer, in the form of interface agents.

In the following section we present the general architecture of the IViHumans platform. In section 3 a brief general description of the GP Layer is presented followed by some detail of the implementation of the movement of the synthetic characters. In section 4 we present the AI Layer, describing the several types of agents that constitute the MAS system, namely the interface agents, the intelligent agents and the monitoring agents. The last section summarizes this work and discusses some future work.

2 Architecture

The IViHumans platform separates the GP layer from the AI layer, an approach that was already found in works such as the JGOMAS system and the ROE3 architecture. However our proposal differs from them by having the amount of responsibility of each layer well balanced. For instance, in what regards sensory honesty, the physical limits to what can be perceived by a character are controlled by the GP layer while the cognitive restrictions reside on the AI layer. Also, the GP layer is responsible for quickly handling low-level aspects, such as collision response, while the AI layer deals with the more complex cognitive behavior, using symbolic representations. As a side effect, this architecture also reduces considerably the communication overhead.

The GP layer is built on top of the rendering engine *OGRE* (www.ogre3d.org) and relies on the rigid body dynamics engine *ODE*(www.ode.org). The Multi-Agent system – the core of the AI layer – is built upon the JADE (jade.tilab.com) platform. The choice of all the underlying software for the platform obeyed well defined criteria, such as the quality of the provided features, the price or the existence of an active community of users.

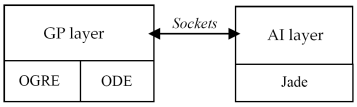

Fig. 1: The Graphical Processing and Artificial Intelligence Layers.

The communication between the two layers (figure 1) is performed by means of TCP sockets. Two reasons underlie this option: first, each layer uses a different implementation language (*C++* for the GP layer and *Java* for the AI layer); second, we intend to use the platform in a distributed environment. The GP Layer is responsible for the representation of all the elements contained in the virtual world, among which the virtual humans are the most important. Because the environments are dynamic, the GP layer must reproduce the appropriate animations that carry the flow of occurrences, consistently enacting the evolution of the world and its components.

3 Graphical Layer

The GP layer is responsible for exhibiting the evolution of the virtual world, representing all the elements contained in it (among which the VHs are the most important) constantly updating their state, and reproducing the appropriate animations.

The IViHumans platform handles VHs whose actions rely, for the time being, on three main skills: perception, movement, and emotion expression. A VH perceives its environment through a ray-casting synthetic vision algorithm [14]. Besides the effects of emotions on their behavior – something that the AI layer may take care of – the VHs can show composite facial expressions to immediately convey their internal emotional state. An explanation of the techniques used to accomplish emotional expression of the VHs can be found at [15].

The motion of the VHs is supported by the concept of *steering behavior*, according to what was introduced by Craig Reynolds at [16]. We also follow his proposal for hierarchically categorizing movement in three layers: locomotion, steering and action selection (in order of growing abstraction). The former two are under the domain of the GP layer, while the last and topmost layer can either be included in the AI layer – which happens for virtual humans – or be absent – so that human users can directly control avatars. The action selection layer operates by activating and parameterizing steering behaviors to achieve the desired goals, according to agents' planning. Locomotion corresponds to the choice of the appropriate animations on the basis of speed and according to rules that are unique for each VH model. In the remainder of this section we detail how steering behaviors are integrated in the GP layer, as a sample grounding base for the explanation of the activity of the AI layer.

3.1 Movement of Characters

To implement the steering of the virtual humans, we closely follow Reynolds' proposals, although we introduce a few ideas of our own. Because movement through steering behaviors could be applied to a myriad of entities, we decided to bring it apart from the implementation of any particular entity. This way, we created a class that models any moving entity as a point mass, as Reynolds did with his vehicle model. This *MovingCharacter* class is not associated with OGRE, except in that it uses some basic data types provided by this library (e.g. vectors).

A *MovingCharacter* is essentially characterized by a mass, a position, a velocity and a vector that specifies his facing direction. This facing vector may be automatically updated when the character moves so that it is always tangent to the path the *MovingCharacter* follows. When the character is still, it is left unchanged. However, we did not want to restrict the movement of a virtual human in this way and so the automatic update of this vector is not mandatory. It should be deactivated if one intends to have the character moving in any way that requires his local depth axis not to be collinear with the velocity vector, that is, if the character must move in one direction while facing other way.

The movement of the *MovingCharacter* is ruled by steering behaviors that specify the forces he should apply on himself. The movement produced by these forces is computed according to the basic laws of classical physics, except for the fact that it is restricted by maximum values for force and velocity. These, as well as some other values that do not vary, are loaded upon construction from a configuration file that is unique for each character.

Steering behaviours are sometimes criticised for being hard-wired into the code [17]. In an attempt to overcome this problem, we separate the character from the actual behaviors, employing polymorphism and object oriented design in general. A *MovingCharacter* may have one instance of *SteeringBehavior*, which is the common interface to all the steering behaviors that may be defined (see figure 2). Any instance of *SteeringBehavior* can be plugged in and out of the *MovingCharacter* at run-time.

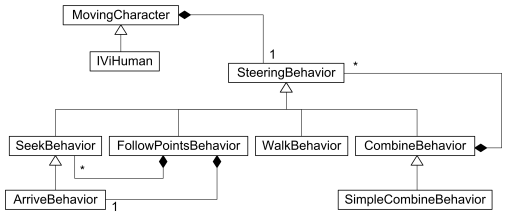

Fig. 2: Conceptual model of the main portion of the GP layer that deals with steering

The behaviors *SeekBehavior* and *ArriveBehavior* are directly inspired in the corresponding behaviors explained in [16]. The former returns a force that fully accelerates the character towards a target point, while the later does the same thing only until the character's distance to the target becomes lower than some predefined threshold. At this point, it will start to produce a force that is opposed to the movement and the character will decelerate until he eventually stops.

The behavior *WalkBehavior* does the same as one of the three rules that underlie the flocking behavior in [2]: it tries to match the velocity of the character with a given target velocity. This behavior is useful to have the AI control a virtual human at a lower level, as well as to have a user directly "piloting" an avatar. We also introduced the behavior *FollowPointsBehavior* that guides the character through a sequence of n targets by making him *seek* the first $n-1$ targets and *arrive* at the nth target. This last behavior is useful, for instance, to make the character follow a path plan. Besides these steering behaviors, many more may be implemented.

Although the *MovingCharacter* can only be associated with one *SteeringBehavior*, there is a special *SteeringBehavior* for combining more than one. It is called *CombineBehavior* and it is an abstract class that has the general functionality for coupling other *SteeringBehavior*s. This class can be extended to implement different ways of combining the forces calculated by encapsulated

behaviors. Currently, we have just one such implementation – which lies in *SimpleCombineBehavior* – that simply adds the forces returned by contained behaviors, returning the net force. In the future, we intend to add other ways of combining *SteeringBehavior*s (e. g. with priorities). This design follows the *Composite Pattern*. In what concerns locomotion, the *IViHuman* objects map

Fig. 3: VHs moving independently

the abstract movement functionality of the class *MovingCharacter* into actual observable movement, by applying the correct transformations to the model that represents the corresponding VH and animating it accordingly (figure 3).

4 Artificial Intelligence Layer

While the GP layer hosts the bodies of the virtual humans, the AI layer manages their minds. Each virtual human is controlled by one or more agents that entitle him with intelligent behavior. The evolution of the world is due to the effects induced by both layers.

In our architecture the AI layer comprises two main categories of agents: interface agents and cognitive agents. The cognitive agents, together with the service agents and meta agents, are the intelligent components.

In order to give some degree of control over the agents, we include another class of agents: monitor agents. These agents act as supervisors, providing a mechanism to manipulate the cognitive agents directly, forcing their behaviour and consequently interfering in the simulation.

4.1 Interface Agents

The interface agents, one for each virtual entity, manage the communication with the virtual entities, receiving sensory information and sending commands (figure 4). Although these agents can act as a raw connection between both layers, they have two additional functions: to provide a sensing/acting cycle that further separates the communication aspects of the control of the virtual

humans from the more complex, and possibly slower, cognitive aspects; and to offer a translation/filtering mechanism between crude data and symbolical representation. We can split these functions in four main components:

- **Sense**: requests sensory information from the GP layer at a defined rate, saving it into a buffer. The various requests from the cognitive agent component for sensor data are obtained from this buffer (the sensor data buffer). This isolates the sensor particulars (refresh rate and cycling) from the higher cognitive levels.
- **Act**: reads the next command from a buffer (the command buffer) and sends it to the GP layer. This feature detaches the agent cognitive level from the physical details, for instance, the number of commands that the GP layer is capable of processing in a time slot.
- **Sensor data translator/filter**: translates raw sensor information in symbolic equivalents or more abstract and constrained representations. This is achieved by splitting the information into clusters of similar data. For instance, a color name can correspond to an interval of values in the rgb gamma.
- **Command translator**: translates the higher level commands used by the cognitive components of the agent into the lower level commands used by the GP layer and saves them in the command buffer. The translation can be achieved by using predefined schemas for action decomposition. Another alternative is to incorporate a planner that produces in real time the desired action sequence.

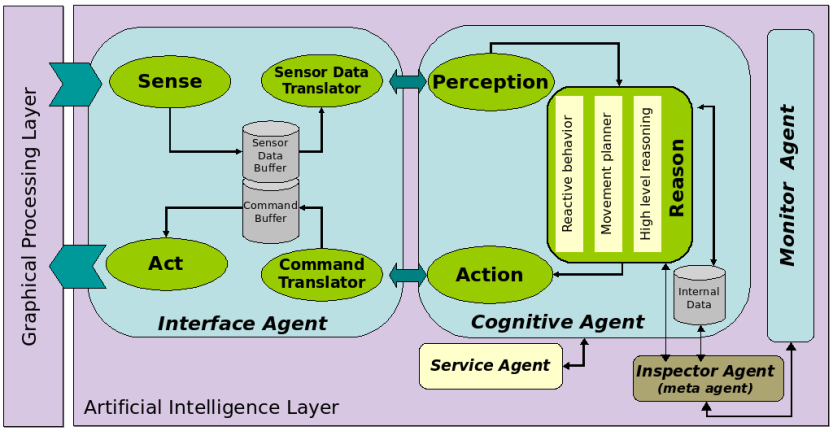

Fig. 4: Artificial Intelligence Layer detailed.

The GP layer, on the other hand, has its own interfacing component, composed by two unique objects: one that deals with the routing of messages to the

adequate recipients and another one that acts as a special receiver for messages that are not directed to any particular entity, being instead targeted at a global manager, capable of operating global changes over the environment.

The concrete entities that must be able to receive orders from the AI layer, such as the virtual humans, will do so through an adapter component that manages to unravel the meaning of messages, translating them to the appropriate method calls.

4.2 Intelligent components

The objective of these components is to provide an intelligent control over the virtual entities. We can classify these components according to their main function into three categories: cognitive agents; service agents; and meta agents. As stated before, each entity in the virtual world is controlled by the intelligent component; this control is performed by a cognitive agent that acts as its mind (figure 4). Each of these agents has access to the interface agent to request sensory information and issue commands. Our architecture does not impose any restriction on how these agents are organized internally. The system designer is in charge of creating the control and coordination mechanisms to achieve the desired behaviours. Although we view a cognitive agent as a single entity, our system allows this agent to act as a representative of an agency, where a group of heterogeneous agents interact in the support of the virtual entity. Our base architectural template for these groups is a centralized society, with a central coordinator, and one external interface agent. In the example of figure 4 the agent main block corresponds to the reason structure, in this case a layered organization of progressively more complex behaviours.

Service agents, as the name suggests, provide extra services to the community. They can be shared by various cognitive agents, providing common facilities, available to all, instead of having each agent support its own version of the commodity. We group these services into three classes: core, aggregation and external services. Core services correspond to simple tasks that a single provider can solve. For instance, a path planner service, from which an agent can request a sequence of actions to reach a desired location. Aggregation services correspond to functionalities that support formation and cohesion of agencies. For instance, an internal message service that allows the agents to communicate. This feature should be carefully used; it allows building virtual communities without regarding the spacial location of the represented entities. In order to prevent undesirable use, we propose to develop this service on top of the platform messaging system incorporating adequate restriction mechanisms. Finally, external services provide agents with links to outside sources. For instance, it could be possible to create a remote web page that controls a virtual entity.

Meta agents act as invisible entities tracking, observing and controlling other agents' performance and behaviour. This kind of agent is not perceived by cognitive and service agents, it can act as an external observer that obtains information data and extracts conclusions. These agents have to be incorporated into the cognitive agency architecture in order to access private data, also they could

provide a direct control mechanism over the other agents. For instance, if the user needs to guide the simulation into a specific track, he has a tool that can be used to directly modify the internal state of an agent, to add information or even to force some desired behaviour.

4.3 Monitoring Agents

We also propose the inclusion of other agents; these provide the user with custom interfaces to the MAS and give him some degree of control over the simulation. These agents use the meta agents in order to access the inner information of the cognitive agencies. An example is a monitor agent acting as an agency supervisor, allowing the user to inspect and control the behaviour of a group of agents, that control a virtual character. Another subgroup of monitoring agents corresponds to automatic triggers in the simulation; if a threshold is passed they can start an action. We could extrapolate some application scenarios: the action can have a direct effect in the simulation itself, for instance, if a new entity is created or a new scenario area is opened, several actions can be initiated; a process of data collection is started to measure some parameters; and the migration of agents to other machines if the overall performance is degraded.

We conclude this section by pointing out some key points of our architecture: independency from the graphical layer — although designed for this particular scenario, it can be used in other graphical environments; distributed characteristics (due to the JADE framework) that allow the simultaneous use different computers; and scalability by adding new agents/features/services and control facilities, tuning the system to a particular simulation.

5 Conclusions

The prevailing opinion of people working in the domain of virtual environments inhabited by virtual humans appears to be that there is still a field for subsequent research, despite the number of existing contributions. This judgment is primarily sustained by the fact that current results are still far from honestly mimicking human behavior in its interaction with the environment and with its peers.

In this paper we present a framework for construction and management of artificial characters in a virtual environment. We have focused mainly on the overall architecture of our system and we have presented, in a shallow form, a number of aspects that deserved to be treated in more depth. These include a complete description of the framework and its internal details of functioning, the methodology to design intelligent behavior and all the available modules built in our system. We also expose features that were so far included on the *IViHumans* platform as well as other possible approaches to its extension.

Our goal is to develop a general graphical visualization platform for multiagent system execution and to apply it to the development of realistic and compelling simulation environments, and to incorporate results obtained in the area

of emotion modeling. In the current stage of development the educational characteristics of the framework are not fully developed. We do believe the platform IViHumans is becoming a valuable tool for training and simulation based design purposes. It will be a working tool in post-graduation courses, used to learn basic concepts on animation and virtual environments. Moreover, it will be used as a growing kernel that is prepared to incorporate new functionalities implemented by students.

References

1. N. Magnenat-Thalmann and D. Thalmann. *Handbook of Virtual Humans*. John Wiley & Sons, 2004.
2. Craig W. Reynolds. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, 21(4):25–34, 1987.
3. W. Swartout, J. Gratch, R. W. Hill, E. Hovy, S. Marsella, J. Rickel, and D. Traum. Toward virtual humans. *AI Mag.*, 27(2):96–108, 2006.
4. S. Marsella, L. W. Johnson, and C. LaBore. *Interactive Pedagogical Drama*, pages 301–308. ACM Press, Barcelona, Catalonia, Spain, 2000.
5. Jeff Rickel and W. Lewis Johnson. Integrating Pedagogical Capabilities in a Virtual Environment Agent. In *Proc. of the First Int. Conf. on Autonomous Agents (Agents'97)*, pages 30–38, New York, 1997. ACM Press.
6. D. Isla and B. Blumberg. New Challenges for Character-Based AI for Games. In *AAAI Spring Symposium on AI and Interactive Entertainment, 2002*, 2002.
7. A. Barella, C. Carrascosa, and V. J. Botti. JGOMAS: game-oriented multi-agent system based on JADE. In *Adv. in Comp. Entertainment Technology*. ACM, 2006.
8. Brian Magerko, John E. Laird, Mazin Assanie, Alex Kerfoot, and Devvan Stokes. AI Characters and Directors for Interactive Computer Games. In *16th Innovative Applications of Artificial Intelligence Conference*, pages 877–883, 2004.
9. Emma Norling and Liz Sonenberg. Creating Interactive Characters with BDI Agents. In *Australian Workshop on Interactive Entertainment*, 2004.
10. J. A. Torres, L. P. Nedel, and R. H. Bordini. Using the BDI Architecture to Produce Autonomous Characters in Virtual Worlds. In *IVA*, pages 197–201. Springer, 2003.
11. J. A. Torres, L. P. Nedel, and R. H. Bordini. Autonomous Agents with Multiple Foci of Attention in Virtual Environments. In *Int. Conf. on Computer Animation and Social Agents*, pages 197–201, 2004.
12. Emma Norling and Frank E. Ritter. Embodying the JACK Agent Architecture. In Brooks M., Corbett D., and Stumptner M., editors, *AI 2001: Advances in Artificial Intelligence*, volume 2256 of *LNCS*, pages 368–377, 2001.
13. R. Evertsz, F. E. Ritter, S. Russell, and D. Shepherdson. Modeling rules of engagement in computer-generated forces. In *Proc. of the 16th Conf. on Behavior Representation in Modeling and Simulation*, pages 123–134, 2007.
14. P. M. Semião, M. B. Carmo, and A. P. Cláudio. Implementing Vision in the IViHumans Platform. In *Ibero-American Symp. in Computer Graphics, SIACG 2006*, pages 56–59, 2006.
15. Ricardo Abreu, Ana Paula Cláudio, Maria Beatriz Carmo, Luís Moniz, and Graça Gaspar. Virtual Humans in the IViHumans Platform. In Dimitri Plemenos, editor, *11th International Conference 3IA 2008, the International Conference on Computer Graphics and Artificial Intelligence*, pages 157–162, 2008.

16. Craig W. Reynolds. Steering Behaviors for Autonomous Characters. In *Game Developers Conf.*, 1999.
17. John Funge, Xiaoyuan Tu, and Demetri Terzopoulos. Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters. In *Siggraph 1999, Computer Graphics Proceedings*, pages 29–38. Addison Wesley Longman, 1999.