Virtual Humans in the *IViHumans* Platform

Ricardo Abreu* Ana Paula Cláudio*
Maria Beatriz Carmo* Luís Moniz* Graça Gaspar*
ricardolafabreu@gmail.com {apc,bc,hal,gg}@di.fc.ul.pt

Abstract

Believable virtual humans need to combine a realistic appearance with a virtual brain that is able to approximate human cognition and behavior. To pursuit this goal, the IViHumans platform is composed of two separate layers, one for graphical processing and one for the artificial intelligence computation. This paper describes some relevant aspects of the experience of developing the IViHumans platform, with a greater focus on the graphical processing layer. We concentrate our description in virtual humans and their abilities to move and to express emotion.

Keywords: Intelligent virtual humans, Steering behaviors, Locomotion, Emotional Expressions

1 Introduction

The concept of virtual environments populated by Virtual Humans (VHs) has a wide range of practical applications. The IViHumans (Intelligent Virtual Humans) platform [1] is the ongoing realization of a research project that aims at building a flexible platform that elaborates and renders scenes with VHs that express emotions.

Believable VHs need to combine a realistic appearance with a virtual brain that is able to approximate human cognition and behavior. The IViHumans platform is therefore composed of two separate layers, one for Graphical Processing (GP) and one for the Artificial Intelligence (AI). A special concern is the interconnection between these two layers because they must be able to control the VHs at different detail levels. Support for relevant aspects such as sensory honesty, as in [2], or attention focusing for effective planning [3], must be distributed among the two layers, maintaining a clear separation of concerns and responsibilities.

*Department of Informatics - University of Lisbon, Portugal

Presently, we have achieved a stable state of development over three main issues: perception, movement and emotion expression. A VH is able to perceive the surrounding environment through the single means, for the time being, of synthetic vision [4]. The motion of the VHs is supported by Craig Reynolds' notion of *Steering Behaviour*, as well as by his hierarchical categorization of movement in locomotion, steering and action selection layers [5, 6]. This last and topmost layer accommodates the essential inner workings of the artificial intelligence and is deeply dependent on several concepts. One of them is emotion, which greatly affects the behavior of real humans. Simultaneously, emotion must be accompanied by the physical expression that it naturally entails, which may comprise changes in body and facial postures. Until now, we have centered our efforts on the graphical aspects of the IViHumans platform. The AI layer – and so, any action selection layer – is yet to be developed.

In section 3 of this paper we describe the global architecture of the IViHumans platform; in section 4 we briefly explain the solution for movement and emotional expression; section 5 presents the conclusions and future work. The next section summarizes part of the research work that created the scientific context that inspires the project IViHumans.

2 Related Work

The area of virtual humans and virtual environments incited much research in the last couple of decades, giving rise to diverse contributions. Some look into the subject in a broad way but most reveal a tendency of specialization, focus-

ing deeply and incisively on specific topics.

Ulicny and Thalmann present a crowd simulation system in which each individual is an independent VH [7]. The system is composed by two clearly separate layers called Model Layer and Visualization Layer. The former deals with all the logic processing whilst the latter is only responsible for the presentation. The system presented in [8] – JGOMAS – is also divided in two modules with analogous duties. Presently it implements a particular simulator for a capture-the-flag type of game. Although several instances of the visualization module can be "attached" to a single MAS – which is independent from them – they do not have any influence over the state of the world, which means that the user can't change the progress of the game.

Torres et. al. use the BDI model to control animated characters in virtual worlds [9]. 3D articulated characters are controlled in real-time by cognitive agents that are clients of the environment. Agents must be implemented in AgentSpeak(L) and they run in an interpreter for this language, each one in a separate process. The main limitation is that everything an agent can perceive must be listed *a priori* in a list of boolean statements and this imposition highly restricts the domain of the data over which the agents can reason. In [10] the authors propose an architecture for an embodied conversational agent that has cognitive and emotional capacities employed to interact with the user. It is graphically represented by a "talking head" that may assume one of the six facial expressions pointed out by Paul Ekman.

Perhaps Reynolds' work is the one that influenced us the most. He introduced the techniques of behavioral animation in [5] and later explored and extended them [6]. He proposed a division of a character's movement in layers of growing levels of abstraction: locomotion, steering and action selection. He then proposed an abstract interface for the general capabilities that the first layer comprises and showed how to build behaviors that belong to the second layer, on top of that. His approach has the goal of simulating the real characters' movement, despite not being physically accurate, unlike, for instance, the models of Tu et. al. [11], intended to more closely mimic nature's complex processes.

Vosinakis et. al. built SimHuman [12], a platform for virtual agents with planning capabilities that run in real-time, in an arbitrary virtual environment. The system contains features as inverse kinematics, collision handling and ray-casting based synthetic vision. It has a physics engine that derives the following state for each object. Multon et. al. propose a framework for animating humans in virtual reality [13], capable of performing real-time motion synchronization, retargeting and adaptation in interactive environments by offering efficient and morphology independent motion representation. The blending of postures is achieved by an algorithm that is driven with priorities and states. Conde et. al. developed the ALifeE environment [14] that equips an Autonomous Virtual Agent with various kinds of perception, used to create internal cognitive maps that are employed in the action processes. The agent's conduct is led by behavioral animation with reinforcement learning.

3 Architecture

The IViHumans platform separates the GP layer from the AI layer, an approach that is already found in works such as the JGOMAS system and the ROE3 architecture [15]. 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 GP Layer is responsible for the representation of all the ele-

ments contained in the virtual world and for reproducing the appropriate animations that carry the flow of occurrences, consistently enacting the evolution of the world and its components. 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 is divided in two main components: the interface agents component and the cognitive agents component. The interface agents, one for each virtual entity, manage the communication with the virtual entities, receiving sensory information and sending commands (figure 1). Although these agents can act as a raw connection between both layers, they were extended with new features. They have two functions: to provide a sensing/acting cycle that further separates the communication aspects of the control of the VHs 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: *sensor buffering*, *action buffering*, *data transfer/filter* and *command translation*.

Sensor buffering 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. **Action buffering** 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. **Sensory data translation/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 translation** 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.

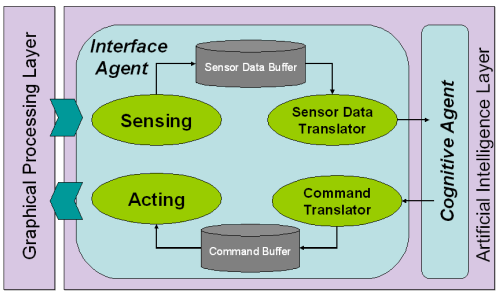

Figure 1: Interface Agents Component detailed.

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 aren't 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 VHs, will do so through an adapter component that manages to unravel the meaning of messages, translating them to the appropriate method calls.

4 Motion and Emotional Expression

The realism of the 3D models and of their animations plays a significant role on what concerns the believability of virtual characters. On the other hand, the 3D models that are used on virtual environments are subject to the restrictions of real-time rendering and should not be defined with an excessive detail, nor depend on too heavy algorithms.

In a preliminary approach, we created a prototypical 3D model for a VH [16]. It is defined as a polygon mesh that is associated with a skeleton. Its materials are only made up of the application of colors and transparency to the faces of the mesh, as well as of the scarce use of color

maps [17]. The model has poses for the six basic expressions that were identified by Paul Ekman. It is easy to exhibit meaningful complex expressions that can be achieved by blending elementary expressions with certain intensities. An offline tool was also created to obtain and record new expressions from the combination of basic ones [18]. This way it is easy to create rich libraries of facial expressions for each model.

4.1 Steering and Locomotion

To implement steering, we closely follow Reynolds' proposals. 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 and so we created a class that models any moving entity as a point mass, as Reynolds did with his vehicle model. This class is called *MovingCharacter* and it is independent of OGRE.

A *MovingCharacter* is essentially characterized by a mass, a position, a velocity and a vector that specifies the direction he is facing. This facing vector may be automatically updated so that it is always tangent to the path. However, this is not mandatory, so that the *MovingCharacter* can also move in ways that require his local depth axis not to be collinear with the velocity vector. The movement of the *MovingCharacter* is ruled by steering behaviors that specify forces he should apply on himself. The movement produced by these forces is computed according to the basic laws of classical physics.

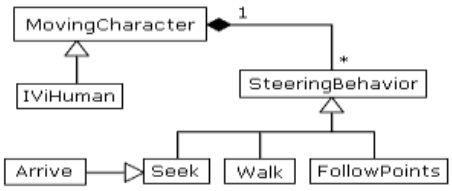

Figure 2: The solution for steering.

Steering behaviors are sometimes criticised for being hard-wired into the code [19]. In an attempt to overcome this problem, we separate the character from the actual behaviors. A *MovingCharacter* aggregates instances of *SteeringBehavior*, which is the common interface to all possible steering behaviors (figure 2). Any instance of *SteeringBehavior* can be plugged in and out of the *MovingCharacter* at run-time.

The behaviors *Seek* and *Arrive* are directly inspired in the corresponding behaviors explained in [6]. The behavior *Walk* does the same as one of the three rules that underlie the flocking behavior in [5]: it tries to match the velocity of the character with a given target velocity. This behavior is useful to have the AI control a VH at a lower level, as well as to have a user directly "piloting" an avatar. We also introduced the behavior *FollowPoints* which is useful, for instance, to make the character follow a path plan. Besides these steering behaviors, many more may be implemented within the current framework.

The VHs are represented internally, on the graphical side, by the class *IViHuman*, which is actually connected to the fundamental objects that enable its graphical representation in OGRE. This class inherits all the steering functionality of *MovingCharacter* and it maps the abstract movement of its parent class onto actual graphical movement and animation, that is, it has to emulate the locomotion layer.

In what regards its position, the *IViHuman* is updated by applying a translation to its 3D model. The translation vector is given by simply subtracting the last position of the *MovingCharacter* from the updated one. To update the rotation of the VH, the minimum rotation that transforms the last facing vector into the new one is applied. The old and new facing vector are always contained in a single plane, unless there is an angle of 180^{o} between them. In this special case the rotation is fixed around the local vertical axis of the VH.

Simultaneously with the raw movement, the VH has to be animated. The animation is chosen on the basis of speed, according to rules that, along with several other parameters, are loaded at runtime. Some of these parameters indicate the proper ratio between the speed of the *MovingCharacter*s and the speed of the animations. The corresponding values are unique for each animation of each VH. The choice and update of the animation is the job of another object that can be plugged in and out at runtime, so that different methods may be applied to achieve dis-

tinct effects. Figure 3(left) shows our prototype moving as a result of the techniques summarized here.

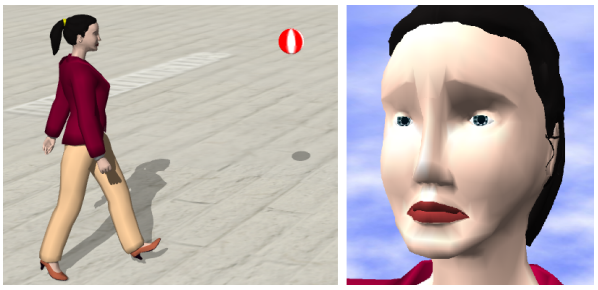

Figure 3: Our prototype, steering to reach a target (left) and showing a sad expression (right).

4.2 Emotional Expression

In the *IViHumans* platform, the VHs can convey emotions through expressions (figure 3 – right). Each VH may have any number of basic expressions that, when blended together, originate more complex expressions. Although the conceptual distinction between basic and complex expressions is essential to the design of the platform (figure 4), their effects are identical. Hence, the class *IViHuman* deals uniformly with basic and complex expressions through a common interface that is contained in the abstract class *Expression*.

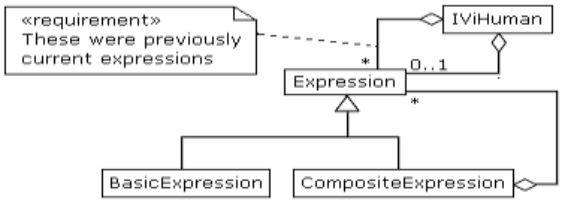

Figure 4: The solution for emotional expression.

The transition between the expressions of a VH must not allow for sudden changes on his face. In our implementation, the smooth transition between expressions relies on different states that the *Expression* objects assume (figure 5). An *Expression* is in the state INACT when it is not yet associated with any *IViHuman*. In this state, it just symbolizes an abstract expression that has a name and a desired intensity. When a request for the *Expression* to become active is made, it becomes necessarily associated with an *IViHuman* and its state is set to ACT. In this state the expression is being activated, that is, its current intensity is gradually increased, a little bit in every call to the method *update*, until it reaches the desired value. At that point, the *Expression* sets itself to the state SKIP and remains constant until it is notified to deactivate, changing to the state DEACT. The deactivation process is the inverse of the activation one and ends when the intensity becomes zero again. Then, its state becomes DONE, until it is explicitly finished with the method *finish*, that has the effect of dissociating the *Expression* from the *IViHuman* that owned it and of turning it into INACT state once again. The *Expression* can also be deactivated when its state is ACT and activated when its state is DEACT or DONE.

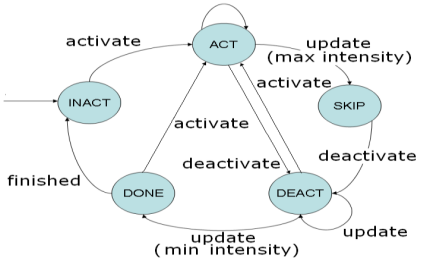

Figure 5: FSM for the activity of an expression.

The class *IViHuman* has two relations with the class *Expression*: the first one for the current expression and the second for the expressions that are being deactivated. The first association is established when a call to *activate* is made. It ends with a call to *deactivate*, giving rise to the second association, which supports multiple *Expressions* per *IViHuman*. The deactivation is enforced by an explicit request from the outside or by the *IViHuman*, when a new *Expression* is supplied to be the current one. Each time step, the *IViHuman* updates all the *Expressions* to which it is related and, when any *Expression* reaches the state DONE, the *IViHuman* will call the method *finish* on it.

The way that the state of an *Expression* is internally translated into a deformation of the mesh depends on its ultimate type. Whilst *BasicExpressions* can be directly translated into an effective expression, *CompositeExpressions* depend on the translation of the basic types that, at the bottom of the tree, compose them.

To materialize a visible expression, each *BasicExpression* relies on a simple animation that gradually intensifies the corresponding pose. Besides being a simpler way of manipulating an expression, wrapping it into an animation encapsulates any other deformation. This way, it is very easy to deal with more complex expressions. There would be no need for any change in the code, for an expression that was built by two poses to be displayed. An expression could also be setup by deforming a skeleton, a lattice or any kind of auxiliary object whose shape can be referenced at a keyframe.

5 Conclusions and Future Work

The prevailing opinion in the domain of virtual environments inhabited by VHs appears to be that there is still a wide open field for subsequent research, despite the broadness of the set 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. With the project IViHumans, we aim at the development of a platform intended to enable easy creation of applications that integrate VHs, modestly participating on the enrichment of the field, by combining a growing set of distinct features.

In this paper we expose features that were so far included on the *IViHumans* platform. Our short-term goals include integrating the capacities of ODE to simulate rigid bodies dynamics, specially when it comes to collision handling, and developing the AI layer and fully integrate it with the GP layer.

References

[1] M. Silvestre, M. Pinto-Albuquerque, M. B. Carmo, A. P. Cláudio, J. D. Cunha, and H. Coelho. Platform for the Generation of Virtual Environments Inhabited by Intelligent Virtual Humans (*student poster*). In *ACM ITiCSE'05*, 2005.

[2] D. Isla and B. Blumberg. New Challenges for Character-Based AI for Games. In *AAAI*, 2002.

[3] L. Morgado and G. Gaspar. Abstraction Level Regulation of Cognitive Processing Through Emotion-Based Attention Mechanisms. *Attention in Cognitive Systems, LNCS 4840/2007*, pages 59–74, 2007.

[4] P. M. Semião, M. B. Carmo, and A. P. Cláudio. Implementing Vision in the IViHumans Platform. In *SIACG*, 2006.

[5] Craig W. Reynolds. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, 21(4):25–34, 1987.

[6] Craig W. Reynolds. Steering Behaviors for Autonomous Characters. In *GDC*, 1999.

[7] B. Ulicny and D. Thalmann. Crowd simulation for interactive virtual environments and VR training systems. In *CAS'01*, pages 163–170, 2001.

[8] A. Barella, C. Carrascosa, and V. J. Botti. Jgomas: game-oriented multi-agent system based on jade. In *Advances in Computer Entertainment Technology*. ACM, 2006.

[9] 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.

[10] M. Longhi, L. Nedel, R. Viccari, and M. Axt. Especificação e Interpretação de Gestos Faciais em um Agente Inteligente e Comunicativo. In *SBC Symp. on Virtual Reality*, 2004.

[11] X. Tu and D. Terzopoulos. Artificial Fishes: Physics, Locomotion, Perception, Behavior. *CG*, (Annual Conference Series):43–50, 1994.

[12] S. Vosinakis and T. Panayiotopoulos. A tool for Constructing 3D Environments with Virtual Agents. In *Multimedia Tools and Applications*, pages 253–279. Kluwer Academic, 2005.

[13] F. Multon, R. Kulpa, and B. Bideau. MKM: A Global Framework for Animating Humans in Virtual Reality Applications. *Presence: Teleoper. Virtual Environ.*, 17(1):17–28, 2008.

[14] T. Conde and D. Thalmann. An Artificial Life Environment for Autonomous Virtual Agents with multi-sensorial and multi-perceptive features. *C. Animat. Virtual Worlds*, 15(3-4):311–318, 2004.

[15] R. Evertsz, F. E. Ritter, S. Russell, and D. Shepherdson. Modeling rules of engagement in computer-generated forces. In *Behavior Representation in Modeling and Simulation*, pages 123–134, 2007.

[16] R. Abreu, A. P. Cláudio, and M. B. Carmo. Humanos Virtuais na Plataforma IViHumans – a Odisseia da Integração de Ferramentas. In *15º EPCG*, pages 217–222, 2007.

[17] Steve Rabin, editor. *Introduction to Game Development*. Charles River Media, 2005.

[18] J. Faustino, A. P. Cláudio, and M. B. Carmo. Faces – Biblioteca de Expressões Faciais. In *Interacção*, 2006.

[19] J. Funge, X. Tu, and D. Terzopoulos. Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters. In *Siggraph*, 1999.