Framework for Simulating the Human Behavior for Intelligent Virtual Agents. Part I: Framework Architecture

F. Luengo[1,2] and A. Iglesias[2⋆]

[1] Department of Computer Science, University of Zulia, Post Office Box #527, Maracaibo, Venezuela
fluengo@cantv.net

[2] Department of Applied Mathematics and Computational Sciences, University of Cantabria, Avda. de los Castros, s/n, E-39005, Santander, Spain
iglesias@unican.es
http://personales.unican.es/iglesias

Abstract. This paper is the first in a series of two papers (both included in this volume) describing a new framework for simulating the human behavior for intelligent virtual agents. This first paper focuses on the framework architecture and implementation issues. Firstly, we describe some requirements for such a framework to simulate realistically the human behavior. Then, the framework architecture is discussed. Finally, some strategies concerning the implementation of our framework on single and distributed CPU environments are presented.

1 Introduction

One of the most exciting fields in Computer Graphics is the simulation and animation of intelligent virtual agents (IVAs) evolving within virtual 3D worlds. This field, also known as *Artificial Life*, has received increasing attention during the last few years [1,2,3,4,5,6,12,14]. Most of this interest has been motivated by its application to the entertainment industry, from virtual and augmented reality in digital movies to video games. However, the range of potential applications also includes Architecture, Science, Education, advertising and many others.

One of the most interesting topics in the field concerns the realistic animation of the behavior of IVAs emulating the human beings. The challenge here is to provide the virtual agents with a high degree of *autonomy*, so that they can evolve freely with a minimal input from the animator. In addition, this evolution is expected to be *realistic*, in the sense that the IVAs must behave according to reality from the standpoint of a human observer.

In a previous paper [10] the authors presented a new behavioral framework able to reproduce a number of the typical features of the human behavior. The system allows the IVAs to interact among them and with the environment in a quite realistic way. A subsequent paper [8] extended the original approach by

⋆ Corresponding author

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3039, pp. 229–236, 2004.

introducing some functions and parameters describing new internal, physical and mental states. The performance of that framework was also discussed in [11].

We would like to remark, however, that such a framework was exclusively designed for behavioral simulation purposes only and, consequently, it can be substantially improved in several directions. For example, neither the graphical output nor the computational efficiency did play a significant role in its design. On the other hand, it was pointed out that the use of Artificial Intelligence (AI) tools, such as neural networks and expert systems, can improve the performance of the behavioral animation schemes dramatically [7,15]. These and other extensions are the core of the present work.

This is the first in a series of two papers (both included in this volume) describing a new framework for simulating the human behavior for intelligent virtual agents. Although originally based on that introduced in [10], the current framework incorporates so many additions and improvements that it can actually be considered as a new one. Its new features concern fundamentally to the architecture and the behavioral engine. The new architecture is based on the idea of decomposing the framework into the physical and the behavioral systems and, subsequently, into their respective subsystems which carry out more specific tasks. In addition, specialized computing tools have been applied to these subsystems, so that the performance has been greatly improved. On the behavioral engine, powerful Artificial Intelligence techniques have been applied to simulate the different behavioral processes. As it will be shown later, these AI tools provide the users with a higher level of realism. Because of limitations of space, the architecture of the new framework will be described in this first paper, while the second one will focus on the application of AI tools to the behavioral engine.

The structure of this paper is as follows: in Sect. 2 we describe the main requirements of a framework to simulate the human behavior for IVAs. Then, Sect. 3 describes the architecture to fulfill those requirements. The agent's design, software tools and programming environments that have been used to implement such an architecture are also discussed in this section. Finally, Sect. 4 presents some strategies concerning the implementation of our framework on single and distributed CPU environments.

2 Framework Requirements

In this work, an *Intelligent Virtual Agent* (IVA) is the graphical representation of a virtual creature able to emulate the behavior of a living being autonomously, i.e., without the animator's intervention. Due to its inherent complexity, it is convenient to decompose our framework into different (simpler) components, which can be rather assigned to one of the following sytems:

1. the *physical system* (PS): it is responsible for the physical elements, including the 3D graphical representation of virtual agents, their motion and animation and the interaction among them and with the world's objects.
2. the *behavioral engine* (BE): it will provide the agents with emotions, feelings, thoughts, needs and beliefs (about themselves, others or the environment). Depending on their particular values, different plans will be designed by

this engine in order to accomplish the agents' goals. Although the human senses (vision, hearing, etc.) are usually associated with physical parts of our body (eyes, ears, etc.), the cognitive process itself happens at our brain, so mental routines related to perception are also included in this component. By the same reason, the different cognitive tasks related to the agent's motion control are performed at this behavioral engine[1].

Reasons for this decomposition become clear if you think about our ability to distinguish between what we are physically and mentally. In fact, we can easily assign any physical object of the 3D world (even our own body itself) to the physical system, while our emotions, beliefs, feelings or thoughts would be assigned to the behavioral engine. This separation is also extremely useful from a computational point of view. On one hand, it allows the programmer to focus on the specific module he/she is dealing with at one time. Clearly, it makes no sense to worry about the graphical motion routines when you are modifying the behavioral ones, and vice versa. On the other hand, specialized programming tools can be independently applied to each module. As a consequence, the framework's performance can be drastically optimized, provided that an adequate choice of such tools is made.

Note, however, that both systems must be strongly interconnected so that each modification in the behavioral engine (for example, if the agent is becoming tired his/her next goal might be to look for a seat to sit down) is subsequently reflected on the physical counterpart (the physical motion towards the seat) and vice versa, just as our body and brain also work as a whole. To this aim, some kind of communication between both systems must be defined. Furthermore, the better we define how these systems work and how they communicate with each other, the more effective the framework will be.

Of course, each system can be broken up into smaller subsystems, associated at its turn with more specific routines such as obstacle avoidance or path determination for the physical system, or goals or internal states for the behavioral engine. By this way, we can either work on each subsystem individually or hand out them to different people to work on. However, we should be careful with the number of levels in this sequence: indeed, too few levels will yield large codes difficult to test and debug, while too many levels will unnecesarily increase the complexity of the system.

3 Framework Architecture and Tools

3.1 Virtual Objects

The virtual agents evolve in a 3D virtual world which also comprises different kinds of objects to interact with (see Fig. 1). Basically, they can be classified into two groups: static objects and smart objects. By *smart objects* we understand those objects whose shape, location and status can be modified over time, as

[1] Note that the physical motion routines themselves still belong to the physical system. What is actually included in the behavioral engine is the simulation of the mental process that yields the orders for motion from the brain to the muscles.

opposed to the static ones. This concept, already used in previous approaches [9,13] with a different meaning, has shown to be extremely helpful to define the interactions between the virtual agents and the objects. For example, a table lamp or a radio are smart objects simply because they might be turned on/off (status set to on/off) and so are a pencil or a bottle (as they can be relocated). We point out that saying that an object is static does not mean it has null influence on the agents' actions. For instance, a tree is a static object but it should be considered for tasks such as collision avoidance and path planning.

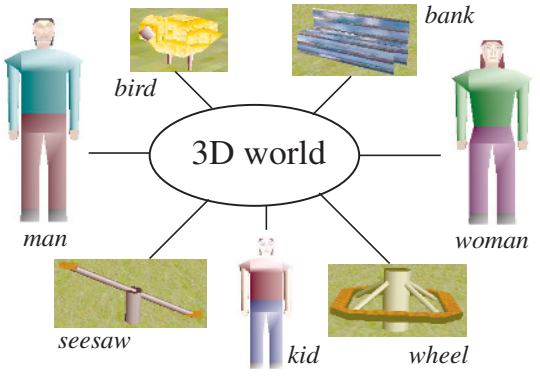

Fig. 1. The 3D world includes different kinds of virtual objects and agents

3.2 Behavioral Engine

Because the behavioral engine also includes some behavioral routines that strongly influence the graphical output (such as those for perception), we decided to split it up into the *physical control system* (PCS) and the *behavioral system* (BS), as shown in Fig. 2.

The PCS comprises two subsystems for perception and motion control tasks. The perception subsystem obtains information from the graphical environment (the virtual world) by identifying its elements (static objects, smart objects, other agents) and locations. In other words, it captures the geometry of the virtual world as it is actually done by the human beings through their senses, in which the perception subsystem is based on. On the other hand, the motion control subsystem is responsible for the conversion of the agents' plans into physical actions, as described below. At its turn, the BS (that will be described in detail in a second paper in sequence) includes several subsystems designed to perform different cognitive processes. The arrows in Fig. 2 show the information flow: the perception subsystem captures information from the virtual world which is subsequently sent to the behavioral system to be processed internally. The corresponding output is a set of orders received by the motion control subsystem,

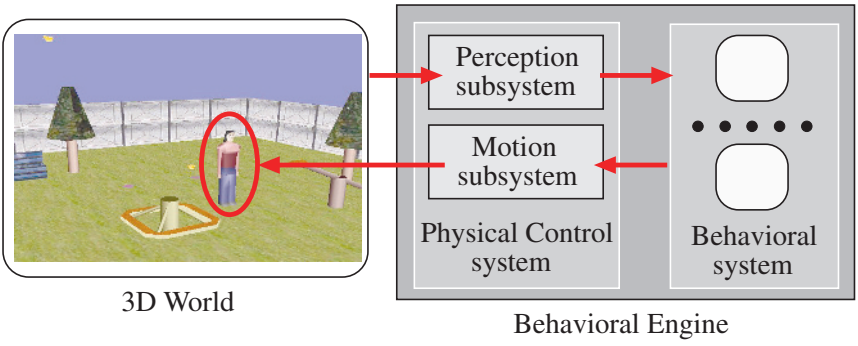

Fig. 2. Scheme of the behavioral engine of a virtual agent

which transform them into agent's physical actions animated by the physical system[2], just as the orders of our brain are sent to our muscles.

We would like to remark that this behavioral engine decomposition into the PCS and the BS is both *reasonable* and *useful*. It is reasonable because agents' reactions and decisions are mostly determined by their "personality" rather than by their physical body. Of course, the physical is also involved in "who we are", but our personality lie in another "level" of ourselves and should be analyzed separately. The usefulness comes from the fact that it is possible to reuse the BE for different virtual worlds. This leads to the concept of *adaptation*: a realistic simulation of a human being implies that the BE must be able to perform adjustments by itself in order to adapt to the changing environment. Similarly, different BE can be applied to the same virtual world. This leads to the concept of *individuality*: no two virtual agents are exactly the same as they have their own individual personality. In computational terms, this means that each virtual agent has his/her own behavioral engine, which is different from any other else.

3.3 Agents Design

As usual in Object Oriented Programming (OOP) environments, each virtual agent is represented by a class called AVA, consisting of attributes and methods. In our case, the attributes are: *AgID*, that identifies the agent, *AgSt* that accounts for the current status of the agent, and *AgVal* that stores some parameters for rendering purposes (position, direction, etc.). The methods include the *Render* method for graphical representation and those for updating the agent's attributes as a consequence of interactions with objects. Moreover, the class AVA encapsulates the attributes and methods related to the perception and the motion control subsystems. Additional methods are considered for the communication

[2] We should warn the reader about the possible confusion between "physical system" (PS) and "physical control system" (PCS). The PCS is a part of the behavioral engine, while the PS contains the routines for the graphical representation and animation of the virtual world.

from the perception subsystem to the behavioral system (*Send*) and from it to the motion control subsystem (*CallBack*). Finally, the method *Think* is used to trigger the behavioral process.

3.4 Programming Languages and Environments

Regarding the programming languages, Table 1 shows the different architecture modules of our framework as well as the software tools and programming environments used to implement such modules. The first module is the Kernel, which drives the main sequence of animation. The use of a powerful graphical library would allow the programmer to improve graphics quality dramatically with relatively little effort. By this reason, the kernel has been implemented in Open GL by using the programming environment GLUT (Open GL Utility Toolkit). The graphical representation of the virtual world (the physical system) is also a CPU demanding task. Therefore, we decided to use C++ to assure the best performance. Another reason for this choice is the excellent integration of Open GL with the C++ layer (to this purpose, we used the Visual C++ environment as programming framework). This combination of C++ with Open GL has also been used for the User Interface.

Table 1. Architecture modules of our framework and the software tools and programming environments used to implement them

Module	Software tools	Programming environment
Kernel	Open GL	GLUT
User Interface	C++ & Open GL	Visual C++ & GLUT
Physical System	C++ & Open GL	Visual C++ & GLUT
Physical Control System	C++	Visual C++
Behavioral System	C++ & Prolog	Visual C++ & Amzi! Prolog

As mentioned above, our framework consists of a physical system (PS) and a behavioral engine (BE). While the combination of C++ and Open GL works well for the physical system, the BS requires more specific tools. In particular, it has been implemented in C++ and Prolog by using the programming environment "Amzi! Prolog" (developed, at its turn, in C language). At our experience, Amzi! Prolog is an excellent tool to generate optimized code which can easily be invoked from C/C++ via Dynamic Link Libraries (DLLs), providing an optimal communication between the PCS and the BS for *standalone* applications. Furthermore, this choice provides a good solution for TCP/IP communication protocols for distributed environments, as discussed in Sect. 4.

4 Implementation on Single and Distributed CPU Environments

The framework presented in the previous sections can be developed by using only a processor or several ones. For the first case, we can either use a dynamic list of objects AVA (as shown in Fig. 3(left)) or to run each AVA in a separate process or *thread* (see Fig. 3(right)). In both cases, we must wait until all AVAs have executed to get the next animation frame. Note also that the communication between the object AVA and the behavioral system is achieved via DLLs to optimize the execution speed, avoiding other alternatives such as TCP/IP, best suited for distributed systems and networks.

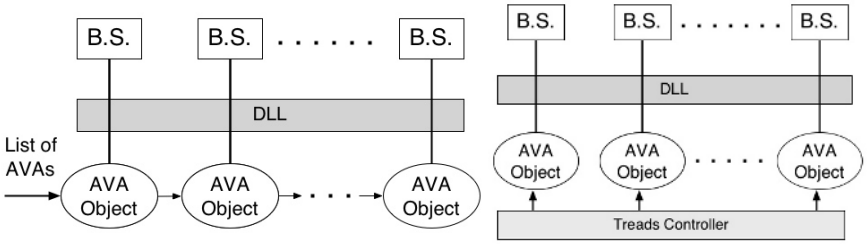

Fig. 3. Framework architectures for a single processor

Figure 4 shows the framework architecture for distributed systems. In this case, we use threads to run the different AVAs, which are connected to their corresponding BS by using sockets and TCP/IP connection. Note that parallel programming can also be applied here. For instance, we can assign each IVA behavioral system to a single processor for maximal performance.

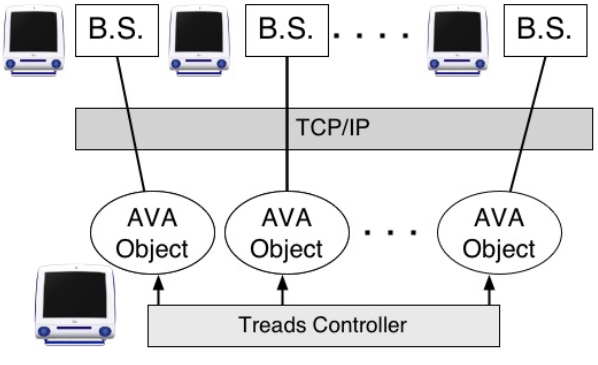

Fig. 4. Framework architecture for distributed systems

The previous single and distributed CPU architectures have been successfully implemented on PC platform (Pentium III processor). Technical details on implementation have had to be omitted because of limitations of space and will be reported elsewhere. In the second paper some interesting questions regarding the behavioral engine will be discussed.

References

1. Badler, N.I., Barsky, B., Zeltzer, D. (eds.): Making Them Move. Morgan Kaufmann, San Mateo, CA (1991)
2. Badler, N.I., Phillips, C.B., Webber, B.L.: Simulating Humans: Computer Graphics Animation and Control. Oxford University Press, Oxford (1993)
3. Blumberg, B.M., Galyean, T.A.: Multi-level direction of autonomous creatures for real-time virtual environments. Proc. of SIGGRAPH'95, ACM, New York (1995) 47-54
4. Cerezo, E., Pina, A., Seron, F.J.: Motion and behavioral modeling: state of art and new trends. The Visual Computer, **15** (1999) 124-146
5. Funge, J., Tu, X. Terzopoulos, D.: Cognitive modeling: knowledge, reasoning and planning for intelligent characters, Proceedings of SIGGRAPH'99, ACM, New York (1999) 29-38
6. Granieri, J.P., Becket, W., Reich, B.D., Crabtree, J., Badler, N.I.: Behavioral control for real-time simulated human agents, Symposium on Interactive 3D Graphics, ACM, New York (1995) 173-180
7. Grzeszczuk, R., Terzopoulos, D., Hinton, G.: NeuroAnimator: fast neural network emulation and control of physics-based models. Proceedings of SIGGRAPH'98, ACM, New York (1998) 9-20
8. Iglesias A., Luengo, F.: Behavioral Animation of Virtual Agents. Proc. of the Fourth International Conference on Computer Graphics and Artificial Intelligence, 3IA (2003) 99-114
9. Kallmann, M.E., Thalmann, D.: A behavioral interface to simulate agent-object interactions in real-time, Proceedings of Computer Animation'99, IEEE Computer Society Press, Menlo Park (1999) 138-146
10. Luengo, F., Iglesias A.: A new architecture for simulating the behavior of virtual agents. Springer-Verlag, Lecture Notes in Computer Science, **2657** (2003) 935-944
11. Luengo, F., Iglesias A.: Animating Behavior of Virtual Agents: the Virtual Park. Springer-Verlag, Lecture Notes in Computer Science, **2668** (2003) 660-669
12. Maes, P., Darrell, T., Blumberg, B. Pentland, A.: The alive system: full-body interaction with autonomous agents, Proceedings of Computer Animation'95, IEEE Computer Society Press, Menlo Park (1995) 11-18
13. Monzani, J.S., Caicedo, A., Thalmann, D.: Integrating behavioral animation techniques, Proceedings of EUROGRAPHICS'2001, Computer Graphics Forum, **20**(3) (2001) 309-318
14. Perlin, K., Goldberg, A.: Improv: a system for scripting interactive actors in virtual worlds, Proceedings of SIGGRAPH'96, ACM, New York (1996) 205-216
15. Van de Panne, M., Fiume, E.: Sensor-actuator networks, Proceedings of SIGGRAPH'93, Computer Graphics **27** (1993) 335-342