Framework for Simulating the Human Behavior for Intelligent Virtual Agents. Part II: Behavioral System

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 second 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 second paper focuses on the application of Artificial Intelligence (AI) techniques to the simulation of the human cognitive process. The paper discusses some important issues involved in this process, such as the representation and identification of objects, the information acquisition and its conversion into knowledge and the learning process. The paper also describes how some standard AI techniques (expert systems, neural networks) have been applied to tackle these problems.

1 Introduction

In the first part of this work we have reviewed some features of the architecture of a new framework for simulating the human behavior for intelligent virtual agents. In addition, we analyzed the software and programming environments used to implement such a framework, with emphasis on the graphical part. Fortunately, the huge number of software applications for 3D graphics and animation allow us to apply well-known standarized tools. The challenge is to develop a similar "machinery" for human behavior simulation.

So far, little effort was placed upon the simulation of the human cognitive processes (learning, memory, recognition, etc.) from the viewpoint of Computer Graphics. Notable exceptions are the works in [2,3,4,7,8,9,10]. In contrast, this is the primary goal of the Artificial Intelligence (AI) field. After all, most of the AI techniques (such as neural networks or expert systems) are based on the idea of reproducing the structure and behavior of the human brain. Consequently, it seems very reasonable to apply them to the simulation of the intelligent virtual agents (IVAs). This is actually the core of this paper. In particular, the paper

⋆ Corresponding author

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3039, pp. 237–244, 2004.

discusses some important issues involved in this process, such as the representation and identification of objects, the information acquisition and its conversion into knowledge and the learning process. The paper also describes how some standard AI techniques (expert systems, neural networks) have been applied to tackle these problems.

2 General Scheme of the Behavioral System

The realistic simulation of the behavior of virtual agents implies that they must be able to carry out an intelligent exploration of the surrounding environment. By *intelligent* we mean that the IVAs need to walk through three main steps:

1. to identify the different objects from the virtual world (*object recognition*)
2. to obtain information from the environment (*information acquisition*)
3. this information is subsequently processed so that the agents can effectively acquire new knowledge and/or update the current one (*knowledge acquisition*).

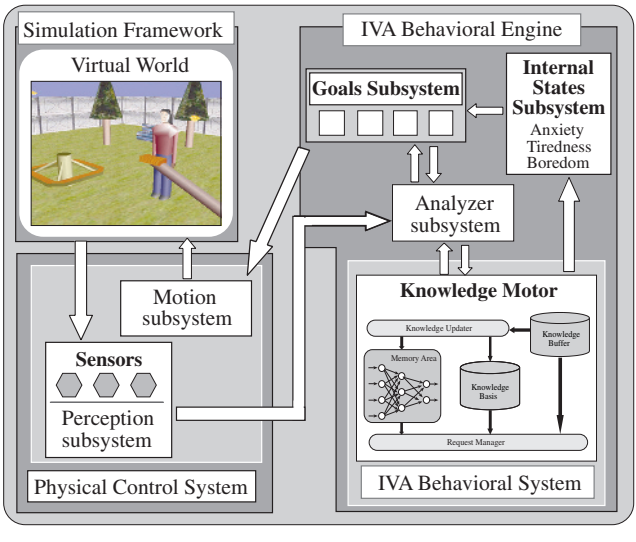

Fig. 1. General scheme for information acquisition and its conversion into knowledge

All these tasks are performed by specific subsystems and modules, as depicted in Fig. 1. The *perception subsystem* (PSB) applies routines to identify the objects and to extract information from the 3D world. Such information is subsequently sent to the *analyzer subsystem*, where the information is processed and transformed into knowledge. The *internal states subsystem* handles the information about agents personality and his/her "emotional state". With that

information, the *goal engine subsystem* updates the goals list, thus determining what the agent wants to do. Finally, the *action engine subsystem* takes a decision about the best way to achieve those goals, updates the agents status, and sends that information to the *motion subsystem* to complete the animation. The following paragraphs will analyze how these tasks have been accomplished.

2.1 Objects Representation and Identification

In order to interact with the 3D world, the IVA must be able to identify its elements, regardless their nature (smart objects, static objects, other agents) and properties (location, status, etc). These properties provide essential information, as they will determine the kind of agent-object interaction and, consequently, the future agent's actions. On the other hand, we would like this object representation to be as simple as possible. This feature is required for efficient manipulation and memory storage.

In this paper we use a representation scheme based on biological concepts such as chromosome and gene. Each object of the 3D world is represented by what we call a *chromosome*. Roughly speaking, it is a collection of multicomponent *sequences* which, at their turn, comprise single fields called *genes*, as shown in Fig. 2. For example, the chromosome in this figure consists of m sequences and n genes ($n > m$).

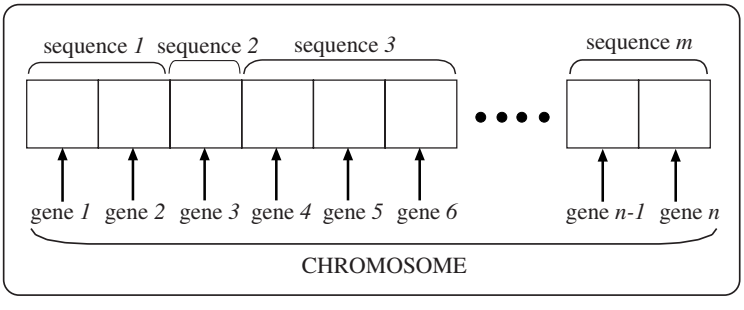

Fig. 2. Structure of the chromosome representing the objects of the 3D world

Each sequence corresponds to a certain characteristic of the object. The sequences are sorted by following a hierarchical structure, as shown in Fig. 3. In this work, we consider that the objects' chromosomes are composed of four sequences, from the most general to the most specific one: the first sequence consists of three genes that account for objects, animals and people (sequences *[1,0,0]*, *[0,1,0]* and *[0,1,1]*, respectively). The second sequence, also with three genes, adds more information about general characteristics, such as the kind of object, animal or person. In this example, the category *person* is subsequently subdivided into *kid*, *adult* and *elderly*. The third sequence consists of one gene

and is associated with the status, size or gender (for object, animal or person, respectively). Finally, the last sequence comprises five genes to store the object's ID, in order to identify an specific element within its own class.

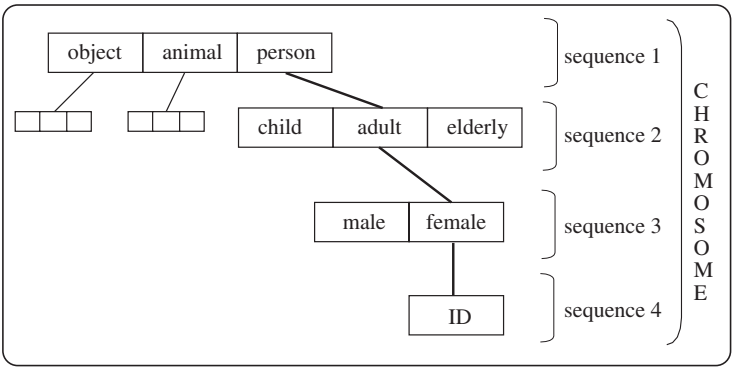

Fig. 3. Sequences of the chromosome

For example, the first woman in our environment is represented by the chromosome *[0,1,1,1,1,0,1,0,0,0,0,1]*. This representation is useful for identification, provided that a mathematical function to compute the distance between two arbitrary elements is defined. Given a pair of elements, the goal of such a function is to determine how close these elements are (in other words, such a function constitutes "de facto" a criterion for similarity). The *distance function* between two elements A and B at a sequence j is defined as follows:

$$dist(j, A, B) = \frac{1}{k} \sum_{i=1}^{k} |A_i^j - B_i^j| \tag{1}$$

where A_i^j denotes the ith gene at sequence j for the chromosome A, and k denotes the number of genes of such a sequence. Note that we can think of sequences in terms of levels in the tree displayed in Fig. 3. The sequence j is simply the level j down the tree at which it appears, with the top of the tree as sequence 1. We will say that two elements A and B are *similar at sequence* (or *at level*) j if $dist(j, A, B) = 0$. Further, they are *similar up to sequence* s if $dist(r, A, B) = 0$, $\forall\, r \leq s$. Note that the hierarchical structure described above imply that an arbitrary object is closer to that minimizing the distance at earlier sequences. For instance, an adult is represented by the sequence $[1, 1, 0]$ which is in-between the sequence for kids $[1, 0, 0]$ and for elder people $[0, 1, 0]$, since $dist(adult, kid) = dist(adult, elder) = \frac{1}{3}$ whereas $dist(kid, elder) = \frac{2}{3}$, meaning that an adult is closer to an elder person than a kid. Therefore, Eq. (1) provides an accurate procedure to classify objects at a glance, by simply comparing them sequentially at each level.

2.2 Information Acquisition

In this step, the *analyzer subsystem* receives the world information acquired by the PSB and then analyzes it to update the *knowledge base* accordingly. As explained in the first paper, the *perception subsystem* has been developed in a software environment different than that for the *behavioral system* (BS). Therefore, it is extremely important to define clearly a communication protocol for information exchange between both systems. In that protocol, carrying-information comprises four fields (see Fig. 4): a parameter specifying the information source (vision, hearing), the object ID or chromosome (see Sect. 2.1 for details), additional information about the location, status, etc. and a parameter called *impact index*. This last parameter is added by the analyzer to account for the impact of a new information on the agent and will be detailed later on.

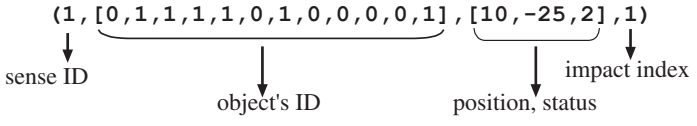

Fig. 4. Information exchange between the perception and the behavioral systems

2.3 Knowledge Acquisition

Once new information is acquired and then processed by the analyzer, it is sent to the *knowledge motor*, whose main components are displayed in Fig. 5. Firstly, the current information is temporarily stored into the *knowledge buffer*, until new information is attained. At that time, previous information is sent to the *knowledge updater*, the new one being stored into this buffer and so on.

The *knowledge base* is actually a based-on-rules expert system, containing facts and inference rules. In addition to the information provided by the updater, the facts include complex relationships among the different elements (personal relationships among agents such as friendship, relative positions of objects, etc). The inference rules, based on deductive schemes such as modus ponens, modus tollens, rule chaining, goal-oriented rule chaining and others(see, for instance, Chapter 2 of [1]), provide the system with the tools to infer new knowledge from the current one. Of course, the system's complexity is mostly determined by the number of rules and the design of the inference engine. Additional subsystems for other tasks (coherence control, action execution) have also been incorporated.

The *memory area* is a neural network that will be applied to learn from data (in our problem, the information received from the environment through the perception subsystem). A *neural network* consists basically of one or several layers of computing units, called *neurons*, connected by links. Each artificial neuron receives an input value from the input layer or the neurons in the previous layer.

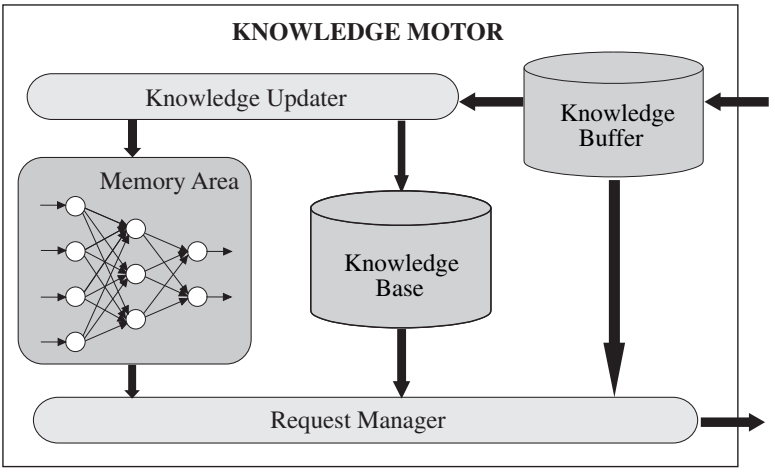

Fig. 5. Scheme of the knowledge motor and its components

Then it computes a scalar output $y = f\left(\sum w_{ik}x_k\right)$ from a linear combination of the received inputs $x_1, x_2, \ldots, x_n$ using a set of weights w_{ik} associated with each of the links and a given scalar function f (the *activation function*), which is assumed to be the same for all neurons (see [5] and [6] for details).

Among the many interesting properties of a neural network, one of primary importance is its ability to learn from the environment and to improve its performance through learning. Such an improvement takes places over time through an iterative process based on adjust the free parameters of the network (the weights). In this paper we consider the *unsupervised learning*, in which the data is presented to the network without any external information and the network must discover by itself patterns, or categories. In particular, we use an autoassociative scheme, since the inputs themselves are used as targets. In other words, the networks tries to learn the identity function, which is a problem far to be trivial as the network contains less neurons than the input/output layers, and hence, the network must perform dimensionality reduction. What the network attempts is to subdivide the chromosome space into clusters in order to associate each chromosome with a specific neuron, the nearest one in our case. To this end, we try to minimize the sum of the squared within-groups residuals, which are basically the distances of the chromosome locations to the respective group centroids. When a new chromosome is received as input, the whole structure is recomputed and the group centroids are relocated accordingly. This problem can be overcome by applying the K-means least-squares partitioning algorithm, a procedure to divide a collection of n objects into K groups. The basic algorithm consists of two main steps:

- compute cluster centroids and use them as new cluster seeds
- assign each chromosome to the nearest centroid.

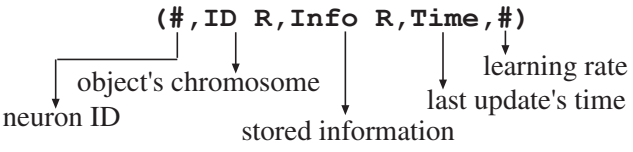

Fig. 6. Information received by the neurons

In our case, each neuron should receive the information shown in Fig. 6, namely, the neuron ID, the object's chromosome, the information to be stored by the neuron, the time at which this information is attained (which will be used for animation purposes), and the learning rate. This last parameter is introduced to describe the neuron's ability to adapt to a new information (and simultaneously, to forget the previous one). Its meaning becomes clear by simply noticing that, in our daily life, we can learn, understand and remember certain things completely, partially and sometimes not at all. In fact, certain things can never be forgotten. This "unforgettable" information is assigned to neurons whose learning rate is set to 0 so that the information is permanently stored. By this way we can deal with information which, although extremely important (i.e., with high impact index), has been received only once.

2.4 Learning Process

Let us suppose that we have a neural network with k neurons and that n data vectors $x_1, x_2, \ldots, x_n$, (with $k < n$) will eventually be perceived at different times. To update the memory area, we employ a K-means procedure for competitive networks, which are a popular type of unsupervised network architectures widely used to automatically detect clusters, or categories, within the available data. A simple competitive neural network is formed by an input and an output layer, connected by feed forward connections. Each input pattern represents a point in the configuration space (the space of inputs) where we want to obtain classes.

This type of architecture is usually trained with a *winner takes all* algorithm, so that only the weights associated with the output neuron with largest value (the winner) are updated. The procedure is based on the following strategy: at the initial stage, all the neurons are available to store new data. Therefore, the first k data vectors are sequentially assigned to these neurons, i.e., data x_i is learned by neuron i, $1 \leq i \leq k$. Simultaneously, time for neuron i is initialized to the moment at which data x_i is learned. Once the next data x_{k+1} is received, it is assigned to the neuron j such that

$$d(x_j, x_{k+1}) \leq d(x_i, x_{k+1}), \qquad \forall i = 1, \ldots, k, \quad i \neq j \tag{2}$$

When this condition is satisfied by several neurons simultaneously, the new data is assigned to that storing the oldest information. Interesting enough is the way

in which the neuron stores the new information: instead of replacing the old data by the new one, what is actually stored is a combination of both data. The basic idea behind this formulation is to overcome the limitation of having more data than neurons by allowing each neuron to store more than one data at the same time. Thus, the neuron does not exhibit a deterministic output but a probabilistic one: what is actually computed is the probability of a neuron to have a particular data at a particular time. This probability is continuously updated in order to adapt our recalls to the most recent data. This leads to the concept of *reinforcement*, based on the fact that the repetition of a particular event over time increases the probability to recall it. Of course, some particular data are associated with high-relevance events whose influence does not decrease over time (or decreases so slowly that it can be considered as a time-independent event). In those cases, the neuron must be able to store this data and maintain its probability regardless the time. The learning rate parameter introduced in Sect. 2.3 is intended to play this role.

Finally, we would like to remark that this scheme improves substantially the deterministic approaches for short-medium-long (SML)-term memory by introducing uncertainty on the agent's recalls. Combination of this scheme and fuzzy logic constitutes a better approach to the human recall process and it is currently being investigated. The conclusions of this study will be the subject of a future publication.

References

1. Castillo, E., Gutiérrez, J.M., Hadi, A.: Expert Systems and Probabilistic Network Models. Springer-Verlag, New York (1997)
2. 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
3. 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
4. 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
5. Haykin, S.: Neural Networks. A Comprehensive Foundation. Macmillan Publishing, Englewood Cliffs, NJ (1994)
6. Hertz, J., Krogh, A., Palmer, R.G.: Introduction to the Theory of Neural Computation. Addison Wesley, Reading, MA (1991)
7. Monzani, J.S., Caicedo, A., Thalmann, D.: Integrating behavioral animation techniques, Proceedings of EUROGRAPHICS'2001, Computer Graphics Forum, **20**(3) (2001) 309-318
8. Ridsdale, G.: Connectionist modeling of skill dynamics. Journal of Visualization and Computer Animation, **1**(2) (1990) 6672
9. Sims, K.: Evolving virtual creatures, Proceedings of SIGGRAPH'94, ACM, New York (1994) 15-22
10. Van de Panne, M., Fiume, E.: Sensor-actuator networks, Proceedings of SIGGRAPH'93, Computer Graphics **27** (1993) 335-342