ALVW: an alife behaviour modelling system

A. Pina
Dpto. de Matemática e Informática, Universidad Pública de Navarra, Pamplona, Spain, and
F.J. Seron, E. Cerezo and D. Gutierrez
Grupo de Informática Gráfica Avanzada, Departamento de Informática e Ingeniería de Sistemas, Instituto de Investigación en Ingeniería de Aragón, Centro Politécnico Superior – Universidad de Zaragoza, Zaragoza, Spain

Abstract

Purpose – Seeks to present here a system that enables the user to design artificial worlds and inhabitants within these worlds.

Design/methodology/approach – An alife (artificial life) approach is used in order to obtain complex behaviour as the emergence of a set of basic behaviours which interact within the environment and between them. Through the paper, the reader will find the necessary details to understand the architecture and the computational model of the system, as well as the way to use it in a specific application: modelling behaviour for synthetic cockroaches within his own environment.

Findings – A detailed example is given of how this application can be used in simulating ecological environments, where the user can design an inhabited ecosystem and observe (tuning it if necessary) the evolution of the artificial ecosystem. Also outlined is another example of how this research system can be coupled with a commercial 3D computer animation software; the goal in this case is to produce automatically a script for a computer animation, reflecting what is happening in the artificial world.

Research limitations/implications – The paper describes a technical research system that allows the user to design reactive behaviour modelling for cockroaches. The next natural step is to make a cognitive behaviour modelling and to extend it to other animals.

Practical implications – Fusion between research systems and commercial packages (in this case, for example, behaviour modelling and 3D computer animation systems) can be an important issue.

Originality/value – The main interests of the presented system face compared with the existing ones are the coordinated use of several alife techniques, the integral approach (design of both the environment and the inhabitants), a computationally reasonable implementation of the system as an artificial ecosystem, and the possibility of using the results of such a research system as an input for a commercial 3D software in order to obtain 3D realistic computer animations.

Keywords Behaviour, Modelling, Genetic engineering, Fuzzy logic, Action research, Cybernetics

Paper type Research paper

1. Background

The development of behaviour modelling techniques can be applied to different fields like robotics (Mataric, 1997), computer simulation (Maes, 1994), computer animation (Tu and Terzopoulos, 1994) or virtual reality among others. In any case, what we obtain is a synthetic creature, software or hardware, autonomous and adaptive.

Kybernetes
Vol. 35 No. 9, 2006
pp. 1431-1451
© Emerald Group Publishing Limited
0368-492X
DOI 10.1108/03684920610662458

In this context emerges the concept of animat. According to Kodjabachian and Meyer (1994) and Meyer (1997) an animat is an artificial organism, either a simulated animal or a robot, with proper animal characteristics. His structure and functions are based on observed mechanisms in real animals; they are provided with sensors, actuators, and a behaviour model that relies, in an adequate manner, the perception system with the performed actions, making possible the survival of the animat within his environment.

An artificial world (Merelo, 1998) is a computational model used to solve any type of problems, using nature inspired techniques, like evolution, adaptation and competition. These worlds can be settled in a general context of complex adaptive systems, generally inhabited by agents, with simple and well-defined interactions, and with an emergent global behaviour which approximates the real behaviour of the real system.

There are several programs in the literature that simulate biologically inspired virtual worlds based on alife techniques; the reader can find more in www.rennard.org/alife/english/liensgb.html, www.alife.org/links.html. Most of them focuses on very concrete aspects, which makes difficult their use when the user wants to create a general artificial world, defining and controlling both, inhabitants and environment.

In this paper, we propose an integral approach to the design, implementation and use of an alife artefact, in order to offer to the user (an ethologist) a global simulation environment.

Owing to this, the purpose of this paper is to wholly describe the experience of designing, implementing and using an alife based system and to contribute to the scientific community effort in the alife area.

The rest of the paper is organised as follows. Section 2 presents the main features of ALVW system. Sections 3 and 4 describe, respectively, the design on the environment and the inhabitants. Section 5 makes some reflections about the computational complexity of the system. Section 6 shows two applications of the system. Section 7 presents some conclusions and future steps.

2. Introducing the ALVW system

The system for modelling artificial life and virtual worlds (ALVW) is a generic platform that allows the user to design an environment as well as the inhabitants who may live within it.

It is divided into two parts (Figure 1). The first module deals with the creation of a virtual ENVironment (ENV). The second module is specialised in modelling the Behaviour of the INhabitants (BIN) and is based on a perception-analysis-reaction loop.

The environment (modelled by means of a changing environment) is made up of special elements, dynamic objects and static objects:

- *Special elements* are controlled by a pseudo-agent based on genetic algorithms (GA) that controls its evolution over time.
- *Dynamic objects* are those whose position changes with time, following a fixed path with certain velocity and acceleration. Their behaviour is not based on any agent.
- *Static objects* are those whose position do not change with time.

The inhabitants are based on a perception-analysis-reaction scheme and are characterised by an autonomous and adaptive behaviour. They make use of

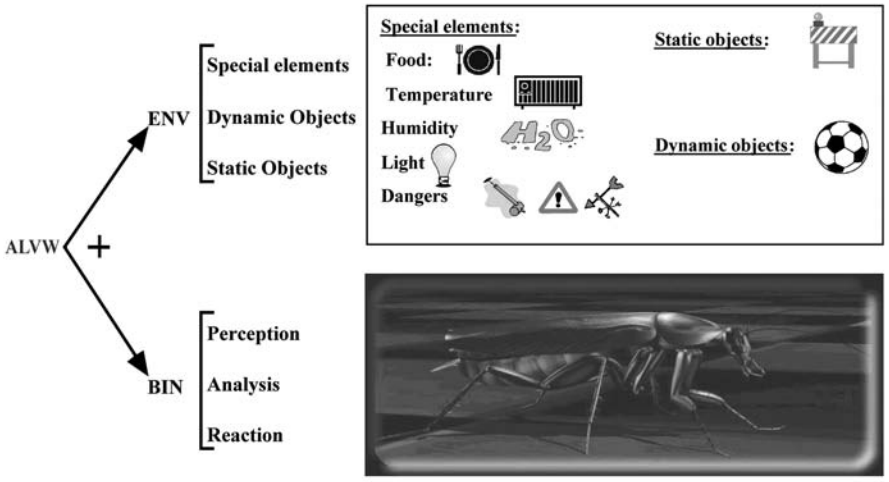


Figure 1.
Architecture of the ALVW system (composed of two modules ENV + BIN)

an intelligent agent composed by two modules: a "fuzzy expert system" (FES), that analyses the values coming from the sensors, and a behaviour selection module (SELEC) based on an action selection mechanism.

The interaction among all these elements is as follows. The inhabitants can perceive all types of inhabitants and objects and can perform actions that modify whatever inhabitant and object. These actions are the materialisation of one or more behaviours. The set of all the behaviours of an inhabitant constitutes his global behaviour.

Therefore, once basic behaviours (for inhabitants) have been defined, a global behaviour emerges when combining those behaviours with a changing environment. The result is a set of autonomous inhabitants able to detect changes in the environment, analyse them, to react and, depending on their behaviour, adapt themselves to the environment.

3. Module ENV: designing the scenario

The purpose of ENV in designing scenarios is to provide a tool, powerful enough, to model any type of object on the scene. We next define the three different types of elements that can be used for modelling the environment: special elements, static objects and dynamic objects.

GA are used to code and control special objects on the scenario. Any individual element is coded by a chromosome, which is a string of n bits (0 or 1). Every bit corresponds to a gene, and is responsible for some part of the individual characteristics. A population with m individuals of a class of special elements will be coded with m chromosomes.

The evolution is assured by the classical genetic operations (Goldberg, 1989): selection, crossover and mutation. This evolution has two important characteristics: a specific long-term goal (fitness function) and an unpredictable behaviour (random crossover and mutation).

For example, a heat source can be coded as a special element characterised by a value, its temperature. But besides this value it may have other attributes like position, direction, velocity (if it is moving), increment or decrement of intensity, … that will be treated as genes. Another example would be a special element meaning danger: besides some genes analogous to the ones in the previous example.

Dynamic objects have spatio-temporal scripts that use procedural motion control in order to move them. This way, a fixed path is defined (although it can be randomly generated) with a given velocity and acceleration, and perhaps with some kind of time-varying specific characteristics. This script can be performed once or can be cyclically repeated. As an example, the effect caused by a lightning, a light line traversing the scenario, can be repeated regularly or it can be something unique and accidental.

Static objects do not have any motion control method, but they can be perceived by the inhabitants. Even though they have some attributes, they have no time or spatial evolution in terms of motion modelling. They can be considered as obstacles in the environment.

4. BIN module: designing inhabitants

An inhabitant is able to show a global behaviour composed of several basic behaviours according to the perceived situation. This global behaviour follows one or more goals and may be influenced by internal motivations. Motor actions (performed by motor skills or commands) allow him to achieve the current behaviour(s).

The scheme for the BIN module is shown in Figure 2. This module performs the perception-analysis-reaction loop and is composed of three main blocks: the

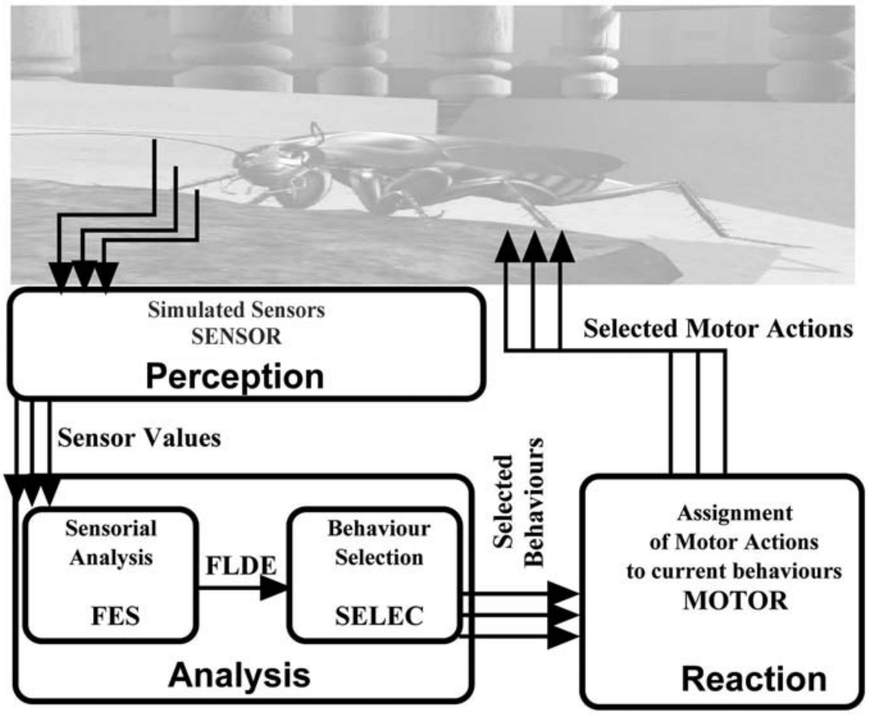


Figure 2.
General scheme for an inhabitant: the BIN module

perception, the analysis and the reaction ones. The first block just reads values coming from the sensors. The second one, the analysis block, is composed of the FES and the SELEC modules. The FES module is based on a fuzzy expert system and processes values from sensors and provides the next layer what we call first-level decision elements (FLDEs). Then, the SELEC module takes the FLDE and, depending on the current behaviour, the goals, the motivations and the behaviour characteristics of the inhabitant, decides which behaviour(s) should be carried out. Once this is decided, the third block associates motor actions to the chosen behaviours.

4.1 FES module: a fuzzy expert system to process environmental data

The main goal of the FES module is to analyse sensor values in order to obtain what we call FLDEs. It provides the inhabitant with the capability of processing single values coming from the sensors and obtaining more elaborated knowledge. This knowledge will be used by the SELEC module to decide next action(s). As an example we may know the temperature (15°C), and the humidity (55 per cent), but we should decide if the ambient, combining temperature and humidity, is very good, or only supportable for the inhabitant. Figure 3 shows the structure of the FES module with the three main stages that have to be accomplished: fuzzyfication, inference and defuzzyfication.

The user has to define the membership functions of the physical values involved, as well as the rules used to infer the FLDE's. Several other options like assigning weights to the rules or activating and deactivating rules are available.

4.2 SELEC module: behaviour selection

The SELEC module takes as inputs the FLDE from the FES module and gives the selected behaviour(s) to the next module. In order to do that, there is a behaviour selection algorithm that uses a behaviour knowledge base (BEHAV), a base of truth facts (BELIEF) and the FLDE, to choose the selected behaviour(s). Figure 4 shows this 3 different elements involved in the process.

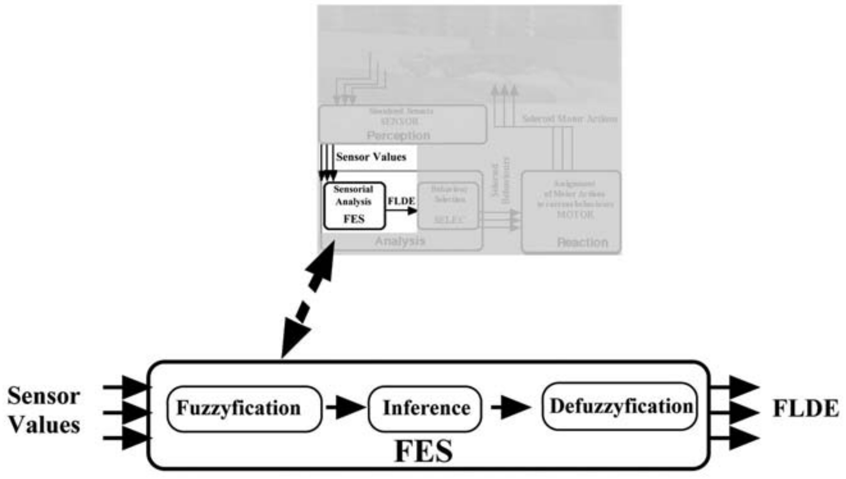


Figure 3.
FES module

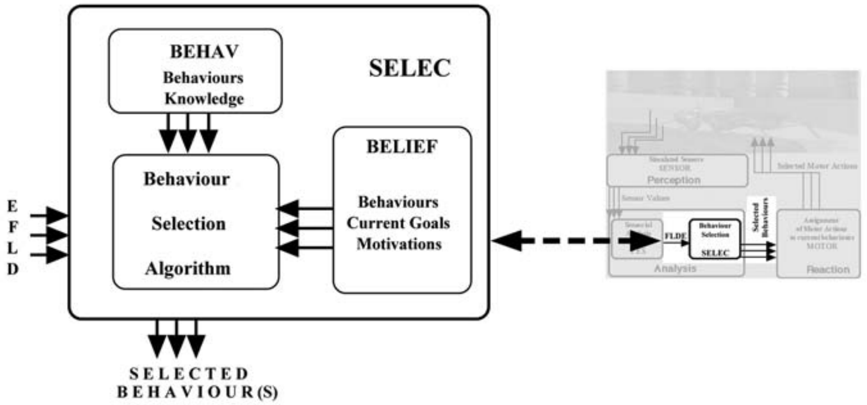


Figure 4.
Detail of the Selec module

The SELEC module builds a behaviour hierarchy considering the following aspects:

- relative importance of all behaviours;
- integral analysis of all the candidate behaviours;
- temporal/permanent goals and motivations fulfilment;
- compatibility of temporal and permanent goals;
- inhibition between behaviours;
- behaviour persistence;
- opportunism; and
- simultaneity.

4.2.1 Computational model of a behaviour. Every behaviour has a computational model based on:

(1) a priority order (a continuous function);
(2) possible inhibitions (continuous functions);
(3) a sequence of subbehaviours (optional);
(4) a set of goals/motivations (a list of goals/motivations and number of occurrences);
(5) a set of coefficients for current goals and/or motivations; and
(6) a degree of activation (a continuous function).

Items (1), (2) and (3) are encoded within the BEHAV behaviour knowledge base, items (4) and (5) are encoded within the BELIEF base of truth facts and item (6) is the result of processing the information given by the FLDE's, BEHAV and BELIEF.

4.2.2 BEHAV module: behaviour knowledge.

4.2.2.1 Priority order. The priority order indicates personal preferences among the different behaviours. This order is established in every instant t depending on the $P_{c_i t}$ priorities of each c_i behaviour in that instant. The priority of each behaviour is mathematically represented by a continuous function that evolves with time, and whose values increase and diminish. This function defines, for every behaviour c_i, a priority $P_{c_i t}$ that is a function of time and of other k behaviours:

$$P_{c_i t} = f(t, P_{c_1}, P_{c_2}, \ldots, P_{c_k}).$$

Using this function, the priorities can take different forms:

- *Constant value.* The priorities remain constant during the simulation. Therefore, the sequence of priority does not change as time passes: $P_{c_i t} = \text{Constant}$.
- *One variable value.* The priorities are only function of time: $P_{c\_it} = f(t)$.
- *Several variables value.* The priorities depend on several variables; besides the time, the priorities of the other behaviours come into play every instant. This reflects situations in which the preference for one behaviour increases and the interest for other behaviours decreases.

4.2.2.2 Inhibitions. This concept reveals incompatibilities among behaviours. An active behaviour can inhibit another behaviour completely or partially. This way, a behaviour can be totally cancelled or gradually damped by other one. Inh_{ij} is the inhibition of behaviour j due to behaviour i. The total inhibition of behaviour j will be the arithmetic mean (or other operation) of all the inhibitions caused by the other behaviours.

The mathematical representation of inhibitions is an adjacence matrix. If there is an inhibition arc between behaviour i and j, there will be a value comprised between 0 and 1 in the (i, j) cell. If there is no inhibition, the value will be 1. This way, for a given behaviour its inhibition will be the arithmetic mean of its column.

4.2.2.3 Sequence of subbehaviours. The subbehaviours sequence establishes the splitting up of certain high-level behaviours into low-level subbehaviours. The behaviour definition is provided once and it does not vary along the simulation. There is no parameter associated to this sequence and the selection algorithm (described in the next section) has only to traverse it following the established order.

4.2.3 BELIEF module: base of truth facts.

4.2.3.1 Goals and motivations. There are two types of goals. The temporal goals, that turn into goals in some moment during the simulation due to the activation of a certain behaviour and the permanent goals, which last all the simulation. These latter goals are called motivations in ALVW. For instance, the behaviour "eating" can be a motivation for a greedy inhabitant and it would be a permanent goal for him. But if an inhabitant, due to a lack of energy needs to seek and ingest food, the behaviour "eating" would be the present goal and, therefore, a temporal goal.

There is a global list, $L_{\text{goals/mot}}$, that contains in every moment present goals and/or motivations. Every list element contains a specific goal or motivation and the number of occurrences (because it is possible that a goal or motivation is favoured by several behaviours). When the simulation begins, the list is initialised with permanent goals. As the simulation moves forward, the list has to be revised in order to add or delete elements.

4.2.3.2 Coefficients for current goals and/or motivations. The goal degree fulfilment coefficient represents the importance of the goals, and for a behaviour c_i it is defined as follows:

$$\text{Fulfilment\_Coef}(c_i) = \begin{cases} \prod_{\text{elt}} (V_{\text{elt}} * \text{Factor}_{\text{elt}}) & \text{If and only If } \exists \text{elt} \in L_{\text{Goal/Mot}} | \text{elt} \in \text{Goals}(C_i) \\ 0 & \text{if not} \end{cases}$$

where

V_{elto} is a constant

elt is a goal or motivation present in the global goal/motivation list

$\text{Factor}_{\text{elt}}$ is the number of occurrences of elt

$\text{Goals}(c_i)$ is the specific goals list of behaviour c_i.

4.2.3.3 Current behaviours. The BELIEF module also monitors the current state of every behaviour (executing or not executing). This is important for the subsequent behaviour selection.

4.2.4 Activation degree. The mathematical representation of the abstract concept of the current importance of a behaviour is a continuous function (one for each behaviour) that evolves increasing or decreasing with time and which models the importance of a behaviour compared with others during a simulation. This function is called the behaviour activation degree.

Each behaviour has associated a preconditions list, that are those FLDEs that can affect its possible activation. Each FLDE is represented by a continuous, increasing or decreasing, function that evolves with time and whose value is defined in the sensorial analysis performed in the FES module. The FLDE has a numerical parameter, called activation threshold, that shows the minimum value so that FLDE can be considered active.

The requirement for a behaviour to be activated and to begin to compete with the rest of active behaviours, is that all those FLDE that conform the behaviour preconditions list are active, i.e. above the activation thresholds:

$$\forall \text{FLDE}_i \in \text{List}(C_i), \text{FLDE}_{it} > U_i$$

where

FLDE_{it} is the FLDE_i value in an instant $t(\text{FLDE}_{it} \in [0, \text{Val}_{\text{Max}}])$

$\text{List}(C_i)$ is the preconditions list of the C_i behaviour

U_i is the FLDE_{it} threshold

Val_{MAX} is the maximum value for a FLDE.

The total value of all FLDE's for an active behaviour c_i, is given by its arithmetical mean:

$$\text{FLDE}s(C_i) = \frac{\sum_{i=1}^{n} \text{FLDE}_{it}}{n}$$

where

FLDE_{it} is the FLDE_i value in instant $t(\text{FLDE}_{it} \in [0, \text{Val}_{\text{Max}}])$

n is the number of FLDE's of the c_i behaviour.

In these conditions, if a behaviour c_i is active, its relevance in the new behaviour selection process is given by the activation degree function, $a_{c_i t}$, which is defined:

$$a_{c_i t} = \text{FLDE } s(C_i)\text{op Fulfilment\_coef}(c_i)\text{op Inh}_{c_i t}\text{op } P_{c_i t}$$

where

FLDE $s(C_i)$, Fulfilment_coef(c_i), Inh$_{c_i t}$and $P_{c_i t}$ have been previously defined
op is the type of operation used (* or other).

A behaviour importance sequence is obtained by sorting the activation degrees of each behaviour at every instant of time (Figure 5). Some simple arcs can be observed; they define the degree of importance of five behaviour nodes at instant t. There are also some double arcs that represent inhibition between node 1 and the nodes 2, 3 and 4. Finally, there are two behaviour nodes, 2 and 3 that are made up of several subbehaviours as defined by the triple arcs that come out from both nodes. Each behaviour node is activated by means of one or more FLDE (in the example each behaviour has only one FLDE).

Summarising, the computational model of every behaviour is made up of the following elements:

- a priority ($\in \mathscr{R}$) that can be a function of time or of other priorities;
- a possible inhibition ($\in \mathscr{R}$), consequence of the inhibition graph;
- a possible subbehaviour sequence, reflected in the subbehaviours graph;
- a set of goals which will be fulfilled in case of activation;

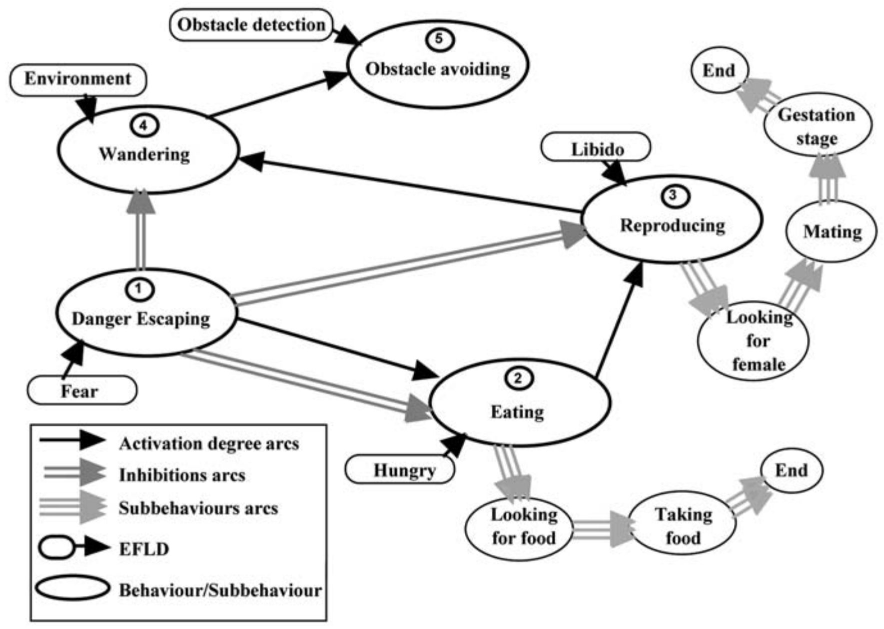


Figure 5.
Behaviour sequence importance at instant of time t

- a coefficient showing the degree of fulfilment that reflects the support of goals and motivations;
- a set of elements of first level of decision ($\in \mathscr{R}$) that determine if a behaviour is active or not;
- an operation type (* or other); and
- an activation degree ($\in \mathscr{R}$).

4.2.5 Behaviour selection and types of behaviour.

4.2.5.1 Behaviour selection algorithm. The behaviour selection algorithm in the SELEC module evaluates all possible next behaviour(s) and chooses the next one(s). The selection algorithm has the following aims:

- to perform an integral analysis of all behaviours;
- to follow temporal/permanent goals;
- to consider inhibitions between behaviours;
- to divide (if possible) behaviours into subbehaviours;
- to take opportunism and/or simultaneity into account; and
- to assure persistence, avoiding dithering, remembering.

We describe now the basic steps of the selection algorithm. First of all, it computes the activation degrees for all the behaviours. Afterwards, all non-inhibited behaviours are analysed, and the one with higher priority is chosen. This may produce a change in the current behaviour(s). If no new behaviour is chosen attention is given to the current behaviour(s). There are three issues that have to be taken into account.

Opportunity. In some situations, it could be interesting to stop, for a short time, current behaviour(s) and to execute another one. For example, if some nice food appears when escaping from a not very huge danger, it may be opportune to stop and eat a little and, afterwards, to continue escaping. That is why the action-selection algorithm, when considering current behaviour, takes a look to the rest of the behaviours to see if there is one candidate to be opportune.

Avoiding dithering. The constancy in selecting and maintaining behaviours is crucial to accomplish goals. To do this, a pile contains information about the tasks that have been interrupted but that should be continued later; using this strategy it is possible to "remember" behaviours that have been suddenly changed and not finished, and it would be possible to resume them later. It is also a way of improving persistence of behaviours.

Simultaneity. Several behaviours can be executed simultaneously, as it is very natural to have more than one behaviour in parallel.

To implement all these aspects several types of behaviour are defined.

4.2.5.2 Types of behaviour. The three main types of behaviour are:

(1) *Normal behaviours*. They are the main block of an inhabitant behaviour.

(2) *Opportune behaviours*. They may interrupt a normal behaviour (for a finite amount of time) allowing the temporal fulfilment of an advantageous behaviour.

(3) *Simultaneous behaviours*. Their activation may be parallel to the activation of a compatible normal behaviour in execution.

Both normal and opportune behaviours can be split up into one or more subbehaviours (conditional or temporal). Therefore, when one of these complex behaviours is selected, their subbehaviours are sequentially executed.

4.2.5.3 Proceeding to change a behaviour: the action selection mechanism that decides the winner. When all the activation degrees have been calculated, the behaviour with the greater one has to become the new behaviour (if it is not already active). Nevertheless, this general rule can be modified in order to favour for instance persistence, goals' consecution or the complete fulfilment of behaviours. Therefore, when the activation degrees of the "winner" behaviour and that of the actual one are similar, there will be a change only in case of emergency. Each behaviour C_i has an attribute to define its emergency. In case of draw between its activation degree, $a_{c_i t}$, and those of other behaviours (active or not) this attribute defines if the behaviour is prior or not (emergency situation). The difference $\mathrm{Diff}(C_i, C_j)$ between the activation degrees of two behaviours C_i and C_j, defined as $|a_{c_i t} - a_{cjt}|$, is used to decide the winner.

Dithering is usually detected among incompatible behaviours (these kinds of hesitations are typical of coupling situations as seeking food/danger or approach to female/danger). The system solves dithering using an hesitation level table, computed during execution.

4.3 MOTOR module

Once the current behaviours have been selected, the motor skills associated to the current behaviours have to be called, as it is shown in Figure 6. The MOTOR module does this.

This module takes the behaviour selected by the SELEC module, and calls the corresponding motor functions that perform pertinent actions on the inhabitant (locomotion), on the environment (eating) or on other inhabitants (reproduction). Depending on the type of motor actions and on the sensorial information they may need, this module would need to consult the perception module.

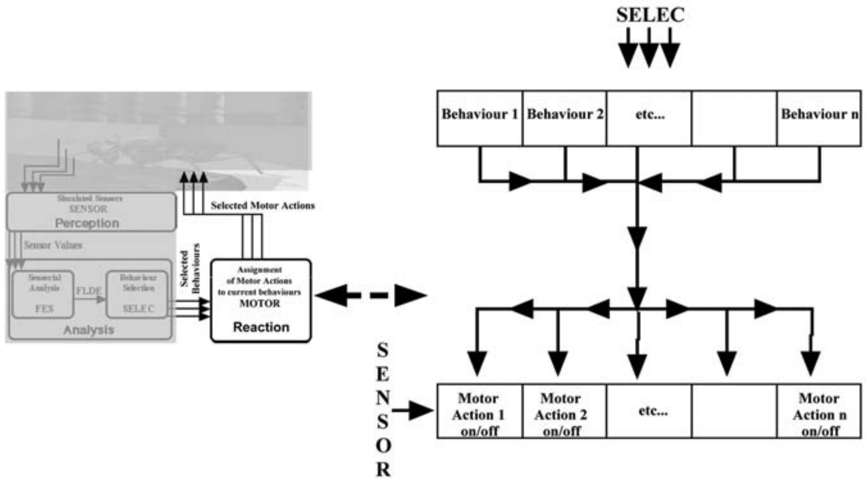


Figure 6.
Motor skills vs behaviours

4.4 User fine-tuning
The user can fine-tune the system by means of files or manipulating directly certain modules to adjust:

- rules, linguistic variables and FES belonging functions;
- behaviour's priority order;
- inhibitions among behaviours;
- goals and motivations fulfilment degree;
- activation degrees (consequence of the four previous ones);
- breaking down a behaviour into subbehaviours;
- sensorial system (fitting up/dropping out or sensor modification); and
- motor system (fitting up/dropping out or motor command modification).

5. ALVW technical specifications and computational complexity
The ALVW implementation has the following features:

- Object oriented
- C, C++
- MAC platform.

Regarding time management, the system clock has been used to control all those simulation aspects that have to deal with the real time passed during the simulation, such as the inhabitant's growth. Nevertheless, the user has been provided with a tool that allows moving forward in time. It can be used, for example, to accelerate the growth of a particular inhabitant. This time acceleration depends on the computer performing the simulation (CPU PowerPC G3, 266 MHz, 256 MB RAM) and on the system complexity, which is a function of the number of inhabitants within the scenario. In Figure 7 the time necessary to perform a complete perception-analysis-reaction loop

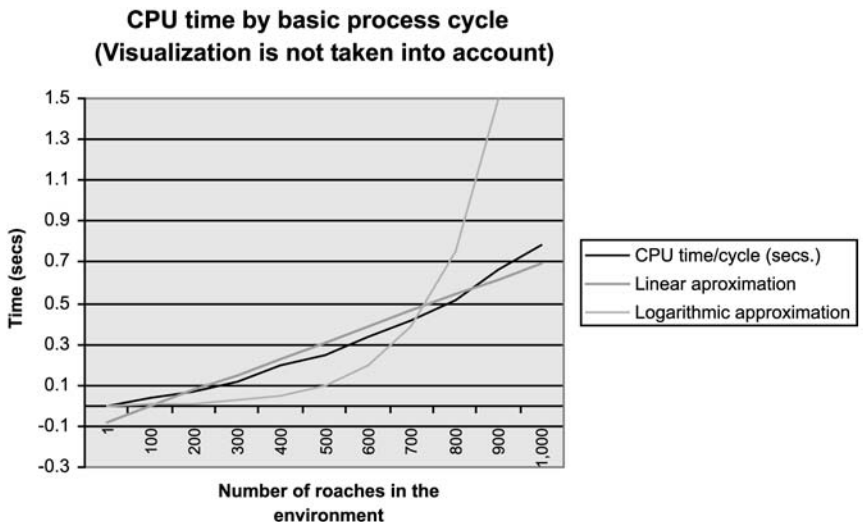


Figure 7.
Computer complexity of the system

(without considering visualisation) is plotted against the number of inhabitants. In the figure a linear and a logarithmic fitting are also shown. As it can be seen, the complexity evolution is almost linear and basically depends on the number of inhabitants.

6. An application of ALVW: Kukasim

In this section we present an application named Kukasim (Figure 8), based in the ALVW system described in the previous sections. This ALVW-based application allows the user to model both inhabitants as well as the virtual environments they are going to live in.

6.1 Kukasim simulations

Kukasim simulations allow to create an artificial environment where the evolution of synthetic cockroaches is studied over a period of time. To run a simulation, a series of rules must be set, to define the behaviours and the special elements, as well as some general parameters such as initial number of cockroaches, maximum allowed number during the simulation or several reproduction parameters for the females.

The complete process for each iteration of the simulation is as follows:

- perception of the environment through the simulated sensors;
- evaluation of the perceived variables through the FES;
- assignation of a behaviour;
- change of motor commands according to the current goal; and
- visualisation of the new situation.

The user can interact with the simulation in several ways:

- adding new elements to the environment;
- eliminating elements;
- changing the climatic conditions by defining a new station for the simulation (spring, summer, fall or winter); and
- fast-forwarding the simulation or forcing concrete situations.

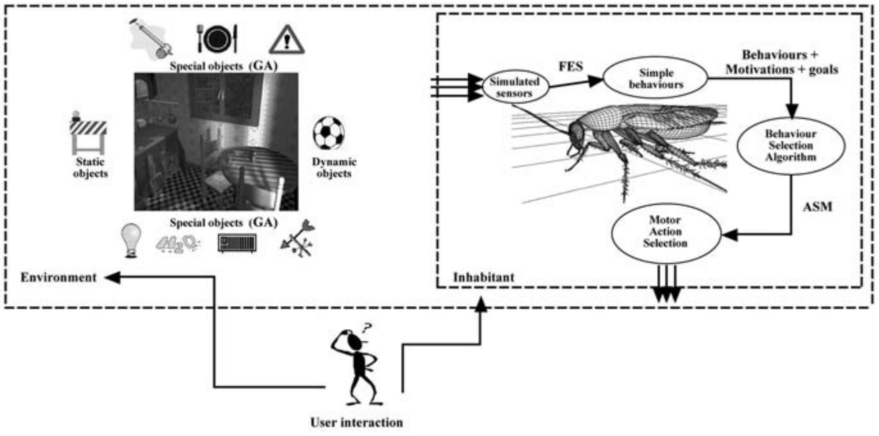


Figure 8. Kukasim: an application based on ALVW

At any given time, the user can visualise (modifying some of them) the following elements (Figure 9):

- behaviours (normal, opportune and simultaneous);
- vital data (energy and age);
- FLDE data; and
- map with an overall view of the artificial environment, indicating the position of the inhabitants.

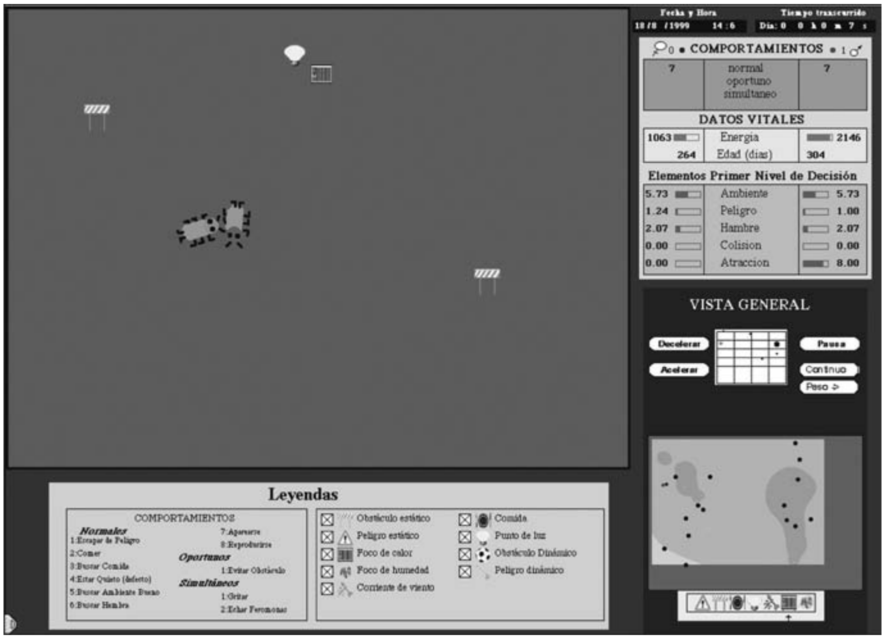
6.2 Experimenting with Kukasim

The ALVW system is a generic and open system which may be used to define any virtual environment and any inhabitant(s) based on a perception-analysis-reaction scheme. In the example we present, it has been used to model the behaviour of cockroaches. The simulator Kukasim allows the animator to define an environment (a kitchen which is the cockroach's habitat), to design inhabitants (cockroaches) who perceive, think and react in this environment. The use of the simulator enables to follow the evolution of the artificial world.

Table I shows the different kinds of elements modelled within Kukasim.

During the simulation the user has visual information of the main properties of the inhabitants (Figure 10).

As it has been explained, performing a simulation implies repeating a perception-analysis-reaction loop. The user can modify the frequency of this loop, and he can even run the system step by step or stop it. These controls allow the user to check some parts of the simulation with the required level of detail.


Fecha y Hora 18/8 /1999 14 :6
Tiempo transcurrido Dia: 0 0 h 0 m 7 s

0 • COMPORTAMIENTOS • 1

7	normal oportuno simultaneo	7

DATOS VITALES

1063	Energia	2146
264	Edad (dias)	304

Elementos Primer Nivel de Decisión

5.73	Ambiente	5.73
1.24	Peligro	1.00
2.07	Hambre	2.07
0.00	Colision	0.00
0.00	Atraccion	8.00

Leyendas

COMPORTAMIENTOS
Normales
1:Escapar de Peligro
2:Comer
3:Buscar Comida
4:Estar Quieto (defecto)
5:Buscar Ambiente Bueno
6:Buscar Hembra
7:Aparearse
8:Reproducirse
Oportunos
1:Evitar Obstáculo
Simultáneos
1:Gritar
2:Echar Feromonas

☒ Obstáculo estático
☒ Peligro estático
☒ Foco de calor
☒ Foco de humedad
☒ Corriente de viento
☒ Comida
☒ Punto de luz
☒ Obstáculo Dinámico
☒ Peligro dinámico

Figure 9.
Kukasim user interface

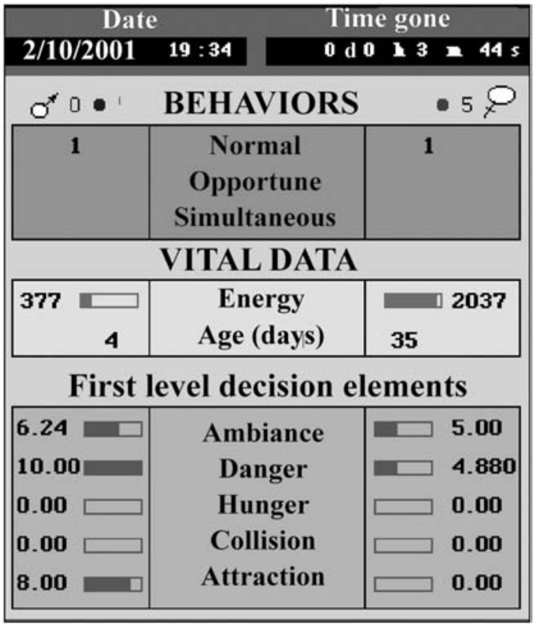
In the following figures several experiments using Kukasim are presented.

Figure 11 shows the percentages of every behaviour and the different types of behaviour in a simulation run. As it can be seen, danger escaping has been the most used behaviour, followed by the eating and looking for good ambience behaviours. The figure gives an idea of how the system is using the different types of behaviour. Most of the time, the roach is using the normal behaviours. In Kukasim, obstacle avoidance, for instance, is defined as an opportune behaviour, so every time the roach is avoiding an obstacle, it uses an opportune behaviour.

In Figure 12, the results of an experiment whose goal was to check the correctness of the exploring behaviour is shown. This behaviour makes a cockroach to wander around pleasant areas when possible. A cockroach decides if an ambience is pleasant or not analysing the quantity of light, the temperature, and the humidity (not very hot, not too dry, and not a too bright defines a pleasant ambience). As it can be observed the roach succeeds in remaining in pleasant areas.

Special elements	Food, temperature, humidity, danger, light
Sensors	Light sensors, chemical sensors, special sensors
Behaviours	Staying quiet, wandering, looking for food, eating, escaping from danger, obstacle detecting/avoiding, looking for female, mating, looking for better ambience
Motor commands	Moving towards/away, setting velocity, mating, eating

Table I. Different elements modelled in Kukasim



Date	Time gone
2/10/2001 19 :34	0 d 0 h 3 m 44 s

♂ 0 ● **BEHAVIORS** ● 5 ♀

1	Normal Opportune Simultaneous	1

VITAL DATA

377	Energy	2037
4	Age (days)	35

First level decision elements

6.24	Ambiance	5.00
10.00	Danger	4.880
0.00	Hunger	0.00
0.00	Collision	0.00
8.00	Attraction	0.00

Figure 10. Visual information during a simulation

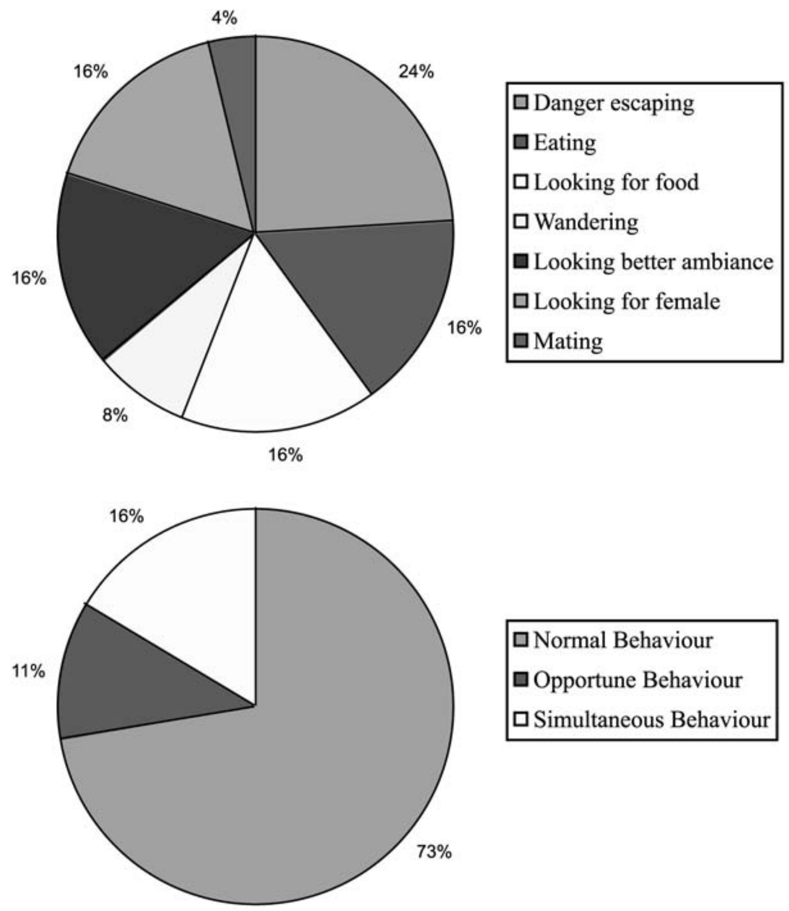


Figure 11.
Percentages of execution of every behaviour and different types of behaviour used during one session

Figure 13 shows the energy evolution for a cockroach during a session. The starting level is a random value, and then, every time the energy is going under a certain level, the roach has to find a food source, and eat in order to gain energy. If the roach remains alive, the energy level varies under a regular interval.

Figure 14 shows life expectancy of a roach (in minutes) related to the number of dangers in the environment. It also shows the number of escaping trials, which grows as the number of dangers increase.

6.3 Using Kukasim in computer animation

The process of creating a 3D animation is by no means a simple one. Typically, the process is done by keyframes: the animator sets property values or keys (such as position of the legs of the cockroach or the situation of its centre of gravity), at specific frames, called keyframes. Values for the frames in between are calculated by the computer by interpolation, in a process known as in-betweening, thus producing the animation. Of course, this is the most basic way of creating computer animations with

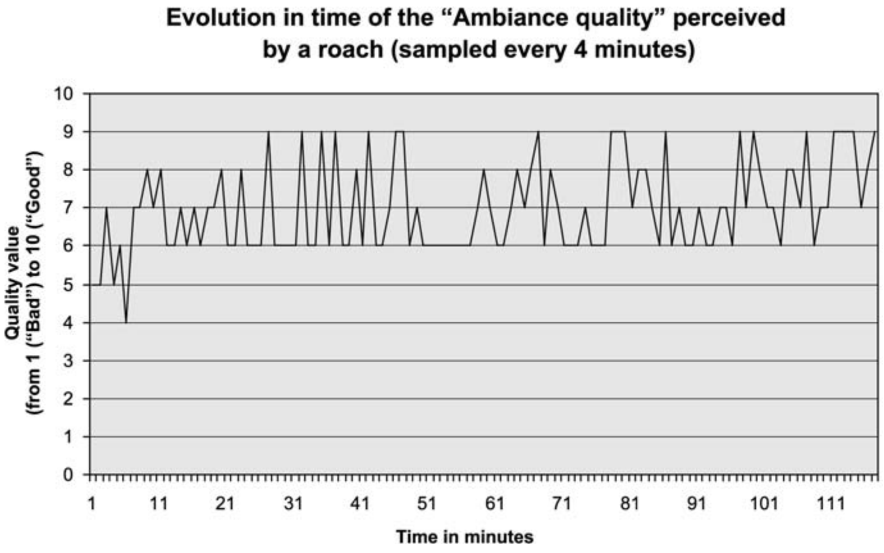


Figure 12. Evolution in time of the "ambience quality" perceived by a roach (sampled every 4 min)

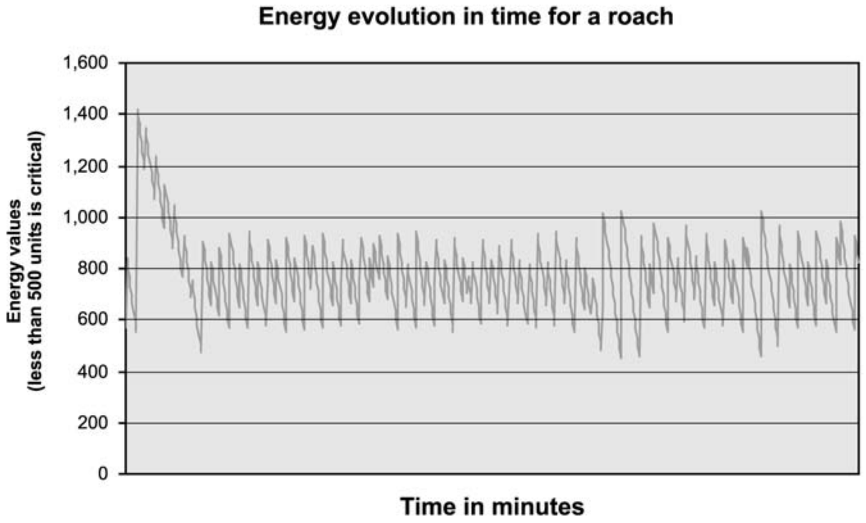


Figure 13. Energy evolution in time for a roach

today's commercial packages. More advanced techniques include inverse kinematics, dynamics, constraints or expressions.

This is a very counterintuitive way for animators to communicate with the computer, especially in the case of animating 3D characters, with several articulations and joints that need to move in a convincing way. Animation studios will usually hire technical directors to design high-level interfaces for the animators that hide the complexities of the animation process, thus reducing the gap between the human and the machine.

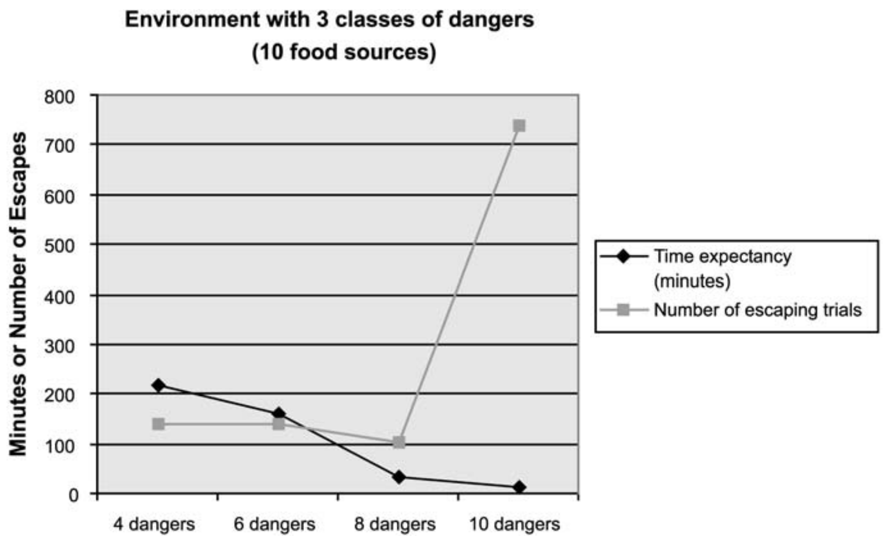


Figure 14.
Dangers and life expectancy (environment with three different classes of dangers and up to ten food sources)

We have tested a novel approach for producing 3D animations that starts with the simulation ran in the Kukasim system. Figure 15 shows an overview of the production of a 3D animation from the stored data of the execution of a simulation.

On the left of the Figure 15 (step 1) the simulator is running, and any relevant change of the state of the virtual ecosystem is stored on disk. Once a simulation is over, this data is integrated with a complete 3D geometric, visual and motion modelling of the cockroach and its environment, stored in a commercial 3D package (steps 2 and 3). The transferring of the data from the simulator to the commercial software is done via a custom plug-in written by the authors that transforms it into a format the package understands and can use. Plate 1 shows one frame of the animation produced in this way.

7. Conclusions and future steps

Our alife behaviour modelling system, named ALVW, allows us to design and manage artificial worlds and the inhabitants who live within it.

Global behaviour emerges from basic behaviours. A bottom-up approach is used: defining several basic and simple behaviours and letting them interact and compete among them, yields to more complex behaviours.

A specific application of the ALVW system, related with the modelling of cockroaches, has also been presented. The simulator, called Kukasim, has been used to design an ethologic simulation and observation laboratory and the results obtained for the moment have been fully satisfactory. We have shown also how to use Kukasim in the computer animation field, as an automatic scripts generator, and how it is possible to embed the Kukasim results within a 3D commercial software.

In spite of the results obtained, there is still a lot of space left for improvement. The creation of intelligent autonomous entities relies on three mainstays: behaviour, learning and cognition. In the paper, behaviour has been profusely treated. We will now make some comments about the other two.

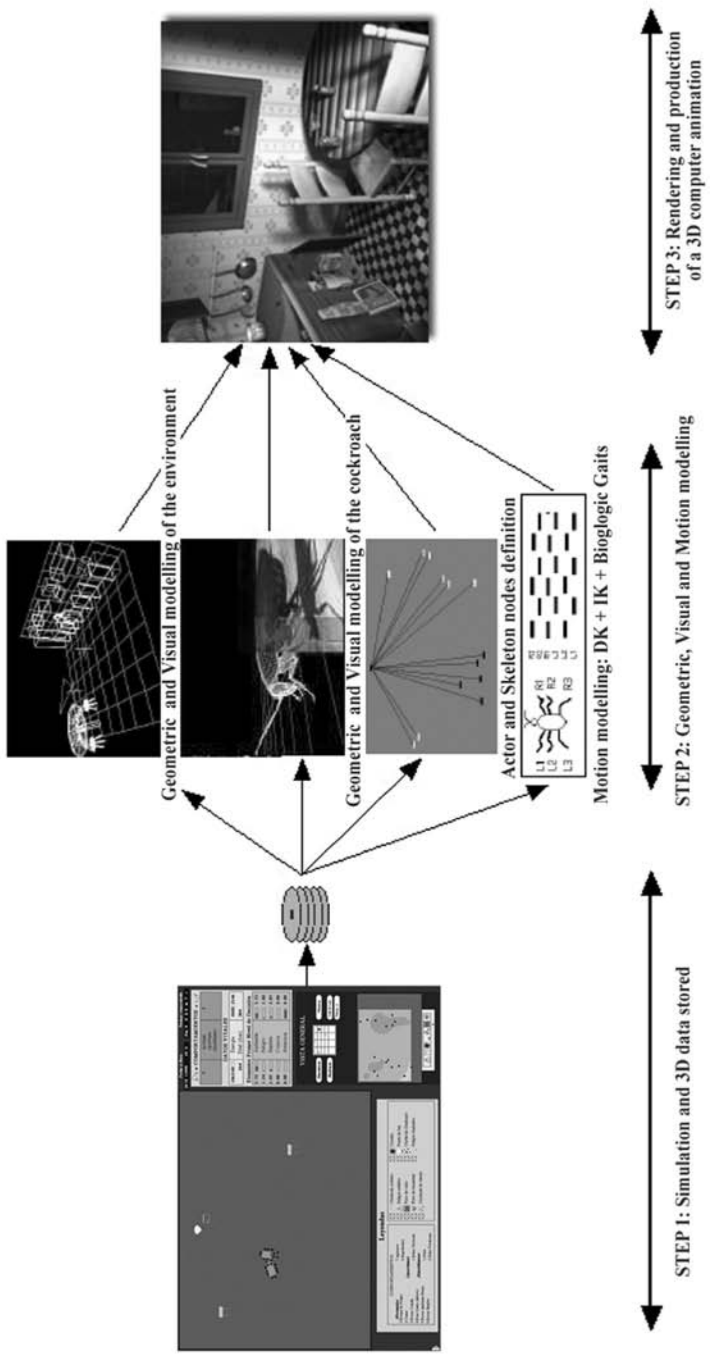

STEP 1: Simulation and 3D data stored
Geometric and Visual modelling of the environment
Geometric and Visual modelling of the cockroach
Actor and Skeleton nodes definition
Motion modelling: DK + IK + Bioglogic Gaits
STEP 2: Geometric, Visual and Motion modelling
STEP 3: Rendering and production of a 3D computer animation

Figure 15.
Steps needed to produce 3D computer animations using Kukasim

Plate 1.
A snapshot of the 3D animation obtained

Learning is the capability, present up to the point in all natural systems, to adapt themselves to their environment. From a computational point of view, it can be understood as the process to obtain some kind of function that allows to achieved, based on previous experiences, general laws about the environment. This way, the entity will be capable of adapting itself to the environment and of interacting with it in a way optimised from the point of view of its goals. In order to obtain a solution characterised by a good generalisation capability, the learning algorithms should allow the construction of a cost function that will be optimised by means of the entity's personal experiences. As learning via experience seams to have a strong statistical component, the problem has a notorious complexity due to the scarce number of variables associated to the input space, and to the shortage, dispersion and uncertainty of the samples (experiences). Relevant works in this subject are those of Vapnik (2000) and Hastre *et al.* (2001).

In more advanced animals there is a more flexible coupling between perception and action due to intermediate processes such as memory, attention, language, … The result of all these processes is what is called cognition. The cognitive architectures are based on some common principles that connect sensorial stimuli with arbitrary sensorial experience associations, and on the selective improvement of this experience via learning and attention mechanisms. Cognition is a complex process that cannot be only understood in terms of stimulus-reaction schemes. The goal of cognitive processes is to translate the sequence of experiences and the environment stimuli into internal representations of reality. These cognitive "maps" are, in fact, subjective models of the external world. Outstanding works in this field are those of López and Connell (2001) and Funge (1999).

References

Funge, J.D. (1999), *AI for Games and Animation: A Cognitive Modelling Approach*, A.K. Peters Ltd, Natick, MA.

Goldberg, D.E. (1989), *Genetic Algorithms in Search, Optimisation, and Machine Learning*, Addison-Wesley, Reading, MA.

Hastre, T., Tibshirani, R. and Friedman, J. (2001), *The Elements if Statistica Learning*, Springer-Verlag, Berlin.

Kodjabachian, J. and Meyer, J.A. (1994), "Development, learning and evolution in animats", *Proceedings of PerAc'94: from Perception to Action*, IEEE Computer Society Press, Las Alamos, CA.

López, L.S. and Connell, J.H. (Eds) (2001), "Semi-sentient robots", *IEEE Intelligent Systems*, (September/October).

Maes, P. (1994), "Modelling adaptive autonomous agents", *Artificial Life*, Vol. 1 Nos 1/2, pp. 135-62.

Mataric, M.J. (1997), "Behavior-based control: examples from navigation, learning, and group behavior", *Journal of Experimental and Theoretical Artificial Intelligence*, special issue on Software Architectures for Physical Agents, H. Hexmoor, I. Horswill, and D. Kortenkamp, (Eds), Vol. 9, Nos 2/3, pp. 323-36.

Merelo, J.J. (1998), "Artificial worlds: computational tools for artificial life", *Summer Course on Artificial Life*, Castilla la Mancha University, Albacete.

Meyer, J.A. (1997), "From natural to artificial life: biomimetic mechanisms in animat designs", *Robotics and Autonomous Systems*, Vol. 22, pp. 3-21.

Tu, X. and Terzopoulos, D. (1994), "Artificial fishes: physics, locomotion, perception, behavior", *ComputerGraphics Proceedings (SIGGRAPH'94)*.

Vapnik, V. (2000), *The Nature of Statistical Learning Theory*, 2nd ed., Springer-Verlag, Berlin.

Corresponding author
A. Pina can be contacted at: pina@unavarra.es