Virtual Life Simulation

Csomai András, Babes–Bolyai Univesity, Intelligent Sistems

Abstract

Natural environment simulation is nothing but creating a virtual reality, an environment model, consisting of a surface relief fauna and flora. The most interesting and difficult problem is the creation of the fauna, which consists of animals with real like behaviors. Hence the characteristics of an animal are almost the same then of an intelligent agent, the most appropriate mode to populate our virtual environment is trough the use of intelligent agents. This paper is focused on the problem of simulating animal behavior (action selection, animal locomotion, navigation etc.). The first chapter presents the characteristics of virtual life, the second presents the main problem of simulating animal behavior, the decision making problem, and in the end we present our proposal, a virtual life form: the virtual ant.

1. Characteristics of a virtual living form and virtual life

To efficiently simulate animal behavior, we need to create a “mental model” of an animal, with certain characteristics [1.]:

Autonomy: Autonomy means, that a virtual life form is able to perceive and analyze the environment, and is able to decide how to react to changes by itself. Autonomy is necessary because of two main reasons: the less work done by the designer (we do not need to preprogram every single possible situation), the more realistic behavior.

Adaptation: Adaptation is the capability of the living form to promote itself in the environment it inhabits. This can be viewed in two views. In a broad view this is nothing but the evolution of the species which means, that in the reproduction process, the offspring is not only a simple copy of the predecessors, but it contains the “best” of the parent’s abilities. This evolution with changes is responsible for the evolution of life. In a narrow view, adaptation is to enhance an organism’s ability by learning so, to survive in more or less unpredictable and dangerous environments.

Interaction: Interaction is the virtual reality user's possibility to query the members of the virtual reality. For a virtual organism is not enough to be able to "live" in its environment, we must be able to monitor and change its state.

2. Decision making problem

The main problems in simulating animal behavior could be grouped in two:

- moving
- behavior

Moving in the virtual environment emerges problems such as: locomotion, collision avoidance, navigation, etc.

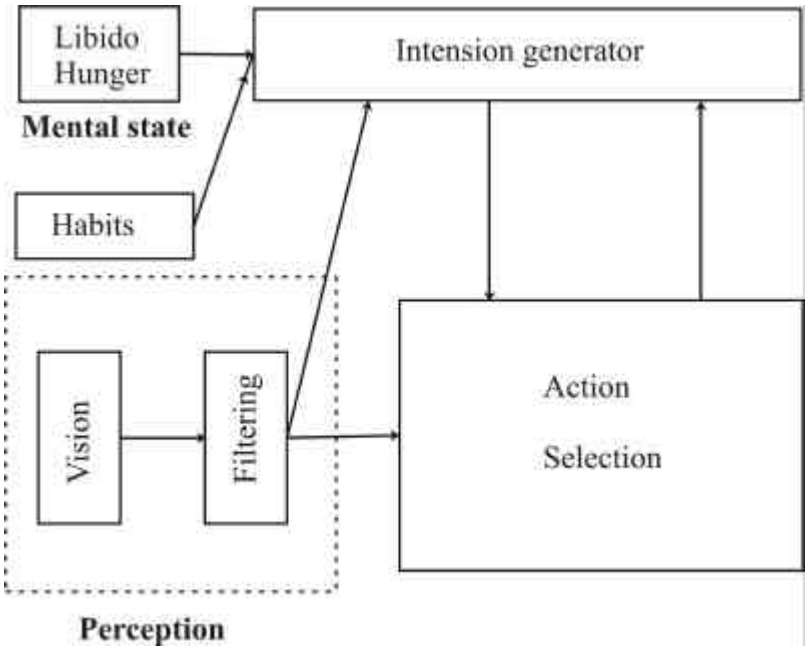
The behavior means the set if actions such as mating, reproducing, food foraging, eating, exploration, escaping from predators, etc.

In every situation the organism has to be able, to make the right decision to survive and propagate itself. This problem can be solved in many different ways. The classic reinforcement learning (Q learning), is not quite suitable for continuous state space - action mappings, therefore alternative methods must be used.

3. Virtual ant.

3.1 mental scheme

For studying behavior we do not need to concern about a virtual organism's locomotion. So in the visual ant project we do not consider this task. The goal is to create an ant, with real like behavior. Mainly the project is based on [2.] and [5.]. From [2.] we use the idea of the structure of the mind, and from [5.] the neural network represented decision policy. The reason why we decided not to use a rule based decision policy, is that we can create a virtual organism with less predictable behavior, so we can obtain a greater diversity of individuals. The scheme of the ant's mind can be seen below.



Scheme of the ant's mind

Mental state: the ant is a virtual animal driven by its instincts. Eats when hungry, mates when libido is high. So we have to track changes in its physical and mental state. This contributes in creating intensions.

Habits: habits are complex sets of actions, which are not implicit such as collision avoidance, but rather species specific, such as bringing food to the nest etc. These cannot be learnt implicitly, we have to "force" the ant to do it.

Vision: the virtual ant perceives the environment trough vision. It has a predefined angle of view, and distance. The artificial vision tasks such as object recognition are not the goal of this project, so these are not to be implemented; the ant would perceive the objects directly.

Filtering: Not every object in the field of vision is relevant to the current intension, so these are to be filtered out. The ant will focus only on objects that must be avoided (obstacles), and objects that are a goal (potential mate, nearest food source).

Action selection: The most important part of the ant's artificial brain is the action selection component. This is nothing but a policy, represented by a neural network. The network takes as inputs the intension and the identified objects gained from its sensors.

Intention generator: This is a way to control the virtual organism's actions. This is only guidance for the action selection part. In the virtual fish scheme its role

was essential, because the fish chose certain action routines based on its intentions. In our case, there are no action routines, so if we want to maintain some persistence in the ant's actions, we need to define its intentions. The eat and mate intensions are calculated as follows:

$$eat = \frac{hunger}{\max\_hunger\_value} + \left(1 - \frac{hunger}{\max\_hunger\_value}\right)\left(\frac{abs(dist - visibility)}{visibility}\right)$$

$$mate = \frac{libido}{\max\_libido\_value} + \left(1 - \frac{libido}{\max\_libido\_value}\right)\left(\frac{abs(distm - visibility)}{visibility}\right)$$

where *hunger* and *libido* are the mental variables, *max_hunger_value* is the maximum possible hunger value (in our case 100), *max_libido_value* is the maximum possible libido value (in our case 100), *dist* is the distance to the nearest food source, *distm* is the distance to the nearest possible mate, *visibility* is the view distance, or the number of grids, the ant can perceive around it (in our case 6). The true function of the intension generator is to maintain balance between the ant's basic instincts, in other words it has the role of not letting the ant be driven by only one instinct. If looking for food, a nearby potential mate could make the ant to mate instead of just walking by.

3.2 Physic part

The "physic part" of the ant (which is not represented on the mind scheme), takes the directions from the action selection block, and moves, refreshes the information in the vision block and the mental state variables. If the ant is over food, eats, if it is over a mate, mates. Adjusts the corresponding mental state variables (e.g. decreases hunger after eating, increases hunger if wandering or foraging). To ease the implementation, we decided to use a grid world. The size of the world is *n x n* usually *n=60*. The objects are represented as follows:

- food: *2*
- nest: *N*
- obstacles: *1*
- track:*

The world is surrounded by obstacles, the nest is always in the middle, the food sources are placed at random, after the disappearances of a food source, another is placed instead, also at random.

3.3 Action Selection

The action selection is reduced to finding the desired direction of walking. If the ant is hungry, looks for food, otherwise explores or brings food back to the nest. The policy is represented by a Multilayer Perceptron, with 7 inputs:

- *mate*: the strength of the mating instinct, it is produced by the intention generator, and it is calculated based on it's current mental state, more precisely the libido, and the distance of the nearest potential mate. Even if the libido is quite low, the presence of a nearby potential mate could cause a high mate desire.

- *eat*: it is calculated as the mate input, also by the intention generator

- *forage*: if the ant is not hungry, and not willing to mate (its personal needs are fulfilled), carries food back to the nest. This action is rewarded. During foraging, its other instincts are disabled.

- *fooddir*: the direction to the nearest food source. The direction is simply represented by the left, ahead, right directions, relative to the ant's direction, converted into real numbers: 0.0 , 0.5 , 1.0 . This information is provided by the filter in the perception block.

- *matedir*: the direction to nearest the possible mate. The direction is simply represented by the left, ahead, right directions, converted into real numbers:

- *nestdir*: the direction to the nest. It is represented as the previous two directions. This information is always known.

- *stepok*: this information shows, weather it is ok to step ahead or not, more precisely weather it is an obstacle ahead or not.

According to these inputs, the action selection method has nothing to do but to chose the direction to follow, and to decide, weather it should move or not. So the two outputs are:

- *dir*: the direction to follow, the response is converted into directions, such as left right and ahead;

- *step*: it is converted in a binary form, it shows, weather we move or not.

The perceptron has a hidden layer consisting of 12 neurons. The transfer function is sigmoidal.

3.4 Training

After experimenting with several classic genetic algorithms for neural network training, with different selection, mutation and crossover operators and not obtaining the expected results, we decided to solve the training of the neural network trough evolutionary programming [7. , 8.]. We use the real valued coding method, so the chromosomes consist of a real valued vector, describing the weights of the network. The architecture of the network is not changing during the evolution, so the mapping between the weights and a chromosome is easily done.

In evolutionary computation the only genetic operator used is the mutation. There are two main mutation operators, the Gaussian and the Cauchy perturbation. we use the first one. In this case, we need to extend the chromosome, which also has to contain the variance parameters for every gene.

The training algorithm is the following:

1. generate initial population of *n* individuals, in the form of (w_i, h_i), where $i = \overline{1,n}$. The individuals are built from random numbers, generated with Gaussian distribution.
2. each individual (w_i, h_i), $i = \overline{1,n}$ creates a single offspring (w_j', h_j'), $j = \overline{1,k}$

$$h_i'(j) = h_i'(j)\exp\left(r'N(0,1) + rN_j(0,1)\right)$$

$$w_i'(j) = w_i(j) + h_i'(j)N_j(0,1)$$

where $w_i'(j)$, $w_i(j)$, $h_i'(j)$, $h_i(j)$ denote the *j*-th component of the vectors w_i', w_i, h_i', h_i. $N(0,1)$ denotes a normally distributed random number with zero mean and 1 variance, *k* is the length of a vector. $N_j(0,1)$ indicates, that the random number is regenerated for every new value of *j*. The parameters r and r' are set to $\left(\sqrt{2\sqrt{n}}\right)^{-1}$ and $\left(\sqrt{2n}\right)^{-1}$.

3. Determine the fitness of every individual.
4. Build the new population by choosing n individuals from all the parents and offsprings with tournament selection. For every individual *q* opponents are selected at random, with uniform distribution. The individual is compared to its opponents, if the opponent's fitness is less or equal than the individual's fitness, it receives a "win". The new population will be built from the individuals with most "wins". The selection becomes more elitist, by increasing the value of *q*.
5. if stop criteria not satisfied, go to step 2.

The mentioned **fitness calculation** (step 3.) is being carried out according to a certain reward function. More precisely the individual is mapped into the neural network, therefore we gain an "Ant", which is placed into a random environment, and its performance is evaluated. To avoid the birth of pure "lucky" ants *(in some cases for example if the ant is placed in the middle of the environment, crowded with food sources, and never hits any obstacle, it's score might be to high in comparison to it's true capabilities)*, the ant is placed back *m* times. After every action taken, a certain award is gained, according to the reward function. The final score is the sum of all the awards and penalties obtained trough a run. The fitness of an individual is the average of the final scores obtained over the *m* runs.

The reward function gives the award according to the last action taken and its consequences, as follows:

1. if the ant steps receives the *step=1* award. This serves as a motivation to move.
2. if the ant steps into a cell never visited before, receives the *explore=5* award. The explore award motivates the ant to explore new territories, therefore finding new food sources.

3. if the ant eats, receives the *eat=50* award.
4. if the ant mates, receives the mate*=50* award.
5. if the ant hits an obstacle receives the *hitpenalty= -100* penalty.
6. if the ant is at hunger maxima receives the *hungpen=-3* penalty.
7. if the ant is at libido maxima receives the *libidopen=-3* penalty.
8. if the ant carries food back to the nest receives the *forage=900* award. The very high value of this award is explained by the very low probability of crossing the nest *(one cell from the n x n world, where n is usually 60).* To obtain this award, the ant has to cross the nest, and also has to have food on him. If the award would not be very high (relative to other awards), the ant would never learn this function.

3.5 Training Parameters.

In the training process the following training parameters were used:

- iterations: 1000
- population size: 150
- number of runs (*m* in 3.4): 3
- number of steps per run: 500
- tournament size (*q* in 3.4): 15

4. Bibliography

Most of the papers were downloaded from ***www.citeseer.com***

1. Fang Wang, Eric Mackenzie: *Virtual Life in virtual environments*.
2. Demetri Terzopoulos: *Artificial animals in realistic virtual worlds*.
3. Mark Humphrys: *Action selection methods using reinforcement learning*.
4. Fang Wang, Eric Mackenzie: *A Multi-agent based Evolutionary Artificial neural network for general navigation in unknown environments*.
5. Dario Floreano, Francesco Mondada: *Automatic creation of an autonomous agent: genetic evolution of a neural network driven robot*.
6. David E. Moriarty, Alan C. Schultz, John J. Grefenstette: *Evolutionary algorithms for reinforcement learning*. Journal of Artificial Intelligence Research 11/1999 199-229
7. D. Dumitrescu, B. Lazzerini, L.C. Jain, A. Dumitrescu: Evolutionary Computation.
8. Xin Yao: Evolving Artificial Neural Networks