MultiAgent System Architecture in Orphibs II

Conference Paper · July 2014

DOI: 10.7551/978-0-262-32621-6-ch095

3 authors:

Nuno Barreto
University of Coimbra
7 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Luis Macedo
University of Coimbra
89 PUBLICATIONS **322** CITATIONS
SEE PROFILE

Licinio Gomes Roque
University of Coimbra
74 PUBLICATIONS **202** CITATIONS
SEE PROFILE

DOI: http://dx.doi.org/10.7551/978-0-262-32621-6-ch095

Multiagent System Architecture in Orphibs II

Nuno Barreto[1], Luís Macedo[1] and Licínio Roque[1]

[1] CISUC- Centre for Informatics and Systems of the University of Coimbra
Department of Informatics Engineering
University of Coimbra
{nbarreto, macedo, lir}@dei.uc.pt

Abstract

Orphibs II is an Artificial Life game where the player takes the role of a caretaker of alien-like creatures called Orphibs. Each orphib is an autonomous agent that has its own personality, given by internal utility functions that can be genetically transmitted. In this paper, we present the architecture behind the agents present in the game we developed, as they feature a new mechanism built on top of a Goal-Based Behavior approach: genetic personalities. Genetic personalities introduce behavior variability for A-Life games and account for new gameplay mechanics such as artificial behavior selection. We start by surveying the state of the art regarding approaches to Multiagent Systems in videogames, in general, and Artificial Life games, in particular. We then describe the architecture behind the agents in Orphibs II, including their reasoning system and evolutionary mechanism. Finally, we present a discussion and potential future work.

Introduction

Inspired by biology, Artificial-Life (A-Life) games are a subgenre of Simulation games that simulate virtual life, where the main gameplay focus is on the interactions that the game's entities (usually represented as virtual living beings) have with the world as well as with the player [1]. Typically, A-Life games provide the player with a sandbox that can be used as a test bed for biological and, to some extent, social experiments. Several games, including Creatures [2] and The Sims [3], have shown that A-Life has market appeal and, at the same time, incites scientific curiosity [2]. There is also a research field behind A-Life which aims to develop techniques and approaches to create virtual agents that appear to be alive, or even sentient.

In this paper, we present a thorough analysis of the agent architecture and the mechanics behind Orphibs II, a game we developed (created to showcase the tools in [4]). Orphibs II is an A-Life prototype game where the player takes the role of a creature caretaker. It borrows some elements from Creatures and The Sims, such as creature evolution and goal-oriented behavior, but, at the same time, introduces new mechanics, namely the use of utility functions that can be genetically transmitted to account for genetic personalities. Moreover we provide a simple experiment to verify how a population of randomly generated Orphibs behave, i.e., if there any emergent behaviors and, in addition, the evolution of the population throughout the experiment.

The main contribution in this implementation lies in Genetic Personalities: an improvement upon the regular goal-oriented behavior which allows for behavior variability, as well as accounting for new gameplay mechanics such as artificial behavior selection.

This paper is structured in the following manner: Section II will present the State of the Art of A-Life games and approaches, Section III overviews Orphibs II and especially its architecture; Section IV showcases an experiment that we conducted as well as its results; and Section V describes future work; and, finally, Section VI, presents the conclusions.

State of the Art

Internally, A-Life games are composed of Multiagent Systems (MAS)which model the living beings the game is attempting to simulate. As such, agent development in this type of games follows an Agent Centric, or Bottom Up [5], approach. This is a method where the agents are entities that populate the game world. Here, agents are created with an internal reasoning system that allows them to interact with the world. Since each agent's reasoning system is independent from the others', this approach supports emergent behaviors. There is another agent development approach in videogames, the System Centric one, that is beyond the scope of this paper, because this type of approach is best suited for tactical behaviors [6] or game engine components [7].

While there are some techniques, including Finite State Machines [8], Behavior Trees [8] and Fuzzy Logic [8], which are commonly used in other genres, the A-Life genre employs techniques such as Rule-Based Systems [8], Neural Networks [8], Cellular Automata [8], BDI Architecture [9] and Goal-Oriented Behavior [8, 10].

Rule-Based Systems (RBS) [8] are a model in which the designer defines a set of condition-action pairs that result in simple behaviors. RBS are present in [11], where the authors developed a MAS to account for the emergence of circadian rhythms through evolution. Although most of the experiments showed that the agents did not present a circadian rhythm, they did, however, demonstrate interesting emergent behaviors. Aylett and Cavazza [12]illustrate another application for RBS, in their work. They explain that rule based systems have been used in crowd behavior. A sub-set of RBS, Cellular Automata [8] are a discrete model that consist in a grid of cells, where each cell has a value attributed according to a set of rules that takes into consideration its neighbors' values. Conway's Game of Life [13] is a direct usage of this approach.

Neural Networks (NN) [8] are inspired by how the brain's cellular structure works. An artificial brain is composed of layers of artificial neurons with weight links between them. In order to calculate the links' weights, feedback mechanisms may be used at run time, making this approach suitable for adaptive behaviors. The main disadvantage of NN is that they are a black box model. This means it is not clear to a human how a given Neural Network works, thus making it hard to debug. For this reason, they are not used in most games. In Creatures [2], agents use an architecture which includes NN to allow them to learn activities such as walk and communicate with the player.

Belief-Desire Intention (BDI) Architecture [9] is another approach in modeling agents. It was created to be akin to how humans make decisions. In this architecture, an agent contains a desire, belief and intention set. Beliefs represent the agent's perceptions, desires symbolize preferable end-states and intentions are the list of possible actions. This architecture has additional functions: the belief update function that allows beliefs to be refreshed according to new perceptions, the options function that uses agent's beliefs and intentions in order to produce desires and the filter function in which beliefs, desires and intentions are used to generate intentions. These mechanisms underlying the BDI architecture result in a plan, or a set of actions, for an agent to follow. This architecture, in conjunction with PSI psychological theory [14] for the display of emotions, is used by [15] with the purpose of creating autonomous non-playable characters in an educational Role Playing Game. Other uses of the BDI architecture include the design of agents present in Unreal Tournament 2004 [16] by [17] and [18]. In both cases, the authors argue that this approach yields agents with human-player-like behavior.

Goal-Oriented Behavior [8, 10]can be seen as a simplification of the BDI Architecture. Here, agents have a set of goals, usually depending on internal needs, and a set of actions. In this method, the agent performs actions to fulfill its current goals which, in turn, are chosen according to the agent's current needs. This method also supports utility functions. In this case, goals are ranked according to a function that maps needs to a score, called utility. As [10] explains, this is the technique used by The Sims videogame series.

A-Life Games

Although the A-Life genre is not as popular as others, there are several accounts of commercial and non-commercial games as well as research projects available.

Introduced as a mathematical game, the Game of Life, created by John Conway [13], consists in a 2D cellular automaton with a specific rule-set that states that each cell can either be dead or alive given its internal state as well as its neighbors'. The game can be played by creating the automaton's initial state and observing the results.

In 1994, Sims [19] developed a system where creatures would compete against each other in order to evolve morphologies and animation controllers that would adapt to a given environment. This work would later serve as basis for the game called 3D Virtual Creatures Environment [20, 21] that works in a similar manner.

Creatures [22, 2], developed by, among others, Grand is a game where the player takes care of virtual creatures called Norns. Norns can learn to adapt to their environment through NN linked to sensors and an artificial hormone system [2], Grand et al. even argue that social behaviors have been observed to emerge in Norns, such as playing collective sports yet they explain that due to the blackbox nature of NN, they could not verify the veracity of their claims. Recently, Grand has begun developing an A-Life called Grandroids [23]. Unlike Creatures, Grandroids are meant, according to Grand, to have a more complex artificial brain.

Little Computer People [24], one of the earliest Human A-Life commercial games, is a game that simulates the life of a single person inside a two-story house. The person acts autonomously and can even interact with the player through verbal communication and a poker mini-game. The player, on the other hand, can also persuade the virtual person to perform some actions through simple text commands.

Inspired by Little Computer People, The Sims [3] is a well-known A-Life series of videogames. Here, the player influences the lives of virtual people called Sims. Zubek [10] explains that sims are driven by needs. Each need, or drive, has an underlying function that converts drives into a global happiness value. Sims then select their next action in order to maximize their happiness. The Sims 2 also introduces the concept of personalities, which are implemented by attributing different weights to drives.

Another well-known series is Petz [25]. Petz is composed of the subseries Dogz, Catz, Babyz, Oddballz and others. Each subseries follows the same basic gameplay structure present in A-Life games: the player has to take care of pets or humans (both called petz by the creators), including dogs, cats, babies and alien-like creatures, while they develop. Petz are able to perform a variety of actions as well as display emotions through body language and facial expressions [26]. Internally, this is done through a set of mechanisms that control a specific feature, including breathing, emotions and multiple planning. Additionally, the Babyz subseries also features learning mechanisms [27] so babyz can learn to understand simple words.

Unlike the previous games, Chao Garden [28, 29] is not a game in itself, but rather a meta-game present in both Sonic Adventure and Sonic Adventure 2. Chao Garden introduces a different gameplay mechanic: the ability to train the game's virtual creatures, called Chao, for competition, by maximizing attributes such as strength and stamina. While independent, Chao cannot develop by themselves the attributes necessary for competing and require to be given special items by the player. Other particularity of this game is the fact that Chao can develop personalities according to the way they are treated, or mistreated.

Finally, Species: Artificial Life, Real Evolution [30] is a game where several species of creatures walk around a procedurally generated virtual world. The player can then feed them, kill them, and even extract their DNA to splice into another creature's DNA. Nevertheless, without the player's interaction, the creatures can still be part of a natural selection process.

Orphibs II

Game Overview

Orphibs II is a game inspired by Creatures and The Sims where the player takes the role of a caretaker for alien-like creatures called Orphibs. Much like other A-Life games, the player does not control an avatar but rather is represented by a hand that can pick-up objects, including Orphibs, scroll through the game world and observe the internal state of the agents. Figure 1 shows a screenshot of the prototype game:

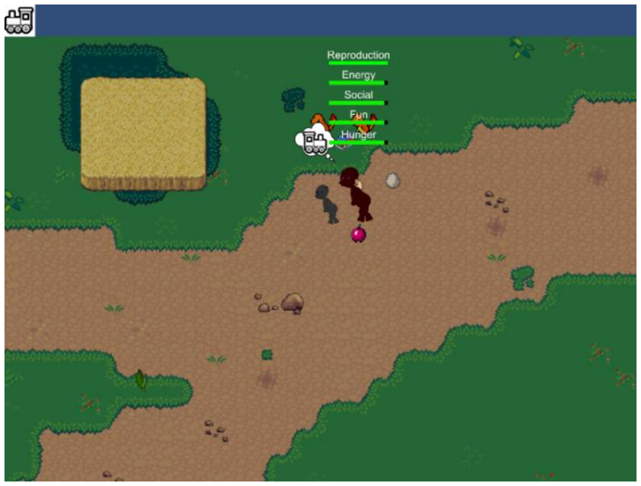

Fig. 1. Screenshot from Orphibs II. It illustrates two orphibs (one of them showing its attributes), an egg and several of the in-game objects: toys, an apple and the straw bed. It also shows one of the Orphibs' satisfaction levels as well as its current action.

The game takes place in a 2D top-down forest-like pathway map where both the player and the orphibs are constrained to the map's limits. The game world is populated by several objects including a square straw bed, a tree that produces a random number of apples at a constant rate and three toys. Orphibs can use each of these objects except for the tree. Additionally, the world is subject to a day-night cycle.

Orphibs Mechanics Overview

Orphibs are anthropomorphic bipedal oviparous and genderless creatures. They have some internal needs [8], called drives, that influence their actions. These needs are a simplification of the drives introduced in The Sims [10]. As such, Orphibs feature one physical need, hunger, and three mental needs: energy, fun and social. In addition, Orphibs have the need to reproduce.

Drives regulate which actions the orphibs can perform at a given time (see subsection Reasoning System for additional details). Currently, Orphibs can play with toys, eat apples, sleep on the straw bed, talk to each other, wander around the game world and reproduce. Whenever an Orphib performs an action that requires it to hold an object, it first travels to the object's position then it checks if the object is not being used by any other Orphib. If the object is indeed being used, the Orphib waits for its turn or looks for something better to do. This is accomplished with the help of an eyesight system that varies according to the orphibs' eye colors and the world's light intensity at a given time.

As previously mentioned, Orphibs are oviparous and, while they are genderless, they feature a sexual reproduction mechanism. This means that when they reproduce, they lay eggs which result in an offspring that receives genetic material from each of its progenitors (subsection Evolutionary Mechanism provides a more in-depth analysis of the Orphibs' reproductive system).

The Orphibs can also mature over time. They grow into a maximum height given by the function presented in Figure 2. This function was created empirically using data-points to simulate a growth rate that is not linear. Additionally, when growing, Orphibs display body discoloration.

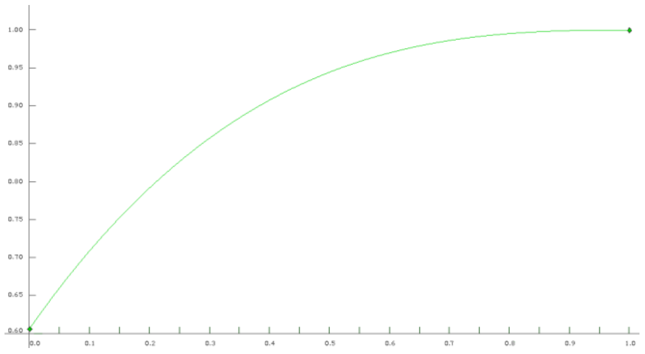

Fig. 2. Screenshot of the Growth Function in Unity3D's Curve Editor. The x-axis, ranging from 0 to 1, represents the Orphib's relative age, while the y-axis, also ranging from 0 to 1, represents the Orphib's relative height.

Orphibs also die during their lifetime. Every 5 seconds, a probability check is done for each Orphib by generating a value from a random uniform distribution and comparing it with the result of the death probability formula, which is given by a linear interpolation of the Age-Death Probability (AD) function and Energy-Death Probability (ED) function, expressed in the formula $p = 0.99 * AD(x) + 0.01 * ED(y)$. It is worth noting that both the functions and the interpolation factor where chosen empirically as a game design decision in order to avoid Orphibs to die too frequently. Figure 3 and Figure 4 illustrate the curves for both the ED and AD functions respectively:

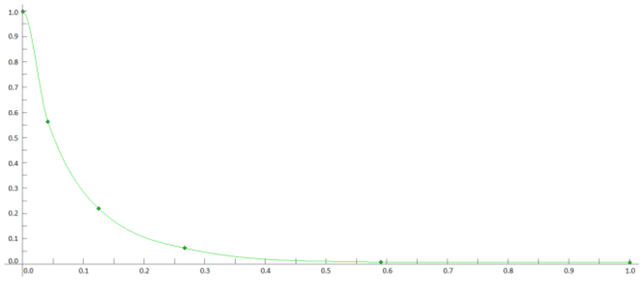

Fig. 3. Screenshot of Energy-Death Probability in Unity3D's Curve Editor. The y-axis represents the probability of dying, while the x-axis represents the energy need values.

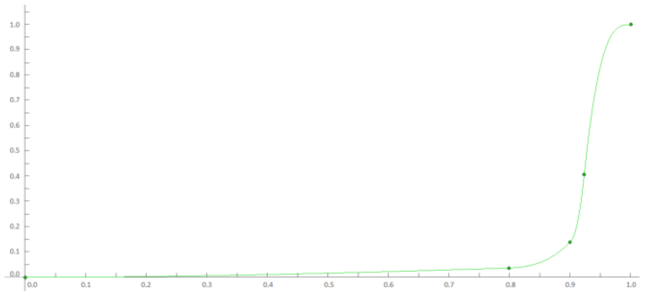

Fig. 4. Screenshot of the Age-Death Probability in Unity3D's Curve Editor. The y-axis represents the probability of dying, while the x-axis represents the Orphib's relative age.

Finally, when the player places the cursor above an Orphib, its attributes are shown on the screen. More specifically, the player can observe the creature's drive levels, in color-coded bars raging from green to red according to the drives' values, as well as the Orphibs' current and future actions.

Agent Architecture

The Orphibs II prototype was created using the Unity3D [31] game engine and, as a result, its architecture is component-based. On a side-note, due to the scope of the paper, this subsection will mainly focus on the agents behind the Orphibs.

Internally, each Orphib, game item (apple, tree, bed, egg and toys) and the map are composed of a hierarchy of Unity3D's GameObjects. For instance, the Orphibs contain a balloon object displaying the creature's current action, an object with the graphical representation of the satisfaction levels and a model object comprised of the creature's body parts for skeletal animation purposes.

Architecture Diagram. Figure 5, illustrates an Orphib's internal agent architecture which involves the following components:

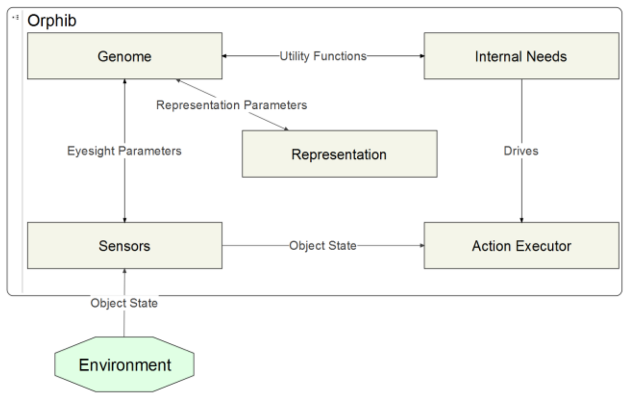

Fig. 5. The agents' architecture diagram.

- **Action Executor -** Module responsible for making Orphibs deliberate and execute actions (described in Reasoning System). It contains all of the available actions as well as a steering behavior mechanism [8], allowing the Orphibs to seek their destinations. Moreover, the Action Executor receives information from the Sensor and Internal Needs modules.
- **Representation –** Module containing the Orphibs' visual representation: eye and body pigmentation, animations and the aging mechanism (explained in more detail in subsection Orphibs Mechanics Overview). Physical traits of the Orphibs, such as their eye and body color, are stored in the genome module.
- **Sensors -** Module that perceives the world's objects as well as their external state. In the current version, the Orphibs only possess a visual sensor, i.e. virtual eyes, that varies according to day light intensity.
- **Internal Needs -** A module that stores and updates the satisfaction levels of the Orphibs' needs at a constant rate during every update time frame. Each satisfaction decays linearly according to a percentage with a fixed variable, $decayRate$: hunger decays with a rate of $1.1 * decayRate$, social decays with a rate of $1.7 * decayRate$ and fun decays with a rate of $0.7 * decayRate$. Reproduction and energy decay differently: the former only starts to decay when the Orphib reaches 10% of his total life span, to simulate puberty, with the rate of $4 * \frac{maxAge}{1+(age-0.1*maxAge)} * decayRate$. This means that while the Orphib ages, its satisfaction decaying rate decreases. On the other hand, energy changes according to the expression $decayRate * (1.7 - eyesight)$, where eyesight is the total relative eyesight given the Orphibs' eyes reaction to light intensity. This means that when Orphibs see less, they need to spend more energy.
- **Genome –** Module that stores the Orphib's complete genotype that is used during reproduction (for more details seesubsection Evolutionary Mechanism.

Reasoning System. The reasoning system behind the Orphibs is an implementation of the Goal-Based Behavior [8]. More specifically, it is the version present in The Sims 2 game [10]. As such, the algorithm is as follows:

1. Iterate through an action queue.
2. If there is an action, execute it and get rewarded, if applicable.
    a. If empty, select a new action:
    b. Iterate through every world object that can be acted upon.
    c. Rank each of the object's available actions using a score function.
    d. Add actions with scores above a given threshold.
    e. Sort all of the actions by rank.
    f. Select randomly one of the top 3 actions.

3. If still empty execute a random fallback action.
4. Repeat step 1.

This implementation takes into account some of the upgrade suggestions presented in [10], namely the fact that step e. was introduced in order to reduce deterministic behavior and how the action rank is calculated. Indeed, the score function used is given by the expression:

$$\mathrm{score}(action) = \\ = \sum^{needs} [U_{need}(current) - U_{need}(future)]$$

Where, for a given need, current is the Orphib's current satisfaction level, future is the sum of the current satisfaction level with the object's satisfaction fulfillment and U_{need} is the respective utility function.

Additionally, Orphibs can perform a meta-action "Wait" during the execution of another action. For instance, this happens when Orphibs want to play with toys that are being used by other Orphibs. In this case, the Orphib stops and waits for its turn. However, while waiting, the Orphib checks for better actions, by performing the selection sub-algorithm with a threshold equivalent to the current score of the action it is executing. Moreover, the halted action's score decays 10% every time the Orphib performs an unsuccessful selection. All in all, this decreases deterministic behaviors and reduces the probability of racing conditions.

In order to avoid unexpected behaviors, whenever an Orphib is picked up by the player's hands, it stops its current action and, in addition, aborts the current action of other Orphibs when engaged in group activities, such as talking.

Finally, the Orphibs' reasoning system also implements a basic personality model. This is accomplished by having different utility functions for each need [10], on each Orphib. As previously explained, utility functions are used to score actions on the reasoning system. They map a need value to a value that ranges from 0 to 1. Utility functions in Orphibs are created as data points that describe curves. This was chosen instead of a mathematical expression in order to better support genetic personalities (see subsection Genetic Personalities).

Evolutionary Mechanism

Since Orphibs support reproduction, they have an internal evolutionary mechanism consisting in a simplified Genetic Algorithm (GA) [32]. Nevertheless, some steps of the GA were not considered. For instance, Orphibs do not have an explicit fitness function, instead, their fitness is represented by their ability to survive long enough to reproduce. In addition, parent selection and survivor selection are not performed in a system-centric manner as the literature suggests but rather by the Orphibs' internal reasoning system and maturing mechanism respectively.

Algorithmically, the Orphibs' reproduction works in the following manner:

1. Approach potential mate.
2. Perform mating dance/animation.
3. Create egg.
4. Inseminate egg.
5. Recombine the genetic material from both progenitors.
6. Fulfill the reproduction need of both Orphibs by the same value.
7. Hatch the egg after 10 seconds have passed.

For recombination purposes, it is considered that the Orphib that engages in the reproduction action is the alpha progenitor, while the other is the beta progenitor. It is worth noting that the beta progenitor does not get to choose if he participates in the reproduction or not.

Genome. The Orphibs' genotype is composed of a data structure containing several data types. While it could have been implemented as a floating point vector, using a data structure increased its readability. Therefore, the Orphibs' genotype is as following:

- **Maximum Size –** Floating point that represents the Orphibs' maximum size.
- **Maximum Age -** Floating point that represents the Orphibs' maximum age.
- **Maximum Vision -** Floating point that represents the farthest an Orphib can see under ideal conditions.
- **Minimum Vision -** Floating point that represents the farthest an Orphib can see under non-ideal conditions.
- **Eye Color –** Floating point vector (for RGBA colors) that represents the Orphibs' eye color.
- **Body Color -** Floating point vector (for RGBA colors) that represents the Orphibs' body color.
- **Personality –** Vector of data point vectors. This represents the Orphibs' personality.

Additionally, these attributes are grouped so that the recombination operator is applied to each group independently. The groups are: one containing maximum size and maximum age, one composed of maximum vision and minimum vision, one comprising the eye color, one containing the body color and one for the personality.

Genetic Personalities. As mentioned, Orphibs have a simple personality profile given by their utility functions. These utility functions are transmitted from parents to offspring using the variation operators described in the next section. As such, Orphibs feature genetic personalities. Genetic personalities account for an interesting gameplay mechanic because players can use artificial selection to their advantage in order to produce Orphibs that behave the way they like. Artificial selection can be achieved by picking up an Orphib with the need to reproduce and placing it near another Orphib. The potential alpha progenitor will then attempt to reproduce with the closest Orphib in its range.

Variation Operators. The Orphibs' evolutionary mechanism uses both a crossover operator and a mutation operator. Recombination is applied to each group of the genome independently with a probability of 0.99 (although the literature suggests typical values of 0.9, this was not considered a typical application of GA). This means that each group has a 1% change of being cloned from the alpha progenitor's genome. The recombination algorithm used for most groups was Whole Arithmetic Recombination [32] with an alpha factor of 0.5 while the personality group uses an Uniform Crossover [32] with a p value of 0.5.

Unlike crossover, mutation is applied to each attribute, with a probability of 0.3 (this value was chosen in order to provide some local variability). The operator used was Non-Uniform Mutation with a Gaussian Distribution [32] with the previous value of the genome attribute as means and standard deviation of 1.

Experiment

Experimental Setup

We conducted an experiment to verify how a population of Orphibs behaves without the player's intervention, including if the population increases or stagnates overtime, if the average lifespan changes or if behaviors emerge.

The experiment consisted in running the Orphibs prototype during a 14-minute run with 15 initial Orphibs, randomly generated. It is worth noting that most of the Orphib's attributes were generated from a set of value ranges. For instance, the maximum size ranged from 0.1 to 1.1; the maximum age ranged from 1 to 6; finally, the eyesight's minimum and maximum values ranged from 0.5 to 10.5. These ranges were chosen to avoid unexpected events such as, for example, Orphibs dying during initialization due to a life span of 0. However, attributes such as the body and eye colors and the utility functions were generated without any additional boundaries, besides the ones imposed by their domains (for example, colors could only be generated inside the values permitted by the RGB model)

During the experiment, we recorded the population's size and the Orphibs' average age and maximum life span. This gives an idea of how the population evolves during the experiment. Other attributes such as average color, size, etc., were discarded as they did not have any impact on the population's evolution. In addition, we observed the Orphibs in order to access if any kind of behaviors emerged.

Experimental Results and Discussion

Figure 6 illustrates the Orphib population's evolution during the test:

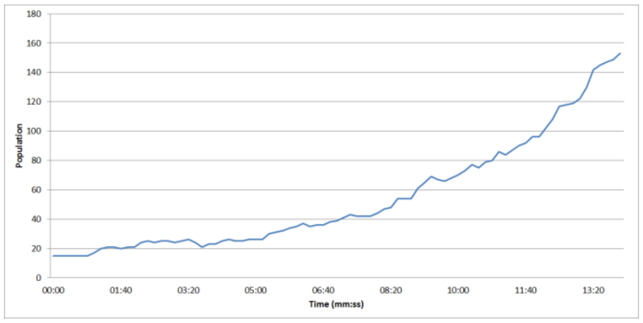

Fig. 6. Orphibs' population evolution over time during the test run.

As can be observed, the population tends to increase non-linearly without signs of stabilizing. In fact, during the tests it was observed that Orphibs developed a compulsive reproductive behavior, even though the reproduction's satisfaction levels' decaying rate decreases as the Orphibs age. Indeed, by the time the test ended, most Orphibs presented a reproduction utility function close to 1. This means that regardless of the reproduction need, its utility would always be high.

Another factor contributing for this increase is that, according to the Orphibs' death probability function, Orphibs are most likely to die of old age than from other events (The Future Work section presents some suggestions to circumvent this).

Figure 7 presents the Orphibs' average age and respective standard deviation:

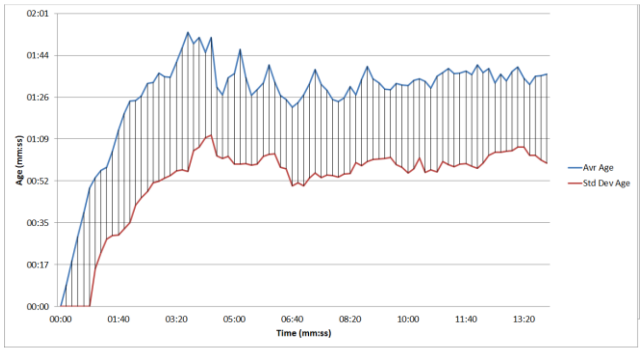

Fig. 7. The population's average age and respective standard deviation, expressed in minutes, during the test run.

There is a time slice, from 00:00 to roughly 05:00, where the average age continuously increases. This can be explained by the fact that, during this time, the population consists of the initial Orphibs. However, after 05:00, the average age tends to fluctuate between 01:26 and 01:44. Nevertheless, its standard deviation ranges between 00:52 and 01:09, meaning that there is some variety in the population's age.

The Orphibs' population average maximum life span is presented in the next figure:

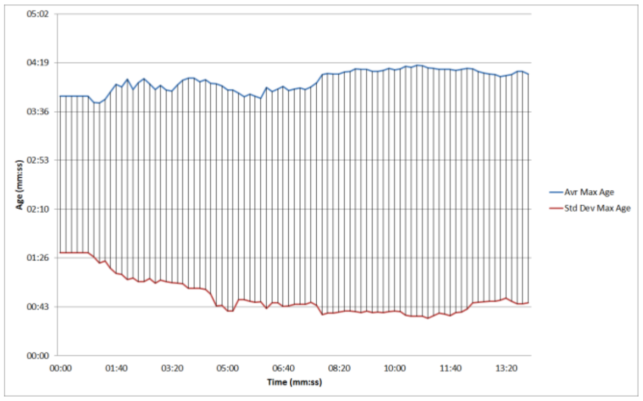

Fig. 8. The population's maximum average life span and respective standard deviation, expressed in minutes, during the test run.

Unlike the population's average age, its maximum life span is more stable: while the average ranges from 03:36 to 04:19, becoming closer to 04:19 at roughly 08:20, the standard deviation decreases to values less than 1 minute.

From the results in Figure 7 and Figure 8 we can conclude that, on average, the Orphibs were close to their midlife and, consequently, above the age where they start to have the need to reproduce. In fact, this corroborates the results in Figure 6 as there are no strong indications that the population is aging. Therefore, it would likely continue to increase in size.

Future Work

As was observed in the experiment, the Orphibs' population increased non-linearly. This was mostly caused by an insufficient population control mechanism. Suggestions of population control mechanism include the manipulation of the Orphibs' internal mechanisms or even the environment.

One of the most basic solutions, is to tweak the death probability function, either by changing the probability curves or by attributing more weight to the energy. Additionally, this function could be augmented to include hunger as a factor. However, balancing the function in order to stabilize the population can be difficult to achieve as reproduction is also dependent on the Orphibs' reasoning system.

Another method is to implement a choosing mechanism for mating. Currently, there is no potential mate criteria, unless a mate is provided by the player. By introducing some criterion that, for instance, restricts which Orphibs can be used for mating another one, reproduction may be more controlled.

Manipulating the environment to control the Orphibs ever-increasing population can be done by introducing the concept of predators, however this is beyond the scope of the prototype. Another way, is to introduce resource scarceness.

In addition to the system described in this paper, there were some planned features, such as implementing an emotional model, which, we believe, are interesting for future research but that, due to time constraints, had to be cut from the prototype.

Currently, the Orphibs are omniscient. This means that they have all of the information regarding the world state. However, initially, they were planned to have limited knowledge, and would use their eye sensors to discover landmarks and items. Additionally, they would talk with other Orphibs to swap information about the world. Moreover, they would use their memory to find objects. This could be implemented using the architecture described in [33] where agents explore the environment in order to fulfil their desire to minimize hunger, maximize information gain and maximize surprise

Another feature that was not implemented in the prototype was emotions. Implementing computational emotional models such as the Computational Belief-Desire Theory of Emotion (CBDTE) [34] could be achieved by combining the Orphibs' reasoning system with a memory mechanism. More specifically, the Orphibs' knowledge of the world could be converted into beliefs while the actions to be performed could be converted into desires. This way, beliefs and desires could be fed to the CBDTE in order to produce emotional responses. Consequently, the emotions could be used to influence actions.

Finally, the DNA mechanism could be improved further. It was planned to contain additional physical morphology traits such as body parts' textures. Computationally this would translate to texture morphing. In addition, the prototype was planned to contain the concept of species, i.e. Orphibs whose DNA differ from each other above a given threshold. Specie classification was to be implemented using a clustering algorithm. Moreover, it was thought to be introduced as a gameplay mechanic: the player would increase a score by creating and maintaining different species.

Conclusion

We presented an overview and agent architecture of the A-Life game prototype we developed: Orphibs II. As a game, Orphibs provides gameplay mechanics present in other A-Life games. However, while the prototype draws some inspiration from other A-Life games, such as The Sims and Creatures, Orphibs II introduced a mixing of concepts which originated in genetic personalities: personalities that are transmitted from parents to offspring.

We also performed a 14-minute test run where 15 randomly generated Orphibs populated the game world. Results indicated that the population tended to increase non-linearly as a compulsive reproductive behavior emerged in the population, during the experiment. Nonetheless, we presented, as future work, several means to potentially solve this problem and provide some population stabilization.

Acknowledgements. This work was supported by the CISUC ICIS project CENTRO-07-0224-FEDER-002003.

References

[1] Rollings, A. and Adams, E. (2006). Artificial Life, Puzzle Games, and Other Genres. Fundamentals of Game Design. Prentice Hall, New Jersey, pages 477-498.

[2] Grand, S., Cliff, D. and Malhotra, A. (1996). Creatures: Artificial Life Autonomous Software Agents for Home Entertainment. *Autonomous Agents and Multi-Agent Systems*, 1(1):39-57.

[3] Maxis (2000). *The Sims*, Electronic Arts.

[4] Barreto, N. (2013). *A Language for Game Design and Choreography*, MSc. thesis, Department of Informatics Engineering, University of Coimbra.

[5] Epstein, J. M. and Axtell, R. (1996). *Growing Artificial Societies: Social Science from the Bottom Up*. Brookings Institution Press, Washington.

[6] Monteiro, I. M. and Alvares, L. O. (2009). Uma Infraestrutura de Teamwork para Ambientes Dinâmicos. In *VII Encontro Nacional de Inteligência Artificial*, Rio Grande do Sul.

[7] Ocio, S. and Brugos, J. A. L. (2009). Multi-agent Systems and Sandbox Games. In *AISB 2009*, Edinburgh.

[8] Millington, I. and Funge, J. (2009). *Artificial Intelligence for Games, Burlington*. Morgan Kaufmann Publishers.

[9] Wooldridge, M. (2011). *An Introduction to MultiAgent Systems*. Wiley, Glasgow.

[10] Zubek, R. (2010). Needs-Based AI. *Game Programming Gems 8*. Course Technology PTR, pages 302-311.

[11] T. Baptista and E. Costa, "Evolution of a multi-agent system in a cyclical environment," *Theory in Biosciences*, vol. 127, no. 2, pp. 141-148, 2008.

[12] Aylett, R. and Cavazza, M. (2001). Intelligent Virtual Environments - A State-of-the-art. In *Eurographics 2001, STAR Reports*, pages 87-109.

[13] Gardner, M. (1970). Mathematical Games: The fantastic combinations of John Conway's new solitaire game

"life". *Scientific American*, (223):120-123.

[14] Bach, J. (2009). *Principles of Synthetic Intelligence. PSI: An Architecture of Motivated Cognition*. Oxford University Press, Oxford.

[15] Lim, M. Y., Dias, J., Aylett, R. and Paiva, A. (2009). Intelligent NPCs for Educational Role Play. In *Agents for Games and Simulations: Trends in Techniques, Concepts and Design*, pages 107-118. Springer, Berlin.

[16] Epic Games (2004). *Unreal Tournament*. Atari.

[17] Köster, M., Novák, P., Mainzer, D. and Fuhrmann, B. (2009). Two Case Studies for Jazzyk BSM. In *Agents for Games and Simulations: Trends in Techniques, Concepts and Design*, pages 33-47. Springer, Berlin.

[18] Hindriks, K. V., van Riemsdijk, B., Behrens, T., Korstanje, R., Kraayenbrink, N., Pasman, W. and de Rijk, L. (2011). Unreal GoalBots: Conceptual Design of a Reusable Interface. In *Agents for Games and Simulations II: Trends in Techniques, Concepts and Design*, pages 1-18. Springer, Berlin.

[19] Sims, K. (1994). Evolving 3D Morphology and Behavior by Competition *Artificial Life*, 1(4):353-372.

[20] Graham, L. (2007). *3D Virtual Creatures Evolution*.

[21] Dubrofsky, E. (2008). Evolution Mutates Beyond Biology. [Online]. Available: http://www.cs.ubc.ca/~dubroe/journalism/evolution.pdf. [Accessed 21 October 2013].

[22] Millennium Interactive (1996). *Creatures*, Mindscape.

[23] Grand, S. (2013). *Grandroids*.

[24] Activision (1985). *Little Computer People*, Activision.

[25] PF Magic (1995). *Petz*, PF Magic.

[26] Frank, A., Stern, A. and Resner, B. (1997).Socially Intelligent Virtual Petz. In *Socially Intelligent Agents AAAI Fall Symposium*, Cambridge.

[27] Stern, A. (2003). Virtual Babyz: Believable agents with Narrative Intelligence. In *Narrative Intelligence*, pages 215-227. John Benjamins Publishing Company, Amsterdam.

[28] Sonic Team (1998). *Sonic Adventure*, Sega.

[29] Sonic Team (2001). *Sonic Adventure 2*, Sega.

[30] Schumacher, J. (2013). *Species: Artificial Life, Real Evolution*.

[31] Unity Technologies (2013). *Unity3D*, Unity Technologies.

[32] Eiben, A. E. and Smith, J. E. (2007). Introduction to Evolutionary Computing, Springer, Berlin.

[33] Macedo, L. (2006). *The Exploration of Unknown Environments by Affective Agents*. Ph.D. thesis, Department of Informatics Engineering, University of Coimbra.

[34] Reisenzein, R. (2009). Emotional experience in the computational belief-desire theory of emotion. *Emotion Review*, 1(3):214-222.