2014

Universidade do Minho

Rui Gonçalves da Silva
PG25228

[ARTIFICIAL LIFE]

Study and implementation of an Artificial Life system.

Conteúdo

Introduction

Artificial life (henceforth referred to as “Alife”) is a field that studies systems related to life, its mutations, evolutions and interactions between agents for example, when the system is created through artificial means, be it from an computer generated system of digital organisms or laboratory created DNA. It is a very broad subject, with many and varied applications in many fields, from biology, robotics, industry and even philosophy. Despite initial controversy and criticism, nowadays Alife is becoming more accepted by the scientific community, with articles in magazines like “Nature” and “Science”, and several prizes in practical applications.

A lot of work has been made in this field, leading to the creation of several tools and engines for the creation of Alife systems. We will be making a brief analysis of the subject, its diverse variants and what has been made in the field, with the final goal of building our own Alife program. This article shall then be divided as such: we will begin with a small overview of Alife and its history, following with a brief study of its variants, and analysis of a few practical applications. Afterwards, we will begin the implementation of the algorithm proper.

Artificial Life

Artificial Life is a subject that has since ancient times fascinated humans, and history is full of stories of mundane objects and human constructs that gained life: homunculi, tsukumogami, the tale of Pygmalion, Pinocchio, the monster of Frankenstein and many, many others are good examples of this. In modern days, this fascination remains strong in the form of robots and other automatons. Science fiction is full of intelligent man-made machines, capable of thinking for themselves and interacting with humans.

Before the dawn of computers, Alife manifested itself in the form of mechanic automatons, made with complex pneumatics and hydraulics like al-Jazari's humanoid robots, and Jacques de Vaucanson's artificial duck. The invention of computers and the potential they provided turned Alife into a real science. Preliminary work on the subject can be dated as far as the late 1940’s, but only in the late 80’s did Cristhopher Langton coined the term “Artificial Life”.

Artificial Life and Artificial Intelligence are related subjects; however, in order for something to be alive, it does not necessarily need to be intelligent. The definition of life itself can be quite complex, and there are several philosophical debates over whether a simulation can or cannot be considered alive outside a chemical solution. When creating an intelligent system, we are preoccupied with creating a system that can solve problems, and therefore, when implementing one we tend to start thing from a “Top-down” approach, starting from what we want to happen. Some types of “soft Alife” can be considered Artificial Intelligence. When creating Alife systems, we usually take a “bottom-up” approach, defining a set of rules for the system and observing how it evolves. In fact, that is the very essence of Alife, a system that can grow, reproduce, evolve and adapt, simulating what naturally occurs in nature, with the advantage of allowing the study of a big population over long periods of time, without having to actually wait that long. Systems like this are often used to study topics like adaptive and social behavior, evolutionary biology, model development, system complexities, learning capabilities, behavior and construction of robotics, and philosophic studies.

Types of Artificial Life

There are three main types of Alife, **soft** or software-based, like digital organism simulators, **hard** or hardware-based, like robots, and **wet** or wetware-based, like the creation of synthetic DNA.

Soft Alife

Soft Alife typically exists in the form of ecology simulators, populated by digital organisms. Evolutionary algorithms are frequently used to simulate natural selection, evolving digital organisms to improve their efficiency. Each individual has a genome that not only influences its attributes but also the decisions that the system takes, like who will reproduce or who will die. The way that decisions are made can vary with different algorithms, like neural networks or fuzzy cognitive maps. Larry Yaeger's "PolyWorld" is a famous example of this approach. In it, trapezoid agents simulate life like characteristics, like hunting and mating. Another technique commonly used is based on cellular automata, from which John Conway's "Game of Life" is one of the most famous examples. Several engines like "breve" where produced to allow easy creation of soft alife systems for study.

Other approaches for soft Alife are the following:

- Program-based, where each organism is a program that is alive when being executed. (e.g. multi-agent systems)
- Module-based, where the behavior of each organism can be modified by the addition of a module, allowing for easy user personalization.

Hard Alife

Hard Alife refers to hardware implementation of alife systems, from which robots are the most common use. Similarly to artificial intelligence, we try to make the robots adapt to the external environment, however, we take a biologic approach letting the physical environment be responsible for generating the robot's behavior. Alife can also be used to design physical components like sensors, control systems or even actual robot design.

Wet Alife

Wet Alife is a strong field of synthetic biology, and its final goal is the creation of artificial cells out of biochemicals. An example of wet Alife is the creation of artificial genes in a laboratory without the need for preexisting DNA sequences. A practical application of this would be the creation of genes for vaccine research or genetic modification.

Practical implementations

Alife systems have been applied in practical uses with great success. The following are but a few examples of accomplishments of Alife in real-world problems.

PLANTWORLD

Jacqui Dyer and Peter Bentley of UCL (University College London), unsatisfied with traditional numeric models used by ecologists to capture the behavior of evolution of environments prone to disasters like fires or earthquakes, created an Alife system that simulates the evolution of a dynamic population of plants. Each plant grows through various stages of life and needs moisture to live, which varies by time and location. Maps for water locations and rain can be added to the simulation, also each plant can have a gene that dictates when the plant should be active (thus growing and consuming resources) or dormant. The objective of the program is to observe the evolution of a great population of plants over a grand number of years, and in diverse environments. As an example, experiments with the program have proved that plants tend to evolve into species that grow dormant in dry seasons.

Model Design

Alife can also be used to generate optimal structures based on specific rules. As an example, Pablo Funes used an Alife system to generate structures made of LEGO pieces. One such example of structure was a robotic LEGO crane capable of lifting 0.5kg of weight. Jordan Pollack and Hod Lipson used a 3D printer and an Alife program to generate bodies for robots.

An even better example is Steven Manos' work in fibre optic cables. Specific hole patterns in cables can help certain light frequencies travel further. Since the number of patterns can be infinite, and there are some manufacturing constrains, to find the best combination he used an Alife program to find the optimal solution. His work was very successful and won several awards, leading to the creation of his own company to take advantage of this technology.

Mosaic World

Similar to PLANTWORLD, this Alife system was used to investigate the evolution of color vision. A world with several types of color "food" is generated, and artificial agents evolve to eat the color they prefer. As one color becomes extinct, the agents must evolve to eat different colors. A variation of this project allows the "food" to grow like plants and also evolve different colors to avoid extinction.

Evolutionary Robotics

Evolutionary robotics is an area of robotics that used Alife algorithms to engineer a robot's characteristics, from its design, using model design as seen above, to its behavior, like the way it reacts to certain situations or stimuli. An example of this is the Embodied Cognition in a Compliantly Engineered Robot or ECCE Robot, a project that designed a robot that uses flexible elements to simulate muscles and tendons that it uses to move. Another example is BigDog, from Boston Dynamics, a robot capable of moving in rough terrain, with his behavior defined through Alife algorithms.

Problems and Solutions

Despite the advantages that Alife presents, it also has its drawbacks. It's almost impossible to represent a real life situation with full accuracy due to its complexity. Real life systems depend of numerous factors, and due to the evolution factor, even small differences between the real life model and our simulation can create very big changes in the final results. In fact, even if we represented perfectly the system, it is possible for it to evolve in a completely different direction. Because of this, we usually use simplified models and several repetitions of the study to draw our conclusions. For example, if instead of trying to study the evolution of a small group of plants in Australia that are going extinct, we try to understand the way plants can interact with their environment in similar conditions, we can obtain a deeper understanding of their ecosystem and the reasons for their extinction. As seen in the above examples, controlled study and use of Alife systems can generate excellent results and, as our knowledge and technology grows, our Alife systems get more and more complex and reliable.

Implementation

The final project draws inspiration from Mosaic World and PlantWorld, but attempts to create a generic framework that is simple to use and adapt to several different problems. The programming language chosen to create our application was Python, a simple to use language but that can run slowly when compared to other languages like C. As such, one of the goals of the project was to make a lightweight application. The library **Pygame** is used to create a graphical representation of the world.

The system has four modules: **Creature**, the class that represents each individual in our world, **Tile**, the basic components of the world, **World**, which defines the world proper, along with several of its rules and **Alipy**, the main module of the program.

Tile

The world is composed of tiles. Each tile represents a small space of our world and can be inhabited by a single creature at a time. Each tile can have different property values to type and moisture, representing different types of land, like desert, forest and lake.

Creature

Creatures are the individuals that inhabit our world. Each is represented by a unique id, and has three main properties: creature type, representing the race of the creature, predator, indicating the creature type of the creature's natural predator, and prey, which indicates the creature type that the creature feeds off. The code for prey can also be 0, indicating that this creature feeds off "moisture". This moisture can differ by tile and can represent actual moisture or other things like the density of plants or insect populations. It mainly exists to feed the creatures at the bottom of the food chain. Each creature also has a genome, which defines several characteristics, energy, and two values for x and y coordinates, indicating its position in the world. The genome is a list of genes, values between 0 and 255 that define attributes in the following order: red color value, green color value, blue color value, speed (how many tiles a creature can move), eye type (aura or tunnel), eye range (how far it can see), mutation rate, and reproduction type (sexual or asexual).

Two types of vision are implemented, aura which is able to see all tiles surrounding the creature up to a range defined by eye range, with a smaller limit for range, and tunnel, which is capable of seeing farther but only in a straight line. Two types of reproduction are also implemented, asexual in which a creature replicates itself, mutating its gene with a chance defined by its mutation rate, and sexual, were a creature must seek a partner to mate. In this case, the offspring will be a product of a crossover between the parents' genomes, with a chance for being mutated.

The color value defines the creature's color. Each creature has a type of color it prefers eating. The higher the difference between a predator's and a prey's color, the lower the chance a predator can eat its prey.

Each creature has several needs. First is the need to reproduce, being the first thing they do when they have a good amount of energy and the need to hunt, which they do when they start to get hungry. They can also run if the spot a predator nearby. If the see nothing of interest they will move randomly to an empty tile.

World

One of the main module s of the program, the world module defines the world and its rules. It has a configurable number of rows and columns, composed of tiles with configurable width and height. Two types of world are implemented, truncated worlds stop at the edges while circular worlds loop around to the other side. Several attributes define the prices and rewards for actions creatures can take, like moving, eating and breeding. Worlds also have lists of every tile they're composed of and of every creature that lives in them. Populations and world parameters can be read of csv format files.

Alipy

The main module of the program, it's the one that starts the system. It uses the library Pygame to draw the world in a window. It can print information of all creatures in the console by pressing the letter "p" on the keyboard, or can create a file in the csv format containing the same information by pressing "w". Several statistics can be printed to the console by pressing "s", like average speed of the population, average eye range and most common vision type.

Results

The result of our work is an application that is lightweight, easy to use, configure and edit, and that can be used to represent different ecosystems. Each run of the program can end in very different results, and it's interesting to watch how the population evolves. We can see how the colors of the creatures change as they are forced to evolve to escape being hunted or running out of food. Due to the complexity of a system like this, it can be hard to create a stable ecosystem, as even the smallest of changes can have drastic consequences, usually resulting in a total extinction of the entire population.

Conclusion

This was a very interesting project to work in, with innumerous ways to implement, and that taught me a lot about the theme of Alife. It was somewhat unfortunate that I didn't end up having a lot of time to dedicate to this project, as I would like to create something a bit more complex. In the future, I hope to have the chance to revisit this project, implementing several ideas that came up as I developed the application, like neural networks for the creature’s decision making, and possibly looking up threading to speed up the program.

Bibliography

1. Wikipédia

2. *Artificial Life, Evolutionary Robotics, Real World Applications*. UCL.

3. *Artificial Life*. Mark A. Bedau.

4. *Artificial Life*. Atool Varma and Nathan Erhardt, 1996.

5. *NMCS4ALL: Artificial Life*. Dave Ackley, https://www.youtube.com/watch?v=YJRRu4dJnTI