The Tool for Modeling of Evolution of the Artificial Life

Iva Bartunkova

August 1, 2007

Abstract

Target of thesis is to create a software simulator allowing experiments with the evolution of simple organisms, run several experiments and describe results.

Chapter 1

Introduction to Artificial life

1.1 What is Artificial life?

Artificial Life, (commonly **Alife** or **alife**) is a field of study and art form that examines systems related to life, its processes and its evolution through simulations using computer models, robotics, and biochemistry (called "soft", "hard", and "wet" approaches respectively [1]).

In this thesis the term "Artificial Life" is often used to specifically refer to soft alife.

The term Artificial life was first coined by Christopher Langton in the late 1980s at the first "International Conference on the Synthesis and Simulation of Living Systems" (otherwise known as Artificial Life I) at the Los Alamos National Laboratory in 1987. He envisioned a study of life as it could be in any possible setting.

Artificial life studies "natural" life by attempting to recreate biological phenomena from scratch within non-living media like computers or RNA structures of molecules. Alife complements the traditional analytic approach of traditional biology with a synthetic approach in which, rather than studying biological phenomena by taking apart living organisms to see how they work, one attempts to put together systems that behave like living organisms.

The process of synthesis has been an extremely important tool in many disciplines. Synthetic chemistry - the ability to put together new chemical compounds not found in nature - has not only contributed enormously to our theoretical understanding of chemical phenomena, but has also allowed us to fabricate new materials and chemicals that are of great practical use for industry and technology.

Artificial life amounts to the practice of "synthetic biology" and, by analogy with synthetic chemistry, the attempt to recreate biological phenomena in alternative media will result in not only better theoretical understanding of the phenomena under study, but also in practical applications of biological principles in the technology of computer hardware and software.[2]

The seminal novelty of ALife lies in its synthetic approach. Whereas traditional research is essentially analytic, breaking down complex systems into basic components, Alife attempts to construct complex systems from elemental units.[4]

For example, answering of questions like how the simple rules of Darwinian evolution lead to high-level structure, or the way in which the simple interactions between ants and their environment lead to complex trail-following behavior promises to provide novel solu-

tions to complex real-world problems, such as disease prevention, stock-market prediction, and data-mining on the Internet.[3]

1.2 History of Artificial life

Important events at development of mathematics and computer science that later led to ideas of artificial life were presented in late 1940s at the Hixon Symposium in California. Math and computer prodigy John Von Neumann delivered a lecture titled "The General and Logical Theory of Automata" where he defined the term "automaton" and said that natural organisms would in the end be found to follow similar simple rules. He postulated a machine, "kinematic automaton", thatcould create an identical machine. Later he crated one with Stanislaw Ulam, a purely logic-based automata, not requiring a physical body but based on the changing states of the cells in an infinite grid - the first cellular automaton (CA). It was extraordinarily complicated compared to later CAs, having hundreds of thousands of cells which could each exist in one of twenty-nine states.

Celular automata were first field of artificial life in computer science and they are important up to present. A cellular automaton is a discrete model that consists of a regular grid of cells, each in one of a finite number of states. The grid can be in any finite number of dimensions. Time is also discrete, and the state of a cell at time t is a function of the states of a finite number of cells (called its neighborhood) at time $t - 1$. These neighbors are a selection of cells relative to the specified cell, and do not change. Every cell has the same rule for updating, based on the values in this neighbourhood. Each time the rules are applied to the whole grid a new generation is created.[5]

The following important step towards Artificial life was realized by Edgar F. Codd, who simplified Von Neumann's original twenty-nine state monster to one with only eight states.

The history of Artificial life dates back to 1987 when the "International Conference on the Synthesis and Simulation of Living Systems" (otherwise known as Artificial Life I) was held at the Los Alamos National Laboratory. Researcher Christopher Langton defined here his idea of new science that had barely existed up to that time.

In 1977 founded Ed Fredkin the Information Mechanics Group at MIT, `http://www.ai.mit.edu/projects/im/`. This group created a computer especially designed to execute cellular automata, eventually reducing it to the size of a single circuit board. This "cellular automata machine" allowed an explosion of alife research among scientists who could not otherwise afford sophisticated computers.

In 1982, computer scientist Stephen Wolfram turned his attention to cellular automata. He explored and categorized the types of complexity displayed by one-dimensional CAs, and showed how they applied to natural phenomena such as the patterns of seashells and the nature of plant growth.

Computer animator Craig Reynolds similarly used three simple rules to create recognizable flocking behavior in groups of computer-drawn "boids" in 1987. With no top-down programming at all, the boids produced life-like solutions to evading obstacles placed in their path. Computer animation has continued to be a key commercial driver of alife research as the creators of movies attempt to find more realistic and inexpensive ways to animate natural forms such as plant life, animal movement, hair growth, and complicated organic textures.

The Unit of Theoretical Behavioral Ecology at the Free University of Brussels applied the self-organization theories to model behavior of swarms and colonies of organisms.

A conference in May of 1985 called "Evolution, Games, and Learning" focused Alife to tie to the emerging field of complex adaptive systems. Key figure was J. Doyne Farmer working at the Center for Nonlinear Studies.

In 2000s the field is underway to create cellular models of artificial life. Initial work on building a complete biochemical model of cellular behavior is underway as part of a number of different research projects, namely BlueGene which seeks to understand the mechanisms behind protein folding.

The current proggres in Artificial life is regulary presented at Alife -International Conference on the Simulation and Synthesis of Living Systems and ECAL - European Conference on Artificial Life organized by International Society of Artificial Life (ISAL), `http://www.alife.org`.

1.3 Abeetles - simulator of artificial life

This thesis is bounded within software approach to Artificial life. Its main target is to design and create software simulator Abeetles. Abeetles runs life of simple organisms in its specific environment and is concerned with evolution of their features and behavior. The system avails experiments with evolution of the organisms under various conditions and offer overviews and statistics of the proccess. Experiments performed with the system will be described in chapter Experiments.

1.4 Related terms

Abeetles concerns with evolution and genetics. They originate from biology and therefore their definition is also biological. Following terms will be used in this thesis: gene, genome, genotype and phenotype.

Gene is a structural unit of inheritance in living organisms. A gene is, in essence, a segment of DNA that has a particular purpose, i.e., that codes for (contains the chemical information necessary for the creation of) a specific enzyme or other protein.[6]

In biology the genome of an organism is its whole hereditary information and is encoded in the DNA (or, for some viruses, RNA). This includes both the genes and the non-coding sequences of the DNA. The term was coined as a portmanteau of the words gene and chromosome.[8]

The genotype of an organism is the class to which that organism belongs as determined by the description of the actual physical material made up of DNA that was passed to the organism by its parents at the organism's conception. For sexually reproducing organisms that physical material consists of the DNA contributed to the fertilized egg by the sperm and egg of its two parents. For asexually reproducing organisms, for example bacteria, the inherited material is a direct copy of the DNA of its parent.

The phenotype of an organism is the class to which that organism belongs as determined by the description of the physical and behavioral characteristics of the organism, for example its size and shape, its metabolic activities and its pattern of movement. [7]

Chapter 2

Abeetles in context of Alife simulators

The family of software simulators of artificial life is numerous. Therefore to find a position for a new simulator needs some classification. This thesis will expand on three classification possibilities of Alife simulators. First, cathegorisation according to the method of creature definition will be used. Second, simulators will be distinguished with respect to the feature or features of Artificial life, that they simulate. Third, the criterion of purpose of the simulator and target group of users will be used. The fourth section places Abeetles into these classification systems and thereby specifies sphere of Artificial life that will be dealt in this thesis.

2.1 Classification by creature definition

In existing systems individual agents are modeled and constructed in many different ways that rank them roughly into following cathegories:

- **Program Based** - In program based simulators an individual is represented by a program which substitutes biological DNA and make up the genome of the agents. Language of the program is usually Turing complete. Assembly derivatives are the most common languages used. Tom Ray's Tierra is a famous example of a program based simulator.

- **Module Based** - An agent in a module based system is a composition of individual modules. These modules modify the creature's behavior and characteristics either directly, by hard coding into the simulation (leg type A increases speed and metabolism), or indirectly, through the emergent interactions between a creature's modules (leg type A moves up and down with a frequency of X, which interacts with other legs to create motion). Generally these are simulators which emphasize user creation and accessibility over mutation and evolution.

- **Parameter Based** - If an organism is constructed to have defined and fixed behavior that is controlled by various parameters that mutate, the system is referred as a parameter based. It means that each organism contains a collection of numbers

or other finite parameters. Each parameter controls one or several aspects of the organism in a well defined way.

- **Neural Network Based** - These systems simulates processes of creatures that learn and grow using neural networks or a close derivative. Emphasis is often, although not always, more on learning than on natural selection.

2.2 Classification by simulated phenomenon

Another grouping of simulators can be done according to phenomenon, that they simulate. It is presented at the web page of Monash Universitys Complexity Virtual Lab. `http://vlab.infotech.monash.edu.au/`

- **Networks** - Relationships in complex natural systems are simulated according to network theory (or diktyology) as networks or graphs and their statistical and topological properties are analyzed.

- **Nonlineality** - Nonlineal systems are systems that cannot be mathematically described as a sum of their components. While certain assumptions can be made for lineal systems, that often make the mathematical modelling of such systems easy, mathematical modelling of nonlineal systems is often very difficult or impossible. As a result, nonlineal systems are often studied through use of simulations.

- **Swarms** - A swarm is a group of independent agents that gather together in order to collectively carry out a certain task. Typically, each agent exhibits a very simple behavior pattern that is influenced by direct or indirect interactions with other swarm members. As a result, the swarm as a whole may exhibit complex and intelligent behavior patterns. In nature, swarms can be observed in social insects, fish schools, but also in primitive single cell organisms.

- **Evolution** - Evolution is the process of development or grows by accumulation of small advantageous changes. The study of all forms of evolutionary processes is one of the primary goals of ALife. This includes the study of biological evolution of species as well as other evolutionary processes in natural, artificial and social systems.

- **Cellular Automata** - The term cellular automaton (CA)is described in the previous chapter. It is not directly a phenomenon of natural life, but it is a discrete model of mutual influence of neighboring elements, which can be in the nature observed.

2.3 Classification by purpose

Apparently, the main purpose of Artificial life simulators is to simulate artificial life. The target can be the complexity of simulation as well as the concentration on a small selection of its attributes. As the simulator is a software system, it is designed for a certain group of users and expected usage. And a decision in this field influences interface, adjustability and output of the program. Also availability of the system is related to the purpose.

- **Games** - Frequent purpose of simulators is entertainment, because game industry is constantly researching for new ideas to animate artificial characters. Simulators from this class are usually supplied with attractive user interface, but neither artificial life techniques and settings nor source corde are accessible to examination. The game Creatures is a well-known example of an artificial life computer program series, created in the mid-1990s by English computer scientist Steve Grand. The program is regarded as an important breakthrough in the advancement of artificial life research.

- **Scientific simulators** - A simulator can be designed primarily for use as a platform in Artificial Life research. Such programs allow to perform experiments in certain subfields of Alife, e.g. evolutionary dynamics, theoretical biology etc. They are usually highly adjustable and configurable and afford opportunities to gather high-quality statistics. The grafical output is not expected to be the most important feature in comparison with games. They are often open source. Tom Ray's system Tierra and Avida from Devolab are such simulations.

- **Simulators for teaching purposes** - Efforts of creators of a simulator can be also concentrated on the idea of demonstration of natural life processes using a computer with the objective that through experimentation and interactive play users learn underlying patterns of life. Intriguing example is the simulator Mitozoos where through mutation of spider-like agents can anybody explore relationship between genetic code and respective life forms.

- **Related systems** - Many systems are closely related to Artificial life simulators, but they are not considered to be Artificial life systems, because they only use techniques, that originate from Alife like genetic algorithms or ant colony optimization. Their purpose can be various. The primary difference lies in the fact that these programs explicitly define the fitness of an agent by its ability to solve another problem than find food, reproduce and avoid death as it is typical for real life and its simulations.

2.4 Classification of Abeetles

As far as definition of agents is concerned, target class of Abeetles are parameter based systems. Genome of agents is defined as a set of parameters that directly influence their life and behavior.

In classification by simulated phenomenon Abeetles can be ranked among systems concerned with evolution. Abeetles evolves population of agents under various conditions and enables to monitor the development.

The purpose of Abeetles is to be a scientific simulator. But not only for users from comunity of computer scientists, but also for those who besides fast results also appreciate user friendly interface with visualisation, even if it decreases performance of the system.

Chapter 3

Targets of Abeetles and their origin

The main target of Abeetles, as well as of other similar programs, is to simulate artificial life. But artificial life is rather a general idea, because to simulate all natural life as we know it on the Earth is with contemporary means impossible. Therefore every simulator chooses just a restricted subset. As stated above, Abeetles is concerned with evolution. The next step is thus to specify what will be evolved and how will look like the world, where it will be evolved. In the following text the subject of evolution will be called agents and the world will be called environment.

Abeetles is one of the simulators where both environment and agents are very simplyfied abstractions that do not have any concrete model in the nature. Abeetles does not endeavor to simulate any existing organisms, but to explore features of natural processes like evolution or natural selection on organisms that are in essence virtual. The question, that arises from this kind of simulation of evolutional processes is, what should parameters of agents and environment, whose models have little in common with real life, look like, so as to cause that the results and course of evolution would resemble patterns of natural organisms in their environment.

Browsing in the Internet, one easily ascertains that there are already many simulators of artificial life. Therefore it is necessary to define targets of Abeetles in the context of other artificial life simulators. Attributes of several of these "brother programs" will be described in details and afterwards features of the environment and agents of Abeetles will be stated after careful comparison. For this purpose will serve five existing simulators: Avida, Bitozoa, Gene Pool, Mitozoos and Primordial life. Reasons for choice of each of them will be described. Common attribute of all of them is that their binaries are accesible freely or on demand. Abeetles is also free software and is intended to examine contributing set of features of natural life only within free software.

3.1 Brother simulators

The simulators are classified by twenty-seven features. Features are sorted to six cathegories: Agents, Mating, Environment, User interface settings and Implementation. Symbol (?) in the classification means that features of the simulator from this aspect are unknown.

Cathegory Agents contains following features: Body(= description of body of agents), Moving(= how do agents move), Behavior(= what they do besides basic activities of life

- moving, mating and eating), Features(= features that change during life), Life-span, Genes(= representation of genes), Phenotypic space(= space of all possible phenotypes of agents), Learning(= whether some ability to learn is included in the model of agents)

Cathegory Mating comprises: Number of offspring in one reproduction, (Genetic algorithm(= algorithm used for creation of genes of a descendant), Choice of partner(= features that a partner must have to be chosen for mating), Conditions of mating(= what conditions must be satisfied before mating), Investment in offspring(= the amount of some resource, usually energy, that is given to a descendant by its parents at its birth)

Cathegory Environment includes: Description(= what does the environment look like), Features(= parameters of the environment, that are not represented by objects contained in the environment),Elements(= objects, that inhabit or are placed in the environment together with agents)

Cathegory User Interface Settings encompasses attributes by which a user influences the run of life in the simulator. They are sorted to three cathegories: Mating, Life of individual and Environment.

Cathegory Statistics is standalone.

Cathegory Implementation holds: Implementation(= features of implementation of the simulator), Distribution of computation(= whether and how it is possible to distribute the computation of the artificial life to more computers), Speed settings(= whether and how it is possible to influence the speed of computation), Available sources(= whether source file of the simulator are available), Availability of binaries(= under what conditions are binaries available, e.g. on registration, free, ect.), Programming language(= programming language used and key libraries)

3.1.1 Avida

Avida is an often presented example of a program based simulator. It studies evolution of programs. For definition of targets of Abeetles Avida's implementation is important - two programs, console and GUI version - serve to scientific purposes, the first one can run very fast and the other one can offer graphical output. Examined release was Avida version 2.0b7 2003 for OS Windows. Avida is a joint project of the Digital Life Laboratory at the California Institute of Technology, `http://dllab.caltech.edu/`, (headed by Chris Adami) and the Digital Evolution Group, `http://devolab.cse.msu.edu/`, at Michigan State University. Web page of the project is `http://dllab.caltech.edu/avida/`.

Agents	segments of code in a simple language called strings
Body	one cell in the grid of the environment
Moving	no
Behavior	execution of the code
Features	content of stack and registers
Life-span	not restricted
Genes	string is itself a genome
Phenotypic space	any finite string constructed of instruction of the language.
Learning	no

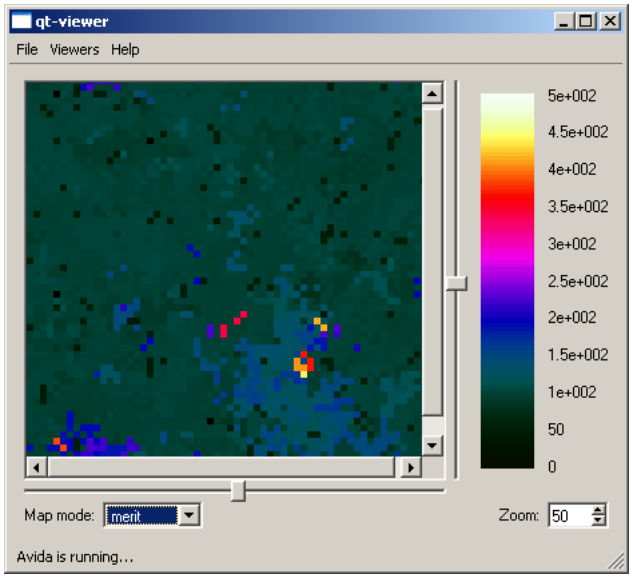

Figure 3.1: Avida 2.0b7 - graphical viewer.

Mating	
Number of parents	1
Number of offsprings	1
Genetic algorith	Poisson-random mutation
Choice of partner	no
Conditions of mating	no
Investment in offspring	no

Environment	
Description	a grid of cells in the form of torus
Features	resources and tasks, completion of a task gives a bonus to the string
Inhabitants	strings

UI settings	
Mating	mutation rate
Life of individual	initial genomes of initial individuals
Environment	resources and tasks of the environment

Statistics	instruction viewer, string details, etc.

Implementation	
Distribution of computation	no
Speed settings	console version versus gui version
Available sources	yes
Availability of binaries	free
Programming language	C++

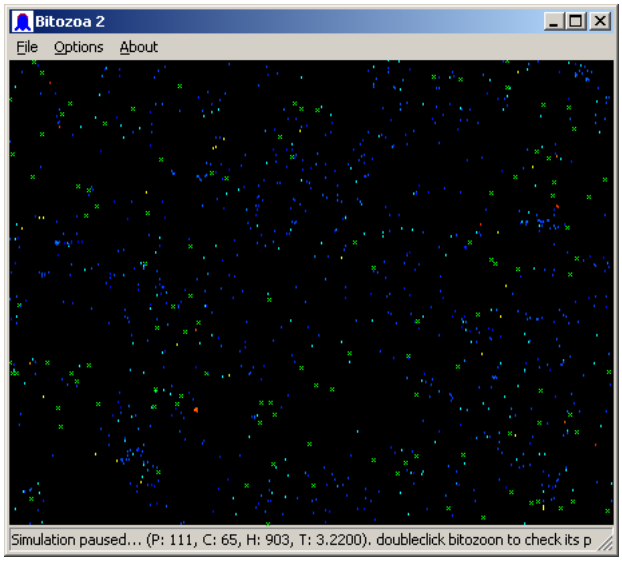

Figure 3.2: Bitozoa 2

3.1.2 Bitozoa

Bitozoa is a simulator created by M. Borkowski, that uses for representation of agents neural network in combination with a set of parameters. It evolves two types of agents, herbivores and carnivores. It was created for purpose of amusement and education. Used version is Bitozoa 2 - artificial life and artficial intelligence simulation, 2000. Web page of the project is `http://www.bpp.com.pl/bitozoa2/bitozoa2.html`.

Agents	Bitozoa - herbivores(eat plants) and carnivores(eat herbivora)
Body	ball with 5 eyes, 2 flagella, various number of neurons, neural network with various topology. Eyes see all object in the world.
Moving	moves using flagella
Behavior	no
Features	energy level - for moving and work of neurons, burned with speed according to age
Life-span	(?)
Genes	(?)
Phenotypic space	angle of flagella connection to body, number of neurons, topology of neural network, sensitivity of eyes
Learning	no

Mating	
Number of parents	2 or 1
Number of offsprings	1-n (2parents) nebo 1(1parent)
Genetic algorithm	random number of genes from both parents + mutation
Choice of partner	same species and high enough stamina
Conditions of mating	meeting of two bitozoa
Investment in offspring	both parents give the same fixed amount

Environment	
Description	space in form of toroid
Features	number of spots, where food grows
Elements	food growing on special spots, herbivores and carnivores

UI settings	
Mating	breed distance, breed level of stamina, inherited stamina
Life of individual	asexual reproduction allowed, costs of an action, flagella angle and efficiency, view of bitozoa, adding or killing of a bitozoa, stamina from food, strength of carnivore, initial values of: eye sensitivity, number of neurons and stamina
Environment	number of bitozoa and food spots at the beginning, time increment, viscosity, random numbers generator seed

Statistics	graph of population, energy flow and energy flow averaged, simulation description

Implementation	
Distribution of computation	no
Speed settings	animation on-off
Available sources	partial
Availability of binaries	permission from author
Programming language	C++ Win32API

3.1.3 Gene Pool

Gene Pool was created by Jeffrey Ventrella, current version is 5. It is a parameter based simulator, dealing with evolution of swimming creatures in a virtual Darwinian aquarium. It is a game with intriguing graphical interface, san serve as an entertaining learning tool, but author demonstrated with it also several interesting results concerning influence of atractiveness of partners on results of evolution.[11] [12]

Figure 3.3: Gene Pool 5

Agents	swimbots
Body	mouth, genitals and 2-10 parts for moving
Moving	swimming using parts, algorithm common for all swimbots
Behavior	nothing
Features	energy
Life-span	not restricted
Genes	(?)
Phenotypic space	length, width, phases, amplitudes and attachment of every part
Learning	no

Mating	
Number of parents	2
Number of offsprings	1
Genetic algorithm	crossover algorithm with mutation
Choice of partner	”in round ””view horizon”” at one snapshot. Swimbot chooses at one snapshot in his ””view horizon”” one mate, that most satisfies his attractiveness criterion”
Conditions of mating	swimbots mate when at least one of them is pursuing the other and the distance between their genitals is less than the length of the genital vector
Investment in offspring	50% of actual energy

Environment	
Description	rectangular pool
Features	constant amount of energy in environment, cyling: food-¿swimbots-¿pool-¿food
Elements	food bits, swimbots

UI settings	
Mating	attraction criterion
Life of individual	location in environment, creation a swimbot, hunger threshold, energy for offspring
Environment	food growth - birth delay, spread radius, energy from 1 piece

Statistics	graph of number of swimbots vs. food, features of individual, the best one in attraction/mating/eating

Implementation	
Distribution of computation	no
Speed settings	no
Available sources	no
Availability of binaries	yes
Programming language	(?)
do uvodu:	how good is a swimbot in reproducing, which means either being good at swimming or being attractive to other swimbots

3.1.4 Mitozoos

The author of project Mitozoos is Spanish software company Bestiario. Purpose of the simulator is educational, as author states "Mitozoos is an interactive artificial life model created with the objective that through experimentation and play participants will understand the relationship between genetic code and life." The web presentation of the project can be found at `http://bestiario.org/mitozoos/english/index.html`.

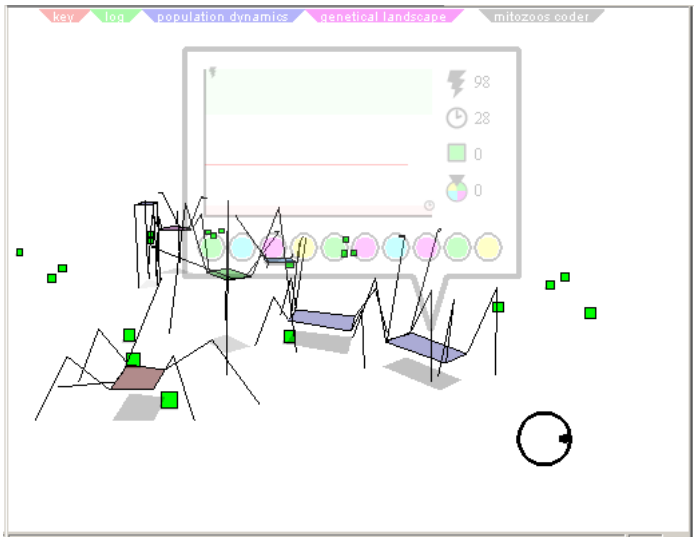

Figure 3.4: Mitozoos

Agents	mitozoos
Body	spider-like, with 2 eyes, body in shape of tetrahydron and four bent legs
Moving	walking using legs, algorithm common for all mitozoos
Behavior	strategy of eating, frequency and duration of resting periods
Features	energy
Life-span	not restricted
Genes	ten genes, each gene has four bases represented by colors
Phenotypic space	length of parts of legs, procreation threshold, eating strategy
Learning	no

Mating	
Number of parents	2
Number of offsprings	1
Genetic algorithm	crossover algorithm with mutation
Choice of partner	no
Conditions of mating	meeting of two mitozoos with energy higher than procreation threshold
Investment in offspring	a fixed amount of energy

Environment	
Description	space in form of a circle
Features	rate of food growth
Elements	mitozoos, food bits

UI settings	
Mating	no
Life of individual	special application, which avails user to create a mitozoos by setting its genes and add it to running life.
Environment	initial number of mitozoos and food pieces, food growing rate, mutation rate

Statistics	graph of number of swimbots vs. food, features of individual, genetic landscape, log of events, on reproduction join crossjoin of genotypes is shown

Implementation	
Distribution of computation	no, but coders can run on different computers.
Speed settings	no
Available sources	no
Availability of binaries	yes
Programming language	Action Script, (?)

3.1.5 Primordial Life

For following evaluation Primordial Life 3.0 by Jason Spofford was tried. It can be downloaded at `http://www.io.com/~spofford/prim30.html`. Primordial life is a shareware parameter based artificial life screen saver writen for purpose to capture the principles of evolution in an interestion and visual way.

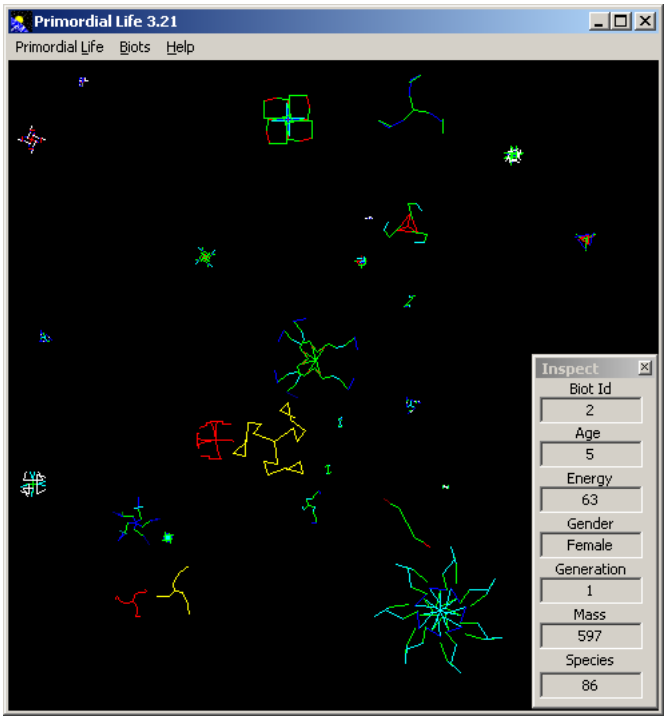

Figure 3.5: Primordial Life 3.21

Agents	biots
Body	pattern of vectors of different colors, color of a part determines its special function.
Moving	moved by environment or by its motor (light blue parts)
Behavior	fighting with each other
Features	energy level
Life-span	restricted
Genes	64 numbers, describe lines color, orientation and length. Certain genes have special meaning detailing how many lines a biot has, its symmetry, whether it has mirrored or radial symmetry, how many children it should have and whether or not it should disperse its children after they are born. Complete description available.
Phenotypic space	number and symmetry of vectors, color, orientation and length of each vector, number and dispersion of offspring in a birth
Learning	learn from collision with other biots

Mating	
Number of parents	2 nebo 1
Number of offsprings	1-n
Genetic algorith	genetic crossover with mutation(2 parents), mutation (1 parent)
Choice of partner	no
Conditions of mating	collision of two biots, one of them with white parts
Investment in offspring	yes

Environment	
Description	rectangular space
Features	mixing of biots continuely up and providing of light, absorbable by green parts of biots and used up as energy
Elements	biots

UI settings	
Mating	mutation rate, sexual/asexual/ both reproduction
Life of individual	life span, speed, regeneration rate and cost, attacking of child, battle of sibblings, level of selfchange thanks to collision
Environment	solar intensity, friction, starting population, plague - its duration and lethality

Statistics	ecosystem status (population, death and birth rate numbers), global status of all connected ecosystems(no. of connected ecosystems, population)

Implementation	
Distribution of computation	connection of ecosystems in a network over the Internet.
Speed settings	no
Available sources	no
Availability of binaries	register

3.2 Features of Abeetles

The original idea of Abeetles is to create a successor of existing program Broucci (=Beetles) introduced by RNDr. Tomas Holan Phd. as an example of different attitude to solution of complex problems. It is not an artificial life simulator, but it could be classified as a related system. It uses genetic programming to solve a specifical problem. The assignment of Broucci is that there is a world where beetles live. Beetles can move - make a step, rotate left and rotate right. The step can head for an empty space or to eat a

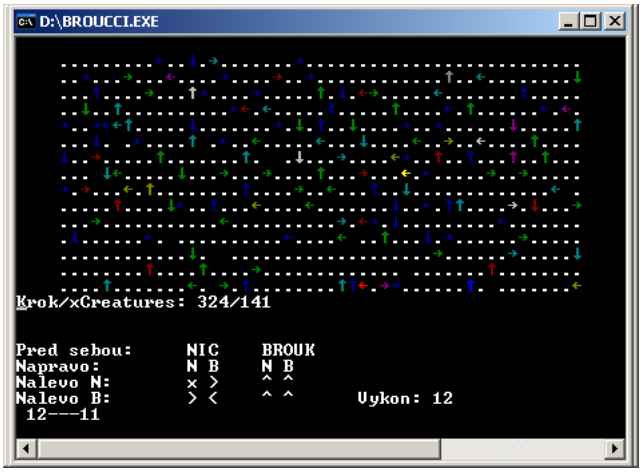

Figure 3.6: Broucci

beetle and occupy its place. They can see what is around them - in the front, on the left and on the right. Task is to find an algorithm that should a beetle use so as not to die of hunger and not be eaten up by another beetle. Solution described by the author is a console program where beetles are visualized as arrows turned in the direction of their sight. Beetles take turns regulary and always when the number of beetles goes down under certain level, new beetles are created from the most successful ones using genetic crossover algorithm. Parents do not meet.[14]

Abeetles takes over following ideas of Broucci: beetles can move by sequence of steps and rotations. According to what they see is in the front, on the left and on the right do they decide which action to take.

"Brother programs" were compared according to 27 different cathegories. Abeetles does not strive to be contributing in all these cathegories. Instead of it, six were chosen as the key ones: moving, learning, choice of partner, distribution of computation, obstacles in environment and aging. The following table compares Abeetles with other solutions only using these cathegories.

	Avida	PLife	Bitozoa	Mit.	GPool	Abeetles
Moving: strategy	x	x	G	S	S	G
Learning	x	O	x	x	x	G/O
Choice of partner	x	x	x	x	O	G
Distribution	x	Y	x	x	x	x
Obstacles	x	x	x	x	x	O
Aging	x	x	x	x	x	O

O - optional (user can influence it), S - static (fixed), G - genetically evolved, x - not present, Y - yes

Moving strategy is a function of an organism, where input is the view and state of the organism and output is a decision what to do. Bitozoa evolve such a function genetically, but bases it on neural networks. Mitozoos uses fixed strategy that is influenced by several parameters. These parameters, e.g. maximal distance of a food bit in the view so as to

head for it, are subject of evolution. Gene Pool has also fixed strategy. Abeetles overtake the idea of evolution of strategy of moving from its predecessor Broucci and intends to explore it.

Learning is a feature that is usually connected with neural networs. Parameter based simulators generally do not use it. But influence of simple learning on evolution of moving might be interesting, therefore it is placed among objectives of Abeetles. The term learning is very general, Abeetles inspired its view of learning with memetic theory.

The term meme was first coined by Richard Dawkins in 1976. Dawkins defined the meme as "a unit of cultural transmission or a unit of imitation"[15]. Memes have as an important characteristic their propagation through imitation, a concept introduced by the French sociologist Gabriel Tarde. Imitation involves copying the observed behavior of another individual. Researchers have observed memetic copying in just a few species on Earth, including hominids, dolphins and birds (which learn how to sing by imitating their parents). The technique of dissemination of memes in population can be formalized and it is described e.g. in lectures of Vlado Kvasnicka from CHTF STU at `ftp://math.chtf.stuba.sk/pub/vlado/Evol_alg_MFF/prednaska1_transp.pdf`.

Abeetles uses imitation of behavior in the way that agents copy some patterns of behavior from other more successful beetles. It would be interesting to explore, to which extent is mutual learning convenient. Abeetle adds probability of learning from another beetle to features of agents and evolves it.

Choice of partner for mating is the next idea Abeetles concerns with. In Primordial Life, Bitozoa and Mitozoos choice of partner is restricted to demand on sufficient level of energy of both parents. Gene Pool is on the other hand interested in influence of criterion of choice of partner on results of evolution. [12] In contrast to it, Abeetles chooses fixed criteria and evolves their values.

Simulator Primordial life offers possiblity to connect simulators and in order to it connect environments together. Plan of Abeetles originally included computation distributed on more computers, but finally it is only single compute r application. Reasons are described later.

Obstacles are incomporated in none of the five simulators. But obstacles and consequenly the shape of the environment can be very interesting for Abeetles, which evolves strategy of motion.

Aging is generally meant as dependence of performance of an agent on age. None of the five simulators implements it. Abeetles makes it possible for a user to set amount of energy obtained from food according to age of a beetle.

Chapter 4

Structure of the thesis in relation to chosen objectives

Chosen objectives are: model of artificial life and implementation of simulator that will run the life and idea of possible experiments and their realization.

The course of following chapters will be:

4.1 Analysis

The first step will be specification of the model of artificial life that will Abeetles use. The model should conform to examination of chosen features of natural life that were described in the previous chapter.

The second step will be description of requirements on the simulator. Which features it should have and which functionality it should offer to users.

4.2 Design

Design of Abeetles and discussion of alternatives.

4.3 Detailed design of Abeetles

4.4 Implementation

4.5 Experiments - description, course and results

4.6 Review of completion of objectives in comparison with other attitudes

Chapter 5

Analysis of the problem from different aspects

5.1 Model of artificial life of Abeetles

Artificial life in simulator Abeetles simulates natural life as environment inhabited by agents. Agents are those elements of environment that are subject to evolution and environment is all the world with other not evolved objects where agents live.

Agents in Abeetles are called beetles and generally they all are of the same species as they can mate with each other and bear descendants, and share the same static part as well as constrains on the variable part of their behavior. Environment is a grid in form of toroid. In the grid three types of objects are placed. Beetles are the only inhabitants of the environment that are able to move. Another type of objects are flowers. Flowers grow in cells of the environment and serve as food for beetles. Neither beetles nor flowers can occur in cells of the grid, where are walls. Walls are static obstacles that serve for forming of the shape of the environment.

Beetles contain static part, that can not be evolved, evolvable part and part that can be influenced during life by learning. Basic features are level of energy, decision mechanism that takes as input what the beetle see and in which state he is and outputs a decision what to do and expectation on a partner for mating.

5.2 Functionality of Abeetles

The simulator runs the artificial model described above. Scenarios of usage can be encapsulated in three particular cases of usage: Running of an experiment, view of an experiment, and learning of possibilities and the artificial life performed by the simulator.

5.2.1 Running of an experiment

The expectations on the system in this case are:

- Speed: Experiment run with the simulator should avail execution of thousands of turns so as to get interesting results of evolution. Also for statistical purposes

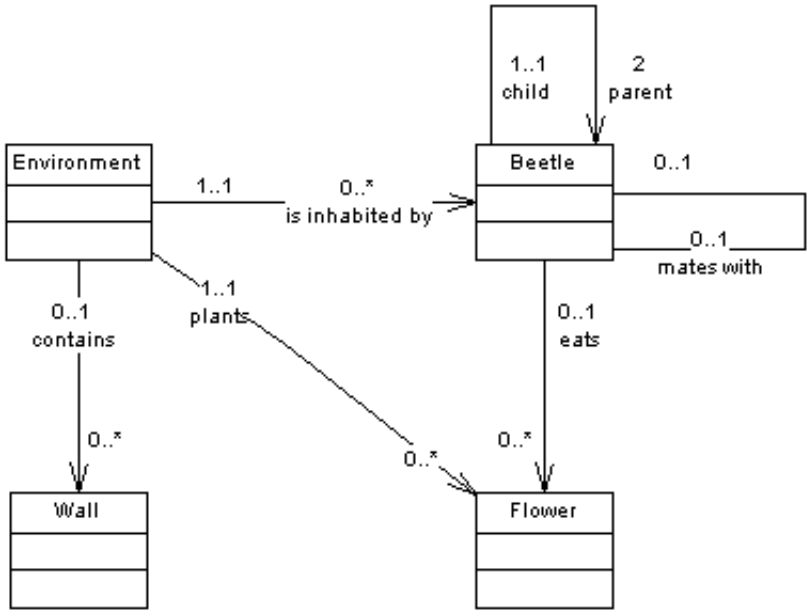

Figure 5.1: Conceptual data model of Abeetles

high number of various inputs must be used when experimenting on random data. Therefore Abeetles should offer possibility to run very fast.

- Input: Start of experiments can not be done manually one by one. Data entry should be read from a script.
- Output data: Results of experiments should be possible to save in an interchangeable format and make their collection automatically.
- Run of the simulator: Simulator should just obtain script at start and then run without need for any input from user.

5.2.2 View of an experiment

- User interface: Interface should offer possibility to look at the environment with all its object so as to see particular situation in the environment that cannot be logged in statistics, e.g. differences between beetles in separate parts of the environment. This could be realized by ability to visualize different aspects of the environment on demand and by ability to zoom various parts of the environment. Also details of individual beetles should be accessible in a concise form.
- Run of simulator: Simulator should be capable of running step by step, of running optional number of turns or of running until being stopped.

5.2.3 Learning about Abeetles

- Speed: A slow run of the simulator helps to see what is happening and understand relationships between events.
- User Interface: The interface should show all the environment with clearly differentiated objects. Legend, explaining what is what, should be easily reachable. A zoom can make possible to see the situation in details as well as to overlook the whole environment.
- Input: Possibility to start default interface with random beetles easily is necessary at the beginning of work with the simulator. Also to save and continue to work with it later.
- Output: To view actual statistics can be performed in any turn.

Chapter 6

Design of Abeetles with discussion of alternatives

6.1 Architecture

6.1.1 Hardware architecture

First, it is necessary to decide, which targets of Abeetles and features recognized by analysis are key for design of architecture of the system. System should be as fast as possible when computing experiments. Simultaneously it should avail to go slowly and display the situation at every step. Speed can be from the point of view of hardware improved by distribution of computation.

But is the model of Abeetles suitable for parallelization of computation? The distribution speeds up the computation only if the task can be separated into independent subtasks and the costs of interchange of data after he completion of each subtask do not prevail the savings from paralel run of the subtasks.

The first question is whether computation of artificial life of Abeetles can be split into independent subtasks. Generally, the run of an artificial life simulator is a sequence of more or less detached updates. In each update each object in the environment gets opportunity to make a small number of actions, usually one, with one input of information from the environment, e.g. snapshot of its surroundings. Interactions in the environment should be only local, so as to split the environment into several parts and each of them compute on a different machine.

In Abeetles the environment is a grid in the form of a toroid. Beetles see only three neihbouring cells. As described later, growth of flowers is dependent only on a setting of each cell. Therefore interactions are only local and paralelization could be covenient.

A possibility of division of the task is splitting of the environment into parts that compute every update separately, run on different computers and after each update send and receive information about the edges that can be in view of objects in the neighboring parts of the environment. Two alternatives of realization of such distribution are: the first, The parts on individual machines are equal applications that communicate peer-to-peer. Each application knows only information about its part of environment and about the borders of neighbors. Despite the peer-to-peer character, this organization needs a server node that starts and stops the other nodes and collects results.

The other one is that there is a central server that splits the environment, sends the parts to individual machines, all the machines compute one update and return the result to the server.

Be all the environment of the shape of a square, split into n smaller squares. Length of an edge of such a square is x. Radius of view of an object in the environment is v. Then in the first case the number of exchanged messages is $4n$ and each message carries information of size av. In the second case the number of messages is $2n$ and size of contained information is a^2.

Advantage of the peer-to-peer attitude is that it is probably faster. Advantages of client - server attitude are possibility to see the all situation and save statistics of the environment without any additional costs. Also control and solving of unexpected situations is much easier.

I would be very interesting to know, whether any of these methods of distribution be so effective would secure such considerable speed-up that would outweigh all the costs of the distribution.

For Abeetles the distibution using peer-to-peer alternative was previously considered, but it would need a great deal of work that exceed the possibilities of this thesis. Nevertheless, implementation of data structure of the grid of the environment is prepared for usage in the peer-to-peer model.

Abeetles is now an application that runs on a single computer.

6.1.2 Software architecture

Modules of Abeetles

Important facts from analysis that influnce choice of layout of modules are usage as a tool for experiments, which expect fast run without visualisation, and usage as a viewer and a learning tool, which expects slow run with elaborate graphical interface. Separation of these two attitudes therefore comes into consideration.

Abeetles could be designed as a tool containing two separate application, that can only share the representation of data. First is a graphical viewer and second is a program run from command line of a console application. This attitude is used by simulator Avida. Avida contains even three kinds of applications: graphical viewer, console application and primitive run with only listing of updates in a console.

Abeetles was first considered to be designed in this way, but paralel development of two applications with many functionality in common was abandoned. The reason is based also on fact, that for Abeetles the program language C++ together with graphical library QT by Trolltech was used, as described in following section. The console application could be, unlike the gui version, written without any gui concerned libraries. These libraries (e.g. QT by Trolltech, Win32API or MFC) contain useful tools for work with files and parsing of text, which are necessary operations in both versions. Develop these operations using non library construct would be more difficult and time consuming and the work does not bring anything new. Do it twice using different instruments would also not be effective. And when in simple application a gui library is also used, then there is no reason to separate the functionality connected with different cases of usage to two independent applications.

Division of application into modules could also follow separation of computation of artificial life from all wrapping functionality. The computation could be called as a dynamical library. For reusage of the core of Abeetles with different interface it would be convenient and it would also be suitable for cooperation of more developers. Abeetles are planned to be prepared for transfer to other operation systems, unix based or MacOS. And dynamical libraries are only usable at one system.

To sum up, Abeetles is an application that contains only one module, that serve for both purposes - running of experiments as well as overlooking individual situations in an environment and learning about Abeetles. Disadvantege lower reusability, but it simplified development and conforms to future compilation at different operation systems.

Choice of platform and programming language

Platform for Abeetles is MS Windows XP.

Speed was the leading criterion when the question of programming language was considered. Model of Abeetles invokes realization by a class based languages, the abstraction is intuitive. Abeetle needs graphical interface and to the chosen language a library for creation of gui should be accessible. Three languages were taken into account: Java, C# and C++. Both Java and C# are, thanks to interpretation of bytecode at runtime, slow in comparison with C++. But a Java application can be run at different OS and is a language of "higher" level than C++. C# applications can run only on Windows, but the future support will be only better. It is as slow as Java. The criterion of speed was considered crucial for Abeetles and C++ was chosen.

Graphical user interface in Abeetles is created using library Qt by Trolltech. Is offers encapsulating class based abstraction of interface and can be used for MS Windows, MacOS Unix and many Unix based platforms. Details are published at Trolltech's webpage `http://trolltech.com/products/qt`. Abeetles uses the open source version of Qt.

Other possibilities were Win32API and MFC. The first is not class based and difficult to use. MFC is a class based library, it is distributed freely with MS Visual Studio, but the level of abstraction is lower then in Qt and future support of this library by Microsoft is not sure.

6.2 Data model of Abeetles

The basic model of life in Abeetles is described in chapter Analysis. Now decomposition proceedes and data structures are designed. The key data structures of Abeetles are beetles and an environment. A beetle has following evolvable features:

- Brain - decision function (situation) -¿ action. The situation is combination of what the beetle sees and his inner states. States of a Beetle are whether he is hungyry or not. It sees what is on the left,in the front and on the right.
- Expectation on parter - A beetle check before mating, whether the partner is fullfils its expectations on amount of energy, investment in children, learning ability and age.
- Hungry threshold - Threshold defines whether the beetle is or is not hungry.

- Investment in children - The beetle invests this amount of energy into its new born offspring.
- Learning Ability - The beetle expects certain level of learning ability from its partner.

Evolvable features are concatenated into a chromosome. When two beetles mate, on their chromosomes is applied genetic algorithm with one-point crossover technique. The other considered alternatives were two-point crossover algorithm, which would be interesting to add to Abeetles and explore differences in results, and cut and splice algorithm, which is not suitable for case of Abeetles because generated chromosomes have different size.

Non evolvable features of beetles are current direction in environment, east, north, west or south and amount of energy.

The Environment of Abeetles is a grid in shape of toroid. Every cell of the grid either contains a beetle, a flower, or a wall or it is empty.

Flowers are objects in environment that serve as food for beetles. Similar source of energy for agents use Bitozoa, Gene Pool and Mitozoos. Gene Pool keeps a fixed amount of energy in environment, the less agents the more food and vice versa. This attitude prevents users from exploring, how would agents develop with higher of lower supplies of food. In contrast to Gene Pool, in abeetles can amount of food vary from zero to full grid of flowers. Due to demand on local interactions in the evironment for purposes of possible parallelization, growth of flower is an individual matter of every cell. Each cell has an optional probability of growth of a flower. If it happens, flower springs up in one update and occupies the cell until it is eaten or dies. Flower can die in any turn with fixed probability.

Environment updates in turns. Therefore both last situation and current situation must be stored. One update goes gradually, cell by cell, in lines. An update of a cell means that if the cell is empty, generator of random numbers decides according to probability of growth in the cell whether a flower springs up. If the cell is occupied by a beetle, the beetle takes a snapshot of three neigbor cells in last update. According to it and his hunger he decides what to do and acts, if possible.

6.3 Components

Functionality of Abeetles is divided into following components: Environment, Run of Script, Run of Gui, Statistics and Configuration Manager. Configuration Manager encapsulates functions for saving and loading of an environment. In 6.1 these components are represented by their key classes.

6.4 User interface

6.4.1 Graphical user interface

The idea of graphical user interface of Abeetles was inspired by interface of Broucci, see Figure 3.6 on page 18, in visualization of beetles. In Abeetles agents are also represented

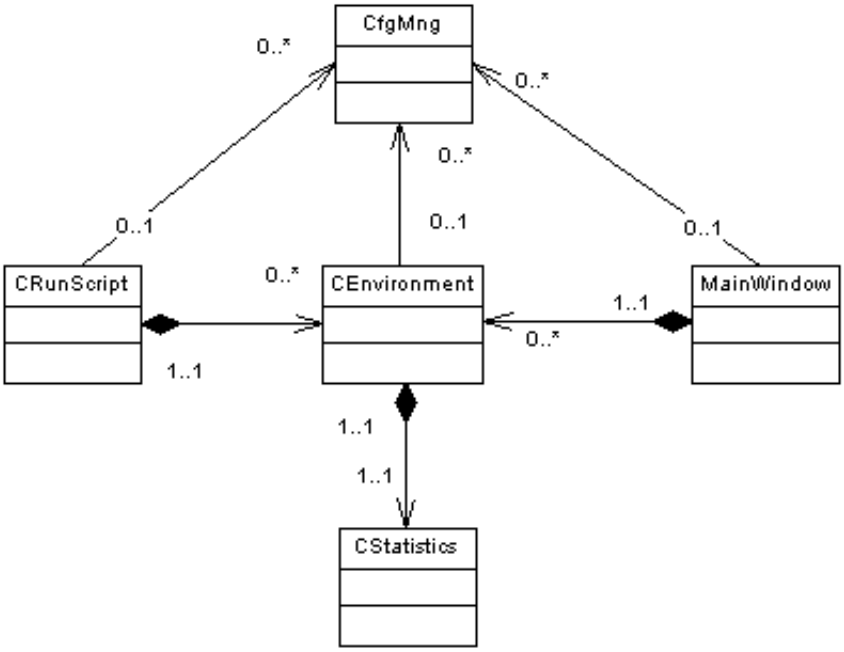

Figure 6.1: Model of main classes of Abeetles

by arrows showing their direction.

Layout of window of GUI of Abeetles resembles the qt-view of Avida, see Figure 3.1 on page 9 . The reason is the similarity of grid base of the environment and intention to serve as a scientific tool.

6.4.2 Statistics

Every artifical life simulator that endeavors after more that being just a game includes some kind of statistics. Systems compared with Abeetles display information either graphically or output them to a file (indicated in brackets):

- Avida: instruction viewer, details of individual strings
- Bitozoa: graph of populations, energy flow and energy flow averaged, information about environment (to file), details about an agent.
- Gene Pool: graph of relationship between number of swimbots and amount of food, features of individual agents, to seek the best agent in chosen criteria.
- Mitozoos: genetic landscape, log of events, on reproduction the crossjoin of genotypes is shown graphicaly, details about an agent.
- Primordial life: information about the environment (population, time death and birth rate numbers), global status of all connected ecosystems, information about an agent.

These simulators prefer to show statistics graphically or textualy in the program. But for further processing, output to files in a standard format might be more useful. Abeetles writes statistics into files, only several actual numbers shows in the graphical interface. Abeetles as well as all described simulators shows details of an individual. First type of statistics offered by Abeetles is actual information about environment - time, number of agents, number of flowers and average values of features of beetles. The second type are graphs of development in time that are very useful for observation of what is going on in the simulator within time. They are also offered by Bitozoa and Gene Pool. In contrast to them, Abeetles does not show graphs directly, but saves values into a file in comma separated value format. Graphs can be then drawn by a special program, e.g. MS Excel. In this format changes of number of beetles, number of flowers and number of births within time are stored. It is also used for graphs displaying frequency of distribution of values of age, learning abilities, investment in children and energy.

6.4.3 Script

Script for Abeetles is intended to be easy to understand. It uses format "command = value" and avails to set any number of runs of the simulator. For each of them all necessary parameters can be defined.

The alternative of usage of command line parameters instead of an Abeetles specific script was considered. One command could start one run of Abeetles and a set of runs could be created in a command line script. But the first alternative was chosen, because it makes it possible to write the script without knowledge of system commands.

Chapter 7

Detailed design of chosen solution

7.1 Data Model of Abeetles Alife

Detailed design of life of Abeetles compounds three classes. CEnvironment, CGrid and CBeetle. UML Class diagram of all three classes and their relationships is in 7.1.

Class CGrid represents the grid of the environment. It stores its size and contents. The most important operations are GetFlowerGrowingProbability and GetCellContent, which returns what is placed in certain cell and in case it is a beetle, it returns reference to it. This class is designed to encapsulate prospective paralelization.

Class CBeetle represents a beetle with all respective features. Beetles make one action in every turn. It can be a step, a rotation or a mating. An attempt to mate happens every time, when a beetle sees another beetle in front of him. If the attemp is not successful, the beetle chooses another action according to his Brain. In the Brain only rotations and steps are included.

The possibility to place mating among other actions in the brain was tried for Abeetles, but it made it impossible for randomly created beetles to evolve, because only a small number of them received randomly the decision to mate in the suitable situation. And the number was not enought to keep the density of population in next generation above the border of extinction.

The most interesting process connected with this class in mating of two beetles. First, demands of both beetles are checked by function IsExpOnPartnerSatisfied(). When beetles find themselves attracted in the other one, CreateChild() operation is called. This method first applies Crossover1Point(),contains the genetic crossover algorithm, and generates two chromosomes. One of them is randomly chosen for the new-born descendant. The descendant then undergoes random changes in operation Mutation(). The new beetle is then placed in one of four cells that are neighboring for both parents. If none of them is empty, the beetle is immediately discarded. Afterwards it starts to live its own life.

Another important process in the environment is the run of life in the environment. Instance of class CEnvironment is called *env*. Its attributes Grid and Grid_Past are of type CGrid. Grid_Past represents the grid in previous update. Using the following code the new env.Grid, representing the grid of the new update is created.

```
for(I=0;I<env.Grid_Past.G_Width;I++)
  for(J=0;J<env.Grid_Past.G_Height;J++)
```

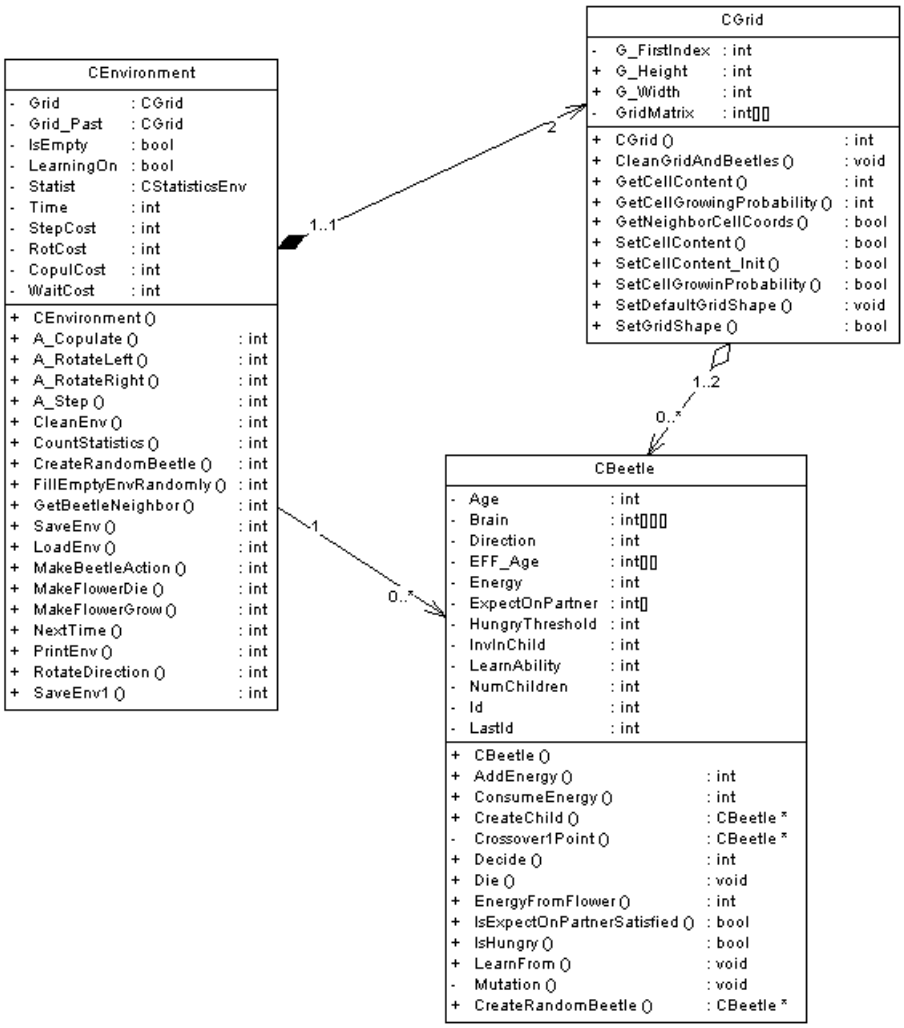

Figure 7.1: Class diagram of Alife in Abeetles

```
{
  if (env.Grid_Past.GetCellContent(I,J)==BEETLE)
    env.MakeBeetleAction(I,J);
  else if (env.Grid_Past.GetCellContent(I,J)==NOTHING)
          env.MakeFlowerGrow(I,J);
       else if (env.Grid_Past.GetCellContent(I,J)==FLOWER)
              env.MakeFlowerDie(I,J);
}
```

The border between two updates is method NextTime() of class CEnvironment. It copied Grid to Grid_Past and counts statistics of the previous turn.

7.2 Design of other components of Abeetles

Components of the system Abeetles and their relationships are in 6.1. The classCfg-Manager encapsulates only input and output methods, it has no attributes. It should be available to all other classes any time. Therefore this class exists in the system only in one instance as a global variable. Class CEnvironment is instantiated by CRunScript as well as by CRunGui. In both it exists in only one piece at a time. Object of class CStatisticsEnv collects information about the run of an environment. So it is designed to be an attribute of CEnvironment, it is constructed and destructed together with it.

7.3 Detailed GUI

7.4 Statistics

As described in previous chapter, there are three types of statistics produced by Abeetles: aggregated statistics, statistics of behavior of environment within time and histogram statistics.

7.4.1 Aggregated Statistics

Aggregated statistics show actual number of beetles, number of flowers, number of new-born beetles and average values of age, energy, hungry threshold, investment in child, learning ability and number of children. These statistics serve to show actual situation in the environment.

7.4.2 Time Statistics

Time statistics show development of environment within time. The included values are number of beetles, number of children and number of new-born beetles. These values are saved to a file comma separated value format. Specialised applications, e.g. MS Excel can paint a graph from it.

7.4.3 Histogram statistics

Structure of population can be observed using histogram statistics. It shows how many beetles have certain value of a particular feature. It contains age, learn ability,investment in children, energy, number of children and hungry threshold. Also expectations of beetles on partners on energy, age, investment in children and learn ability are included. Each value is connected with number of all beetles who have it within their range.

7.5 Script

Abeetles can be run with an input script file. The script is a text file containing description of one or more individual runs of the simulator. Runs are separated by key word *run*. The

complete form of the file is described in following example using C++-style comments. In the script file itself no comments are allowed.

```
run=env3  //name of the run and name of directory to store
          //results
map=env_cfg.bmp //name of bmp picture of the map that should
                //be used
beetles=random,200,20 //Initial set of beetles can be either
  //random or can be read from a file. If it is random, it must
  //be stated by word "random", then follows the seed for
  //a generator of pseudorandom numbers "200" and then number
  //of beetles to be generated. Both numbers can be -1, which
  //means time-based seed and default number of beetles.
eff=EnergyFromFlower.bmp //name of bmp image that describes
  //dependency of beetles energy exploited from an eaten flower.
mutationprob=5//probabality of mutation of genes of new-born
  //beetles. Must be within range 0-10.
costs=1,1,2,0 //cost of actions of a beetle: step, rotation,
  //mating, waiting
endtime=199000 //number of turns of the run of the simulator
aggrstatfn=aggr.txt //name of files where aggregated statistics
  //should be written
histstatfn=hist.csv //the same for histogram statistics
timestatfn=time.csv //the same for time statistics
savetimeaggrreg=150 //every 150 turns will be aggregated
  //statistics saved to file <time><aggrstatfn>
savetimehistreg=200 //the same for histogram statistics

run=env4
map=env_cfg.bmp
beetles="beetles.txt"//name of file of beetles
eff=EnergyFromFlower.bmp
costs=1,1,2,0
endtime=1000
aggrstatfn=aggr.txt
histstatfn=hist.csv
timestatfn=time.csv
savetimeaggrreg=100
savetimeshist=70,500,1000 //enumeration of turns, when
    //histogram statistics should be saved.
savetimesaggr=70,500,1000 //the same for aggregated statistics
```

Chapter 8

Description of prototype implementation

8.1 Description of the model of a beetle

Representation of an agent(beetle) in Abeetles from genetical point of view: Chromosome of a beetle includes following genes:

Brain [2][4][4][4] - Set of actions, each of them is chosen by the beetle in the proper situation. Situation is defined by the quaternion (hunger,what is in the left, the front, the right). Item "hunger" specifies whether or not is the beetle hungry. Values of what is in the three directions from the beetle are four: beetle, flower, wall or nothing. These four possibilities are precisely the four possible contents of cells of the environment.
Actions, for which the beetle can decide in this situations are chosen from the following set: make one step, turn right, turn left.
Originally there were five actions: these three and copulation and waiting. Decision for copulation has sense only in situations, where is another beetle in front of the beetle and when the beetle is not hungry. And, furthemore, it these situations the beetle should always or almost always copulate (add here more rigorous explanation)when he meets another beetle. As a result, it is compulsory action in case a beetle sees another beetle in front of him.
The action waiting was omitted - it gets beetles into deadlock situation. They should never decide for waiting, when they are hungry, because there cannot grow any flower on the spot, where they are staying. And when they are not hungry, it might be useful only as waiting for somebody to come and make child with him. But even in this situation it is not clever - beetle cannot finish it, because it has no memory. So it waits until it gets hungry or situation around changes.
ExpectOnPartner - Age [2] = 2B how much older / younger can be the partner ExpectOnPartner - Energy = 1B how much more than ExpectOnPartner - InvInChild ExpectOnPartner - InvInChild = 1B how much more than InvInChild ExpectOnPartner - LearningAbility [2]= how much less/more can have the parter InvInChild = 1B HungryThreshold = 1B LearningAbility = 1B

Chapter 9

Experiments

With Abeetles numerous experiments can be conducted. The space is set by possible values in cathegories size of map of environment, layout of walls in the map, probability of growth of flowers, initial number and features of beetles, availability of learning, function energy from flower - absolute values as well as the course of the curve, and costs of actions and their ratio.

Focus of experiments can be manifold. As mentioned above, such simulators are used for exploration of paralels between artificial and natural life. Parameters could be set to resemble some natural life feature and then examine what happens or search initial parameters that would cause development with some attributes of natural life. Often problem of artificial life simulators is that the evolution after starting development gets to a balanced point, aroud which it only oscilate but does not change essentialy any more.

In this chapter .. experiments conducted with Abeetles will be introduced and described.

9.1 Experiment1 - Four Caves

Map of environment are four caves connected with narrow corridors. It contains 900 cells. Percentage of growth of flowers is 100%.

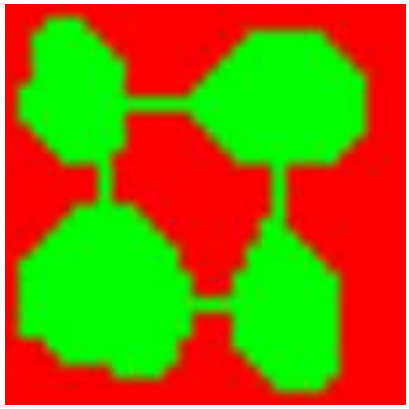

Figure 9.1: Map of environment in Experiment1 - Four Caves.

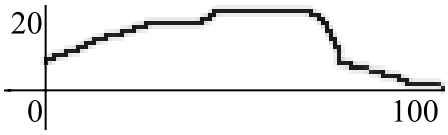

Figure 9.2: Course of function Energy From Flower in Experiment1 - Four Caves. horizontal vertex = time: 0 .. 100th update, vertical vertex = energy: 0 - 20

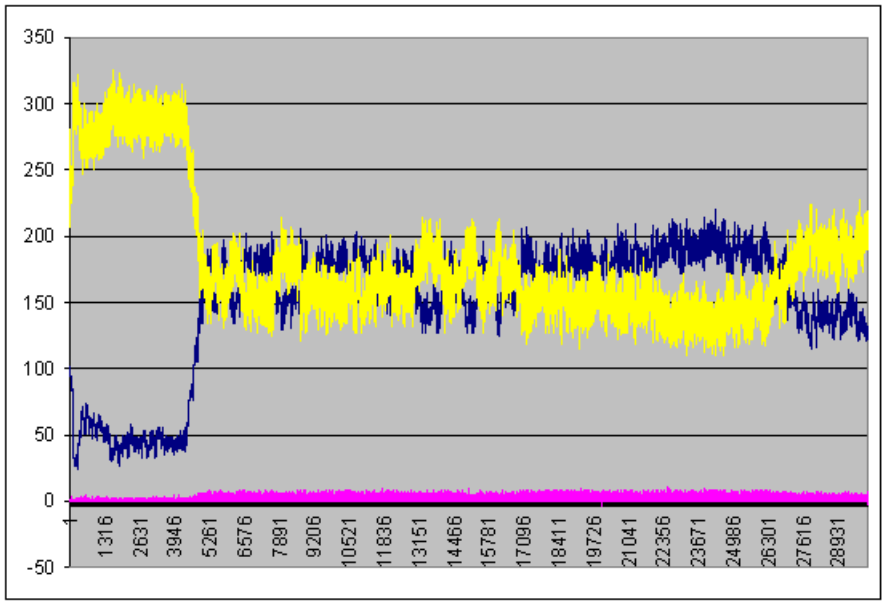

Figure 9.3: Run 1 Experiment1 - Four Caves.

Initial number of beetles is 100, their features were chosen randomly with rule "step on flower". Costs of actions were set to 2 for a step, 2 for a turn, 3 for mating and 1 for waiting. 30,000 updates were made. Function of energy from flower is shown in figure 9.2. The EFF function restricts the age of beetle to approximately 100 years.

The experiment contains four runs with seeds 300, 350, 400 and 450 for generator of pseudorandom numbers. Course of experiments are in figures 9.3, 9.4, 9.5 and 9.6. Number of beetles, number of flowers and number on new born beetles is shown in figures.

Differences between courses show, that coincidence influences the course of matters to a great extent. The map of four caves causes that course of number of agents is not smooth. As a free cave is inhabited, the number of beetles jumps by about 50. Inhabitation means that the density of population in the cave is high enough to be independent on immigration from other caves. In other words there are enough beetles to meet, mate and thereby keep the population. On the other hand, if the population shinks under certain size, it becomes extinct in the cave. For example, in update 26,500 in run 1, 9.3, in top left hand corner beetles became extinct, while the other three caves are populated.

Initial features of beetles were random, histogram of run 1 at the beginning: 9.7. The Situation after 30,000 updates is in 9.8.

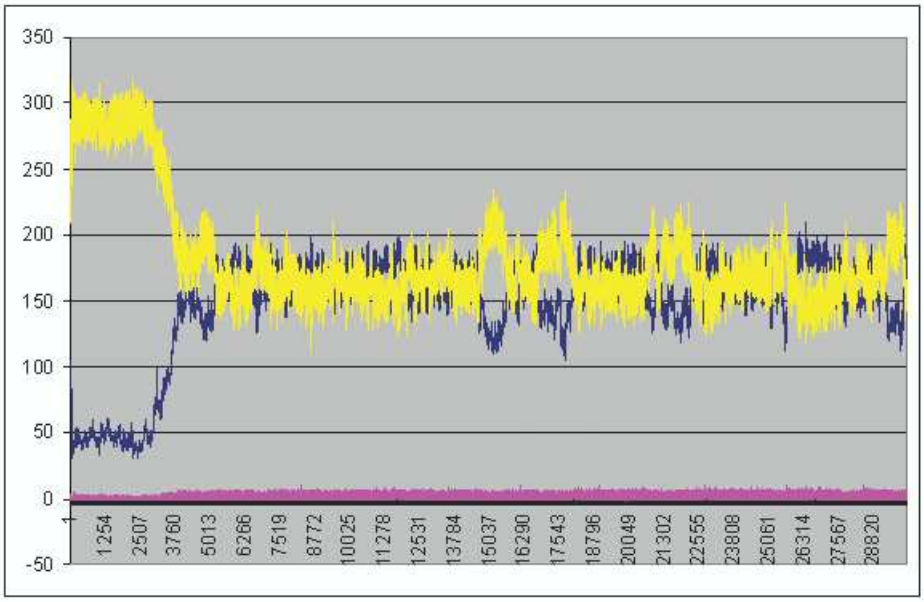

Figure 9.4: Run 2 Experiment1 - Four Caves.

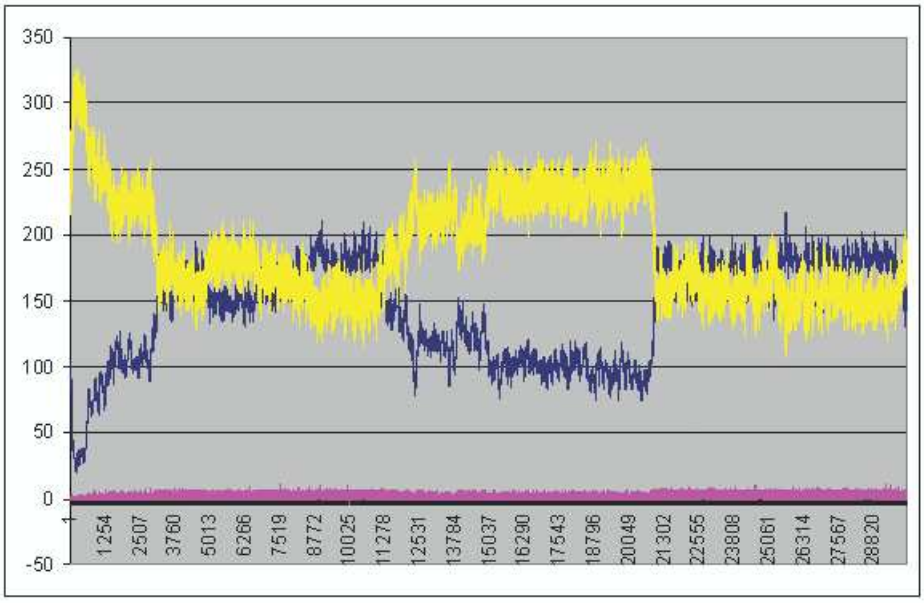

Figure 9.5: Run 3 Experiment1 - Four Caves.

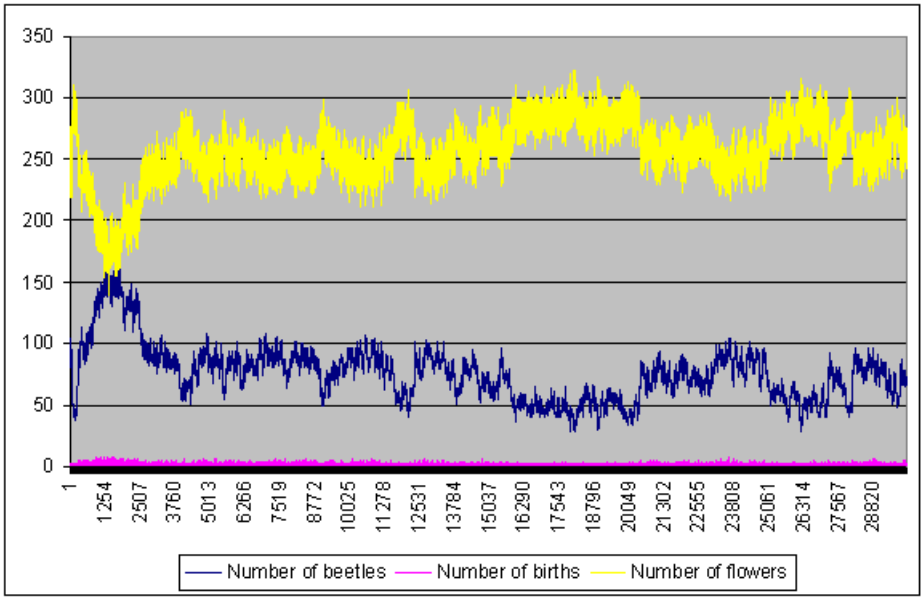

Figure 9.6: Run 4 Experiment1 - Four Caves.

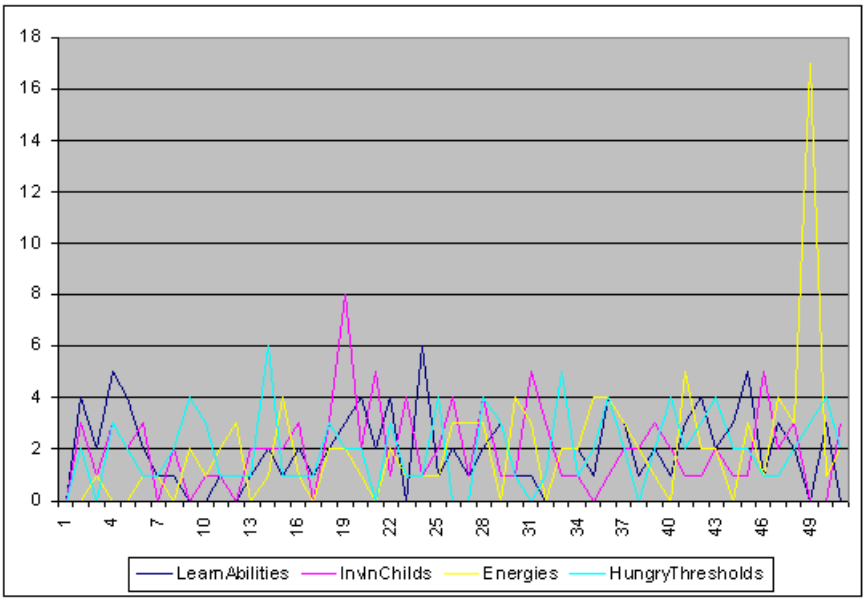

Figure 9.7: Run 1 Experiment1 - Four Caves, histogram of features of beetles at the beginning.

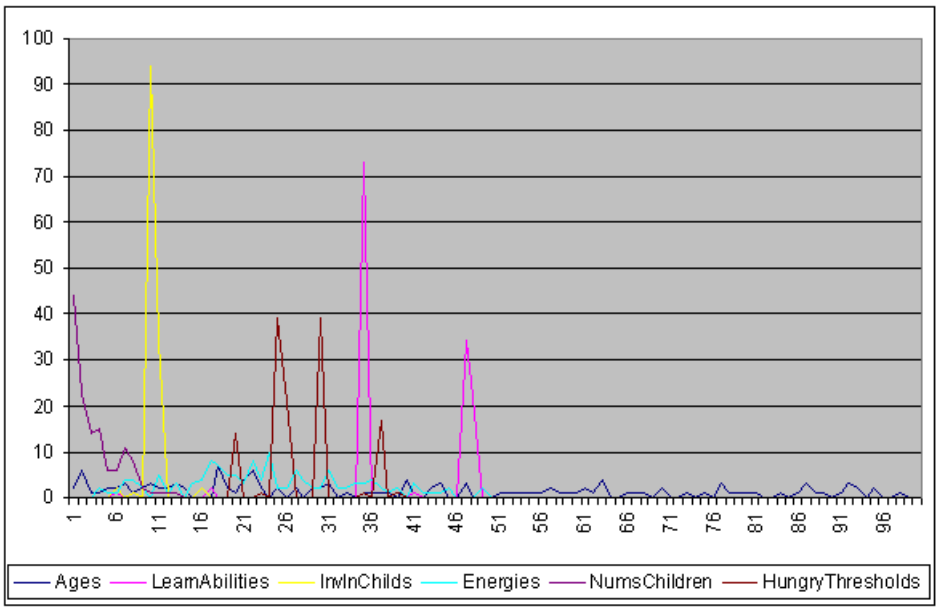

Figure 9.8: Run 1 Experiment1 - Four Caves, histogram of features of beetles after 30,000th update.

Chapter 10

Resume of fulfilling of objectives and other known attitudes

Chapter 11

Resume with brief summon of results and contribution

Chapter 12

References to related web pages

`http://myxo.css.msu.edu/cgi-bin/lenski/prefman.pl?group=al` - Evolution experiments with digital organisms

Chapter 13

Implementation

13.1 Paralelisation

13.1.1 I

dea of paralelisation in Abeetles: Paralelisation should distribute the computation of the grid environment for abeetles on several computers. As the environment is a grid, it can be cut into rectangular parts, each of this executed and evolved on a different source computer. This parts can have versatile size according to performance of the computer. With this schema, the number of fields of one part, that are computed on one computer is up to quadratical to width(heigth) of the part and number of fields of one part, that need communication with other parts is linear (four times) to width(heigth) of the part.
Why paralelisation can make the computation quicker: Beetles are located in a grid, so the interactions between the grid cells are similar to those of a celular automata - each beetle interacts only with its nearest neighborhood. Artificial systems such as Tom Ray's tierra [10] have opened the possibility of studying evolvable systems. The study of evolution in such an information-rich artificial environment requires ever larger and faster systems, and present systems are largely restricted by such limits. Distributing over-multiple processors is not practical on a large scale, because of the non-local interaction between members of the tierran population. But the system Avida developed the idea of the environment towards distribution. Avida is based on an array of cells that interact only with their nearest neighbours and an update mechanism reminiscet of 2D cellular automata.[9]
Nevertheless Avida system does not include distributed computation in its latest implementation, neither in older ones. Abeetles should exploit ideas presented in the article about Avida, follow suggestions about local interaction and other improvements of Avida and implement them in a system, that will execute in a paralel way. But unlike Avida, Abeetles will be a parameter based system, not a program based.

13.1.2 C

hoice of tool for Abeetles: There are two main tools used for paralelisation of computation. PVM(Paralel virtual Machine) and MPI(Message Passing Interface). Comparison of features has been published in an article [16]

Topology:

MPI provides higher level of the message-passing topology. In MPI a group of tasks can be arranged in a specific logical interconnection topology. PVM does not support such an abstraction, the programmer is to manually arrange tasks into groups with the desired communication organization. For Abeetles the concept of MPI would be more convenient. But the organization of squares as parts of a toroid should be simple enough to realize using PVM.

Fault Tolerance:

Fault tolerance is important for any long-running simulation. PVM has supported a basic fault notification scheme. Using this scheme individual tasks can inform other tasks about faults and respond to faults without hanging or failing.

Resource Control:

PVM is inherently dynamic in nature. Computing resources can be added or deleted any time. In a longrunning application in which the user occasionally wishes to attach a graphical frontend to view the computation's progress. Without virtual machine dynamics, the graphical workstation would have to be allocated during the entire computation. MPI lacks such dynamics and is, in fact, specifically designed to be static in nature to improve performance. Abeetles intends to be such an application - fast running, with rare user requests for graphical view of progress (possibly) or graphs and metrics of the course of the computation. Thus PVM should serve better for the case of Abeetles.

Process Control:

Ability to stop and start tasks in included both in MPI(v.2.0) and PVM.

Portability versus Interoperability:

MPI programs are only portable, which means that they can be ported from one architecture to another, compiled and executed without modification. PVM programs can be dealt in the same way, but moreover can also communicate with each other.

Context for Safe Communication:

Offered by both MPI and PVM (v.3.4.)

Speed:

MPI is expected to be faster within a large multiprocessor. Reason is MPI's simplicity.

13.2 Genetical Algorithm

13.2.1 Choice of genetical algorithm for Abeetles

Representation of an agent(beetle) in Abeetles from genetical point of view: Chromosome of a beetle is the set of variables, representing the beetle:

Brain [2][4][4][4] = 128B

I will consider the individual brain decisions as noninteracting and availe placing of a crossover point anywhere after finished set of decisions

I want following variables to be logicaly noninteracting and thus it should be possible to place a crossover point anywhere among them. Hence I represent the expectations on a partner as relative divergence from other values.

ExpectOnPartner - Age [2] = 2B how much older / younger can be the partner ExpectOnPartner - Energy = 1B how much more than ExpectOnPartner - InvInChild ExpectOnPartner - InvInChild = 1B how much more than InvInChild ExpectOnPartner - LearningAbility [2]= how much less/more can have the parter InvInChild = 1B HungryThreshold = 1B LearningAbility = 1B

For implementation of crossover step the possible crossover points must be carefully chosen. The target is to find points, that availe separation of the genome into two noninteracting groups. Otherwise the crossover is not important for evolution and may be omitted.

Points can be placed between any pair of adjacent variables. Two created parts will be independent

Generally:

- 1.selection
- 2.crossover
  - One-point crossover: This algorithm seems to be useful for the case of Abeetles. I will implement it as the first.
  - Two-point crossover: This algorithm seems to be useful for the case of Abeetles. I will implement it.
  - "Cut and splice": This technique is not suitable for case of Abeetles, because created children have different length of chromosomes.
  - Uniform Crossover and Half Uniform Crossover: These aproaches may also be useful.
  - Crossover for Ordered Chromosomes
- 3.mutation

Sources:

- `http://en.wikipedia.org/wiki/Crossover_\%28genetic_algorithm\%29`
- `http://www.springerlink.com/content/b67qv704abd9mtgt/fulltext.pdf`
- `http://www.springerlink.com/content/b67qv704abd9mtgt/`

- http://www.cs.cas.cz/~roman/ea.html
- http://wiki.matfyz.cz/wiki/AIL025
- http://en.wikipedia.org/wiki/Swarm_intelligence

13.3 Memory of beetles

Beetles without memory have no chance to vary their decision, if the situation around them is constant. And if the decision is anything else than a step, their repeat this until they get hungry, die or the situation changes. But in case their are already hungry and the situation is a for long time constant, they are determined to die. And in the range of whole environment, if there are few beetles in the environment, they become extinct. The other problematic situation ocurrs, when two beetles meet and want to copulate with each other. They might make a child or more, but then they do not have enough energy to create another child and get into deadlock, until situation around them changes or one of them dies. (Ne.. tohle nevadi, protoze se zmeni situace z hladove na nehladovou. Postreh: obecne nema smysl, aby hladovi broucci mohli volit akci $A_COPULATEPostreh2 : Ajenutne, abybroucekprikazdemsetkanisjinymbrouckemprovedl A_COPU$ $omitaction A_COPULATE fromlistofpossibleactionsandincludeitas"hardwired".Sothisactionwillb$

Ways how to solve this problem: 1. Flowers could besides growing also die back. Such a solution would cause the environment to vary in every step, even if beetle don't eat the flowers. 2. Beetles could get aditional memory, to determine a deadlock-like situation. This could be done in several different ways: a) remember n past coords - a beetle which does not move is likely to die. b) remember n past actions - a beetle which does some other action than the step more than 3 times (rotation) or more than once (unsuccessful copulation) does it wastefully.

Chapter 14

my Remarks

http://www.alife.org/-InternationalsocietyforArtificiallife
http://www.scs.carleton.ca/\~soma/biosec/readings/spafford-viruses.pdf - Interesting essay about considerations whether viruses are forms of artificial life or not. Result: No. Paper includes a worthwhile definition of life:

- Life is a pattern in space-time rather than a specific material object.
- Self-reproduction, in itself or in a related organism.
- Information storage of a self-representation.
- A metabolism that converts matter/energy.
- Functional interactions with the environment.
- Interdependence of parts.
- Stability under perturbations of the environment.
- The ability to evolve.
- Growth or expansion

Chapter 15

Ideas

- Pocet rodicu ¿ 2 ... jak to ovlivni evoluci?

Bibliography

[1] Mark A. Bedau (November 2003). Artificial life: organization, adaptation and complexity from the bottom up. TRENDS in Cognitive Sciences. http://www.reed.edu/\~mab/publications/papers/BedauTICS03.pdf

[2] Chris G. Langton(1989) Artificial Life Addison-Wesley. http://zooland.alife.org/

[3] Chris Adami and Titus Brown (2000) "What is Artificial Life?" The Seventh International Conference on the Simulation and Synthesis of Living Systems, Reed College, Portland, Oregon, USA, 2000

[4] Stefan Bornhofen and Claude Lattaud (2006) Artificial Evolution, chapter Outlines of Artificial Life: A Brief History of Evolutionary Individual Based Models, pages 226-237, Springer, April 2006

[5] Marianne Delorme, "An Introduction to Cellular Automata," Cellular Automata: a Parallel Model, Mathematicsan Its Application ,Kluwer. http://citeseer.ist.psu.edu/delorme98introduction.html

[6] Gene. (n.d.). Columbia Electronic Encyclopedia. Retrieved July 14, 2007, from Reference.com website: http://www.reference.com/browse/columbia/gene-ent

[7] Lewontin, Richard, "The Genotype/Phenotype Distinction", The Stanford Encyclopedia of Philosophy (Spring 2007 Edition), Edward N. Zalta (ed.), http://plato.stanford.edu/archives/spr2007/entries/genotype-phenotype/.

[8] Joshua Lederberg and Alexa T. McCray (2001). "'Ome Sweet 'Omics – A Genealogical Treasury of Words". The Scientist 15 (7). http://lhncbc.nlm.nih.gov/lhc/docs/published/2001/pub2001047.pdf

[9] Avida: A Software Platform for Research in Computational Evolutionary Biology; C. Ofria and C.O. Wilke, Artif. Life 10 (2004) 191-229.

[10] Ray, T. S. 1991, "Evolution and optimization of digital organisms", in Billingsley K.R. et al (eds), Scientific Excellence in Supercomputing: The IBM 1990 Contest Prize Papers, Athens, GA, 30602: The Baldwin Press, The University of Georgia. Publication date: December 1991, pp. 489-531.

[11] Jeffrey Ventrella 1998, "Attractiveness vs. Efficiency: How Mate Preference Affects Locomotion in the Evolution of Artificial Swimming Organisms", Artificial

Life VI 1998, MIT Press. http://www.ventrella.com/Alife/Attractiveness/attractiveness_0.html

[12] Jeffrey Ventrella, "Gene Pool: Exploring the Interaction Between Natural Selection and Sexual Selection", Chapter 4 in Artificial Life Models in Software, edited by Andrew Adamatzky and Maciej Komosinski, Springer, 2005

[13] The book of the mitozoos. (http://bestiario.org/mitozoos/english/pdf/mitozoos_en.pdf). Mitozoos World is a project created and developed by Bestiario company. Website of the project is (http://bestiario.org/mitozoos/english/index.html).

[14] Tomas Holan, "Different Programming", http://ksvi.mff.cuni.cz/~holan/jinak/itat.pdf, ITAT 2004

[15] Richard Dawkins, The Selfish Gene, 11. Memes:the new replicators, Oxford University, 1976, second edition, December 1989, ISBN 0-19-217773-7; April 1992, ISBN 0-19-857519-X; trade paperback, September 1990, ISBN 0-19-286092-5

[16] PVM and MPI: a Comparison of Features; G.A.Geist J.A.Kohl P.M. Papadopoulos