# The Framsticks system: versatile simulator of 3D agents and their evolution

Maciej Komosiński
*Institute of Computing Science, Poznan University of Technology, Poznan, Poland*

***Keywords*** *Cybernetics, Simulation, Complexity*

***Abstract*** *Various aspects of the Framsticks system are described. The system is a universal tool for modeling, simulating and optimizing virtual agents, with three-dimensional body and embedded control system. Simulation model is described first. Then features of the system framework are presented, with typical and potential applications. Specific tools supporting human understanding of evolutionary processes, control and behaviors are also outlined. Finally, the most interesting research experiments are summarized.*

## 1. Introduction

Life is one of the most complex phenomena known in our world. Researchers construct various models of life, which serve diverse purposes, ranging from entertainment to medicine. One of such models is *Framsticks*, a system developed since 1997 (Komosiński and Rotaru-Varga, 2000, 2001; Komosiński and Ulatowski, 1997). It is different from most other models in that it does not address a single purpose or a single research problem. On the contrary, it is built to support a wide range of experiments, and to provide all of its functionality to users, who may use this open system in a variety of ways.

Another feature which is special for Framsticks is the significance of *understanding*, which is central for system development. Although this is a much simplified model of reality, it is easily capable of producing more complex phenomena than a human can comprehend (Komosiński, 2000). Thus it is essential to provide as many automatic analysis/support tools, as it is possible. Intelligible visualization is one of the most fundamental means for human understanding of artificial life forms, and this feature is certainly present in the software.

The system is designed so that it does not introduce restrictions concerning complexity and size of creatures. Thus neural networks can have any topology

Kybernetes
Vol. 32 No. 1/2, 2003
pp. 156-173

0368-492X
DOI 10.1108/03684920310452382

Maciej.Komosinski@cs.put.poznan.pl; www.frams.alife.pl
This work has been supported by the State Committee for Scientific Research, from KBN research grant no. 8T11F 006 19, and by the Foundation for Polish Science, from subsidy no. 11/2001.

and dimension, allowing for a range of complex behaviors, some described in Komosiński (2000), section 4. Avoiding limitations is important because Framsticks is ultimately destined to simulation of open-ended evolution, where interactions between creatures and environment are the sources of competition, cooperation, communication, intelligence, etc.

Further sections focus on the following issues: Section 2 – simulation (morphology, control system, environment); Section 3 – general system framework (genotype-phenotype relationship and possible usage of the system); Section 4 – tools that the system provides to support research; Section 5 – sample experiments which have already been performed, as well as some ideas for the future. Section 6 summarizes this paper.

## 2. Simulation

Framsticks simulates a three-dimensional world and creatures. All kinds of interaction between physical objects are considered: static and dynamic friction, damping, action and reaction forces, energy losses after deformations, gravitation, and uplift pressure – buoyancy (in a water environment).

There is always a tradeoff between simulation accuracy and simulation time. We need a fast simulation to perform evolution, on the other hand the system should be as realistic (detailed) as possible to produce realistic (complex) behaviors. As we expect emergence of more and more sophisticated phenomena, the evolution has to be longer and the simulation must be less accurate, but faster[1]. Thus, in order to make the simulation fast and due to the computational complexity, some aspects, like collisions between parts of an organism itself, were discarded.

Artificial creatures in Framsticks are built of body and brain. Body is composed of material points (called *parts*) connected by elastic *joints*. Brain is made from *neurons* (these are signal processing units, receptors and effectors) and neural *connections*. For more detailed description of this model refer to GDK at Ulatowski (2000).

### *2.1 Body*

The basic element is a stick made of two flexibly joined parts (finite element method is used for step-by-step simulation). Parts and joints have some fundamental properties, like position, orientation, weight, and friction, but there may also be other properties, like the ability to assimilate energy, durability of joints in collisions, etc. Articulations exist between sticks where they share an endpoint; the articulations are unrestricted in all three degrees of freedom (bending in two planes plus twisting). Figure 1 shows forces considered in the simulation.

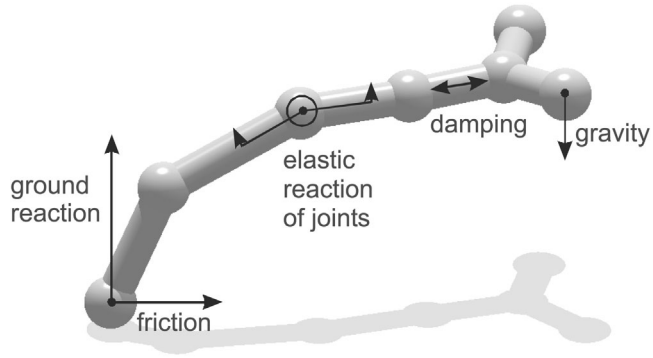


**Figure 1.**
Forces involved in the simulation

*2.2 Brain*

Brain (the control system) is made of neurons and their connections. A neuron may be a signal processing unit, but it may also interact with body as a receptor or effector. There are some predefined types of neurons.

The standard Framsticks neuron is more general than a popular weighted-sum sigmoid-transfer-function neuron used in AI: three parameters are used for each neuron. They influence speed and tendency of changes of the inner neuron state, and the steepness of the sigmoid transfer function. However, in the special case, when the three parameters are assigned specific values, the characteristics of a neuron becomes identical to the popular, reactive AI neuron. In this case, neural output reflects instantly input signals. More information and sample neuronal runs can be found in the simulator details section at Komosiński and Ulatowski (1997).

Another pre-defined neuron is a sinus-generator with frequency controlled by its inputs. Random noise generator neuron is also available. It is easily possible to add custom user-designed neurons.

The neural network can have any topology and complexity. Neurons can be connected with each other in any way (some may be unconnected). Inputs can be connected to outputs of other neurons (also senses), while outputs can be connected to inputs of other neurons (also effectors – muscles). A sample control system is shown in Figure 2.

*2.3 Receptors and effectors*

Receptors and effectors are interacting between body and brain. They must be connected to brain in order to be useful, but they must also be a part of the world. Framsticks have currently three kinds of *receptors* (*senses*): for

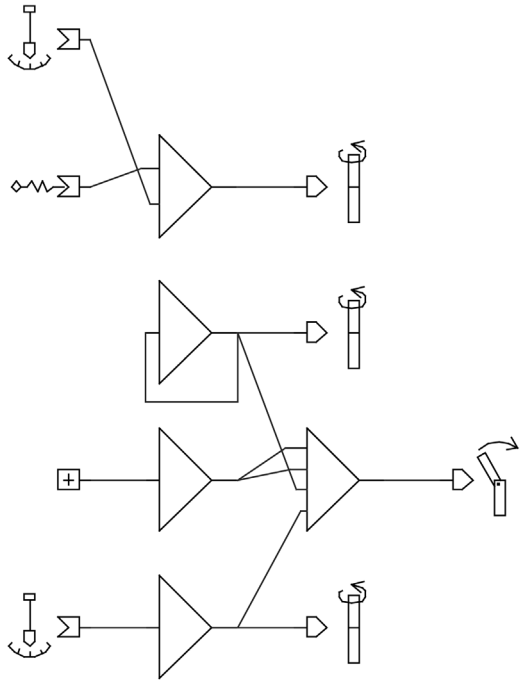

**Figure 2.**
A sample neural network. Triangles are signal-processing neurons. You can see receptors on the left (gyroscope, touch, constant signal) and controlled muscles on the right (rotating, bending). Note recurrent and parallel connections

orientation in space (equilibrium sense, gyroscope), detection of physical contact (touch), and detection of energy (smell). See also Figure 2.

Effectors (muscles) are placed on stick joints. There are two kinds of muscles: bending and rotating. Positive and negative changes of muscle control signal make the sticks move in either direction – it is analogous to the natural systems of muscles, with flexors and extensors. The strength of a muscle determines its effective ability of movement and speed (acceleration). If energetic issues are to be considered, then a stronger muscle consumes more energy during its work.

A sample Framstick equipped with these elements is shown in Figure 3.

### *2.4 Environment*

The world may be flat, build of smooth slopes, or blocks (see Figure 4). It is possible to adjust water level, so that not only walking/running/jumping creatures, but also the swimming ones, emerge. World boundaries may be:

- none (the world is infinite),
- hard (fence: it is impossible to cross the boundary),
- wrap (crossing the boundary means teleportation to another world edge).

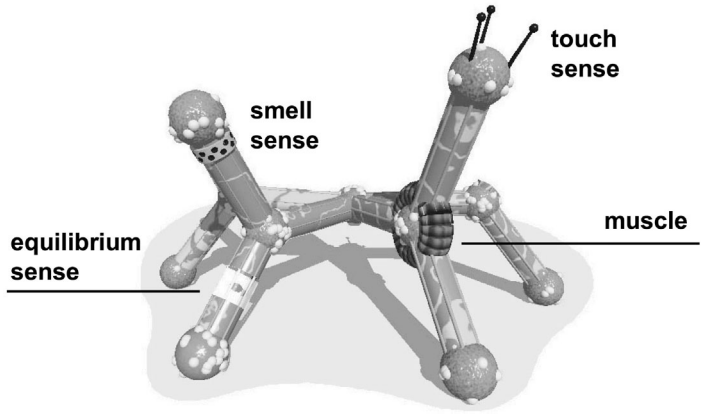


**Figure 3.**
Receptors (equilibrium, touch, smell) and effectors (muscles) in Framsticks

All these options are useful in various kinds of experiments and performance measurements.

## 3. Framework and evolution

### *3.1 Genetics*

There are multiple genetic encodings ("genotype languages") supported by the Framsticks system, each with its own representation and operators. The system manipulates and transforms genotype strings in various representations, and ultimately decodes them into the internal representation used by the simulator.

Any creature can be completely described using a low-level representation ***f*0**, by listing all of its components and attributes. This representation can be treated as a special genotype encoding – special because it is a direct one-to-one mapping. Other higher-level encodings convert their representation into the corresponding ***f*0** version (possibly through another intermediary representation). The reverse mapping into higher-level encodings is difficult

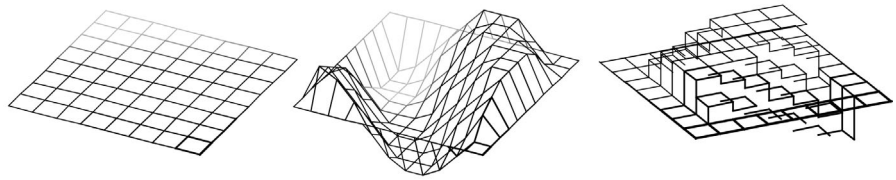

**Figure 4.**
Flat land, smooth slopes, and blocks

to compute, which is also true for biological phenotype encodings. As a consequence, in the general case, it is not possible to convert a lower-level representation into a higher-level one (or a higher-level one into another higher-level one).

Each encoding has its associated genetic operators (mutation, crossover, and optional repair), and a decoding procedure, which translates a genotype into a **f0** representation. A new encoding can be added relatively easily, by implementing these components, without the need to work with internal representations. The Framsticks system is accompanied by the Genotype Development Kit (GDK) to simplify this process (Ulatowski, 2000). Later the most popular encodings are shortly described.

The direct low-level, or **f0**, encoding describes agents exactly as they are represented in the simulator. This encoding is more of a direct representation than a proper encoding, but it is possible to use it as such. It does not use any higher-level features to make the genotype more compact or flexible, and because of this, it is expected that this encoding is not very well suited for evolution. Its useful characteristics are that it has a minimal decoding cost and that it is universal: every possible agent can be described using this encoding. These properties make it possible to use the **f0** encoding as an intermediary representation during the translation from other higher-level encodings.

The direct low-level (**f0**) genotype consists of a list of descriptions of all the objects the agent is composed of: *parts*, *joints*, *neurons*, and *connections*. Every description specifies all the attributes of the object explicitly. Generally, this encoding does not impose any restriction on the phenotypes, and it even allows morphologies with cycles.

The recurrent direct encoding (**f1**) describes all the parts of the corresponding phenotype. Small changes in the genotype cause small changes in the resulting creature. Body properties are represented locally, but propagate through a creature's structure (with decreasing power). That means that most of the properties (and neural network connections) are maintained when a part of a genotype is moved to another place. Control elements (neurons, receptors) are associated with the elements under their control (muscles, sticks). Only treelike structures can be represented (no cycles allowed). This encoding is relatively easy for humans to manipulate and design creatures manually by editing their genotypes.

The developmental encoding (**f4**) is development-oriented, similar to encoding applied for evolving neural networks (Gruau *et al.*, 1996). An interesting merit of developmental encoding is that it can incorporate *symmetry* and *modularity*, features commonly found in natural systems, yet difficult to formalize. **f4** is similar to **f1**, but codes are interpreted as commands by cells (sticks, neurons, etc.). Cells can change their parameters, and divide. Each cell maintains its own pointer to the current command in the genetic code. After division, cells can execute different codes, and thus differentiate themselves.

The final body (phenotype) is the result of a development process: it starts with an *undifferentiated* ancestor cell, and ends with a collection of interconnected *differentiated* cells (sticks, neurons and connections).

Each of these encodings has its own genetic operators (mutation and crossover). Each of them has been carefully designed and tested, and each encoding was based on numerous theoretical considerations. More detailed description can be found in Komosiński and Rotaru-Varga (2001). Examples of simple genotypes and corresponding phenotypes are shown in Figure 5.

### *3.2 General framework*

The most important feature of Framsticks is that you may define your own rules for the simulator. There are no predetermined laws, but a script – *experiment definition*. A script is a set of instructions in some language, which is interpreted by some program and executed.

This script defines behavior of the Framsticks system in a few associated areas.

- Creation of objects in the world. The script defines where, when and how much of what objects will be created. An object is an evolved organism, food particle, or any other element of the world designed by a researcher. Thus, depending on some specific script, food or obstacles might appear, move and disappear, their location might depend on where creatures are, etc.
- Objects interactions. Object collision/contact is an event, which may cause some action defined by the script author. For example, contact may mean energy ingestion, pushing each other, destruction, or reproduction.
- Evolution. A steady-state (one-at-a-time) selection model, where a single genotype is inserted into a gene pool at a time, can be used. But a standard (i.e. generational replacement) evolutionary algorithm approach is also possible (a new gene pool replaces the whole old gene pool). Another possibility is tournament competition for all pairs of genotypes. In general, the script can define many gene pools and many populations, and perform independent evolutions under different conditions.

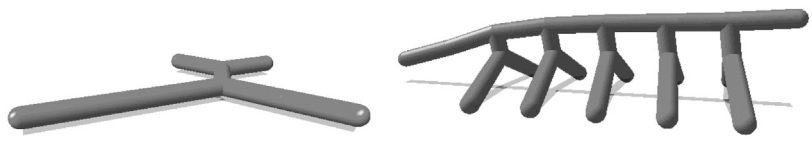

**Figure 5.**
Left: example of ***f*1** genotype `XXX(XX,X(X,X))`. Right: example of ***f*4** genotype with repetition gene: rr < X > #5 < , < X > RR ⋘ llX > LX > LX ⋙ X

- Evaluation criteria are flexible, and do not have to be as simple as the performances supplied by the simulator (but they will have to be based somehow on performances). For example, fitness might depend on time or energy required to fulfill some task, or degree of success (distance from target etc.).

The script is built of "procedures" assigned to system events. These include the following events.

- `onExpDefLoad` – occurs after experiment definition was loaded. This procedure should prepare the environment, create gene pools and populations.
- `onExpInit` – occurs at the beginning of the experiment.
- `onExpSave` – occurs on save experiment request.
- `onExpLoad` – occurs on load experiment request. After this event, system state should resemble the state before onExpSave.
- `onStep` – occurs in each simulation step.
- `onBorn` – occurs when a new organism is created in the world
- `onKill` – occurs when a creature is removed from the world
- `on[X]Collision` – occurs when an object of group [X] has touched some other object.

Thus a researcher may define the behavior of the whole system by implementing appropriate actions within these events. A single script (experiment definition) may use parameters, so it usually allows to perform a whole bunch (class) of diversified experiments.

*3.3 Illustrative example (standard experiment definition)*

The file "standard.expdef" contains the full source for the script used to optimize creatures on a steady-state basis, with fitness defined as a weighted sum of their performances (see also Figure 6). This script is quite versatile and complex. Below its general idea is explained, with much simplified actions assigned to system events:

`:onExpDefLoad`

- create single gene pool "Genotypes",
- create two populations "Creatures" and "Food"

`:onExpInit`

- empty all gene pools and populations
- place the beginning genotype in "Genotypes"

`:onStep`

- if too little food: create new object in "Food"
- if too few organisms: select a parent from "Genotypes"; mutate, crossover, or copy it. From the resulting genotype create an individual in "Creatures"

`:onBorn`

- move new object into a randomly chosen place in the world
- set starting energy according to object's type

`:onKill`

- if "Creatures" object died, save its performance in "Genotypes" (possibly creating a new genotype). If there are too many genotypes in "Genotypes", remove one.

`:onFoodCollision`

- send energy portion from "Food" object to "Creature" object.

## 4. Additional tools

Many research works concern studies of evolutionary processes, their dynamics and efficiency. Various measures and methods have been developed in order to be able to analyze evolution, complexity, and interaction in the observed systems. Other works try to understand behaviors of artificial creatures, regarding them as subjects of survey rather than black boxes with assigned fitness and performance.

Artificial life systems, especially those applied to evolutionary robotics and design (Bentley, 1999; Funes and Pollack, 1998; Lipson and Pollack, 2000), are quite complex and it is difficult to understand the behavior of existing agents in

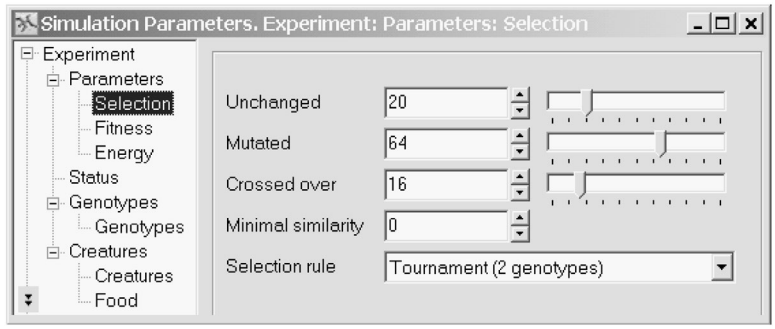


**Figure 6.**
A fragment of software window with experiment parameters

detail. The only way is to observe them carefully and use human intelligence to draw conclusions. Usually, the behavior of such agents is non-deterministic, and their control systems are sophisticated, often coupled with morphology and very strongly connected functionally (Lund *et al.*, 1997).

Thus for the purpose of studying behaviors and populations of individuals, one needs high-level, intelligent support tools (Komosiński and Rotaru-Varga, 2000). It is not likely that automatic tools will soon be able to produce understandable, non-trivial explanations of sophisticated artificial agents. Nonetheless, their role and help cannot be ignored. Even simple automatic support is of great relevance to a human, which becomes obvious after spending hours on investigating relatively simple artificial creatures. It may be that in the future some advanced analysis methods, developed within artificial life methodology, will be useful for real life studies, biology and medicine.

One of the main purposes of the Framsticks system is to allow creating and testing such tools and procedures, and to develop methodology needed for their use. Realistic artificial life environments are the right place for such research.

### *4.1 Clustering of similar individuals*

Similarity seems to be a simple property. However, automatic measures of similarity can help in observation of regularities, groups of related individuals, etc. Similarity can be identified in many ways, including aspects of morphology (body), brain, size, function, behavior, performance, fitness, etc. Whatever definition is used, automatic measure of similarity can be useful for:

- optimization to introduce artificial niches by modification of fitness values (Goldberg, 1989),
- studies of evolutionary processes and the structure of populations of individuals,
- studies of function/behavior of agents,
- reduction of the number of agents to a small subset of interesting, diverse, unique individuals,
- inferring dendrograms (and hopefully, phylogenetic trees) based on distances between organisms.

For Framsticks, we constructed a heuristic method that is able to estimate the degree of similarity of the two individuals. This method treats body as a graph (with material points as vertices and joints as edges), and then tries to match two body structures based on the degrees of vertices as the main piece of information. For a more detailed description of this method see Komosiński *et al.* (2001), and an example is presented in Section 5.

*4.2 History of evolution*

In real life, although we are able to trace genetic relationships within existing creatures, we do not know what happens during mutation and crossing over of their genomes. Moreover, we cannot trace genetic relations in a longer scale and higher number of individuals.

In Framsticks, it is possible not only to remember all parent-child relationships, but also to estimate genetic shares of related individuals (how many genes have mutated, or have been exchanged). This allows to draw a real tree of evolution, as shown in Figure 7. Vertical axis is time, horizontal one reflects local degree of genetic dissimilarity (between a pair of individuals). Vertices in the tree are single individuals.

*4.3 Fuzzy control*

Fuzzy control, nowadays, has become a very popular method of controlling in many domains of our life: washing machines, video cameras, ABS, air-condition, etc. It is also used for controlling non-linear, fast-changing processes, where quick decisions are more important than exact ones (Yager and Filev, 1994). Fuzzy control

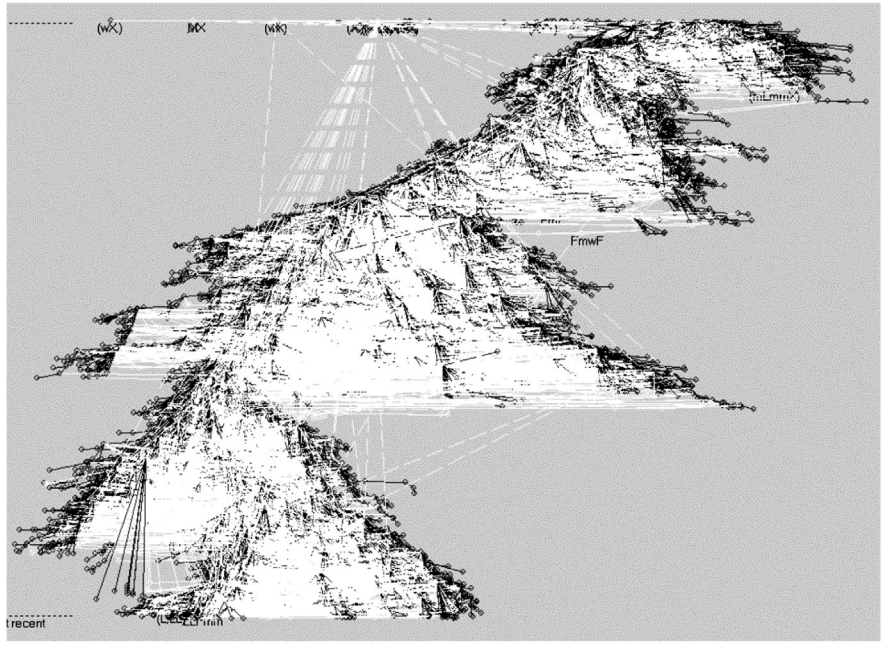


**Figure 7.**
The real tree of evolution. Top: single ancestor and beginning of time. Black lines represent mutations, white ones are crossovers

- allows for linguistic variables. "Drive fast" is much more flexible than "drive with speed 80-100 km/h". In fuzzy approach we would say: 79 km/h is still fast with degree 95 percent
- better comprehension for humans – fuzzy rule: *if X is Big and Y is Small then Z is Medium* is easier to understand than crispy one: *if X is between 32.22 and 43.32 and Y is less than 5.2 then Z is 19.2*
- more fluent, "natural" change of system states – for example, gradual slowing down of a car.

For such reasons, fuzzy control is developed in Framsticks. Therefore, evolved Framsticks' brains may be more understandable for a human. Fuzzy control, similarly to neural networks, can also cope with uncertainty of information – when the process is compound, depends on lottery events or the measurement is burden with error. Fuzzy approach can manage this inconvenience by generalization of information. To incorporate this approach in Framsticks neural network model, three additional types of neurons are needed:

- *difference neuron* – to compare current value of the sensor to the previous one,
- *rule neuron* – to represent a fuzzy rule,
- *aggregate neuron* – to gather answers of the rule neurons and calculate crisp output.

The two sample fuzzy rules are:

IF gyroscope is big+ AND Δgyroscope is small+

THEN bend muscle medium −

IF gyroscope is medium − AND Δgyroscope is zero+

THEN bend muscle big+

where gyroscope is the measurement taken from the gyroscope receptor, Δgyroscope is the change in its value, and linguistic terms "big+" etc. correspond to fuzzy intervals of values. The representation of these two rules in the Framsticks neural network is shown in Figure 8.

### *4.4 Helper software*

Together with continuous development of extensions to the Framsticks system itself, there are some external tools created. *Fred* is a visual Framsticks Editor developed in JAVA, which allows to design a creature (or a structure) in a user-friendly way. *Framsticks Experimentation Center* is an Internet database of genotypes and experiment definitions/proposals, which can be browsed, downloaded and uploaded.

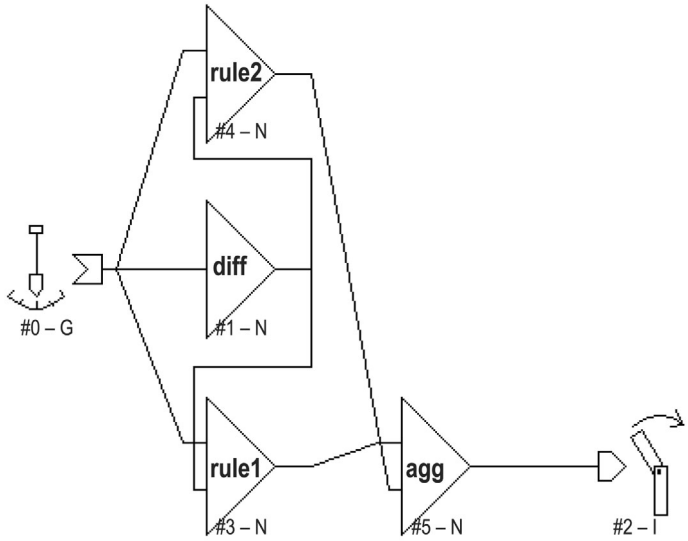


**Figure 8.**
Two fuzzy rules encoded as a Framsticks neural network

## 5. Research experiments

### *5.1 Comparison of genotype encodings*

There are a number of studies of the evolution of simulated creatures equipped with realistic physical behavior. In such systems, the use of a physical simulation layer implements a complex genotype-fitness relationship. Physical interactions between body parts, the coupling between control and physical body, and interactions during body development can all add a level of indirection between the genotype and fitness. The complexity of the genotype-fitness relationship offers a potential for rich evolutionary dynamics.

The most important element of the genotype-to-fitness relationship is the genotype-to-phenotype mapping, or genotype encoding. There is no obvious simple way to encode a complex phenotype – which consists of a variable-size, structured body and a matching control system – into a simpler genotype. Moreover, an evolutionary algorithm can perform poorly when using a certain genotype encoding, and better when using others, for reasons not yet immediately obvious. The employed genotype encoding can have a significant effect on the performance of the evolution.

The *Framsticks* system has been used as the context of analysis of various genotype encodings. The performance of the three encodings described in Section 3 was compared in their optimization tasks: passive and active height, and velocity maximization (Komosiński and Rotaru-Varga, 2001). The solutions produced by evolution are considered to be successful for the given tasks in all three cases. However, there were some important differences in the degree of success. The ***f0*** encoding performed worse than the two higher-level

encodings. The most important differences between these encodings are that ***f*0** has a minimal bias and is unrestrictive, while the higher-level encodings (***f*1** and ***f*4**) restrict the search space, and introduce a strong bias towards structured phenotypes. Based on these results, we conclude that a more structured genotype encoding, with genetic operators working on a higher level, is beneficial in the evolution of 3D agents. The presence of a bias towards structured phenotypes can overcome the apparent limitation that entire regions of the search space are not accessible by the search. This bias may be useful in some applications (engineering and robotics, for example). The significant influence of a chosen encoding can be clearly seen in the obtained agents: those with ***f*0** encoding displayed neither order nor structure. The two encodings restricting morphology to a tree produced more clear constructions, and for developmental encoding segmentation and modularity could be observed (Figure 9).

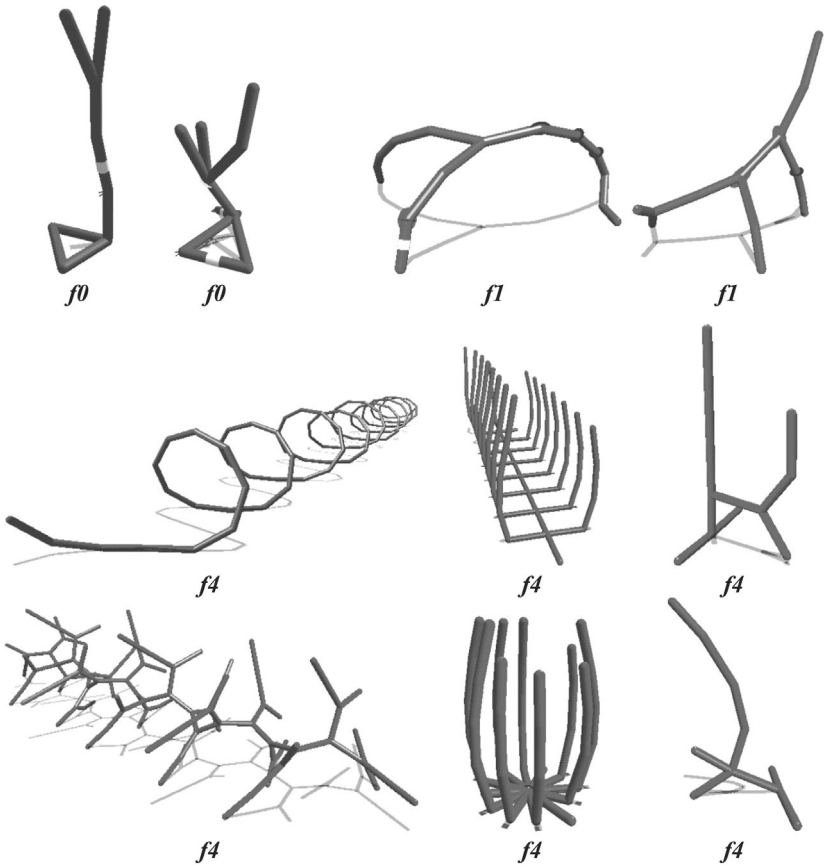


**Figure 9.**
Representative agents for various genetic encodings and height maximization task

*5.2 Automatic optimization versus human design*

Designing agents by hand is a very complex process, in professional applications it requires planning and extensive knowledge about how the control system, receptors, and effectors work, as well as knowledge about the simulator. Designing neural networks for control by hand is especially difficult and tedious. For this reason, human-built agents usually have lower fitness than agents produced by evolution. However, human creations are often interesting qualitatively. Human designs have such properties as explicit purpose, elegance, simplicity (minimum of means), and often symmetry and modularity. These features are opposed to evolutionary results, which are characterized by hidden purpose, complexity, implicit and very strong interdependencies between parts, as well as redundancy and randomness (Komosiński and Rotaru-Varga, 2001).

The difficult process of designing neural networks can be circumvented by a hybrid solution: bodies can be hand-constructed, and control structures evolved for it. This approach can yield interesting creatures (Adamatzky, 2000; Komosiński, 2000; Komosiński and Rotaru-Varga, 2000; Komosiński and Ulatowski, 1997), often resembling in behavior creatures found in nature (Ijspeert, 2000).

*5.3 Clustering with similarity measure*

The similarity measure outlined in Section 4 allowed us to perform various experiments (Komosiński *et al.*, 2001). Figure 10 shows the results of application of UPGMA clustering method to ten individuals from the height maximization task. The figure shows their morphologies together with the clustering tree.

It can be seen that the three big organisms are in a single, distinct cluster. They are similar in size, but different in structure, so the distance in-between them is high. Moreover, the measure captured also functional similarity (hp_1, 3, 6, 9, 7) – all these agents have a single stick upwards and a similar base. The agents hp_0 and hp_4 are of medium size, but certainly closer to the small organisms group than to the big ones. They are also similar in structure; that is why they constitute a separate cluster.

The similarity measure is very helpful for study and analysis of groups of individuals. Manual work of classification of the agents shown on Figure 9 yielded similar results, but it was a very mundane and time-consuming process. It also lacked objectivism and accuracy, which are properties of the automatic procedure.

*5.4 Other possibilities*

Considering open architecture of the Framsticks system, many possibilities exist of defining diverse genetic representations, experimental setups, interaction rules, and environments (see Section 3). Most obvious ideas

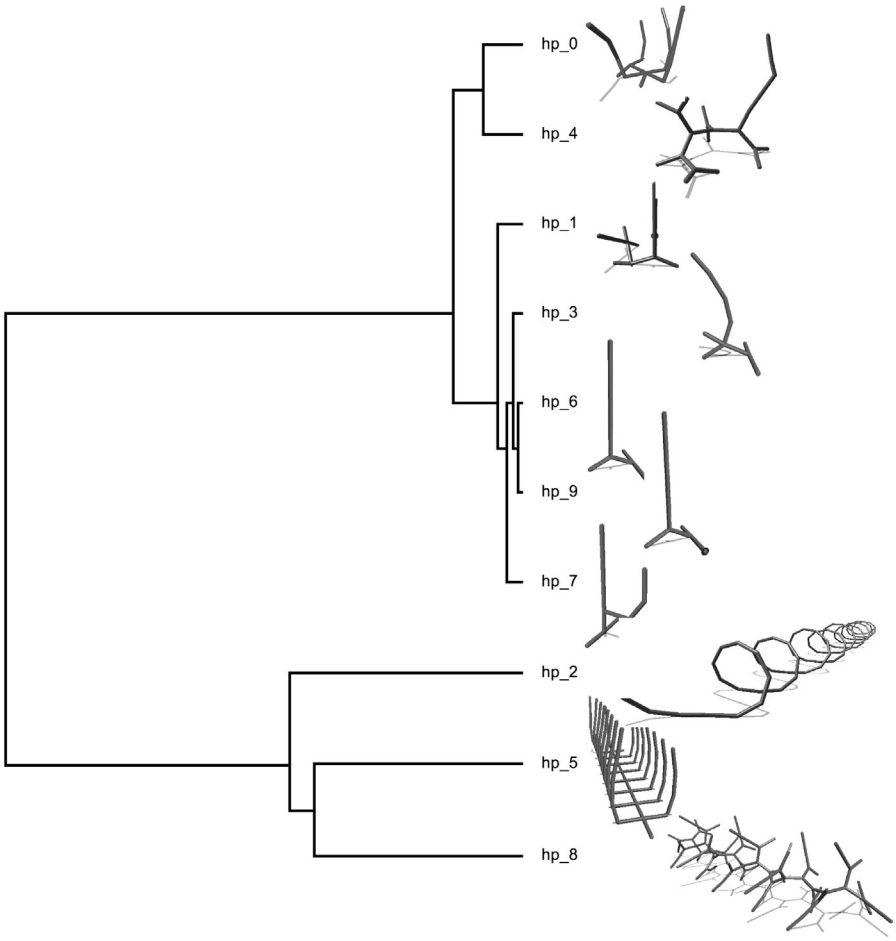


**Figure 10.**
The clustering tree for ten best individuals from the height maximization task

include coevolution of individuals in a population, predator/prey relationships, multiple gene pools and populations, and their specialization.

Some of the possible experiments related to biology include introducing geographical constraints, and then looking at the differences in clusters obtained after a given period of time, or studying two or more populations of much differing sizes. The latter, under geographical constraints, could be used to simulate and understand speciation.

Refer Mandik (2002a, b) for a number of interesting experiments regarding evolutionary and neuro-computational bases of the emergence of complex forms of cognition, and a discussion about semantics of evolved neural networks, perception, and memory.

## 6. Summary

This paper was devoted to Framsticks, a general tool for modeling, simulation, and optimization of virtual creatures. Although the system is versatile and complex, one should remember that it may be simplified when some features are not needed. For example, control systems can be neglected if only static structures are to be considered. Genetic encoding may only allow for two-dimensional structures if 3D is not required. Local optimization framework may be used if the task does not require evolutionary algorithms, etc.

Complexity is useless when it cannot be understood or used. This is why Framsticks software tries to present information in a human-friendly and clear way. It is not only employed in research experiments, but is also popular in education. The system is used by computer scientists, biologists, roboticists, and other scientists, but also by students and laypeople of various age.

### Note

1. However, there are plans to integrate a very accurate simulation engine into the system when it is required; such engines are used in some works (Taylor and Massey, 2001).

### References

Adamatzky, A. (2000), "Software review: Framsticks", *Kybernetes: The International Journal of Systems and Cybernetics*, Vol. 29 No. 9-10, pp. 1344-51.

Bentley, P. (1999), *Evolutionary Design by Computers*, Morgan Kaufmann.

Funes, P. and Pollack, J.B. (1998), "Evolutionary body building: adaptive physical designs for robots", *Artificial Life*, Vol. 4 No. 4, Autumn, pp. 337-57.

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

Gruau, F., Whitley, D. and Pyeatt, L. (1996), "A comparison between cellular encoding and direct encoding for genetic neural networks", in Koza, J.R., Goldberg, D.E., Fogel, D.B. and Riolo, R.R. (Eds), *Proceedings of the First Annual Conference*, Genetic Programming 1996 MIT Press, Cambridge, MA, pp. 81-9.

Ijspeert, A.J. (2000), "A 3-D biomechanical model of the salamander", in Heudin, J.-C. (Ed.), *Proceedings of 2nd International Conference on Virtual Worlds (VW2000), LNAI 1834*, July 2000, Springer-Verlag, Paris, France, pp. 225-34.

Komosiński, M. (2000), "The world of Framsticks: simulation, evolution, interaction", in Heudin, J.-C. (Ed.), *Virtual Worlds. Lecture Notes in Artificial Intelligence 1834*, Springer-Verlag, pp. 214-24.

Komosiński, M. and Rotaru-Varga, A. (2000), "From directed to open-ended evolution in a complex simulation model", in Bedau, M.A., McCaskill, J.S., Packard, N.H. and Rasmussen, S. (Eds), *Artificial Life VII*, MIT Press, pp. 293-9.

Komosiński, M. and Rotaru-Varga, A. (2001), "Comparison of different genotype encodings for simulated 3D agents", *Artificial Life Journal*, Vol. 7 No. 4 Fall, pp. 395-418.

Komosiński, M. and Ulatowski, S. (1997), Framsticks Web Site, http://www.frams.alife.pl

Komosiń, ski, M., Koczyk, G. and Kubiak, M. (2001), "On estimating similarity of artificial and real organisms", *Theory in Biosciences*, Vol. 120, pp. 271-86.

Lipson, H. and Pollack, J.B. (2000), "Automatic design and manufacture of robotic lifeforms", *Nature*, Vol. 406 No. 6799, pp. 974-8.

Lund, H.H., Hallam, J. and Lee, W.-P. (1997), "Evolving robot morphology", *Proceedings of IEEE 4th International Conference on Evolutionary Computation*, (Invited paper) IEEE Press, NJ.

Mandik, P. (2002a), "Synthetic neuroethology", *Metaphilosophy*, Vol. 33 No. 1/2, pp. 11-29, http://www.wpunj.edu/cohss/philosophy/faculty/mandik/papers/synthneur.pdf

Mandik, P. (2002b), "Varieties of representation in evolved and embodied neural networks", *William Paterson University Cognitive Science Technical Report* 2002-03. http://www.wpunj.edu/cohss/philosophy/faculty/mandik/papers/vreenn.pdf

Taylor, T. and Massey, C. (2001), "Recent developments in the evolution of morphologies and controllers for physically simulated creatures", *Artificial Life*, Vol. 7 No. 1, Winter, pp. 77-88.

Ulatowski, S. (2000), Framsticks GDK (Genotype Development Kit), http://www.frams.alife.pl/dev

Yager, R.R. and Filev, D.P. (1994), *Foundations of Fuzzy Control*, Wiley, New York.