ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/2766574

On Genetic Algorithms and Lindenmayer Systems

Article *in* Lecture Notes in Computer Science · November 1998

DOI: 10.1007/BFb0056876 · Source: CiteSeer

CITATIONS
49

READS
247

1 author:

Gabriela Ochoa
University of Stirling
165 PUBLICATIONS **3,384** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project DAASE Project View project

All content following this page was uploaded by Gabriela Ochoa on 08 August 2013.

The user has requested enhancement of the downloaded file.

On Genetic Algorithms and Lindenmayer Systems

Gabriela Ochoa

COGS – School of Cognitive and Computing Sciences
The University of Sussex
Falmer, Brighton BN1 9QH, UK

Abstract. This paper describes a system for simulating the evolution of artificial 2D plant morphologies. Virtual plant genotypes are inspired by the mathematical formalism known as Lindenmayer systems (L-systems). The phenotypes are the branching structures resulting from the derivation and graphic interpretation of the genotypes. Evolution is simulated using a genetic algorithm with a fitness function inspired by current evolutionary hypotheses concerning the factors that have had the greatest effect on plant evolution. The system also provides interactive selection, allowing the user to direct simulated evolution towards preferred phenotypes. Simulation results demonstrate many interesting structures, suggesting that artificial evolution constitutes a powerful tool for (1) exploring the large, complex space of branching structures found in nature, and (2) generating novel ones. Finally, we emphasize that Lindenmayer systems constitute a highly suitable encoding for artificial evolution studies.

0. Introduction

Natural computation techniques can play a vital role in computer graphics, animation, and virtual reality. Several models have been developed that realistically emulate a broad variety of living beings, from lower organisms all the way up the evolutionary ladder to humans (Dawkins, 1986; Oppenheimer, 1986; Sims 1991, 1994; Reynolds 1987). In Particular, the work of Karl Sims (1991) illustrates the potential of artificial evolution as a tool for the creation of procedurally generated structures, textures and motions. Evolution turns out to be a method for exploring and creating complexity that does not require human understanding of the very last details of the process involved.

Little work has been done in computer-simulated evolution of plants. Among the few is the work of the botanist Karl J. Niklas (1985, 1988, 1997). Niklas' model aims to simulate the evolution of branching patterns in early land plants. Firstly, he stated some specific hypotheses concerning plant evolution, then developed mathematical techniques for quantifying the hypothesized competitive advantages offered by various features. To encode a plant branching pattern, three parameters or characteristics were used: (1) probability of branching, (2) branching angle and (3) rotation angle. Using these characteristics, plant growth is simulated through several branching cycles. For modeling evolution, a deterministic scheme for searching among the nearest neighbors in the tree-parameter space is employed. The fittest of the explored neighbors becomes the starting point for the next search cycle. This process is reiterated until the computer has identified a set of morphological characteristics that is more efficient than any immediate neighbor in the search space. Niklas' simulation model has some limitations. Clearly, the three parameters are an oversimplification of plant geometry. Many more factors may influence plant shape. Other limitations concern the way evolution is simulated. The search method proposed can easily get stuck in local minima. Furthermore, a single organism instead of a population is maintained, and sexual reproduction is not considered.

In this paper we describe a novel use of genetic algorithms and Lindenmayer systems with the aim of evolving artificial plant morphologies. The model described simulates the evolution of 2D plant morphologies. Virtual plant genotypes are inspired by the mathematical formalism known as Lindenmayer systems (L-systems). The phenotypes are the branching structures resulting from the derivation and graphic interpretation of the genotypes. The system allows for two types of artificial evolution. Interactive selection, based on human perception of the plant-like structures, allows the user to direct simulated evolution towards preferred forms. Alternatively, automated evolution is simulated using a genetic algorithm with a fitness function inspired by current evolutionary hypotheses concerning the factors that have had the greatest effect on plant evolution.

Previous work has been done in combining artificial evolution and L-systems. Jacob (1994) presents the “Genetic L-systems Programming” (GLP) paradigm, a general framework for evolutionary creation and development of parallel rewriting systems, which demonstrates that these systems can be designed by evolutionary processes. However, Jacob’s example targets a somewhat simple problem: generate L-systems that form space constrained structures with a predefined number of branches. Here, we suggest a more complex fitness function with the aim of evolving structures that resemble natural plants.

Next section describes the formalism of L-systems and their graphic interpretation. Section 2 describes the proposed model: how L-systems are used as genetic encoding, the characteristics of the genetic algorithm employed, the genetic operators designed, and the fitness function's inspiration and design. Section 3 shows simulation results. Finally, section 4 discusses conclusions and suggests future work.

1. Lindenmayer Systems

L-systems are a mathematical formalism proposed by the biologist Aristid Lindenmayer in 1968 as a foundation for an axiomatic theory of biological development. More recently, L-systems have found several applications in computer graphics (Smith, 1987; Prusinkiewicz and Lindenmayer, 1990). Two principal areas include generation of fractals and realistic modeling of plants.

Central to L-systems, is the notion of rewriting, where the basic idea is to define complex objects by successively replacing parts of a simple object using a set of rewriting rules or productions. The rewriting can be carried out recursively. The most extensively studied and the best understood rewriting systems operate on character strings. Aristid Lindenmayer’s work introduced a new type of string rewriting mechanism, subsequently termed L-systems. The essential difference between the most known Chomsky grammars and L-systems lies in the method of applying productions. In Chomsky grammars productions are applied sequentially, whereas in L-systems they are applied in parallel, replacing simultaneously all letters in a given word. This difference reflects the biological motivation of L-systems. Productions are intended to capture cell divisions in multicellular organisms, where many divisions may occur at the same time.

1.1. D0L-systems

In this section, we introduce the simplest class of L-systems, termed *D0L-systems* (Deterministic and context free). To provide an intuitive understanding of the main idea, let us consider the example given by Prusinkiewicz et al. (1990). See Fig. 1.

> Lets us consider strings built of two letters *a* and *b* (they may occur many times in a string). For each letter we specify a rewriting rule. The rule $a \rightarrow ab$ means that the letter *a* is to be replaced by the string *ab*, and the rule $b \rightarrow a$ means that the letter *b* is to be replaced by *a*. The rewriting process starts from a distinguished string called the axiom. Let us assume that it consist of a single letter *b*. In the first derivation step (the first step of rewriting) the axiom *b* is replaced by *a* using production $b \rightarrow a$. In the second step *a* is replaced by *ab* using the production $a \rightarrow ab$. The word *ab* consist of two letters, both of which are simultaneously replaced in the next derivation step. Thus, *a* is replaced by *ab* , *b* is replaced by *a*, and the string *aba* results. In a similar way (by the simultaneous replacement of all letters), the string *aba* yields *abaab* which in turn yields *abaababa,* then *abaababaabaab,* and so on.

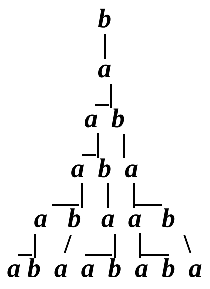

Fig. 1. Example of a derivation in a D0L-system.

1.2. Graphic interpretation of strings

Lindenmayer systems were conceived as a mathematical theory of development. Thus, geometric aspects were beyond the scope of the theory. Subsequently, several geometric interpretations of L-systems were proposed in order to turn them into a versatile tool for fractal and plant modeling. An interpretation based on turtle geometry, was proposed by Prusinkiewics et al. (1990). The basic idea of turtle interpretation is given below.

A state of the turtle is defined as a triplet *(x, y, α)*, where the Cartesian coordinates *(x, y)* represent the turtle's position, and the angle α, called the heading, is interpreted as the direction in which the turtle is facing. Given the step size *d* and the angle increment δ, the turtle can respond to the commands represented by the following symbols:

F Move forward a step of length d. The state of the turtle changes to (x', y', α), where x' = x + d cos α and y' = y + d sin α. . A line segment between points (x, y) and (x, y') is drawn.

f Move forwards a step of length d without drawing a line. The state of the turtle changes as above.

+ Turn left by angle δ. The next state of the turtle is (x, y,α +δ).

- Turn left by angle δ. The next state of the turtle is (x, y,α -δ).

To represent branching structures, the L-system alphabet is extended with two new symbols, '[' and ']', to delimit a branch. They are interpreted by the turtle as follows:

[Push the current state of the turtle onto a pushdown stack.

] Pop a state from the stack and make it the current state of the turtle.

Given a string *v*, the initial state of the turtle (x_0, y_0, α_0), and fixed parameters *d* and δ, the turtle interpretation of *v* is the figure (set of lines) drawn by the turtle in response to the string *v*. This description gives us a rigorous method for mapping strings to pictures, which may be applied to interpret strings generated by L-systems.

An example of a bracketed L-system and its turtle interpretation, obtained in derivations of length *n* = 1 - 4, is shown in Fig. 2. These figures were obtained by interpreting strings generated by the following L-system:

{*w: F, p: F* $\rightarrow$ *F[-F]F[+F][F]*}.

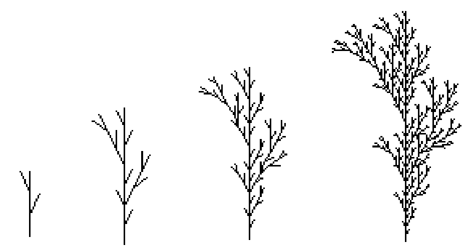

Fig. 2. Generating a plant-like structure.

2. The Model

Bracketed D0L-systems are used for encoding virtual organisms. A chromosome is constituted by a D0L-system with a single rewriting rule whose axiom (starting symbol) is always the symbol *F*. More precisely, the chromosome is the successor of the rule, there is no need to store the predecessor because it is always the symbol *F*. For example, the D0L-system showed in Fig. 2 is encoded as: *F[-F]F[+F][F]*. The phenotypes are the structures produced after deriving and interpreting the L-systems following the turtle graphic method.

2.1. Genetic Operations

When using a special genetic encoding for organisms, one must define special reproduction operations as well. These operations are often more elaborated than those in the canonical Genetic Algorithm. Our chromosomes have a well-defined syntactic structure stemming from the L-Systems formalism. To allow for a proper derivation and interpretation of genotypes, our genetic operations must produce offspring with valid syntactic structures. Three main operators were designed: crossover and two types of mutation.

- **Crossover**: The designed crossover is inspired by the Genetic Programming crossover operation (Koza, 1992). Koza's Lisp subtrees can be considered analogous to correctly bracketed substrings within an L-system. Fig. 3 shows an example of crossover. The hierarchical representation of the parents can be illustrated as:

```
F | +   |  - FF  |                          F   | + |    - FF  |    |   |
-FF  FFF   -F-F                               +F  -F-F     +F  -F  F
```

  where the underlined substrings are to be interchanged.

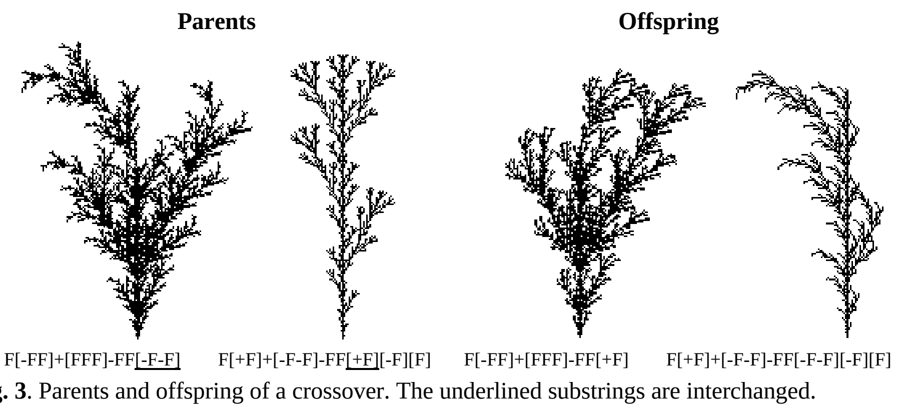

Fig. 3. Parents and offspring of a crossover. The underlined substrings are interchanged.

- **Mutation**: The mutation operation introduces random variations in structures of the population. Two types of mutation were designed. Each one acting on distinct parts of chromosomes (Fig. 4).

- **Symbol Mutation**: A randomly selected symbol of the chromosome in the set *{F, +, -}* is substituted by a random but syntactically correct string.
- **Block Mutation**: A randomly selected block in the chromosome is substituted by a random syntactically correct string.

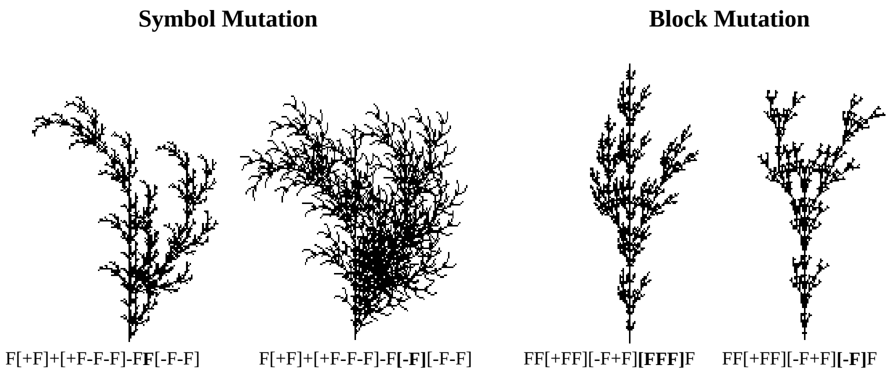

Fig. 4. Parent and offspring of the two types of mutation. The mutated segments are indicated in bold font.

2.2. The Genetic Algorithm

The implemented GA differs from the canonical GA (Goldberg, 1989) in several ways. Firstly, rather than binary fixed length string encoding, our genotypes are based on L-systems. They are of variable lengths and have a defined syntactic structure. Moreover, steady-state selection (Mitchell, 1996) is employed; only *1/5* of the population, the least fit individuals, are replaced in each generation. Given that several genetic operations were designed, a scheme of selecting operators for each reproductive event, according to given proportions, was also employed (Davis, 1991).

2.3. Fitness Function

Several researches modeling the evolution of morphological aspects in artificial organisms (Dawkins, 1986; Oppenheimer, 1986; Sims, 1991) have pointed out the difficulty of automatically measuring the aesthetic or functional success of simulated objects. It is trivial to select organisms according to a particular formula if you have access to all their genes. However, natural selection doesn't act directly upon genes, but rather upon their effects on organism bodies or phenotypes. The human eye is good at selecting phenotypic effects, but to construct computer programs that directly select phenotypic patterns is a difficult and sophisticated task. So, the usual practice is to rely on human perception as the selective pressure to evolve preferred forms.

Here, we pursue a higher degree of automation. So, the design of an adequate fitness function is necessary. In order to have a fitness function that indeed guides the simulated evolution towards structures resembling natural plants, we have to formulate hypotheses concerning the factors that have had the greatest effect on plant evolution. The hypotheses employed in our model are those formulated by Karl Niklas in his work (Niklas, 1985):

> [...] the majority of plants can be seen as structural solutions to constraints imposed by the biochemical process of photosynthesis. Plants with branching patterns that gather the most light can then be predicted to be the most successful. Consequently changes in the plant's shape or internal structure that increase its ability to gather light should confer competitive advantages.
>
> To be effective competitors for light and space, plants must perform certain other tasks. In particular they must be able to stay erect: to sustain the mechanical stresses involved in vertical growth. A second hypothesis, then, might be that evolution of plants was driven by the need to reconcile the ability to support vertical branching structures.

Thus, the designed fitness function is based on these hypotheses. To model the features there mentioned: light gathering ability, and structure stability, in an explicit analytic procedure, we constructed a function made of the following components: (a) phototropism (growth movement of plants in response to stimulus of light), (b) bilateral symmetry, (c) light gathering ability, (d) structural stability and (e) proportion of branching points. Simple algorithmic techniques have been developed for quantifying the competitive advantages offered by these features.

Selective pressures act upon phenotypes. So, before we can evaluate an organism, its encoding L-system must be derived and geometrically interpreted. Each component of the fitness function is quantified by a procedure that uses as input the geometric information produced while drawing the figure; and returns a real number between 0 and 1. A brief description of the procedures, is given below.

Let us consider a 2D Cartesian coordinate system, the origin of this system is the figure starting point. Each vertex in a figure will be represented as a pair *(x, y)*. The five features mentioned are quantified as follows:

- **Positive phototropism (a)**: High fitness is given to structures whose maximum *y* coordinate is 'high'. While low fitness s given to structures whose maximum y coordinate is 'low'. This is intended to force the "growth" of structures toward light. Furthermore tall structures are supposed to be better at disseminating seeds.
- **Bilateral Symmetry (b)**: The 'weight' balance of the structure is estimated. The absolute values of vertices' x coordinates at left and at right of the vertical axis are added up. Higher fitness is given to structures whose left to right ratio is closer to one, in other words, to better balanced structures.
- **Light gathering ability (c)**: The ability to gather light is estimated by quantifying the surface area of the plant 'leaves' --- ending segments --- exposed to light. The leaves exposed to light are those that are not shadowed by other leaves when we assume that the light rays are vertical lines from top to bottom.
- **Structural stability (d)**: The branches starting from each branching point in the structure are counted. It is assumed that branching points possessing too many branches are unstable. Thus, plants possessing a high proportion of this type of

nodes are rated low, while plants possessing a majority of 'stable' branching points are rated high.

- **Proportion of branching points (e)**: The total number of branching points with more than one branch leaving is calculated. This number is in direct proportion to the total number of branches in a structure. It is assumed that plants with a high number of branches are better at gathering light and disseminating seeds.

Weight parameters *(w_a, w_b, w_c, w_d, w_e)* are then used for tuning the effect of each component on the final fitness function:

$$F(Phenotype) = \frac{aw_a + bw_b + cwc_c + dw_d + ew_e}{w_a + w_b + w_c + w_d + w_e}$$

This fitness function takes us one step further in automating the selection of phenotypic traits. However, the human participation has not been eliminated altogether. Our model maintains the user determination of the fitness function weights.

3. Simulation Results

Many experiments were carried out. A typical experiment consisted of running the GA starting from a random generated population. The values used for GA parameters are: population size = 50, number of generations = 100, generation gap = 20 %, and chromosome length range = 7-30.

Distinct plant-like morphologies were obtained depending on the selected fitness function weights. Figures 5 and 6 show some of the fittest structures for different fitness function weights.

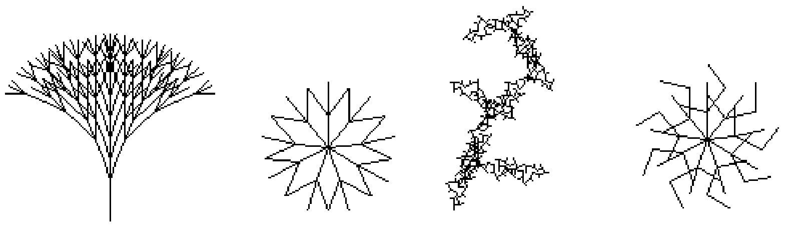
Fig. 5. Fitness function with weight values of 50 for all components

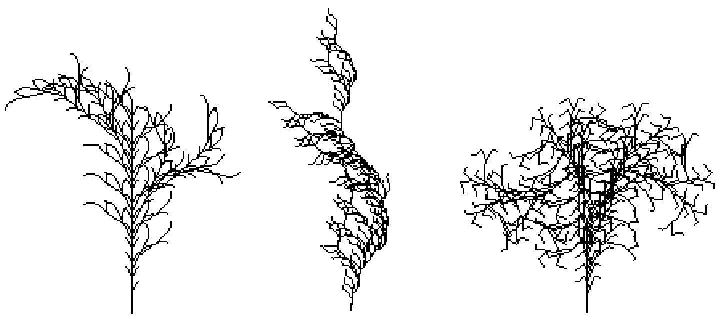
Fig. 6. Fitness function with component weights of: a = 100, b = 90, c = 40, d = 20, e = 30.

Finally, structures that resemble animals were also obtained with a fitness function favoring bilateral symmetric organisms (Fig. 7).

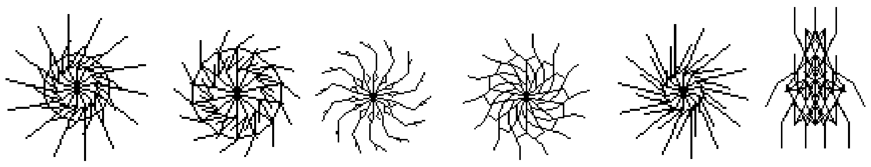
Fig. 7. Organisms obtained with fitness function favoring bilateral symmetric structures.

4. Discussion

A model has been described that can generate complex 2D branching structures without requiring cumbersome user specifications, design efforts, or knowledge of algorithmic details. We argue that L-Systems constitute an adequate genetic representation for studies which simulate natural morphological evolution. They allow the necessary, and very convenient, distinction between genotype and phenotype, and provide a well-defined process (*morphogenesis*) to generate the latter from the former. Moreover, they satisfy most of the important properties identified by Jefferson et al. (1991) for genetic encodings in biologically motivated studies. Among them: (a) L-systems provide a *simple, uniform model of computation*, because derivation and turtle interpretation of strings constitute a well defined way to go from genotypes to phenotypes; (b) they are *syntactically closed* under the designed genetic operations; and (c) they are *well conditioned* under genetic operators. This last requirement is not formally defined. Essentially, it requires that “small” mutational changes should (usually) cause “small” phenotypic changes, and that crossover usually produces offspring whose phenotypes are in some sense a “mixture” of the parents’ phenotypes, with occasional jumps and discontinuities.

The model has employed the simplest type of L-systems (D0L-systems). Further studies may be done using complex ones, considering, for example, genotypes with several rules, context sensitive L-systems, and inclusion of 3D morphologies

(Prusinkiewicz and Lindenmayer, 1990). Simulation results indicate that L-systems constitute a suitable encoding for artificial evolution studies. Thus, the evolution of other biological structures may be modeled using L-systems as genotypes. Finally, the model shows considerable creative power in generating novel and unexpected morphologies.

Acknowledgments. Thanks are due to J. A. Moreno for useful guiding and support. Many thanks, also, to A. Meier, H. Buxton and I. Harvey for valuable suggestions and critical reading.

References

1. Davis, L.: *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, (1991)
2. Dawkins, R.: *The Blind Watchmaker*. Harlow Longman, (1986)
3. Goldberg, D.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading (1989)
4. Jacob, C.: *Genetic L-system Programming*. Parallel Problem Solving from Nature III (PPSN'III), Lecture Notes in Computer Science, Vol. 866, Ed. Y. Davidor and P. Schwefel. Springer-Verlag, Berlin (1994) 334 - 343
5. Jefferson, D., Collins, R., Cooper, C., Dyer, M., Flowers, M., Korf, R., Taylor, C., Wang, A.: *Evolution as a Theme in Artificial Life: The Genesys / Tracker System*. In: Artificial Life II: Proceedings of the second workshop on the synthesis and simulation of living systems, Vol. X, SFI Studies in the Sciences of Complexity, Ed. C. Langton, C. Tylor, J. D. Farmer, and S. Rasmussen. Addison-Wesley, Redwood City (1991)
6. Koza, J.: *Genetic Programming: on the Programming of Computers by Means of Natural Selectio.* MIT Press, (1992)
7. Lindenmayer, A.: *Mathematical models for cellular interaction in development*. Parts I and II. Journal of Theoretical Biology, 18 (1968) 280-315.
8. Mitchell, M. 1996. *An Introduction to Genetic Algorithms*. MIT Press, (1996)
9. Niklas, K.: *Computer Simulated Plant Evolution*. Scientific American (May 1985), (1985)
10. Niklas, K.: *Biophysical limitations on plant form and evolution*. Plant Evolutionary Biology, Ed. L. D. Gottlieb and S. K. Jain. Chapman and Hall Ltd, (1988)
11. Niklas, K.: *The Evolutionary Biology of Plants*. The university of Chicago Press, Chicago (1997)
12. Oppenheimer, P.: *Real Time Design and Animation of Fractal Plants and Trees*. Proceedings of SIGGRAPH ‘86, in Computer Graphics. ACM SIGGRAPH, 20(4) (1986) 55-62
13. Prusinkiewics, P., Lindenmayer, A.: *The Algorithmic Beauty of Plants.* Springer-Verlag, (1990)
14. Reynolds, C.: *Flocks, Herds, and Schools: A Distributed Behavioral Model*. Proceedings of SIGGRAPH ’87, in Computer Graphics. ACM SIGGRAPH, 21(4) (1987) 25-34
15. Sims, K.: *Artificial Evolution for Computer Graphics*. Proceedings of SIGGRAPH ’91, in Computer Graphics. ACM SIGGRAPH, 25(4) (1991) 319-328
16. Sims, K.: *Evolving Virtual Creatures*. Proceedings of SIGGRAPH ’94, in Computer Graphics. ACM SIGGRAPH, 28(4) (1994) 15-22
17. Smith, A.: *Plants, Fractals, and Formal Languages*. Proceedings of SIGGRAPH ’84, in Computer Graphics. ACM SIGGRAPH, 18(4) (1984) 1-10

View publication stats