GREEN: A TOOL FOR MODELLING NATURAL ELEMENTS[1]

J.Lluch*, M.J.Vicent*, R. Vivó*

R. Quirós#

*Department of Sistemas Informáticos y Computación
University Politécnica de Valencia, Camino de Vera s/n
46022 Valencia
Spain
e-mail: jlluch@dsic.upv.es

#Department of Informática
University Jaume I, Campus de Riu Sec
12071 Castellón
Spain

ABSTRACT

In this article we present a prototype tool for the modelling of natural elements based on procedural modelling using Random L-systems, or RL-systems. RL-systems are L-systems which are extended using random variables, which permits us to obtain different individuals of the same species based on the same system. The proposed tool is flexible, easily modifiable and extensible due to its implementation using an object-oriented methodology which permits the incorporation of system modules with diverse functionality, such as the representation of specific parts of the element, simulation of tropism or pruning functions. Additionally, the proposed tool permits different interpretations to obtain different file formats depending on the desired viewer application, which permits the connection to other applications for modelling and visualisation of populations.

Keywords: image synthesis, procedural modelling, RL-systems.

1. INTRODUCTION

The utilisation of models permits the description of distinct aspects of a concrete or abstract entity. The function of a model is to permit the visualisation and the understanding of the structure and/or the behaviour of the entity being represented, as well as provide a medium within which to experiment with and predict the effects of distinct inputs or possible changes in the model.

In computer graphics a model should describe principally the geometry of the element being studied along with other properties such as its visual aspects or physical characteristics.

The work described here follows a line of investigation based on the modelling with RL-systems and its application to the creation of tree-like structures. RL-systems, as described in section 2, are framed within procedural modelling and, specifically, within rewriting systems. Procedural models allow us to represent natural phenomena which are difficult to represent as geometric models based on an explicit description of the objects using individual surface or volume primitives.

Green is a prototype of a natural element modeller based on Random L-systems (RL-systems) [Lluch93][Quiro96]. These systems are an extension of Parametric L-systems [Prusi90a], widely utilised to generate self-repeating natural structures, and to which we add random variables so

[1] Partially supported by CICYT TIC-0510-C02-01

that from a single system unique individuals may be generated. As in parametric L-systems, parametric strings, or a list of models formed by symbols with a series of associated parameters, are applied. The string of models resulting from the derivation process may then be interpreted in various manners to permit portability of the graphic specification of an individual to different description languages such as VRML, OpenInventor, POV, etc.

The principal objectives of creating the tool were the following:

- Unification of the initial software for RL-systems, utilising object-oriented design.
- Permit the connection to different visualisation standards.
- Provision of an easily-modifiable and extensible tool.

Although the proposed system may be used to model a wide variety of natural objects, the first and principal application is the modelling of tress and plants.

2. L-SYSTEMS

In 1976 Lindenmayer introduced parallel rewrite systems to describe the growth of living organisms [Linde76]. The theory related to these systems, called Lsystems, developed very quickly. However, it was not until 1984 that Aono and Kunii [Aono84] and Smith [Smith84] applied the theory to the generation of plants and trees. Almost at the same time Siromoney and Subramanian [Sirom83] combined L-systems and string codes to generate certain types of space-filling curves.

Even though L-systems a1low representation of a large class of self-repeating objects [Prusi86], their application is fairly limited due to their lack of support for time-varying representations. To avoid this problem, Lindenmayer proposed associating a set of parameters to the symbols of an L-system. The new system was thus called the parametric L-system and Prusinkiewicz and Hanan in [Prusi90b] first formalized it.

Parametric L-systems use parametric strings, which are lists of modules consisting of symbols with an associated set of parameters. Symbols belong to an alphabet V. Parameters belong to the domain of the real numbers. We denote by $A(a_1, a_2, ...,a_n)$ a module with symbol $A \in V$ and parameters $a_1, a_2, . . ., a_n \in \Re$, where $\Re$ is the set of the real numbers. Each module belongs to the set $M = V \times \Re^*$, where $\Re^*$ represents the set of all possible finite parameter sequences. The set of all possible module strings is thus denoted by M^*, with M^+ being the set of all possible non-empty module strings.

The actual parameters that appear in the strings match the set of formal parameters used in the specification of the system's production rules. If Σ is the set of formal parameters of the system then $C(\Sigma)$ denotes a logical expression on the elements of Σ and $E(\Sigma)$ denotes an arithmetic expression on the elements of Σ. Both expressions consist of numeric constants and formal parameters combined with arithmetic, relational, and logical operators, included functions and parenthesis. Finally, the empty logical expression is assumed to always evaluate to true.

If $C(\Sigma)$ and $E(\Sigma)$ represent the sets of all possible logical and arithmetic expression, respectively, then we define a parametric L-system by: G < V, Σ, ω, P >, where:

- V is the set of symbols,
- Σ, is the set of formal parameters,
- $\omega \in M^+$ is a non-empty string called axiom, and
- $P \subset (V \times \Sigma^*) \times C(\Sigma) \times (V \times E(\Sigma)^*)^*$ is a finite set of production rules.

We use the symbols ":" and "→" to separate the three components of a production rule: the predecessor, the condition (or guard) and the successor. A production matches a module of a parameter string if and only if: (i) the symbol of the module and the symbol of the predecessor of the rule are the same, (ii) the number of actual parameters of the module matches the number of formal parameters of the predecessor of the rule, and (iii) the condition of the rule evaluates to true when its formal parameters are substituted by the actual parameters of the module.

A matching production can then be applied to a module by generating the new string of modules according to the successor of the production rule. To do so, parameters in the successor are substituted in the same order as they appear in the predecessor. For example, the production:

$$A(t) : t > 5 \rightarrow B(t + 1)CD(t * 0.5)$$

can be applied to the module A(10.0) to generate a new module string B(11.0)C D(5.0).

Random L-SYSTEMS

The parametric L-systems we have described so far use deterministic substitution rules. There are some other applications, however, that require the use of non-deterministic rules, like simulating natural phenomena or modelling craft works. Non-deterministic perturbations add in these two cases substantial realism to the rendered images. Furthermore, sometimes it is crucial to be able to obtain irregular patterns for improved modelling power. These features can be added to our system as follows:

- allow the same symbol to be substituted in different ways by applying productions stochastically.
- permit local perturbations of a portion of the geometry obtained by a substitution (local rules).
- permit global deformations of the model using 3-D functions (global rules).

Yokomori [Yokom68], who associated a probability to each production of the L-system, first proposed stochastic application of productions. This way, different productions with the same predecessor have different probabilities of occurring. L-systems constructed that way are called stochastic L-systems. They were later reviewed and validated by Eichhorst and Savitch [Eichh80].

Unfortunately, stochastic L-systems are inadequate to simulate local perturbations in productions. One solution includes pseudo-random functions in the expressions used to, combine the parameters of the model to generate the local geometry [Prusi92]. This approach, however, does not allow easy tuning of the model, and it is not clearly based on the concept of random-variable. For these two reasons random L-systems were first introduced by Lluch et al. [Lluch93], and later applied to the generation of plants by Chover [Chove95].

Random L-systems include the features of parametric L-systems plus a set of random variables that take values according to a specific probability distribution. These variables are combined with the parameters associated with the symbols in the system in order to generate different strings according to a particular user-selected distribution function.

We define a random L-system to be a tuple $G = \langle V, \Sigma, \vartheta, \omega, P \rangle$, where ϑ represents a set of random variables. Each of the variables in ϑ can be a continuous or a discrete random variable. We allow the following distributions for the variables in S:

Continuous	Discrete
Constant	Constant
Uniform	Uniform
Exponential	Binomial
Normal or Gaussian	Poisson

Distributions for random variables
Table 1

Random variables are implemented as described in [Jones91]. When a module needs to be derived, random deviates are obtained for all variables in ϑ. Those deviates remain constant during the matching process of a production. Stochastic L-systems can thus be easily simulated by random L-systems.

Once a matching production is found, we obtain different deviates for all the occurrences of random variables in the successor of the production rule. This allows the generation of local perturbations. Hence we can summarize the advantages of random L-systems with respect to stochastic L-systems as follows**:**

- we can rewrite any stochastic L-system as a random L-system (the converse not necessarily being true),
- random L-systems are more versatile, since they allow us to define a much larger set of objects,
- production rules are more compact and, easier to specify using random L-systems, and
- we can generate local perturbations without resorting to external functions, since random variables are an integral part of a random L-system.

The syntax of the specification language of a random L-system is presented in [Quiro96]. The system includes the usual arithmetic and logic functions, plus other graphics specific functions to compute normal vectors, apply global deformations, *etc.* [Barr94]. To better control the application of production rules we use two variables *NumIt* and *MaxIt* that store, respectively, the current iteration number and the maximum number of iterations. Using these two variables we can define production rules to be applied only to certain generations of an object. For example, a production like:

$$A() : NumIt = MaxIt \rightarrow B()$$

will only be applied during the last iteration. This simple technique allows easy subdivision of an object's evolution into different phases.

Context-sensitive systems

In context-sensitive RL-systems the applicability of a rule depends of the context of its predecessor, so that a production would have the following format:

lc < pred > rc : cond → suc

where lc, pred and rc form the left context, the strict predecessor and the right context. The only required elements are the strict predecessor and the successor. For example, the subsequent L-system is composed of the axiom and three productions:

SYSTEM
DECLARATIONS
    CONTINUOUS p UNIFORM(0,1)
AXIOM
    A(1)B(3)A(5)
RULES
    A(x): p<0.4 → A(x+1)
    A(x): p>0.4 → B(x-1)
    A(x) < B(y) > A(z) : y<4 → B(x+z)[A(y)]
END.

The two first productions have a stochastic application, so that they substitute the module A(x) for A(x+1) or B(x-1) with a probability of 0.4 and 0.6 respectively. The final production is context-sensitive and will substitute the module B(y) with a left context A(x) and a right context A(z) for B(x+z)[A(y)], providing that y<4. Therefore, the first pass of the derivation would have the following result:

A(1)B(3)A(5) ⇒ A(2)B(6)[A(3)]B(4)

In this derivation it is assumed that as a result of applying a random election the first production has been applied to the module A(1) and the second to the module A(5). The third production is applied to the module B(3), because it has the required context and the condition 3 < 4 is met.

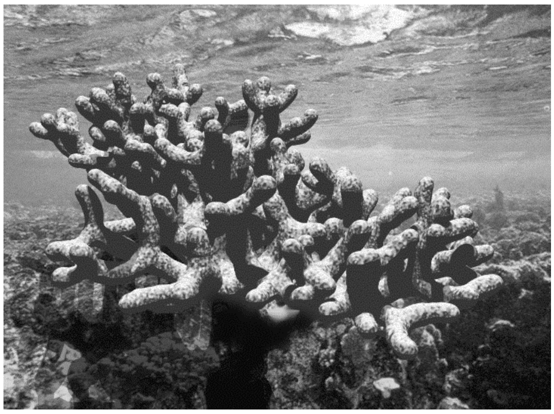
Interpretation of strings

The graphic interpretation of the strings resulting from the derivation process are realised using the 3D Turtle described in [Prusi90a]. However, when we consider the environmental conditions, the position and orientation of the turtle is important to the modelling of phenomena such as collision detection, exposition to light, or the pruning of the individual. The extension of L-systems to the sensitivity of the environment makes these parameters accessible during the derivation process [Prusi94]. The string which is generated is interpreted after each derivation pass and the attributes of the turtle resulting from the interpretation are returned as parameters of the reserved interrogation modules. Syntactically the interrogation modules take the form ?X(x,y,z), where X= P,H,U o L. Depending on the symbol X, the values of parameters x,y,z represent the position or an orientation vector.

Applications of RL-systems

RL-systems have a wide field of application in that they include L-systems, stochastic L-systems and parametric L-systems, so that any applications of these may also be realised with RL-systems. Therefore, one of the principal applications is the generation of vegetable species as shown in [Lluch93], where the architectures proposed by Hallé, Oldeman y Tomlinson were utilised, as well as for specific herbacious plants, with and without flowers [Quiro94].

The similarity which exists between the structure of certain species of coral and basic tree-like architectures permits their generation using an RL-system. The principal differences between a tree and a coral, regarding its structure, relates to the geometry of the basic primitives composing it [Quiro97].



Coral formation
Figure 1

RL-systems have also been utilised to obtain chaotic attractors [Quiro94] and in the generation of objects using the geometric substitution [Quir96a].

3. GREEN

Green is a prototype system, implemented in Smalltalk, for the modelling of objects based on context-sensitive RL-systems, permitting us to obtain, from an initial grammar, distinct individuals of the same species. To realise the modelling of an individual we begin with a string of initial modules (called axioms) upon which we apply a process of derivation resulting in a string of final modules to be interpreted by different viewers.

Description

The Green system has been designed utilising object-oriented methods, and programmed in Smalltalk to benefit from its versatility and the structure of its available classes. Green is a new class within the Smalltalk system, which in turn is composed of several subclasses. This structure of classes and associated methods facilitates the process of object modelling using RL-systems.

The principal characteristics of the system are the following:

- Includes the context-sensitivity of n symbols to the right and left.
- Permits the incorporation of diverse functions such as pruning, mathematical functions, noise functions, etc.
- Permits the utilisation of logical and arithmetic expressions for both the rules and the module parameters.
- Permits the creation of new modules which improve flexibility, control and performance of the derivation process
- Permits the realisation of operations on vectors and points
- Permits the determination at any moment, of the position and orientation of the turtle, via interrogation modules.

Structure of classes

The structure of the classes may be grouped into four sets. First we have the auxiliary subclasses, formed by:

- Collection: collection of objects of any class, supplied by the system.
- Vector: contains the elements of a vector.
- Turtle: composed of a stack of turtles forming a collection, the position and orientation (vectorH, vectorU y vectorL) of the turtle.
- Function: contains the subclasses of all the functions which define the behaviour of the random variables, the pruning functions, the complex mathematical functions, and whatever additional functions needed by the system.

Another set is formed by the following basic subclasses:

- Expression: contains a list of symbols (constants, random variables, operators and functions) that may be evaluated, and may return values.
- Module: formed by a name and a list of parameters which are expressions. Included are modules to which special behaviours may be assigned.

At the intermediate level are the following sets:

- String: a list of modules.
- Rule: contains a condition that is an expression and the predecessor and successor of the rule that are strings. The predecessor has a central module and a left and right context.

And finally, the highest level is formed by the following classes:

- RL System: contains an axiom that is a string, functions and random variables defined in the system and the rules from which it is formed.
- Deriver: contains the string of modules that are being derived, the context of derivation, and the RL System.
- Interpreter: formed by the turtle, the context of interpretation, and the string to be interpreted.

Because of the methodology used it is possible to incorporate additional modules at will. Presently a series of special modules have been developed, whose functionality permits the derivation of a single string in different manners, which has made Green quite a powerful tool.

This structure in classes also permits us to model different characteristics of trees and plants (or tree-like structures in general) including sensitivity to the environment, tropisms and pruning. The different modules implemented may be classified as follows:

- Those which act on the state of the turtle (for example turns or advances),
- Those which define the geometry of the different parts of the object (such as cylinders or spheres),
- Those which interrogate the state of the turtle (such as position),
- Those which allow control of the actual derivation; among these include the modules *Switch*, *Pendulum*, *Repetition* or *Prune*.

The module Switch utilises a list of modules as parameters. In each iteration it derives the next module in the list. Once the last of the list has been derived the module returns to the beginning.

The Pendulum module permits that a list of modules is derived directly or inversely during each iteration.

The Repetition module allows the derivation of a string the number of times indicated in a parameter, which may be a random variable.

Finally the Prune module eliminates from the derivation string all those modules which pertain to the branch to be pruned.

Flow of execution

The process of execution begins from an RL-system, which contains the declaration of random variables to be used, the initial sting (axiom) and the production rules utilised in the derivation pass. This system will define the geometry of the generated object.

First the system file is read and the syntax is validated. Then the system is derived a determined number of times. In each derivation we begin with an initial string and we obtain an output string. In the first iteration the initial string is the system axiom, and in subsequent iterations the input string is the output of the previous iteration.

To derive a string we step through it module by module. For each module we check for a rule with which to make a *match*, that is, a rule with a predecessor central module which coincides with the module to derive, the left and right contexts match, and the condition evaluates to true. Otherwise, if no rule complying with these conditions is found, the module derives itself, and if the rule is then found the module to derive is substituted for the successor of the rule, substituting the formal parameters for the actual parameters in the output string.

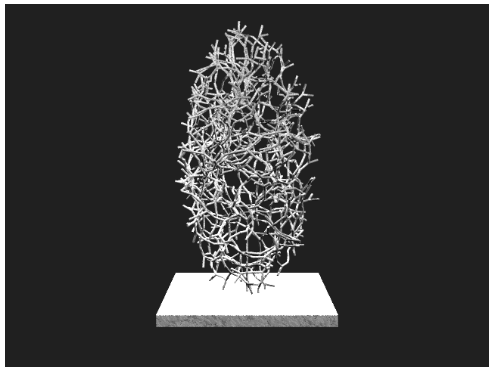
When the derivation process is completed, the resulting string is interpreted, that is, we step through module by module and they are interpreted using the system's turtle. During this process there are modules which are not interpretable, and therefore are ignored by the system. The interpreted modules create a scene file in an appropriate format depending on the desired viewer software: VRML, OpenInventor or POV.

Examples

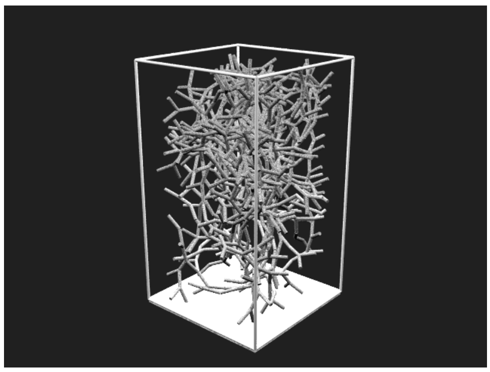
In the following section we show some of the objects obtained using the Green tool:

Ternary tree
Figure 2

Palm tree
Figure 3



Pruning against an ellipsoid
Figure 4



Pruning against a parallelepipedon
Figure 5

4. ALTERNATIVE TOOLS

In this section we compare well known tools such as Lparser [Lpars] and Xfrog [Linte96] [Linte98], with the Green tool introduced here. Both the Lparser and Xfrog are used for modelling plants and are based on rewriting techniques.

Xfrog

Regarding Xfrog we can say that it utilises a technique of object-oriented modelling in which the structural and geometric information, along with the necessary methods, are encapsulated in objects which combine to form a hierarchy that defines the model. The object hierarchy represents a free context system based on the production rules.

Xfrog permits the utilisation of recursion, geometric deformation of objects, parameters which may vary randomly within an interval, simulate tropisms and pruning as well as model exceptions. Thus, the Xfrog is a powerful tool for creating a wide range of different object types.

Lparser

The Lparser system creates models based on rewriting, specifically L-systems. The application reads a file with a description of an L-system and it processes it to create a string of symbols that can be interpreted in different forms to obtain different file formats.

Lparser uses parameterised commands which in turn use L-system rules to permit manipulation of the orientation and movement of the turtle, of the parameters which define the angles, lengths and thickness, and may even apply forces of gravity to the branches or cause mutations during the plant's growth.

In Table 2 we show a comparison of the characteristics of the Xfrog, Lparser and Green tools:

	Xfrog	Lpars	Green
Mutations	YES	YES	YES
Pruning	YES	YES	YES
Tropism	YES	YES	YES
Expressions	NO	NO	YES
Functions	NO	NO	YES
Operations with points and vectors	NO	NO	YES
Control of derivation	NO	NO	YES
O-O Programming	YES	YES	YES
Interactive modelling	YES	NO	NO
Context sensitivity	NO	NO	YES
Definition of view	YES	NO	NO
Definition of geometric primitives	YES	YES	YES
Properties of materials	YES	NO	YES
Depth of recursion	YES	YES	YES
Surfaces of revolution	YES	NO	NO
Free deformations	YES	NO	NO
Use of parameters	YES	YES	YES
Combination of recursive components	YES	YES	YES
Random parameters	YES	NO	YES
Modelling of exceptions	YES	NO	YES
Output in different formats	NO	YES	YES

Comparison of tools
Table 2

5. CONCLUSIONS

Our work has three fundamental parts. The first corresponds to the description of the RL-systems. The second corresponds to the description of the Green tool proposed here, and whose main characteristics are summarised below:

- Uses RL-systems to model natural elements
- Permits the creation of various interpretations of an object to obtain different file formats depending on the viewer software desired
- Permits the incorporation of sensitivity to the surroundings (pruning, tropism) and modules that permit us to control the derivation process
- Is a system which is easily modified and extended
- Uses object-oriented programming

Finally, we have describe two other, similar, tools for modelling plants , the *Lparser* and *Xfrog* tools, and can conclude that the Green tool presents the following improvements which makes it an ideal tool for the modelling of trees and plants:

- Permits the use of context-sensitive systems
- Permits the use of expressions composed of constants, variables and functions
- Permits us to associate conditions to the production rules
- Permits the creation of modules that control the derivation, such as Switch, Repetition or Pendulum.

6. FUTURE WORK

At this time the Green project is in the development phase. Some of the characteristics are being implemented at the time of this writing, and others which are to be implemented in the near future include the following:

- Creation of models from real objects
- Incorporation of modelling of different elements such as leaves, fruit, and flowers
- Application of a model of illumination for natural, exterior light sources
- Incorporate textures
- Simulate atmospheric effects, such as the effects of wind on trees
- Application of generated models for the interactive visualisation of populations using multi-resolution techniques.

REFERENCES

[Aono84] Aono, M. and Kunii, T. L., Botanical tree image generation. *IEEE Computer Graphics and Applications, 1984, 4, 10-34.*

[Barr94] Barr, A. H., Global and local deformations of solid primitives. *Computer Graphics (Proceedings of SIGGRAPH94),* 1994,18(3), 21-30.

[Chove95] Chover, M., Vivó, R., Quirós, R. and Lluch, J., Texture, Displacement and Immersion: A Model for Tree Rendering. In *Proceedings of the Winter School of Computer Graphics and Visualization, 1995 (WSCGV '95), Vol. 1,* 1995, pp. 69-78.

[Eichh80] Eichhorst, P. and Savitch, W. J., Growth functions of stochastic Lindenmayer Systems. *Information and Control,* 1980, 217-228.

[Jones91] Jones, H., Saupe, D., *Stochastic Methods and Natural Phenomena*, Eurographics'91 Tutorial Note 2, 1991

[Linde76] Lindenmayer, A. and Rozemberg, G., *Automata, Languages, Development,* North-Holland, *1976.*

[Linte96] Lintermann, B., Deussen, O.. *Interactive modelling and animation of natural branching estructures*. Computer Animation and simulation'96, pag. 139- 151. Springer-Verlag 1996.

[Linte98] Lintermann, B., Deussen, O., *A Modelling Method and User Interface for Creating Plants*. Computer Graphics forum 17(1), pag. 73-82. 1998.

[Lpars] Lparser. www.xs4all.nl/~ljlapre/lparser

[Lluch93] Lluch, J., Quiros, R., Castellanos, B., *Aplicaciones de los Sistemas Paramétricos Aleatorios a la generación de árboles y plantas herbáceas*, III Congreso Español de Informática Gráfica (Actas del Congreso), Granada, 1993, Junio, pág. 195-207

[Prusi86] Prusinkiewicz, P., *Graphical applications of L-systems*. In Proccedings of Graphics Interface86, 1986, pp. 247253.

[Prusi90a] Prusinkiewicz, P., Lindenmayer, A., *The algoritmic beauty of plants*. New York, De. Springer Verlag, 1990.

[Prusi90b] Prusinkiewicz, P. and Hanan, J., Visualization of botanical structures and processes using parametric L-systems. In *Scientific Visualization and Graphics Simulations,* ed. D. Thalmann. 1990, pp. 183-201.

[Prusi92] Prusinkiewicz, P. and Hanan, J., L-systems: from formalism to programming languages. In *Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology,* cds G. Rozenberg and A. Salomaa. Springer-Verlag, Berlin, 1992, pp. 193-212.

[Prusi94] P. Prusinkiewicz, M. James, R. Mech, "Synthetic Topiary", Computer Graphics, 1994, pp. 351-358.

[Quiro94] Quirós, R., *Modelado de especies vegetales mediante Sistemas-L Aleatorios*, Informe de investigación, C.E.R.F.A.C.S, Toulouse, 1994

[Quiro96] Quirós, R., Lluch, J., Chover, M., Vivó, R.., *Geometric substitution using random L-Systems.* Computer& Graphics, 1996, pag. 713-721.

[Quiro97] Quirós, R., Lluch, J., Vicent, M.J., Huerta, J., *Modelado y visualización de formaciones de coral*, Actas del VII Congreso Español de Informática Gráfica, Barcelona, 1997, Junio

[Sirom83] Siromoney, R. and Subramanian, K.G., Space-filling curves and infinite graphs. In *Graph Grammars and their Application to Computer Science, 2nd International Workshop,* Springer-Verlag, Berlin, *1983, pp. 380-391.*

[Smith84] Smith, A. R., Plants, fractals and formal languages. *Computer Graphics, 1984, 18(3), 1-10.*

[Yokom68] Yokomori, T., Stochastic characterizations of EOL languages. *Information and Control,* 1968, 280-315.