Animation based on the Interaction of L-systems with Vector Force Fields

Hansrudi Noser, Daniel Thalmann, Russell Turner

ABSTRACT

This paper discusses the use of rewriting systems for animation purpose. In particular, it describes the design of timed parameterized L-systems with conditional and pseudo-stochastic productions. It proposes a formulation to integrate the various features of L-systems into a unique L-system. It also introduces into the symbolism a way of using vector force fields to simulate interaction with the environment. The implementation, based on an object-oriented methodology, is described and animation examples are presented.

Keywords: rewriting systems, L-systems, production rules, vector force field, animation

1. INTRODUCTION

One way of describing growth phenomena is the use of *rewriting* mechanisms. The technique basically consists of starting with an initial shape or configuration and successively replacing parts using a set of rewriting rules. Several approaches have been proposed: fractals, cellular automata and L-systems.

1.1. Fractals

One of the first examples of mathematical objects defined by a rewriting process is the Koch curve (von Koch 1905), which is obtained by recursively scaling an original curve and using the reduced pattern to replace the middle third portion of the curve. Such objects have been intensively studied by Mandelbrot (1975, 1977, 1982) to define the concept of fractals. Fractal geometry has been introduced by Mandelbrot to describe complex natural phenomena such as coastlines, mountains, galaxies or clouds. He shows that objects constructed in his geometry, called *fractal objects*, can represent the shape of mountains, the pattern of stars in the sky and fluid networks in the organism. Fractal-geometry has become a new field of mathematics and it seems to provide a clearer view of much of our natural universe. As a mathematical model for random fractals, Mandelbrot and Van Ness (1968) have introduced *fractional Brownian motions* (*fBm*), which is an extension of the well-known concept of Brownian motion used in physics. fBm denotes a family of one-dimensional Gaussian stochastic processes, providing useful models for natural time series. Mandelbrot uses these models to approximate a wide range of natural phenomena such as terrains and landscapes. However, for producing numerous frames as in computer animation, the direct application of this theory based on fractional Brownian motion (fBm) is very expensive. For this reason, Fournier et al. [1982] introduce a simplification of the Mandelbrot methods. They propose a *recursive subdivision algorithm* to generate approximations of the sample path of one-dimensional Brownian motion. The method is much more efficient than previous methods; however, as pointed out by Mandelbrot, it sacrifices mathematical purity for execution speed in its approximation to fBm. This method corresponds to the construction of a "fractal polyline" primitive from an initial deterministic line segment. It recursively subdivides the interval and generates a scalar value at the midpoint. This value is taken as a displacement of the midpoint at each step in the recursion and is used as an offset from that midpoint along a vector normal to the original segment.

To model plants and trees, Demko et al. (1985) introduced fractals based on *iterated function systems*. An IFS on a k-dimensional space is defined by a set of maps and a set of probabilities. A random walk in the k-dimensional space is then defined using these maps and probabilities. Each IFS has a unique attractor, which is typically a fractal set. More formally it is the set theoretical union of the images of itself under the functions in the IFS. Intuitively, this attractor is the set about which the random walk eventually clusters. Demko et al. also introduce a P-balanced measure of the IFS which quantifies this clustering by ascribing a sense of density to the attractor.

1.2. Cellular automata

Conway's popular Game of Life (Gardner 1971, 1983) is in fact a trivial and well-known application of a more general theory called the *theory of cellular automata* (Wolfram 1984). Game of Life provides a good example for explaining the conceptual framework of the theory. John von Neumann (1966) originally developed the theory of cellular automata. He used it to prove the capacity of a machine to reproduce itself. This was, however, only a specific component of what would have been a more general theory of complex natural and artificial automata. A *cellular automaton* (Demongeot et al. 1985; Lobry and Reder 1985) is a cellular space or set of cells with the following properties:

1. The cells are regularly distributed in an N-dimensional space
2. Each cell can possess, in a given generation, a state chosen from a finite set
3. A configuration of the cellular automaton is defined as the set of states of all the cells in a given generation
4. The state of a cell in a generation depends exclusively on the states of the neighboring cells
5. A transition function defines the state of a cell from the states of those in its neighborhood in the previous generation

An important feature of cellular automata is that all state transitions occur at the same time in discrete time steps, forming successive generations as the model evolves. The aim of the game is to find an initial configuration which evolves over many generations. Cellular automata can serve as explicit models for a wide variety of biological, chemical, and physical processes: biological pattern formation, snowflake formation, turbulent fluids.

Thalmann (1986) introduced *analog cellular automata* as homogeneous spatial organizations in one, two, or three dimensions of interconnected finite state automata. The state of each automaton is characterized by the use of real numbers that correspond to some physical characteristics of the system being modeled (position, velocity, color, surface normal). State transitions functions are defined on these real numbers and are applied to all automata at the same time using information from the state of neighboring automata. Each application of these rules forms a generation. To implement the model, we set up an initial configuration assigning a certain value to some automata. The rest of the configuration remains in a stable state. In computer graphics, analog automata may be easily implemented in structures corresponding to MxN polygonal meshes, such as ruled surfaces (cylinders, revolution surfaces) and free-form parametric surfaces (Coons, Bézier, or ß-spline surfaces). One application of cellular automata to represent a natural system involves the modeling of a water surface.

1.3. L-systems

Lindenmayer (1968) showed that the branching patterns of primitive filamentary organisms can be formalized using his extended L-systems (DOL systems). A description of L-systems and DOL-systems has been summarized by Aono and Kunii (1984). L-systems are similar to Chomsky (1957) grammars, except that rules are applied in parallel to all symbols in the current state. The original DOL-system is defined as a triplet G = <V,w,P> where V is a alphabet, w the axiom or starting symbol, and P a set of production rules. The rules are applied to give shapes to a set of generated strings generated for either type of branching pattern. Each alphanumeric character of the generated string denotes a filamentary cell (represented by a line segment) of different length, while a pair of parentheses denotes a branch and its direction. Aono and Kunii proposed four new geometric models (GMT1, GMT2, GMT3, and GMT4) to generate botanical tree images. They analyze and compare their models with the L-system approach. With the help of this approach, they proposed the A-system, which produces tree images with leaves, shadows, and shades.

Smith (1984) also described a modeling technique called *graftals*, which represents plants using Lindenmeyer systems. Graftals are based on an extension of the formal languages of symbol strings to the formal languages of labeled graphs. The main difficulties of the technique are how the replacement rules work and how to define connectivity before and after. Smith does not propose a general solution but shows how to solve the problem for very simple graphs.

In (Prusinkiewicz and Lindenmayer 1990), several kinds of L-systems are described using a LOGO-style turtle (Abelson and diSessa 1982):

- DOL-systems — as already mentioned, deterministic context-free L-systems are the simplest class of L-systems;
- Bracketed OL-systems: — a turtle stack allows branching structures
- Stochastic L-systems: — allows variations of L-structures by randomization of the turtle interpretation and the L-system. A function $\prod(p)$, called the probability distribution, maps the set of productions into the set of production probabilities.
- Parametric OL-systems: — parameters are associated with each L-system symbol; actual parameters should correspond to formal parameters used in the specification of L-system productions. The application of a production may depend on conditions
- Timed OL systems: — the derivation function allows continuous growing and development of L-structures (objects). In timed OL-systems, the derivation length is expressed with the total elapsed time since the beginning of the observation rather than a number of derivation steps.

- Braced L-systems: the use of a polygon stack allows the development of surfaces.

Production systems and L-grammars, as introduced by Lindenmayer are very powerful tools for creating images. They can really take advantage of the power of the computer to represent objects in 3D with any perspective. The creative work of the artist consists, in principle, of defining an axiom and production rules. From these initial data, the computer creates the corresponding images with a complexity only dependent on the number of times the productions are applied. The resulting images are generally aesthetically appealing, because they contain a form of order and regularity present in the nature. The theory of these grammars is strongly related to genetics and the growth of living beings. The theory of L-structures has been mainly used for the visualization of the development and growth of living organisms like plants (Prusinkiewicz and Lindenmayer 1990), trees (Aono and Kunii) and cells.

In this paper, we propose the use of these theories for animation purposes. The animator may create any realistic or abstract shapes, play with fascinating tree structures and generate any concept of growth and life development in the images. The creative work of the animator is performed in two different spaces: the formal space of grammars and the shape and color space.The grammar space can be changed by modifying the axiom and/or the production rules. The shape of color space can be alterd by changing the geometry and color values of terminal nodes. In addition, a number of global parameters can be adjusted to control aspects of both spaces. While modifying the shape and color space can greatly alter the physical appearance of a particular grammar, the real creativity is more in the design of production rules.

2. PARAMETRIC TIMED CONTEXT-FREE L-SYSTEMS

Our objective was the design of a system allowing an integration of various characteristic described by Prusinkiewicz and Lindenmayer which required first a modified formal description. Our system is based on the introduction of timed parametric context-free L-systems (or parametric tOL-systems) with conditional and pseudo-stochastic productions in interaction with vector force fields. All grammar symbols are parameterized by three parameters and three growth functions.

These parametric tOL-systems may be formalized as follows:

1. V an alphabet
2. C a set of constants
3. R^+ the set real positive numbers
4. R^3 the 3D space
5. Σ the set of formal parameters
6. $E(\Sigma)$ the arithmetic expressions in Polish notation
7. $C(\Sigma)$ the logical expressions in Polish notation
8. ω an axiom
9. P $P = \{p_{a,i} | a \in V, i \in N\}$ the set of productions
10. Π $p_{a,i} \rightarrow [o,1]$ a pseudo-statistical distribution with $\sum \pi(p_{a,i}) = 1$
11. $(a,x_0,...,x_6) \in V \times R^7$ parameterized symbol

The first parameter x0 represents time or symbol age. The interpretation of other parameters is not fixed. It generally depends on the application, which means it depends on the semantics given to the symbols by the visualization procedure.

Let

$P \in V \times R^7 \times C(\Sigma) \times \prod \times (V \times E(\Sigma)^7)^*$

$p_{a,i} \in P$

$$p_{a,i} : (a,\alpha,x_1,\ldots,x_6) \quad \xrightarrow[\prod (p_{a,i})]{\text{Cond}(\alpha,x_1,x_2,x_3,T)} \quad \begin{matrix} (a_1,f_{10},\ldots f_{16}) \\ . \\ . \\ . \\ (a_n,f_{n0},\ldots,f_{n6}) \end{matrix}$$

for a given i with $\sum_i \cong(p_{a,i}) = 1$.

α:	maximal symbol age
f_{jo}:	initial age of symbol a_j
x_1,x_2,x_3:	3 parameters
$f_{j0}+t$:	local age of the symbol
T:	global time
$f_{j1}\ldots f_{j3}$:	parameter expressions
$f_{j4}\ldots f_{j6}$:	growth functions
f_{jk}:	$f_{jk}(f_{j0},x_1,x_2,x_3,t,T)$ a conditional and pseudo-statistical production.

The derivation function D is defined by the following axioms:

The derivation function D:
$(V \times R^7)^* \times R \cong\cong\cong\cong\cong\cong> (V \times R^7)^*$

A_1: The Development of each symbol is independent of each other in time
$D((a_1,x_0,\ldots,x_6) \ldots (a_n,x_0,\ldots x_6),t) =$
$D((a_1,x_0,\ldots,x_6),t) \ldots D(a_n,x_0,\ldots x_6),t)$

A2: The Growth of a symbol before terminal age
If $x0+t < \alpha$
$D((a_1,x_0,\ldots,x_6),t) = (a,x_0+t,x_1,x_2,x_3,f_4,f_5,f_6)$
with $f_j = f_j(x_0+t,x_1,x_2,x_3,T)$, j=4,5,6
growth functions

A3: Application of stochastic production at terminal age
If $x0+t > \alpha$ and $\text{Cond}(\alpha,x_1,x_2,x_3,T)$ = TRUE with a probability $\cong(p_{a,i})$
$D((a,x_0,\ldots,x_6),t) =$
$D((a_1,f_{10},\ldots f_{13},x_4,x_5,x_6)$
.
.
$(a_n,f_{n0},\ldots,f_{n3},x_4,x_5,x_6),t - (\alpha-x_0))$

A4: Selection of random numbers for the productions
array_random_values [(x+y[x]) mod z]

with array_random_values = array of size z with uniform random values between 0 and 1
x = recursion depth of D
y[x]: position of the production in the derivation tree of D at the x depth
The values of x, y[x] are provided by D.

This recursive definition of the derivation function implies an implementation using recursivity. The nature of this derivation function is normally deterministic. To add a stochastic aspect, we introduce an array of random numbers. This array has a fixed size and it is initialized by a uniform random number generator, which should be itself initialized by a seed during the image specifications. The use of this array, which does not change during an animation, is explained by an example in Fig.1.

axiom: [z(0] + [z(0)] + [z(0)]
production: z(1) -> c z(0)

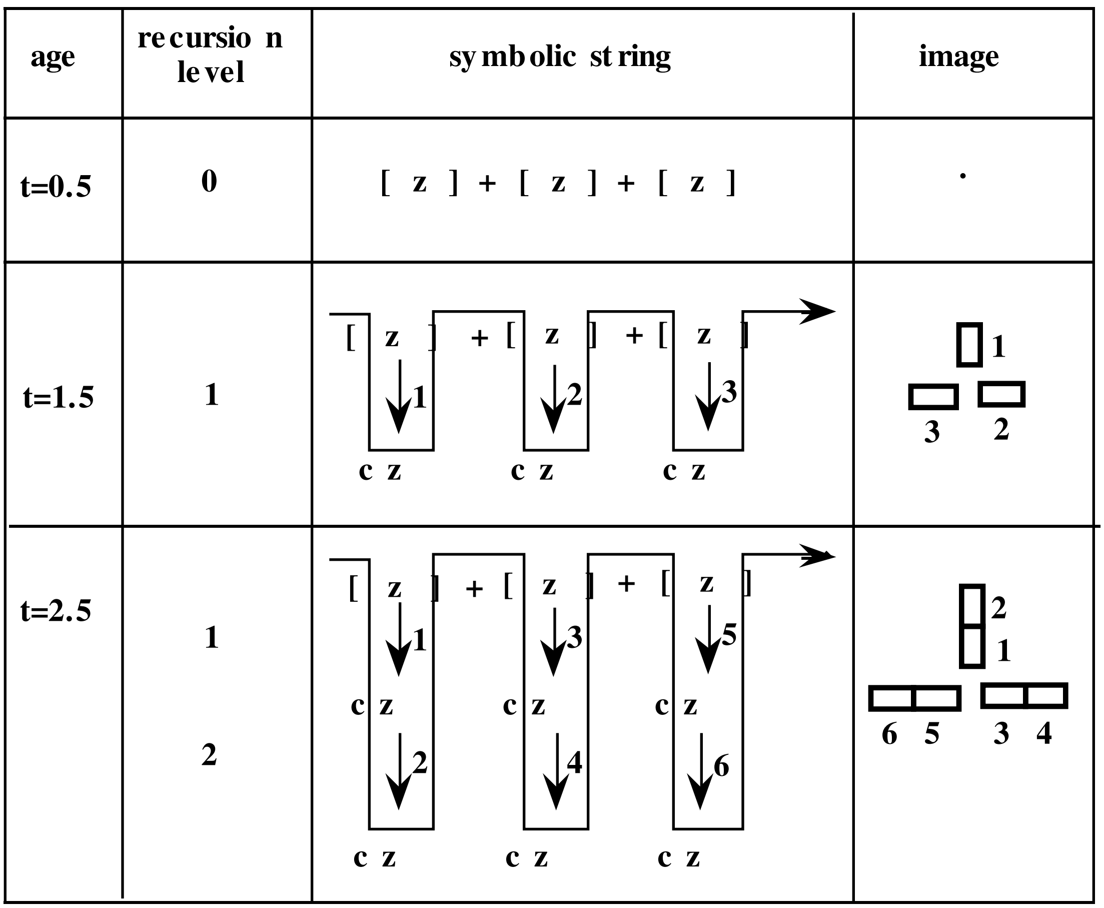

Fig.1. Mechanism of the derivation function D

In Figure 1, the symbol z is non-terminal, which means that it is not drawn or interpreted. The symbols '[' and ']' represent the turtle stack operations push and pop to save and read the turtle stack. The symbol '+' rotates the turtle around one of its axes. To simplify the presentation the parameter expressions and growth functions of the symbols are omitted. The symbol c corresponds for example to a cube or a rectangle. Its color is given by a probability distribution. If the symbol z reaches the age 1, the production z(1) -> c z(0) is applied. Fig.1 shows the development of the object at three different times. When t=0.5, there is no production (indicated by arrows) and the axiom is drawn, which correspond to nothing visible. When t=1.5, for each z, a production is applied in the order of numbering 1, 2 and 3. All symbols below the line are interpreted and we obtain the image consisting of three rectangles. A random number determining the color of the rectangle had been assigned to each production. When t=2.5, there were 6 productions in the temporal order 1,2,...,6. Again, everything below the line is interpreted and the result is an image with 6 rectangles. Moreover, a random number is again associated with each production. To ensure a continuity in the development, it is necessary for example that the production 2 at the time t=1.5 has the same associated random number as the production 3 at the time=2.5. As the order of productions at different times is not the same, it is not possible to use a sequential random generator producing a number for each production. In order to associate with each production the right random number, we use the fact that a production is determined by the depth and width of the recursion. With these two values, it is possible to calculate an index into an array of uniform random numbers. By using this method, which is contained in Axiom A4, the continuous growth of objects is ensured.

It should be noted that because of the exploration in depth of the derivation function, it is not possible to introduce a context dependence without storing the whole derivation tree.

Neither the formal object, nor the graphical object need to be stored completely in memory. Only the formal description of the current state of the derivation function needs to be maintained on a stack, resulting in memory requirements proportional to the recursion depth of the derivation function.

3. OBJECT DESCRIPTION

Objects in the system are defined using text descriptions. A description is composed of four parts:

1) Basic parameters including camera characteristics, object translation and rotation, seed for the random generator.

e.g. reference_point 0 0 0

2) Constants

A further parameterisation of the constants (of the expressions for the three parameters and the growth functions) by a global time allows the animation and deformation of L-structures (objects).

We introduced types of constants that vary linearly, periodically and exponentially with time.

3) The Axiom

This is defined by a sequence of operators followed by parameters. Some of the important operators used are the 3D turtle operators described in Section 3.4.

e.g. + () () () () (h) () ()	means a rotation by an angle h around the first axis. (h) represents the first growth function of the symbol. The initial age of the symbol is zero which is the value of the default expression '()'. The three parameters and the two other growth functions represented by the empty expression '()' are unused.

4) The Productions

They are also defined as a sequence of operators followed by parameters.

The most important operators used are the following 3D turtle operators as described by Prusinkiewicz and Lendenmayer (1990) and shown in Fig.2:

+ (turn left), **-** (turn right), **&** (pich down), **^** (pitch up), **$** (roll left), **/** (roll right),

f (forward)

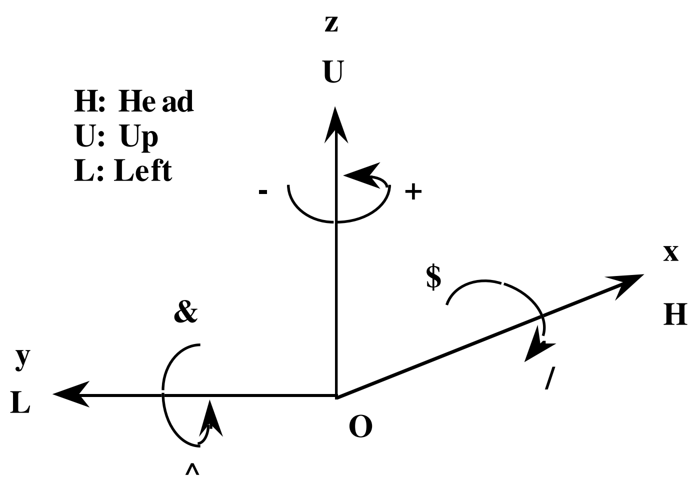

Fig.2. The 3D turtle operations

An operator **F** (detach) has been introduced, as explained in section 4, the effect is dependent on a flag MODE; two cases are possible, the turtle is linked to the initial position by invisible links or it is detached from this initial position and moves in 3D space according to given differential equations.

Two stack operations are also defined as in bracketed L-systems:

[push the current state of the turtle onto the stack

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

3D shapes may be also created using turtle operators: cubes, pyramids,cylinders, spheres. We have also introduced an operator B to import objects created by the SurfMan sculpting program (LeBlanc et al. 1991). Polygonal meshes may be also created using stack operations as defined by Prusinkiewicz and Lindenmayer (1990).

A certain number of additional features have been also added to the system:

each symbol has associated 3 growth functions as parameters. So every symbol can have several sets of growth functions in one production.

turtle variables have been introduced to increase the flexibility in object design.

Interaction with vector force fields have been also introduced. They are described in more details in the next section.

Animation of the observer, the reference point and the camera are carried out by a special symbol of the L-system. This symbol has to be placed at the beginning of the axiom.

There is a possibility to store not too complex images on files in a standard format and to reuse them in other image definitions with productions.

Figures 3 to 5 show examples.

Fig.3 a-b: Simple moving and growing object composed of cubes. At each iteration a cube is added at the end of a branch. Its length, width and height is controlled by the three growth functions.

Fig.4 a-b: Thin tree-like three-dimensional fractal object growing continuously and periodically bending its branches. The bending of all branches is easily achieved by a simple constant variation in the turtle rotation operator of a production.

Fig. 5 a-b: A whirling and growing group of snow flakes. Each snow flake is a three-dimensional version of "Koch's" snow flake and develops itself from a point to complicated fractal. At the final iteration, each snow flake produces the seeds for two new snow flakes.

4. EXTERNAL FORCES

External forces can also have an important impact in the evolution of objects. Prusinkiewicz and Lindenmayer have proposed two simple cases of external forces. In the first method, the 3D turtle may be easily aligned horizontal to a vector representing the gravity. The second case is specific to plant and tree modelling and allows the simulation of tropism, which is responsible for the bending of branches.

In our extended version, tree structures can be elastically deformed and animated by time and place dependent vector force fields. The elasticity of each articulation can be set individually by productions. So, the bending of branches, for example, can be made dependent of the branches thickness, which makes animation more realistic. The force fields too, can be set and modified with productions (i. e different forces for different parts of a tree structure). This interaction is based on tropism described by Lindenmayer with some modifications.

We introduced a third type of force that affect objects. It simulates the displacement of an object (or part of an object) in any vector field of forces dependent on time and position. The movement is determined by a class of differential equations which can be set and modified by productions. The mass of the turtle (representing the moving object) can be set by a special symbol in the productions. The displacement is determined by the following differential equations:

$$\ddot{x}=f_x(x,y,z,\dot{x},\dot{y},\dot{z},t,X,Y,Z)/m \qquad (1)$$

$$\ddot{y}=f_y(x,y,z,\dot{x},\dot{y},\dot{z},t,X,Y,Z)/m$$

$$\ddot{z}=f_z(x,y,z,\dot{x},\dot{y},\dot{z},t,X,Y,Z)/m$$

x,y,z: position
$\dot{x},\dot{y},\dot{z}$: velocity
X,Y,Z: initial position
m: mass

This vector field force approach is particularly convenient to simulate the motion of objects in fluids (air, water) as described by Wejchert and Haumann (1991). The fluid current is simulated using the force field. This field corresponds to a vector field of current velocities and should be the solution of Laplace equation to be realistic. The link between the fluid velocity and the force applied on a spherical object is given by the Stokes drag equation:

$$F = 6*\pi*a*\eta*v_r \qquad (2)$$

η: viscosity
a: particle's radius
v_r: relative velocity in the fluid

To solve the differential equation system, we use the Runge Kutta method. The initial conditions of position and velocity of the object are given when the field force is activated by the particular symbol F of the grammar. The position is defined by the turtle position, and the velocity is initialized according to three growth functions de F. The intial time is always 0 and corresponds to the initial age of the symbol F. The constants x,y,z (position), u,v,w (velocity), t (local age), T (global time) and X,Y,Z (starting position) are always updated before the evaluation of expressions. Movements are completely calculated from 0 until the actual age of the object. Therefore the moving object is always dependent on the starting point. If this point (e.g. the end of the branch of a tree) is moving, the motion of the object is added to the motion of the starting point.

It is also possible to specify the position and initial velocity when objects should be separated from the starting point (like an apple falling from an apple tree). In this case, the actual position and velocity of the detached and moving object have to be stored in an array or a hash table. The identification of such an object has to be made by the same method (explained in Section 2) as used for the selection of the pseudo-static numbers of the productions. With the actual depth and width of the recursion, a moving object can be identified and its index in the table can be calculated.

With our approach of external forces, the user may use a text editor and specify his own differential equation for a displacement, or his own force field for an elastic deformation in the case of an articulated substructure. Fig. 6 shows an example.

5. CASE STUDY: THE APPLE FALLING FROM THE APPLE TREE

An example of animation produced using our system contains a tree in a vector force field. Gravity and a periodic wind with a moving whirlwind are simulated. Fig.3 shows the force of gravity, which becomes zero on the floor (y=0) and changes of sign by incresing exponentially when y <0. Therefore if an apple falls from the tree, it jumps a little on the floor and stops according to the damp in the apple differential equation. The damping is also increasing when the floor is reached. The apple differential equation is as follows:

$$\frac{d^2y}{dt^2} = i(1-3^{-(y-2)}) - \frac{v}{8} / ((y-2)^4 + 0.01)$$

$$v = \frac{dy}{dt}$$

The moving whirlwind and the periodic wind are represented by the following formulas:

$$f_x = -z / ((x+50-1.8T)^2 + z^2 + 0.5) + a + b*\sin(cT)$$

$$f_z = (x+50-1.8T) / ((x+50-1.8T)^2 + z^2 + 0.5) + d + e*\sin(fT)$$

The leaves and the branches are moving according to all forces including gravity, but only gravity is applied to the falling apples.

Figures 6 to 8 show examples.

6. IMPLEMENTATION

The system has been implemented on IRIS 4D workstations using the C language with an object-oriented methodology. The modular structure and the interaction with the environment is shown in Fig.3. The performance graphics is based on the Silicon Graphics Library *GL*, the text editor is a standard text editor.

Fig.6 a-b-c: Exponential growth of image complexity with time which is typical of production systems. In this particular case, image complexity is multiplied by a factor of six at each iteration. The three objects in each figure develops continuously from a point to the complex structures shown.

Fig. 7 a-b: Small developing forest of apple tree-like structures, animated by a time and place dependent vector force field simulating gravity and aperiodic circulating wind. The whole development of the forest starts with 4 points(seeds of trees). the elasticity of each branch is proportional to its thickness. About ten percent of the apples and leaves are detached from the trees by the wind. Each apple, fallen to the ground, is a starting point for a new tree. The final complexity of this scene is more or less limited only by the calculation time of the computer.

Fig. 8 a-b: A scene with some fractal objects. The forest is growing and animated by the same winds as in Fig. 7. The other objects stopped their development. These movements are caused by simple variations of constants.

The user interface is based on the *5th Dimension Toolkit* (Turner et al. 1990). The module *LM_object* provides methods for creating and deleting objects and generate object at a fixed age according to the probabilistic and parameterized grammar. The axiom, the constants, the production rules and the array of random numbers are provided by the *reader* module, responsible for the analysis of the syntactic description of the object. During the generation, the module *procedures* directly interprets the axiom and the productions and draws the objects using the operations provided by the *turtle* module and the forces exported by the module *forces*. At any time of the evolution, objects may be saved in a format compatible with the SurfMan sculpting software (LeBlanc et al. 1991); this is performed by the module *procedures_sm*. The module *computer* exports procedures for the evaluation of numerical and logical expressions and is responsible for the integration of differential equations using Euler and Runge-Kutta methods.

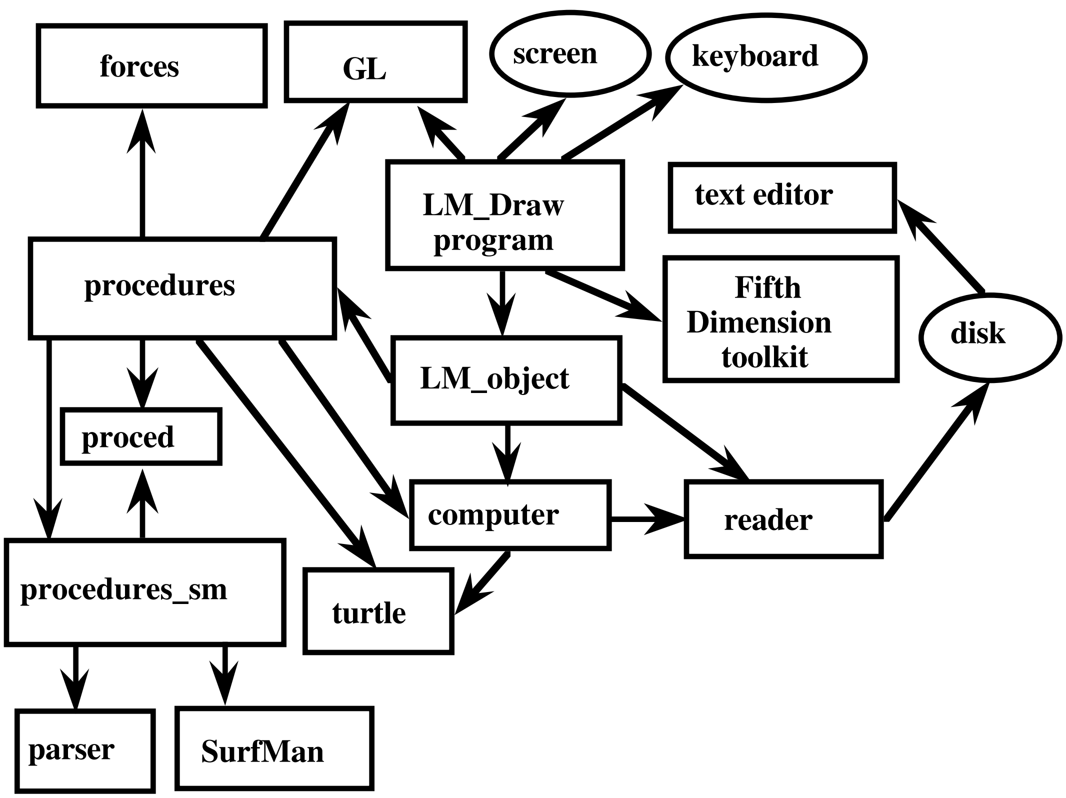

Fig.9. Modular structure and interaction with the environment

The system has been developed for two types of users:

1) the designer who uses a text editor to define axioms and production rules and the LMDraw program to visualize the results

2) the programmer who may use the LM_object primitives in his program to generate objects defined by L-structures

7. CONCLUSION

It has been proved possible to build a tool for creating artistic animation sequences based on L-structures. The system allows the creation of very complex scenes and motions because it integrates timed parametric context-free L-systems with conditional and pseudo-stochastic productions in interaction with vector force fields.

8. REFERENCES

Abelson H, diSessa AA (1982) Turtle Geometry, MIT Press, Cambridge.

Aono M, Kunii TL (1984) Botanical tree image generation, IEEE Computer Graphics and Applications, Vol.4, No5, pp.10-34

Demko S, Hodges L and Naylor B (1985) Construction of Fractal Objects with Iterated Function Systems, Proc. SIGGRAPH'85, Computer Graphics, Vol. 19, No3, pp.271-278.

Demongeot J., Goles E. and Tchuente M. (1985) Introduction Dynamic Behaviour of Automata, in: Dynamical Systems and Cellular Automata, Academic Press.

Fournier A, Fussell D, Carpenter L (1982) Computer rendering of stochastic models. Comm. ACM, Vol.25, No6, pp.371-384

Gander W (1985) Computermathematik, Programm Praxis, Band 3, Birkhaeuser Verlag, Basel

Gardner M. (1971) Mathematical Puzzles and Diversions, Penguin books.

Gardner M. (1983) Wheels, Life and Other Mathematical Amusements, Freeman, San Francisco.

LeBlanc A, Kalra P, Magnenat-Thalmann N, Thalmann D (1991), Sculpting with the "Ball & Mouse" Metaphor, Proc. Graphics Interface '91, Calgary, Canada, pp. 152-159.

Lindenmayer A (1968) Mathematical Models for Cellular Interaction in Development, Journal of Theoretical Biology, Vol.18, pp.280-315.

Lobry C and Reder C (1985) Concerning the Support of the Solutions of Certain Automata, in: Dynamical Systems and Cellular Automata, Academic Press.

Mandelbrot B (1975) Stochastic models for the earth's relief, the shape and fractal dimension of coastlines, and the number area rule for islands. Proc. National Acad.-Sc USA, Vol.72, No10, pp.2825-2828

Mandelbrot B (1977) Fractals:form, chance and dimension. WH Freeman, San Francisco

Mandelbrot B (1982) The fractal geometry of nature. WH Freeman, San Francisco

Mandelbrot B, van Ness J (1968) Fractional brownian motions, fractional noises and applications. SIAM Review, Vol.10, NO4, pp.422-437

Noser H (1992) L-structures, Dipl. thesis, Computer Graphics Lab, Swiss Federal Institute of Technology (in French)

Prusinkiewicz P, Lindenmayer A (1990) The Algorithmic Beauty of Plants, Springer-Verlag

Smith AR (1984) Plants, Fractals, and Formal Languages, Proc. SIGGRAPH '84, Computer Graphics, Vol.18, No3, pp.1-10.

Thalmann D (1986), A Lifegame Approach to Surface Modeling and Rendering, The Visual Computer, Vol.2, No6, pp.384-390

Turner R, Gobbetti E, Balaguer F, Mangili A, Thalmann D, Magnenat-Thalmann N (1990) An Object Oriented Methodology Using Dynamic Variables for Animation and Scientific Visualization, Proceedings Computer Graphics International 90, Springer-Verlag, Tokyo, pp.317-328.

Von Koch H (1905) Une méthode géométrique élémentaire pour l'étude de certaines questions de la théorie des courbes planes, Acta Mathematica, Vol.30, pp.145-174.

Von Neumann J (1966) Theory of self reproducing automata, University of Illinois Press.

Wejchert J, Haumann D (1991) Animation Aerodynamics, Proc. SIGGRAPH '91, Computer Graphics, Vol.25, No4, pp. 19-22.

Wolfram, S (1984) Cellular Automata as Models of Complexity, Nature, Vol.311, No 4, pp.419-424.

Hansrudi Noser is a researcher at the Computer Graphics Laboratory of the Swiss Federal Institute of Technology in Lausanne, Switzerland. He received his diploma in Physics from the ETHZ (Swiss Federal institute of Technology in Zürich, Switzerland). After working as a researcher for the Institute of Applied Physics at the ETHZ (C-MOS process developing) ans as a researcher in the department of Thin Film Electronic of the Balzers AG, Balzers (FL), he received his diplôme d'ingenier informaticien d'ingénieur informaticien from the Swiss Federal Institute of Technology in Lausanne, Switzerland. His research interests include computer animation, computer art.
E-mail: noser@ligsg2.epfl.CH

Daniel Thalmann is currently full Professor, Director of the Computer Graphics Laboratory and Head of the Computer Science Department at the Swiss Federal Institute of Technology in Lausanne, Switzerland. He is also adjunct Professor at the University of Montreal, Canada. He received his diploma in nuclear physics and Ph.D in Computer Science from the University of Geneva. He is coeditor-in-chief of the *Journal of Visualization and Computer Animation*, member of the editorial board of the *Visual Computer* and *the CADDM Journal* and cochairs the EUROGRAPHICS Working Group on Computer Simulation and Animation. Daniel Thalmann's research interests include 3D computer animation, image synthesis, virtual reality and scientific visualization. He has published more than 100 papers in these areas and is coauthor of several books including: *Computer Animation: Theory and Practice* and *Image Synthesis: Theory and Practice*. He is also codirector of several computer-generated films.
E-mail: thalmann@eldi.epfl.CH

Russell Turner is a researcher at the Computer Graphics Laboratory of the Swiss Federal Institute of Technology in Lausanne, Switzerland. He received his B.S. in Physics and his M.S. in Computer and Information Science from the University of Massachusetts at Amherst. He has also worked as a software engineer for V.I. Corporation of Amherst, Massachusetts. His research interests include computer animation, physical modeling, user-interfaces and object-oriented programming. He is a member of IEEE and ACM.
E-mail: `turner@elma.epfl.CH`

The authors may be contacted at:

Computer Graphics Lab
Swiss Federal Institute of Technology
CH 1015 Lausanne, Switzerland
tel: ++41-21-693-5214 fax: ++ 41-21-693-5328