# REAL-TIME INTERACTIVE L-SYSTEM
## *A Virtual Plant and Fractal Generator*

Ludovic Hamon, Emmanuelle Richard, Paul Richard and Jean-Louis Ferrier
*Laboratoire d'Ingénierie des Systèmes Automatisés (LISA)*
*Université D'Angers, 62, Avenue Notre Dame Du Lac, 49000 Angers, France*

Keywords: Virtual plant, L-systems, Real-time interaction, Haptic rendering, Augmented reality.

Abstract: L-system is a rewriting process based on Chomsky grammar that can generate 3D fractals and virtual plants. Its efficiency has been proved for more than twenty years with the works of Lindenmayer and Prusinkiewicz in particular. This paper presents a Real-Time Interactive L-system (RTIL-system) allowing real-time rendering, interactions with a user and/or the environment, haptic feedback and covering most of the important L-systems extensions such as parametric and context sensitive features. Different potentialities of the RTIL-system are illustrated where the user intervention becomes an essential element of plant/fractal evolution through Virtual Reality and Augmented Reality applications.

## 1 INTRODUCTION

Plant modeling is used in various domains like botany, agronomy, forestry, ecology, urban design, and even video games. Virtual plants are computer simulations of the growth, development and deployment of plants in three-dimensional (3D) space. In Virtual Reality (VR) or Augmented Reality (AR) applications, real-time interaction with the virtual plants is required. Moreover, in some applications involving tasks such as pruning, physical properties may be added to enable haptic rendering and therefore the simulation realism. In the last decades, several software applications have been developed. For example, Amap is based on statistic measurements, Markov chains and the Monte-Carlo method (Cirad, 2009). Another software program called Xfrog was specifically developed for trees modeling (Lintermann and Deussenm, 1999). Vlab can generate, order and save an amount of virtual plant models coming from different programs (Federl and Prusinkiewicz, 1999). GreenLab adopted the AMAP discretization scheme and a dual-scale automation (Reffye et al., 2003).

One of the most interesting software applications is L-studio/cpfg/lpfg created in 1999 at the University of Calgary (Canada) by Prusinciewicz and co-workers (Prusinkiewicz et al., 2000; Karwowski, 2002). It is based on L-system (Lindenmayer, 1968; Prusinkiewicz and Lindenmayer, 2004), a parallel rewriting process inspired by Chomsky grammar. This formal mathematical approach can generate 3D virtual plants with realistic morphological and physiological aspects.

Numerous applications are based on the L-system model. However, to our knowledge, none of them are simultaneously: (i) open source, (ii) with a good rendering quality, (iii) in real time, (iv) covering most of L-system essential extensions such as parametric and context sensitive features, and (v) interactive. This paper presents a Real-Time Interactive L-system (RTIL-system) allowing interaction between virtual plants/fractals and both the user and the environment. In addition, our approach takes into account most important L-systems extensions. Collision detection and haptic rendering of the virtual plant/fractal are allowed. Different potentialities of the proposed RTIL-system are illustrated through VR/AR applications such as interactive dynamic plants evolution, pruning, constrained growing and an interactive fractal. In the next section, the formalism and evolution of L-systems focusing on real-time interactive applications are given. Section 3 is dedicated to the basic features and the implementation of our RTIL-System. Real-time interaction with the user and/or the environment are described in section 4 with corresponding applications. Finally a conclusion provides some perspectives.

Hamon L., Richard E., Richard P. and Ferrier J. (2010).
REAL-TIME INTERACTIVE L-SYSTEM - A Virtual Plant and Fractal Generator.
In *Proceedings of the International Conference on Computer Graphics Theory and Applications*, pages 370-377
DOI: 10.5220/0002845703700377

## 2 L-SYSTEMS: FORMALISM AND EVOLUTION

### 2.1 Elementary Notions and Extensions

L-system is a formal language that was firstly used to describe the development and proliferation of simple multi-cellular organisms (Lindenmayer, 1968). Lindenmayer and Prusinkiewicz developed and extended this language to model plants (Prusinkiewicz and Lindenmayer, 2004). L-system is a triplet (V, w, P), V being an alphabet, w being an axiom, and P being a set of production rules. The couple (p,s) belonging to the set production rules is composed of a predecessor (p) and a successor (s). A derivation process replaces the axiom symbols equal to predecessors by the corresponding successors. If there is no corresponding predecessor, then the symbol is replaced by itself. After the axiom is completely reviewed, the string obtained is called "L-string". Parametric L-systems were important improvements in which each symbol (called "module") may take a finite number of real parameters (Prusinkiewicz and Lindenmayer, 2004). Furthermore, a logical condition was added to each production rule that is applied if: $(i)$ the current symbol is equal to the rule predecessor, $(ii)$ the number of parameters of symbol is equal to the number of parameters of the rule predecessor, and $(iii)$ if the rule condition is true (Fig. 1). Thereafter, Lindenmayer and Prusinkiewicz proposed the turtle geometry concept (belonging to the LOGO Language) to give a geometrical interpretation to the L-string. This approach is based on an imaginary turtle that walks and draws lines. In two dimensions, the turtle geometry concept is based on the triplet $(x,y,a)$, $(x,y)$ being the position of the turtle, and $a$ being the yaw angle of the turtle movement. A simple example of context free deterministic D0L-system is given in figure 2(a). The following interpretation symbols are introduced with a stack that saves the turtle states as follows (Prusinkiewicz and Lindenmayer, 2004):

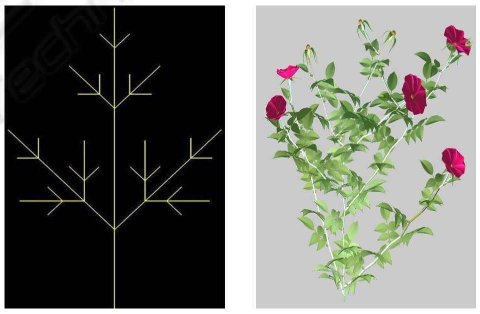
$F$ : move a $d$ distance forward and draw a line.
$(x,y,a) \Rightarrow (x+d*cos(a), y+d*sin(a), a)$
$f$ : move a $d$ distance forward without drawing a line
$(x,y,a) \Rightarrow (x+d*cos(a), y+d*sin(a), a)$
$+$ : turn left of $o$ angle. $(x,y,a) \Rightarrow (x,y,a+o)$
$-$ : turn right of $o$ angle. $(x,y,a) \Rightarrow (x,y,a-o)$
$[$ : push the current turtle state to the stack.
$]$ : pop the previous turtle state from the stack.

$w : B(2)A(4,4)$
$p_1 : A(x,y) : y \leq 3 \rightarrow A(x*2, x+y))$
$p_2 : A(x,y) : y > 3 \rightarrow B(x)A(x/y, 0)$
$p_3 : B(x) : x < 1 \rightarrow C$
$p_4 : B(x) : x \geq 1 \rightarrow B(x-1)$

$$\begin{tikzcd}
\mu_0: & B(2) \arrow[d] & A(4,4) \arrow[dr] & \\
\mu_1: & B(1) \arrow[d] & B(4) \arrow[d] & A(1,0) \arrow[d] \\
\mu_2: & B(0) \arrow[d] & B(3) \arrow[d] & A(2,1) \arrow[d] \\
\mu_3: & C \arrow[d] & B(2) \arrow[d] & A(4,3) \arrow[d] \\
\mu_4: & C & B(1) & A(8,7)
\end{tikzcd}$$

Figure 1: Example of parametric L-system evolution (Prusinkiewicz and Lindenmayer, 2004).

$Lsystem : 1$
$derivation\ length : 4$
$Axiom : A$
$A \rightarrow F[+A][-A]FA$
$F \rightarrow FF$



a b

Figure 2: Illustrations of D0L-systems: (a) a simple model with corresponding axiom and production rules (Prusinkiewicz and Lindenmayer, 2004); (b) a realistic rose-bush (Favre et al., 2007).

In 1970s and 1980s several new forms of L-systems appeared and numerous interpretation symbols were proposed. For example, stochastic L-systems have been developed to add a probability to each production rule (Prusinkiewicz and Lindenmayer, 2004). This feature allows the application of several rules to the same symbol, which was not possible with deterministic L-systems. Pseudo L-systems have rule predecessors that can be made up of several modules (Prusinkiewickz, 1986). Another interesting extension of L-systems is context sensitive L-system (called D1L-systems or D2L-systems). In a left (respectively right) context sensitive L-system a rule is applied only if the left (respectively right) symbol of the current symbol is equal to the left (respectively right) context of the current rule predecessor (Prusinkiewicz and Lindenmayer, 2004).

Different applications with advanced graphical rendering have appeared. For example, Baele and coworkers have generated real time trees with different detail levels, lightning and animation under wind using GPU capabilities (Baele and Warzée, ).

From 1990s to now, virtual plants generated by L-system explicitly have described spatial distribution of plant organs in three dimensions as a collection of interconnected components. Furthermore, they model the morphological and physiological aspects (Favre et al., 2007) and the dynamic of plants population (Bornhofen and Lattaud, 2007; Prusinkiewicz, 2009).

## 2.2 Interactions with the User

Many software applications dedicated to plant modeling exist and provide real-time basic interactions with virtual plant such as rotation or zoom. These interactions use the computer mouse and/or the keyboard (Federl and Prusinkiewicz, 1999; Reffye et al., 2003; Prusinkiewicz et al., 2000; Karwowski, 2002). However, some approaches have focused on advanced interactions techniques. For example, Joanna et al. have developed an interesting interactive L-system called ILSA (Interactive L-String Arranger), based on the Cpfg engine (Joanna et al., 1999; Prusinkiewicz et al., 2000). The user can select plant components using a mouse picking technique. The selection can be done with two points on a branch (a starting and an ending point). Operations like twisting, bending or pruning, are possible. Interactive manipulations are executed in semi real time, using a basic graphic rendering.
More recently, Onishi and co-workers have presented a D0L-system in which 3D turtle cartesian coordinates are directly sent to the environment (Onishi et al., ). They use a magnetic tracking system to generate the trunk and the branches from hand movements. Corresponding L-strings are generated and branches are attached to the trunk by some specified connecting points. The user is also able to specify some geometric primitives or draw silhouettes in which parts of plant will grow up. Furthermore, a component included in the selected zone may be pruned, rotated, etc. These works provide some interesting 3D interactions but are all based on simple non-parametric D0L-systems limiting both the modeling possibilities and animation rendering.

## 2.3 Interactions with the Environment

Another interesting feature of interactive L-systems concerns interaction with the environment. The Cpfg engine includes this possibility using a specific module called ”Communication Module” (Mech, 1997). This module is presented by the following syntax: ”?E(x1,...,xn)”. It allows the program to send and receive information from the environment. For example, suppose the following L-string: ”F(2)?E(1,2)F(3)F(5)F(5)”. Information sent to environment is: position of the communication module in the L-string and its parameters (here 1 and 2), turtle state (position and orientation), name and parameters of the module that follow the communication module (here ”F(3)”). Information received from the environment is the parameters $(x_1,\ldots,x_n)$ that can be modified by this last one. A L-system with a communication module is called Open L-system. Open L-system offers several interesting possibilities such as handling collisions between plants organs or components, pruning, water diffusion in soil, light distribution and several other interactions. However, a communication module is required for each modules to send the entire topological structure to the environment.
An interesting work on L-systems was done by Bornhofen et al. who used a D0L-system to model different plants in a virtual garden. These plants grew according to some environmental factors such as sunlight and soil components (Bornhofen and Lattaud, 2007). In addition, the virtual plants are in competition for resources with respect of morphological and physiological aspects.

## 3 RTIL-SYSTEM: BASIC FEATURES

**RTIL-system** is based on Cpfg concepts and model and uses a quite similar syntax. It is a library written in C/C++ language that allows creating and editing a script file and generating a 3D rendering of the virtual plant using OpenGL library with optional stereoscopic features for depth sensation enhancement ($3DVision^{TM}$ device from NVIDIA was used). RTIL-system takes into account, deterministic, context free, context sensitive, parametric, non parametric and pseudo L-systems features . Homomorphism and decomposition rules are also available (Prusinkiewicz et al., 2000; Karwowski, 2002).

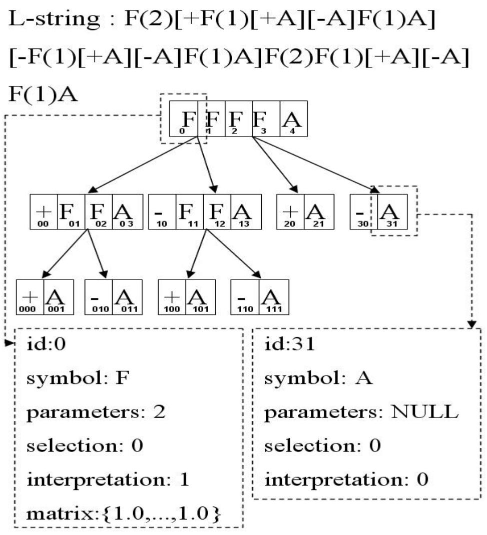


Figure 3: Tree structure of a L-string example.

## 3.1 Data Structure

A classic tree from computer science is used (fig. 3). Each node is composed of a finite set of modules. For each module, the following attributes are recorded: (i) an identifier called *id*, (ii) its symbol, (iii) its parameters, (iv) a boolean variable (selection) that indicates if the current module is selected by the user, (v) a boolean variable (interpretation) that indicates if the current module has a graphical representation. In the case where ”interpretation” is true, some geometrical information is added such as (vi) an OpenGL homogeneous matrix, etc..

## 3.2 Scripting and Syntax

A typical RTIL-system rule is presented as follows: $cg$<pred>$cd$: $C_a$ *cond* $C_b$ $\rightarrow$ succ, where ”pred” is the predecessor, $cg$ is the left context, $cd$ is the right context, *cond* is a logical proposition, $C_a$ and $C_b$ are C langage instructions and ”succ” is the successor. According to the location of C language instructions in the script, their respective executions are triggered as follows: $C_a$ is executed if the predecessor and its context correspond to the current L-string module and its context. $C_b$ is executed if the previous condition for $C_a$ execution is true and the logical proposition *cond* is true. Other C language instructions can be included after the *Extern Start* and *Extern StartEach* keywords. The first one indicates that the C instructions are executed before the beginning of the first derivation process. The second one indicates that the C language instructions are executed before each derivation process. These instructions are limited to simple operators like, allocation, addition, substraction etc.. An example (*script*1) with its graphical representation is given below and illustrated in figure 4.

$Lsystem : script1$
$Derivationlength : 2$
$Axiom : A$
$ExternStart : \{c_2 = 0; s_2 = 0;\}$
$ExternStartEach : \{c_2 = 0;\}$
$A \rightarrow I(1)[+A][-A]I(1)A$
$I(x)$:{$c_2$=$c_2$+1;} x≤10 {$s_2$=$s_2$+1;}$\rightarrow I(x*2)$
$Homomorphism$
$I(x) \rightarrow F(x)$
$A \rightarrow F$

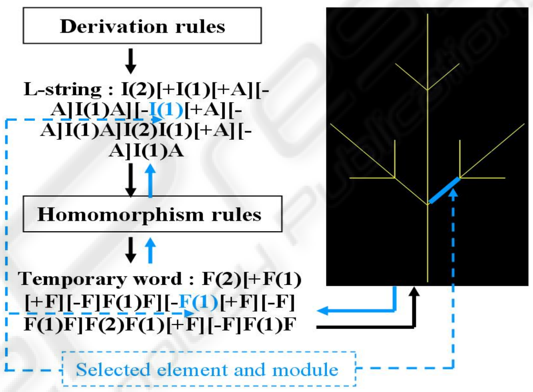

Derivation rules
L-string : I(2)[+I(1)[+A][-A]I(1)A][-I(1)[+A][-A]I(1)A]I(2)I(1)[+A][-A]I(1)A
Homomorphism rules
Temporary word : F(2)[+F(1)[+F][-F]F(1)F][-F(1)[+F][-F]F(1)F]F(2)F(1)[+F][-F]F(1)F
Selected element and module

Figure 4: From L-string to graphical rendering : derivation and homomorphism.

The *derivation length* is the number of required derivation. The global variable ”$c_2$” counts the number of call of second rule during the entire process. The global variable ”$s_2$” counts the application number of second rule. Homomorphism is an optional and additional derivation process with specific rules that follow *homomorphism* keyword. It is executed after the L-string generation and creates a temporary word (see black arrows in fig. 4). Homomorphism process is usually used to isolate the interpretation sequence from the derivation sequence i.e: most of the interpretation symbols are added with the homomorphic derivation. It allows creating an accurate topological formalism in which, for example, each plant component possesses its own symbol, its own corresponding rules as well as its own functions and operators.

## 4 RTIL-SYSTEM INTERACTIVE FEATURES

### 4.1 Data Transmission and Reception

The *Extern Start* and *Extern EachStart* keywords allow global variables definition. Global variables can be seen and edited in real time by programs that include the RTIL library. A function *func(id,*$x_1,\dots,x_n$*)* can be included in the C language instructions and/or in the module parameters. This function calls a function written in a C/C++ file, referred to ”id”, sends *(*$x_1,\dots,x_n$*)* parameters and returns a real value. Thus, advanced algorithms, functions, procedures and data types belonging to C/C++ language may be used. A specific procedure named *com(*$x_1,\dots,x_n$*)* can be called in *C* instructions. It sends *(*$x_1,\dots,x_n$*)* parameters that could be modified by intern or extern programs. With this procedure, local variables and other information are available to an environment that can simulate light, collisions, soil distribution etc. Finally, between two derivation steps, each graphical piece of information linked to each interpretation module (i.e: positions and orientations of the turtle geometry) can be seen by programs including our RTIL-library.

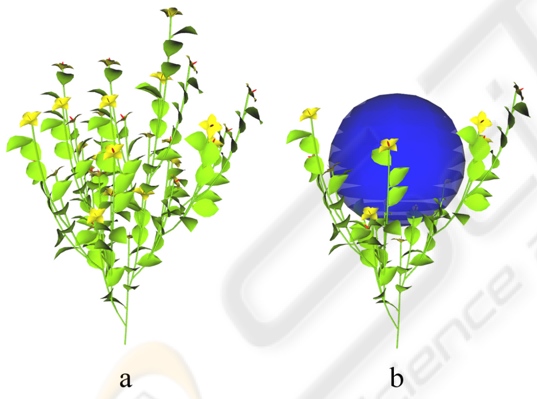


Figure 5: Development of a virtual plant : unconstrained evolution (a), and evolution constrained to the outside of a sphere (b).

An example of interaction with the environment is illustrated in figure 5. In this case the virtual plant growth is constrained to the outside of a sphere. This process is achieved thanks to rules similar as this one: $I(i)$ :$\{com(x);\}$ $(x == 1) \rightarrow I(i+di)$. The procedure *com* calls an ”environment” program that tests if the position of the current graphical component linked to the current internode *I* is inside the sphere or not. If not, then x is set to 1 and the current internode *I* is lengthened by *di*. In the opposite case, the internode stops growing.

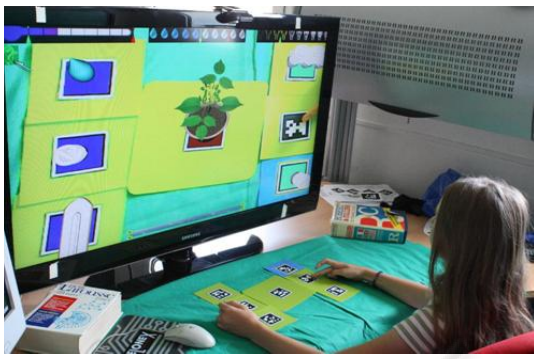


Figure 6: A child feeding a virtual plant using an AR interaction technique.

An interesting application based on our RTIL-system is illustrated in figure 6. The aim of this AR application was to make children (i) learn about some fundamental needs of plants (water, heat and light) and (ii) become aware of differences in plant needs throughout their life cycle. Using a menu-based interaction technique, the child can bring various quantities of these three fundamental needs to a virtual plant, and observe their effect on the germination, growth and reproduction steps. The three needs are defined using global variables that are updated between two derivation steps. Depending on their values, positive or negative rules are applied to the plant evolution. A possible evolution of the virtual plant is illustrated in figure 7. An experiment in which twenty-four K4-K5 children were instructed to make a virtual plant grow was carried out and the results are exposed in a paper called ”Real-Time Interactive L-system for Virtual Plants, An Augmented Reality Educational Application”, submitted for a Special Issue on Augmented Reality on November 2009. (Virtual Reality Journal, 2010).

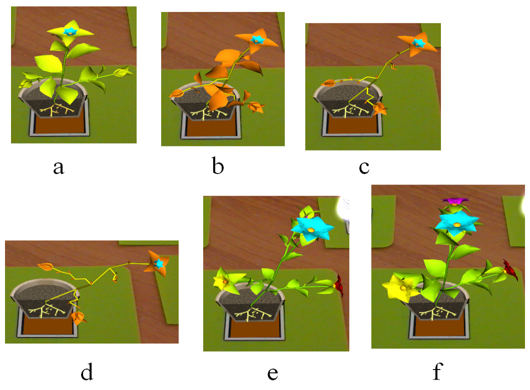


Figure 7: Illustration of a possible evolution of the virtual plant : flowering stage (a), followed by a lack of water and excess of heat (b)(c)(d), and then by a correct feeding (e)(f).

## 4.2 Graphical Interaction and Haptic Rendering

Graphical interaction with the virtual plant requires the three following steps: (i) select one (or more) 3D component(s), (ii) find the corresponding module(s) in the L-string (see blue arrows in fig. 4), and (iii) apply a corresponding action on the selected module(s). For example, a pruning task consists in the removal of a graphic element coupled with the deletion of the corresponding module in the L-string. Some difficulties may appear when linking the graphical component to the corresponding L-string module whereas homomorphism concept and pseudo L-systems allow the creation of different L-system scripts generating the same graphical shape and behavior. Let us consider the *script*1 with the first homomorphic rule $I(x) \rightarrow F(x)$ replaced by this one: $I(x) \rightarrow F(x/2)F(x/2)$. This change leads to the same graphical rendering but the application is not bijective.

Thus, apart from L-systems without homomorphism process, graphical selection tasks necessarily require that the predecessor of each homomorphic rule has the same number of modules as the corresponding successor *(proposition 1)*. Consequently, to preserve the current formalism and to avoid this problem, RTIL-system homomorphic rules are limited to *proposition 1*.

### 4.2.1 3D Interaction Devices

For the selection of a graphic component of the plant, our RTIL-system allows the integration of different types of 3D interaction devices such as the (i) the Polhemus $Patriot^{TM}$ electromagnetic tracking device from Polhemus or (ii) the $PhantomOmni^{TM}$ desk-top haptic interface from Sensable (fig. 10). Selection of plant components is achieved using a collision detection algorithm involving a spherical cursor (fig. 11) and the geometrical model associated with the selected component.

### 4.2.2 Selection: Collision Detection and Haptic Rendering

In order to handle collision detection and provide haptic rendering, RTIL system associates a geometrical primitive to each plant component (Fig. 8): (i) a cylinder for roots, branches, trunks, internodes and fractal components (ii) a sphere for seeds and fruits, (iii) and an ellipsoid for leaflets, leaves and sepals (merged with the (LX,LY) surface plan). Figure 9 shows both graphical and physical models of a virtual flower, and their superimposition.

Thus, the selection of a component of the plant requires that the center X($x_1$,$y_1$,$z_1$) of the spherical cursor with radius $r$ is solution to one of the following equations according to the geometrical primitive, where L($x_0$,$y_0$,$z_0$) is the origin of the local cartesian space (fig. 8) :

Cylinder : $(0 \leq y_1 \leq h_0)$ and
$((x_1 - x_0)^2 + (z_1 - z_0)^2 \leq (r + r0)^2)$
Sphere :
$(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2 \leq (r + r0)^2$
Ellipsoid : $(\frac{x_1 - x_0}{a_0 + r})^2 + (\frac{y_1 - y_0}{b_0 + r})^2 + (\frac{z_1 - z_0}{c_0 + r})^2 \leq 1$

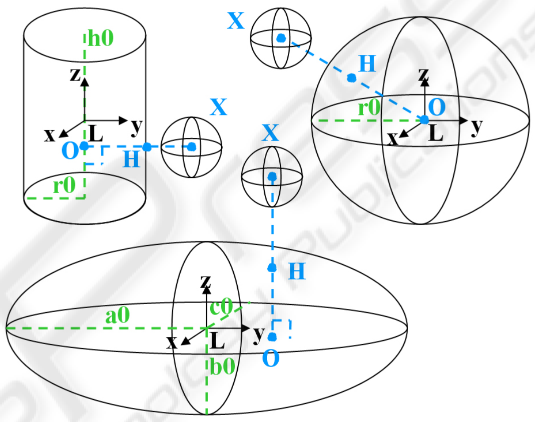


Figure 8: Three geometrical models used for collision detection and haptic rendering : (a) a cylinder for roots, branches, trunks, internodes and fractal components ; (b) a sphere for seeds and fruits ; and (c) an ellipsoidal surface for leaflets, leaves and sepals.

A blue color shows that a component is selected. Force feedback vectors have been implemented when the $PhantomOmni^{TM}$ haptic interface is used thanks to RTIL library. For each physical element, two points have been calculated depending on "X" point location: "O" is the center of the physical sphere (or respectively, the orthogonal projection of X on the cylinder axe, or the orthogonal projection of X on the (LX,LY) ellipsoid plan), and "H" is the intersection of OX line and the physical sphere surface (or respectively the cylinder surface, or the ellipsoid surface) (fig. 8).

With these two points, two simple forces $\overrightarrow{F}$ have been created in the cartesian space: an attractive force that attracts the Phantom Omni inside the physical representation if $\|\overrightarrow{OX}\| \geq (\|\overrightarrow{OH}\| + r)$, and a repulsive force that allows the Phantom Omni sliding along the physical representation if $\|\overrightarrow{OX}\| > \varepsilon$:

$\overrightarrow{NorPos} = \overrightarrow{OX}$, Normalize($\overrightarrow{NorPos}$),
Attractive Force: $\overrightarrow{F} = -\frac{stiffness1}{\|\overrightarrow{OX}\|^2} \times \overrightarrow{NorPos}$

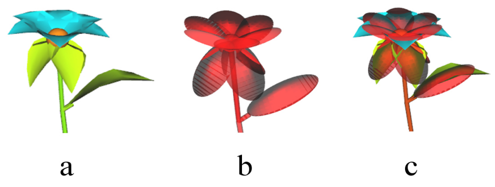


Figure 9: A virtual flower : (a) graphical model ; (b) physical model ; and (c) superimposition of both graphical and physical models.

Repulsive Force: $\overrightarrow{F} = \frac{stiffness2}{\|\overrightarrow{OX}\|^2} \times \overrightarrow{NorPos}$

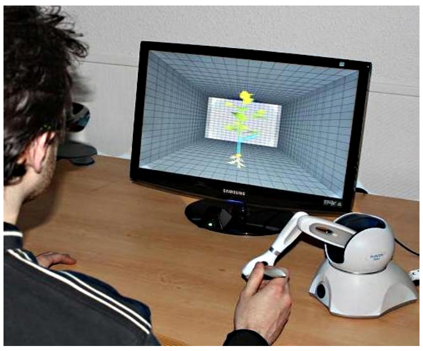


Figure 10: A user working on a virtual plant, with Phantom Omni device, force feedback and stereoscopic viewing.

The first effect constrains the user to navigate inside and along the virtual object. The stiffness coefficient adjusts the attraction force. With the second effect touch sensation appears. The virtual object seems rigid with a high value of stiffness coefficient or soft with a low value. Figure 10 shows a user performing tasks with force feedback in stereoscopic viewing condition.

### 4.2.3 Deletion and Pruning

When one or more graphical components are selected by the user, with the help of dedicated interfaces, they can be pruned. That leads to: (i) the deletion of the corresponding modules in the data-structure (fig. 3), (ii) the deletion of all modules that follow these last ones in the same node, (iii) and the recursive deletion of all "son" nodes that are linked to previous erased "father" modules in (i) and (ii) (fig. 11, (b) and (c)).

### 4.2.4 Partial Interactive Derivation

Our RTIL-system introduces a new keyword called *Interaction* in the script file (fig. 12). It is followed by a finite set of rules called "*Interaction Rules*" that can be applied to selected modules only. With this feature, the concept of *Partial Interactive Derivation*

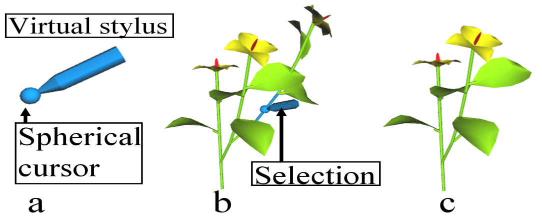


Figure 11: Illustration of a simple pruning task : the virtual Phantom Omni stylus (a), selection of an internode of the virtual plant (b), and deletion (c).

is introduced i.e: one or more derivation steps with *Interaction Rules* on only selected modules. If there are no *Interaction rules*, usual rules replace them. A large set of possible interactive tasks such as bending/twisting components, partial development, elongation/reduction of components, breaking, insertion of elements etc... are available with *Partial Interactive Derivation*.

$$\begin{gathered} Lsystem : script2 \\ Axiom : A+++A+++A \\ A \rightarrow A-A++A-A \\ Interaction \\ A \rightarrow A-A++A-A \\ Homomorphism \\ A \rightarrow F \end{gathered}$$

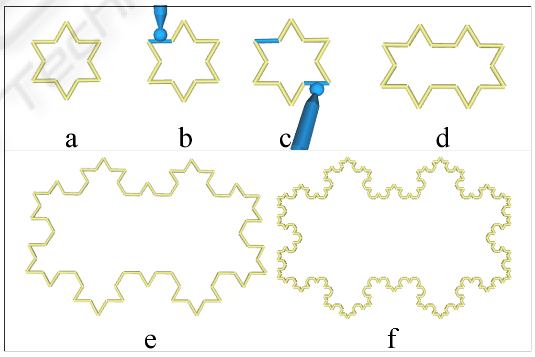


Figure 12: A half divided Koch fractal snowflake.

For example, Figure 12 shows a mutated "Koch Fractal snowflake". From a normal Koch snowflake in (a), user selects two components in (b) and (c). With application of the Koch interactive rule "$A \rightarrow A-A++A-A$" to selected components, a half divided snowflake is obtained in (d). Then (e) and (f) are the results of two classic derivations on the entire object.

Another example illustrating the *Partial Interactive Derivation* process is shown in figure 13. A bug, represented by a red sphere, is placed (b) on a selected component of the virtual plant (a) thanks to the following interaction rule: $I(i) \rightarrow BI(i)$ ($I$: inter-node, $B$:

bug). As a consequence of several normal derivations, the bug climbs along the current axis of the plant and eats the flower (c). This induces an apical dominance phenomenon leading to the outgrowth of a lateral bud (c).

With respect of the L-system concept, "*Partial Interactive Derivation*" lead to open L-system formalism to advanced interactions where user can create, edit, control part or entire derivation process and becomes an important element of the model evolution.

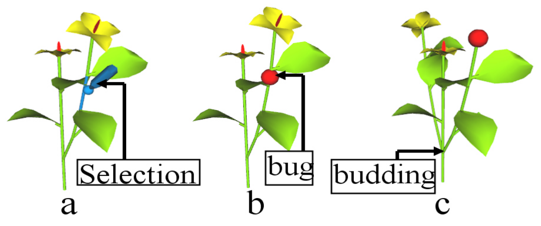


Figure 13: Example of a Partial Interactive Derivation process: a bug is placed (b) on the selected component (a), climbs along the axis and eats the flower, and the result is the outgrowth of a lateral bud (c).

## 5 CONCLUSIONS

This paper presents a Real-Time Interactive L-system (RTIL-system) allowing interactions with both the user and the environment. RTIL-system takes into account most important L-systems extensions such as parametric and context sensitive features. In order to preserve L-systems formalism and allow advanced interactions, the *Partial Interactive Derivation* concept is introduced. It allows the definition of *Interaction Rules* that can be applied on selected modules only. Collision detection and haptic rendering of the virtual plants are also provided. Different potentialities of the RTIL-system are illustrated through VR/AR applications such as interactive dynamic plants evolution, pruning, constrained growing and a mutated Koch fractal. In future work we plan to run experiments that will aim to investigate human performance in tasks involving multimodal interaction with virtual plants. Real time advanced graphical rendering technics will be explored. In addition, we will used our RTIL-system to generate different geometrical forms and interact with force feedback.

## REFERENCES

Baele, X. and Warzée, N. Real time l-system generated trees based on modern graphics hardware. In *International Conference on Shape Modeling and Applications, SMI'05 2005*.

Bornhofen, S. and Lattaud, C. (2007). Evolution of virtual plants interacting with their environment. In *Proceedings of the 9th International Conference on Virtual Reality (VRIC'07)*.

Cirad (2009). Amap website, http://amap.cirad.fr.

Favre, P., Guéritaine, G., Andrieu, R., and Boumaza, R. (2007). Modelling the architectural growth and development of rosebush using l-systems. In *Workshop on Growth Phenotyping and Imaging in Plants*.

Federl, P. and Prusinkiewicz, P. (1999). Virtual laboratory: An interactive software environment for computer graphics. In *Proceedings of Computer Graphics International*.

Joanna, L., Power, A., Bernheim, B., and Prusinkiewicz, P. (1999). Interactive arrangement of botanical l-system models. In *Proceedings of the Symposium on Interactive 3D Graphics*, pages 175–182.

Karwowski, R. (2002). Improving the process of plant modeling: The l+c modeling language. *Ph.D. dissertation, University of Calgary, Canada*.

Lindenmayer, A. (1968). Mathematical models for cellular interactions in development. *in Journal of Theoretical Biology parts I and II*, pages 280–315.

Lintermann, B. and Deussenm, O. (1999). Interactive modelling of plants. In *IEEE Computer and Application*, volume 19, pages 2–11.

Mech, R. (November 1997). Environment using l-systems and their extensions. *PhD thesis, University of Calgary, Canada*.

Onishi, K., Murakami, N., Kitamura, Y., and Kishino, F. Modeling of trees with interactive l-system and 3d gestures. In *International Workshop on Biologically Inspired Approaches to Advanced Information technology*, pages 222–235.

Prusinkiewickz, P. (1986). Graphical applications of l-systems. In *Proceedings of Graphical Interface and Vision Interface 86*, pages 247–253.

Prusinkiewicz, P. (2009). publications 1986 - 2009, http://algorithmicbotany.org/papers/.

Prusinkiewicz, P., Karwowski, R., Mech, R., and Hanan, J. (2000). L-studio/cpfg: A software system for modeling plants. *Nin Nagl M., Schrr A. and Mnch M. (eds.), Applications of Graph Transformations with Industrial Relevance,Lecture Notes in Computer Science*.

Prusinkiewicz, P. and Lindenmayer, A. (1990–2004). The algorithmic beauty of plants. *Springer Verlag*.

Reffye, P., Zhao, X., Yan, H., and Kang, M. (2003). Greenlab: a new methodology towards plant functional-structural model - structural aspec. In *Proceedings PMA03 : 2003' International symposium on plant growth modeling, simulation, visualization and their applications*.