# APPLICATIONS OF L-SYSTEMS TO COMPUTER IMAGERY

*Przemyslaw Prusinkiewicz*

Department of Computer Science
University of Regina
Regina, Saskatchewan, CANADA S4S 0A2

## ABSTRACT.

A method for object modeling is presented and illustrated with examples. It extends to three dimensions a previously described technique for generating two-dimensional pictures using L-systems [Prusinkiewicz 1986]. The objects are modeled in two steps:

- A string of symbols $\mu$ is generated using an L-system,
- $\mu$ is interpreted graphically as a sequence of commands controlling a turtle which maneuvers in three dimensions. The turtle can draw lines of various widths and colors, and trace boundaries of filled polygons.

Examples of synthesized objects are given and the construction of the corresponding L-systems is explained. Attention is focused on the modeling of plants. Stochastic L-systems are introduced to model various specimens of the same species. The turtle interpretation is extended to allow for incorporating predefined curved surfaces in the model. In spite of the apparent complexity of the resulting images, all discussed objects are generated by very concise L-systems.

**KEYWORDS:** L-systems, turtle geometry, fractals, plants, 3D object modeling, computer imagery, database amplification.

## 1. INTRODUCTION.

L-systems are rewriting systems introduced by Lindenmayer [1968] for the purpose of describing the growth of living organisms. They are particularly well suited to present the branching structures of plants. Hand-made drawings illustrating these structures have accompanied the theory of L-systems from its beginnings. However, only the topology of plants was formally defined. The geometry resulted from arbitrary decisions made by the draftsman.

The possibility of automatically generating pictures of plants using L-systems with a formally specified graphical interpretation was first investigated by Hogeweg and Hesper [1974]. They have conducted an exhaustive study of a subclass of propagating deterministic 2L-systems (with brackets). The generated strings were interpreted as three-dimensional branching patterns using the following set of rules:

In: H. Ehrig, M. Nagl, A. Rosenfeld and G. Rozenberg (Eds.): Graph grammars and their application to computer science; Third international workshop. Lecture Notes in Computer Science 291, Springer-Verlag, Berlin, pp. 534-548.

- Substrings between the same pair of brackets form a branch.
- A sidebranch is attached to that symbol of another branch which defines the left-side context of the left-most symbol.
- Branches grow straight, sidebranches leave their main branch at a given angle.
- All sidebranches attached to the same symbol are spread evenly over 360° while the direction of the first sidebranch at the symbol is defined according to a "spiralling coefficient".
- Alphabetic symbols are each represented by a line of given length, ended by a dot.

Hogeweg and Hesper used a plotter as the graphics output device. Consequently, the images they obtained were not realistic - they are perceived as schematic diagrams of plants. The potential of L-systems to generate realistic-looking plants was first realized by A.R. Smith [1978]. His subsequent paper [1984] established L-systems as a modeling tool for computer imagery purposes. Basically, the graphical interpretation of L-systems used by Smith was similar to that of Hogeweg and Hesper. Modifications included a greater variety of branching angles selected by a pseudo-random mechanism (implemented using a look-up table), changes of the branch sizes with the distance from the root, and the use of three-dimensional primitives - cylinders and spheres. The essential improvement in the quality of the generated images resulted from better rendering techniques which provided antialiased color images with shades, shadows, etc.

While the work of Hogeweg, Hesper and Smith concentrated on the modeling of plants, other interpretations introduced L-systems as a tool for generating abstract mathematical objects, specifically fractal curves. Szilard and Quinton [1979] defined a graphical interpretation in which both lines and angles were explicitly represented by string symbols. Prusinkiewicz [1986] noted that such interpretation could be easily described in terms of the turtle geometry [Papert 1980, Abelson and diSessa 1982]. The string symbols were then viewed as commands controlling a LOGO-like turtle (the basic commands being "move forward", "make a left turn" and "make a right turn"). A related interpretation based on chain coding [Freeman 1961] was introduced by Siromoney and Subramanian [1983]. In this case the string symbols denoted line segments drawn in predefined directions: up, down, left, or right. The curves generated by L-systems under these interpretations included the classic space-filling curves (the Sierpiński curve, the Hilbert curve and the Peano curve) and various Koch curves. The book by Mandelbrot [1982] made it possible to consider these curves in a unified way within the theory of fractals.

This paper further explores graphical applications of L-systems. The necessary definitions related to the L-systems are collected in Section 2. Section 3 introduces the three-dimensional graphical interpretation of strings. Section 4 presents examples of objects modeled using L-systems. While the L-systems provide astonishingly concise descriptions of many complex-looking objects (this feature was called *database amplification* by Smith [1984]), the construction of an L-system which would generate a specific object is not always straightforward. To illustrate the process of object modeling, two case studies are explained in detail. Section 5 introduces the notion of stochastic L-systems. They are used to model specimen-to-specimen variation within a species. An extension of the interpretation function which allows for incorporating curved surfaces in the modeled object is outlined in Section 6. Finally, section 7 summarizes the obtained results and presents topics open for further research.

It should be noted that while L-systems provide an effective and biologically well-motivated method for modeling plants, other methods have also been used in computer imagery. They can be divided into two basic categories:

- The methods in which the variety of forms is achieved primarily by modifying the *geometric* aspects of the branching structures. The topology results from the recursive algorithms used to generate these structures and is not controlled in detail. The best known methods of this group were proposed by Kawagushi [1982], Aono and Kunii [1984], Reeves and Blau [1985] and Bloomenthal [1985].
- The methods in which the variety of forms is achieved primarily by controlling the *topology* of the branching structures. The approaches based on the L-systems obviously fall in this category. A different method was developed by Eyrolles [1986] who adapted the Horton-Stahler analysis to generate diverse silhouettes of trees.

## 2. L-SYSTEMS.

This section summarizes fundamental definitions and notations related to L-systems. For their tutorial introduction, see Salomaa [1973], Herman and Rozenberg [1975], and Rozenberg and Salomaa [1980].

Let $V$ denote an alphabet, $V^*$ - the set of all words over $V$, and $V^+$ - the set of all nonempty words over $V$.

*Definition 2.1.* A **0L-system** is an ordered triplet $G = < V, \omega, P >$ where $V$ is the **alphabet** of the system, $\omega \in V^+$ is a nonempty word called the **axiom** and $P \subset V \times V^*$ is a finite **set of productions**. If a pair $(a, \chi)$ is a production, we write $a \rightarrow \chi$. The letter $a$ and the word $\chi$ are called the **predecessor** and the **successor** of this production, respectively. It is assumed that for any letter $a \in V$, there is at least one word $\chi \in V^*$ such that $a \rightarrow \chi$. A 0L-system is **deterministic** iff for each $a \in V$ there is exactly one $\chi \in V^*$ such that $a \rightarrow \chi$.

*Definition 2.2.* Let $G = < V, \omega, P >$ be a 0L-system, and suppose that $\mu = a_1 ... a_m$ is an arbitrary word over $V$. We will say that the word $\nu = \chi_1 ... \chi_m \in V^*$ is **directly derived** from (or **generated** by) $\mu$ and write $\mu \Rightarrow \nu$ iff $a_i \rightarrow \chi_i$ for all $i = 1,...,m$. A word $\nu$ is generated by $G$ in a derivation of **length** $n$ if there exists a sequence of words $\mu_0, \mu_1, ..., \mu_n$ such that $\mu_0 = \omega$, $\mu_n = \nu$ and $\mu_0 \Rightarrow \mu_1 \Rightarrow ... \Rightarrow \mu_n$.

*Definition 2.3.* A **T0L-system** is an ordered quadruplet $G_T = < V, \omega, P, T >$ where $V$, $\omega$ and $P$ are as in the definition of a 0L-system and $T$ is a finite nonempty collection of subsets of $P$, called **tables**. It is assumed that for each table $t \in T$ and each letter $a \rightarrow V$ there is at least one production $p \in t$ with the predecessor $a$.

*Definition 2.4.* A word $\nu$ is **directly derived** from the word $\mu$ in a T0L-system $G_T = < V, \omega, P, T >$ if $\nu$ is directly derived from $\mu$ in 0L-system $G = < V, \omega, P >$ and all production used in this derivation belong to the same table $t \in T$.

The notion of the direct derivation in a T0L-system generalizes to the derivation of length $n$ as in Definition 2.2. All productions used in a given derivation step must belong to the same table but different tables may be used in different derivation steps.

*Convention.* If no production is explicitly specified for a given predecessor $a \in V$, we assume that the **identity production** $a \rightarrow a$ belongs to the set of productions $P$ or to the table $t$ under consideration.

## 3. THREE-DIMENSIONAL TURTLE INTERPRETATION OF STRINGS

This section presents a method for mapping strings of symbols into graphical objects. This method is based on maneuvering a LOGO-like turtle in three dimensions [Abelson and diSessa 1982]. The key concept is to represent current **orientation** of the turtle in space is by three vectors $\vec{H}, \vec{L}, \vec{U}$, indicating the turtle's **heading**, the direction to the **left** and the **up** direction, respectively. Obviously, these three vectors are perpendicular to each other. Additionally, they are assumed to have the unit length, and to satisfy the equation $\vec{H} \times \vec{L} = \vec{U}$. Rotations of the turtle can be then expressed by the equation:

$$\left[\vec{H}'\ \vec{L}'\ \vec{U}'\right] = \left[\vec{H}\ \vec{L}\ \vec{U}\right]\mathbf{R} \tag{3.1}$$

where $\mathbf{R}$ is a 3×3 **rotation** matrix. Specifically, rotations by angle $\alpha$ about vectors $\vec{U}$, $\vec{L}$ and $\vec{H}$ are represented by matrices:

$$\mathbf{R_U}(\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$\mathbf{R_L}(\alpha) = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{3.3}$$

$$\mathbf{R_H}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{3.4}$$

*Definition 3.1.* A three-dimensional **turtle** is a triplet $T = < V, S, N >$ where $V$ is called the turtle's **alphabet,** $S$ is a **set of states** and $N$: $V \times S \rightarrow S$ is a **state transition function.** The alphabet $V$ consists of symbols:

$$F \quad f \quad @ \quad + \quad - \quad \& \quad \hat{} \quad \backslash \quad / \quad | \quad \# \quad ! \quad ' \quad ` \quad \{ \quad \} \quad [ \quad ] \tag{3.5}$$

The **state** of the turtle is a sextuplet $s = < \vec{P}, \vec{H}, \vec{L}, \vec{U}, w, c >$. The position vector $\vec{P}$ represents the turtle's **position** in Cartesian coordinates. Vectors $\vec{H}$, $\vec{L}$ and $\vec{U}$ represent the turtle's **orientation**, as described above. Number $w$ represents the **width** of the line drawn by the turtle. Number $c$ corresponds to the **color** of the lines and polygons drawn (*).

The state transition function has four parameters: the **step size** $d$, the **angle increment** $\delta$, the **width increment** $w\_inc$ and the **color increment** $c\_inc$. Given these parameters, the turtle responds to the symbols of $V$ as follows:

| | |
|---|---|
| $F$ | Move forward a step of length $d$. The position of the turtle changes to $\vec{P}' = \vec{P} + d\vec{H}$. A line segment of width $w$ and color $c$ between points $\vec{P}$ and $\vec{P}'$ is drawn. |
| $f$ | Move forward a step of length $d$ without drawing a line. |
| @ | Draw a dot of diameter $w$ and color $c$ at the current position $\vec{P}$. |
| + | Turn left by angle $\delta$. The orientation of the turtle changes according to formula (3.1), with the rotation matrix equal to $\mathbf{R_U}(\delta)$. |
| − | Turn right by angle $\delta$. The rotation matrix is equal to $\mathbf{R_U}(-\delta)$. |
| & | Pitch down by angle $\delta$. The rotation matrix is equal to $\mathbf{R_L}(\delta)$. |
| ^ | Pitch up by angle $\delta$. The rotation matrix is equal to $\mathbf{R_L}(-\delta)$. |
| \ | Roll left by angle $\delta$. The rotation matrix is equal to $\mathbf{R_H}(\delta)$. |
| / | Roll right by angle $\delta$. The rotation matrix is equal to $\mathbf{R_H}(-\delta)$. |
| \| | Turn away. The rotation matrix is equal to $\mathbf{R_U}(180°)$. |
| # | Increment current line width according to the formula: $w' = w + w\_inc$. |
| ! | Decrement current line width according to the formula: $w' = w - w\_inc$. |
| ' | Increment current color according to the formula: $c' = c + c\_inc$ |
| ‘ | Decrement current color according to the formula: $c' = c - c\_inc$ |
| { | Start saving the subsequent positions of the turtle as the vertices of a polygon to be filled. |
| } | Fill the saved polygon using the current color $c$. |
| [ | Push current state of the turtle into a pushdown stack. |
| ] | Pop a state from the stack, and make it the current state of the turtle. No line is drawn, although the position of the turtle may change. |

All other symbols are ignored by the turtle (the turtle preserves its state).

*Remark 3.1.* The above definition specifies only the essential aspects of the turtle's behavior. Specifically, exception handling is left undefined and may depend on the particular implementation of the turtle. An exception occurs, for example, if the line width decreases to a negative value or if an attempt to pop a state from the empty stack is made.

(*) In the actual software implementation of the described concepts on a SUN 2/160 workstation, $w$ is a parameter to the SunCore function set_linewidth, and $c$ is the index to the color table. (SunCore is a trademark of Sun Microsystems, Inc.)

*Definition 3.2.* Let $s_0 = < \vec{P}_0, \vec{H}_0, \vec{L}_0, \vec{U}_0, w_0, c_0 >$ be the initial state of the turtle, and suppose that $d$, $\delta$, *w_inc* and *c_inc* are fixed parameters. The object (set of lines and polygons) created by the turtle responding to a word $\nu$ is called the **turtle interpretation** of $\nu$.

## 4. OBJECT MODELING USING L-SYSTEMS.

This section presents examples of three-dimensional objects modeled using L-systems, and explains the techniques which were used to construct these L-systems.

*Example 4.1.* Consider a cube divided into 27 equal "subcubes", and remove from it seven subcubes according to Fig. 1. Apply this process recursively to the remaining twenty subcubes. The object resulting from this construction is called the *Menger sponge* [Menger 1932, Mandelbrot 1982]. The following T0L-system generates the sponge shown in Fig. 2. In order to make this system easier to analyze, identifiers corresponding to Fig. 1 are used instead of single letters.

| | | |
|---|---|---|
| $\omega$: | **SPONGE** | |
| *p*1: | **SPONGE** → **[SLICE −1+ slice −1+ SLICE]** | |
| *p*2: | **SLICE** → **[ROW 2 row 2 ROW]** | |
| *p*3: | **slice** → **[row 2 2 row]** | |
| *p*4: | **ROW** → **[SPONGE ^3& SPONGE ^3& SPONGE]** | |
| *p*5: | **row** → **[SPONGE ^33& SPONGE]** | (4.1) |
| *p*6: | 1 → 2 | |
| *p*7: | 2 → 3 | |
| *p*8: | 3 → 111 | |
| *p*9: | **SPONGE** → [{*f*+*f*+*f*+*f*+}*f*&′{*f*+*f*+*f*+*f*+}/′{*f*+*f*+*f*+*f*+}] | |
| *p*10: | 3 → *f* | |

The set of productions $P$ is divided into two tables. Table $t_1$ contains productions $p_1 - p_8$, plus the necessary identity productions ( [ → [ , − → − , etc.). Table $t_2$ contains productions $p_9$ and $p_{10}$ (and the identity productions for the remaining symbols). The angle increment $\delta$ is equal to 90°.

The system operates as follows. First, the productions of table $t_1$ define the structure of the sponge to the desired level of recursion. These productions are applied in a sequence of **phases.** Each phase corresponds to a well-defined modification of the generated object and is characterized by a particular subset of the productions used.

- *Phase 1* (productions $p_1$ and $p_8$).

  Decompose each sponge into three slices according to Fig. 1. (Obviously, there is only one sponge in the first derivation step.) Note that the external slices are different from the internal one. Consequently, they are denoted by distinct identifiers: **SLICE** and **slice**.

- *Phase 2* (productions $p_2$, $p_3$ and $p_6$).

  Decompose each slice into rows.
- *Phase 3* (productions $p_4$, $p_5$ and $p_7$).

  Decompose each row into sponges.

In general, productions $p_1 - p_5$ describe the recursive structure of the sponge, while productions $p_6 - p_8$ are responsible for placing the sponge components at the appropriate distances from each other.

After the sponge structure has been defined to the desired level of recursion, productions of table $t_2$ are used to present this sponge as collection of adjacent small cubes. Actually, the production $p_9$ constructs only three front faces of each cube. The back faces are neglected, since they are always invisible by virtue of being away from the observer.

*Example 4.2*. The following L-system generates the *bush* shown in Fig. 3.

$\omega$: **apex**
*p*1: **apex** $\rightarrow$ **[branch** //////′ **branch** ///////′ **branch]**
*p*2: **branch** $\rightarrow$ **[& stem leaf ! apex]** (4.2)
*p*3: **stem** $\rightarrow$ *F* **leaf**
*p*4: *F* $\rightarrow$ *F*///// **stem**
*p*5: **leaf** $\rightarrow$ [′′′^^{−*f*+*f*+*f*−|−*f*+*f*+*f* }]

(The angle increment $\delta$ is equal to 22.5°.) The system operated as follows. Production $p_1$ creates three branches from an apex. Production $p_2$ describes a branch as consisting of a stem, a leaf and the apex which will subsequently create three new branches. Productions $p_3$ and $p_4$ specify the growth process of a stem: in the subsequent derivation steps the stem gets longer and produces new leaves. Finally, production $p_5$ specifies a leaf as a filled polygon with six edges.

The growth of the stem requires an additional comment. Basically, productions $p_3$ and $p_4$ have the structure of the simple L-system:

$$\begin{aligned} \omega&: \quad a \\ p1&: \quad a \rightarrow b \qquad (4.3) \\ p2&: \quad b \rightarrow ba \end{aligned}$$

It is known that L-system (4.3) generates strings of lengths expressed by consecutive terms of the Fibonacci series (1,1,2,3,5,8,...) [Salomaa 1973]. Consequently, the lengths of the stems generated by (4.2) increase according to the same series.

Although various manifestations of the Fibonacci series occur frequently in nature [Stevens 1974], the particular growth rate characterizing the bush shown in Fig. 3 was chosen purely because of aesthetic reasons. The object shown in Fig. 3 was supposed to look like a bush, but no attempt was made to model any existing species.

*Example 4.3*. An L-system modeling a *plant* with leaves and flowers (Fig. 4) is given below:

```
ω:    plant
p1:   plant → stem + [plant + flower] − − //
                [− − leaf ] stem [+ + leaf ] −
                [plant flower] + + plant flower
p2:   stem → F seg [//&& leaf ] [//^^ leaf ] F seg
p3:   seg → seg F seg                                          (4.4)
p4:   leaf → ['{+f−ff−f+|+f−ff−f}]
p5:   flower → [&&& pedicel '/ wedge //// wedge ////
                wedge //// wedge //// wedge]
p6:   pedicel → FF
p7:   wedge → ['^F][{&&&&−f+f |−f+f}]
```

(The angle increment $\delta$ is equal to 18°). This L-system can be described and analyzed in a way similar to (4.2). The exponential growth of the stems (specified by production $p_3$) resulted in the "best-looking" plant in this case.

## 5. STOCHASTIC L-SYSTEMS.

Figures 3 and 4 show single specimens of plants generated using L-systems. A natural question is how to model more specimens (for example, in order to create a field covered with flowers). An obvious approach is to use a given L-system repetitively. Unfortunately, due to the deterministic nature of the L-systems considered, all generated plants would be identical. An attempt to include them in the same figure would produce an artificial, eye-striking regularity. In order to prevent this effect it is necessary to introduce specimen-to-specimen variations.

Specimen variation can be achieved by randomizing the interpretation function, the L-system, or both. Randomization of the interpretation function alone has a limited effect. While the geometric aspects of a plant - such as the stem sizes or the branching angles - are modified, the underlying topology remains unchanged. In contrast, a stochastic application of productions may affect both the topology and the geometry of the plant (by randomizing the branching patterns as well as the growth rates of the plant parts). Consequently, this approach was selected for implementation. At present it is not clear whether a randomization of the L-system *and* the interpretation function would offer any further improvement.

*Definition 5.1*. A **stochastic 0L-system** is an ordered quadruplet $G_\pi = < V, \omega, P, \pi >$. The alphabet $V$, the axiom $\omega$ and the set of productions $P$ are defined as in a 0L-system (Definition 2.1). Function $\pi: p \rightarrow R^+$, called the **probability function**, maps the set of productions into the set of positive real numbers. The values of $\pi$ are called the **probability factors**.

*Definition 5.2*. Let $\hat{P}(a) \subset P$ denote the subset of all productions of $P$ which have the predecessor $a$. We will call the derivation $\mu \Rightarrow \nu$ a **stochastic derivation** in $G_\pi$ if for each *occurrence* of the letter $a$ in the word $\mu$ the probability of selecting production

$p_i \in \hat{P}(a)$ is defined as:

$$prob(p_i) = \frac{\pi(p_i)}{\sum\limits_{p_k \in \hat{P}(a)} \pi(p_k)} \tag{5.1}$$

*Note 5.1.* According to the above definition, different productions with the same predecessor can be applied to various occurrences of the same letter in one derivation step.

*Note 5.2.* For another method of introducing probabilistic aspects to the L-systems see Jürgensen [1976].

*Example 5.1.* A simple example of a stochastic L-system is given below.

$$\begin{array}{lll} \omega: & F & \\ p1: & F \rightarrow F['+F]F['-F]F & 0.3 \\ p2: & F \rightarrow F['+F]F & 0.3 \\ p3: & F \rightarrow F['-F]F & 0.3 \end{array} \tag{5.2}$$

The probability factors are listed after the productions. According to the formula (5.1), each production can be selected with the same probability:

$$prob(p_1) = prob(p_2) = prob(p_3) = \frac{0.3}{0.3 + 0.3 + 0.3} = 0.33 \tag{5.3}$$

Examples of objects generated by the L-system (5.2) in derivations of length 5 are shown in Fig. 5. Note that these objects look like different specimens of the same species.

*Example 5.2.* The *flower field* presented in Fig. 6 consists of four rows and four columns of plants. All plants are generated by a stochastic modification of the L-system (4.4). The essence of this modification is to replace production $p_3$ from (4.4) with the following three productions:

$$\begin{array}{lll} p3': & \textbf{seg} \rightarrow \textbf{seg}\ [//\&\&\ \textbf{leaf}\ ]\ [//\hat{}\hat{}\ \textbf{leaf}\ ]\ F\ \textbf{seg} & 0.3 \\ p3'': & \textbf{seg} \rightarrow \textbf{seg}\ F\ \textbf{seg} & 0.3 \\ p3''': & \textbf{seg} \rightarrow \textbf{seg} & 0.3 \end{array} \tag{5.4}$$

Thus, in any step of the derivation, the stem segment **seg** may grow and produce new leaves (production $p_3'$), grow without producing new leaves (production $p_3''$), or not grow at all (production $p_3'''$). All three events occur with the same probability.

## 6. EXTENDING THE TURTLE INTERPRETATION WITH CUBIC SURFACES.

One of the essential limitations of the modeling mechanism considered so far is related to the representation of surfaces. The concept of filling the area inside the contour traced by the turtle is well defined only in the case of planar, closed and non-self-intersecting polygons. In biological applications, flat polygons can be considered as adequate approximations of small leaves or petals (Figs. 4, 6 and 7). However, in other cases the use of curved surfaces may be essential.

The leaves of the plant showed in Fig. 8 were outlined by the turtle moving in three dimensions. The interior areas of these leaves were filled on the projection plane. Unfortunately, such approach has several drawbacks. A contour alone does not completely define a curved surface which, in consequence, cannot be properly rendered. The problem of specifying a closed three-dimensional line as a sequence of turtle commands is difficult. Furthermore, the obtained shape cannot be easily modified (as opposed to a real leaf which can be bent, twisted, etc.)

A more general and flexible approach makes use of the standard method for representing curved surfaces with cubic patches [Foley and Van Dam 1982]. Their shape is predefined using an interactive patch editor. The alphabet of the L-system is extended to include symbols representing different patch shapes. When the turtle encounters a patch symbol while interpreting the string, the corresponding patch is incorporated into the model. Its position, orientation and color are determined by the current state of the turtle. Thus, the L-system controls the appearance of the surfaces and their distribution in space, while the shape of the surfaces is defined outside the conceptual framework of L-systems.

*Example 6.1.* (prepared by Jim Hanan) Fig. 9 shows a wire-frame representation of a Bezier patch which models a leaf. A plant incorporating these leaves is shown in Fig. 10. Its flowers were modeled using Bezier patches as well.

## 7. CONCLUSIONS.

This paper presents a method for modeling three-dimensional objects using L-systems. The basic idea is to generate a string of symbols using an L-system and to interpret this string as a sequence of commands which control a LOGO-like turtle. Examples of the generated objects include a three-dimensional fractal and several plants. The basic concepts are extended in two directions. Stochastic L-systems are applied to model many specimens of the same species. Cubic patches are used to enhance the turtle interpretation by incorporating curved surfaces into the model. In general, the presented method makes it possible to generate complex objects from concise and intuitive specifications.

Many interesting problems are open for a further study. The first group of problems is related to the modeling and rendering of plants. While the cubic patches present an obvious improvement over flat polygonal surfaces, their application to the modeling of biological structures is still limited. For example, patches are inconvenient to use when modeling flowers with strongly wrinkled petals or leaves with the toothed margins. Patches also lack the texture which characterizes various surfaces of real plants (although it can be added using texture or bump mapping, cf. [Bloomenthal 1985] and [Oppenheimer 1986]). Consequently, a developmental model properly representing the shapes and textures (venations) of leaves as well as the color structures of flowers would find an immediate practical application in computer imagery.

Since L-systems describe the process of growth, they appear to be well suited for the computer animation of plant development. The problem is that the subsequent frames must be relatively similar to each other in order to produce a good impression of continuous growth. Sequences of plant images generated by L-systems usually do not satisfy this condition. In other words, the plant development stages corresponding to the derivations of length *n* and *n*+1 are too different to appear in consecutive frames. It is therefore important to devise a technique which would provide for an arbitrarily small rate of change occurring between the consecutive frames. Additionally, the overall growth curves should correspond to those observed in nature (cf. [Vitányi 1986]).

The faithful modeling of plants actually observed in nature is yet another problem. In the context of this paper, its solution involves a good description of the plant in terms of an L-system. However, only a relatively small number of plants have been described this way so far (for examples see [Frijters and Lindenmayer 1976, Frijters 1978a and 1978b]).

Graphical applications of L-systems present a number of interesting mathematical problems. One of them is to find the L-systems which generate space-filling curves in three dimensions. Several other problems were listed by Szilard and Quinton [1979] and Prusinkiewicz [1986].

## ACKNOWLEDGMENT.

I would like to thank Prof. Zdzislaw Pawlak for his crucial encouragement. Stimulating discussions with Prof. Aristid Lindenmayer, Dr. Hermann Lück and Dr. Jacqueline Lück clarified the relationship between my work and the existing results in developmental models. Dr. Alvy Ray Smith brought to my attention the paper by Hogeweg and Hesper and kindly provided explanations related to his program Gene. Dr. Grzegorz Rozenberg brought to my attention the paper by Szilard and Quinton. This work could not have been done without the support and facilities provided by the Department of Computer Science at the University of Regina. The support from the Natural Sciences and Engineering Research Council of Canada (grant number A0324) is also gratefully acknowledged.

## REFERENCES.

Abelson, H., and diSessa, A. A. [1982]: *Turtle geometry*. M.I.T. Press, Cambridge and London.

Aono, M, and Kunii, T. L. [1984]: Botanical tree image generation. *IEEE Computer Graphics and Applications* **4**, Nr. 5, pp. 10-34.

Bloomenthal, J. [1985]: Modeling the Mighty Maple. *Computer Graphics* **19**, Nr. 3, pp. 305-311.

Eyrolles, G. [1986]: *Synthèse d'images figuratives d'arbres par des méthodes combinatoires*. Ph.D. Thesis, Université de Bordeaux I.

Foley J.D., and Van Dam, A. [1982]: *Fundamentals of interactive computer graphics*. Addison-Wesley, Reading.

Freeman H. [1961]: On encoding arbitrary geometric configurations. *IRE Trans. Electron. Comput.* **10**, pp. 260-268.

Frijters, D. and Lindenmayer, A. [1976]: Developmental descriptions of branching patterns with paracladial relationships. A. Lindenmayer and G. Rozenberg (Eds.): *Automata, languages, development*, North-Holland, Amsterdam - New York - Oxford, pp. 57-73.

Frijters, D. [1978a]: Principles of simulation of inflorescence development. *Annals of Botany* **42**, pp. 549-560.

Frijters, D. [1978b]: Mechanisms of developmental integration of *Aster novae-angliae* L. and *Hieracium murorum* L. *Annals of Botany* **42**, pp. 561-575.

Herman, G. T., and Rozenberg, G. [1975]: *Developmental systems and languages*. North-Holland, Amsterdam and Oxford.

Hogeweg, P., and Hesper, B. [1974]: A model study on biomorphological description. *Pattern Recognition* **6**, pp. 165-179.

Jürgensen, H. [1976]: Probabilistic L-systems. In A. Lindenmayer and G. Rozenberg (Eds.): *Automata, languages, development*, North-Holland, Amsterdam - New York - Oxford, pp. 211-225.

Kawagushi, Y. [1982]: A morphological study of the form of nature. *Computer Graphics* **16** , Nr. 3, pp. 223-232.

Lindenmayer, A. [1968]: Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology* **18**, pp. 280-315.

Mandelbrot, B. B. [1982]: *The fractal geometry of nature*. W. H. Freeman, San Francisco.

Menger, K. [1932]: *Kurventheorie*. Leipzig - Berlin.

Oppenheimer, P. [1986]: Real time design and animation of fractal plants and trees. *Computer Graphics* **20** , Nr. 4, pp. 55-64.

Papert, S. [1980]: *Mindstorms: children, computers, and powerful ideas*. Basic Books, New York.

Prusinkiewicz, P. [1986]: Graphical applications of L-systems. *Proceedings of Graphics Interface '86 - Vision Interface '86*, pp. 247-253.

Reeves, W. T., and Blau, R. [1985]: Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Computer Graphics* **19** , Nr. 3, pp. 313-322.

Rozenberg, G., and Salomaa, A. [1980]: *The mathematical theory of L-systems*. Academic Press, New York and London.

Salomaa, A. [1973]: *Formal languages*. Academic Press, New York and London.

Siromoney, R., and Subramanian, K. G. [1983]: Space-filling curves and infinite graphs. In H. Ehrig, M. Nagl and G. Rozenberg (Eds.): *Graph grammars and their application to computer science*. Springer-Verlag, Berlin - Heidelberg - New York - Tokyo.

Smith, A. R. [1978]: About the cover: "Reconfigurable machines". *Computer* **11**, Nr. 7, pp. 3-4.

Smith, A. R. [1984]: Plants, fractals, and formal languages. *Computer Graphics* **18**, Nr. 3, pp. 1-10.

Stevens, P. S. [1974]: *Patterns in nature*. Little, Brown and Co., Boston.

Szilard, A. L., and Quinton, R. E. [1979]: An interpretation for DOL systems by computer graphics. *The Science Terrapin* **4**, pp. 8-13.

Vitányi, P.M.B. [1986]: Development, growth and time. In G. Rozenberg and A. Salomaa [Eds.]: *The book of L*, Springer-Verlag, Berlin - Heidelberg - New York - Tokyo, pp. 431-444.

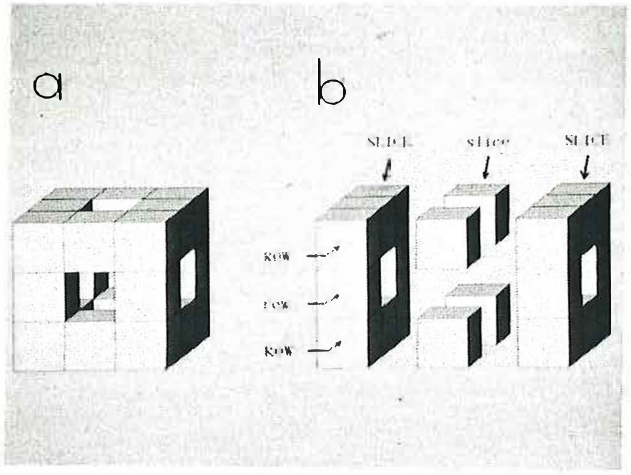

Fig. 1. Construction of the Menger sponge.

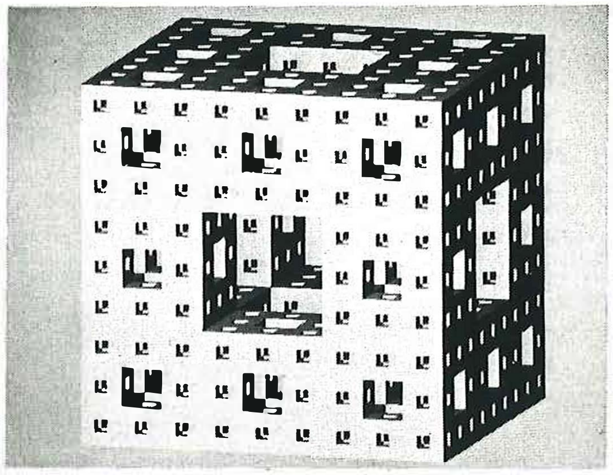

Fig. 2. The Menger sponge.

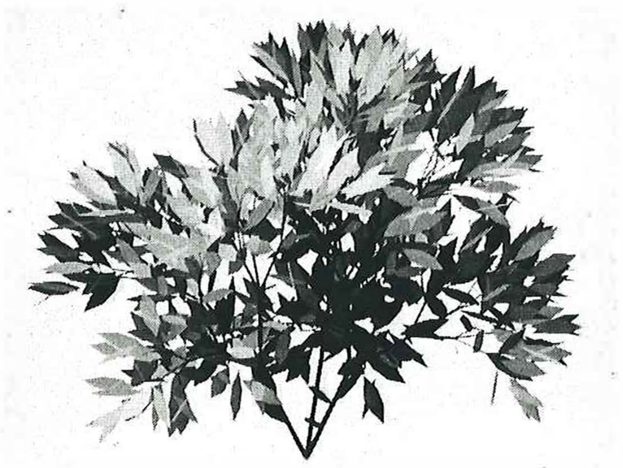

Fig. 3. The bush generated by L-system (4.2).

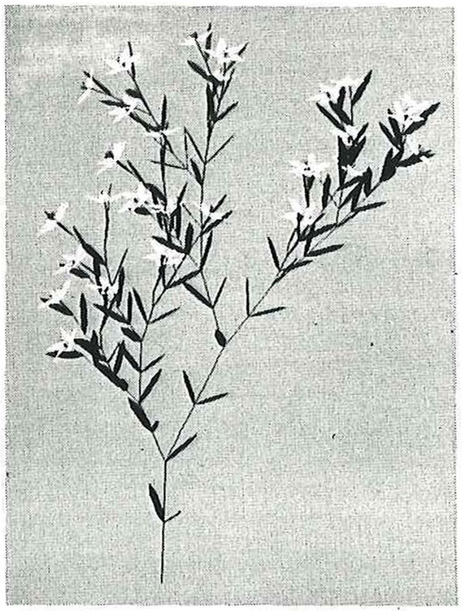

Fig. 4. The plant generated by L-system (4.4).

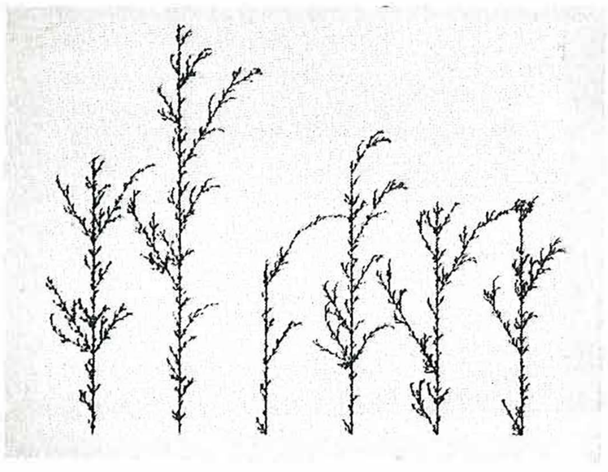

Fig. 5. Sample objects generated by the stochastic L-system (5.2).

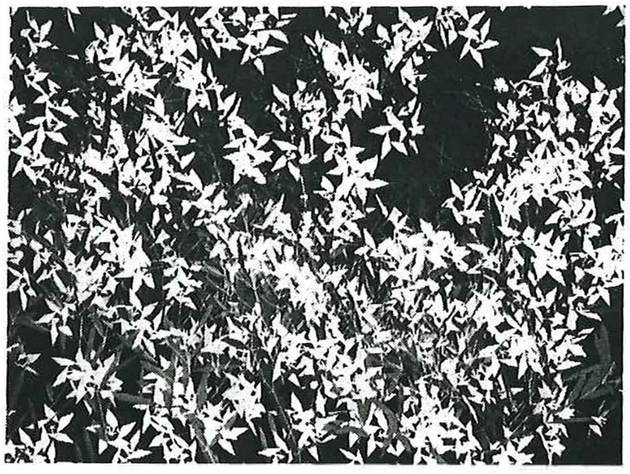

Fig. 6. The flower field.

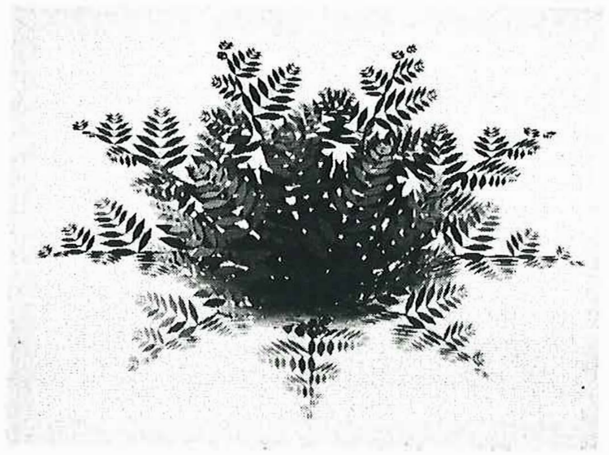
Fig. 7. A plant with small leaves approximated by flat polygons.

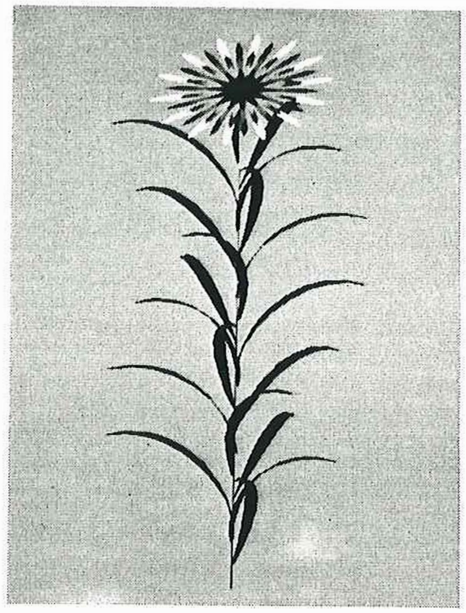
Fig. 8. A plant with curved leaves.

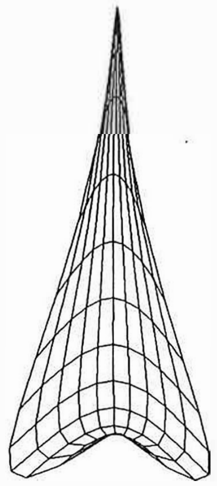
Fig. 9. A Bezier patch modeling a leaf.

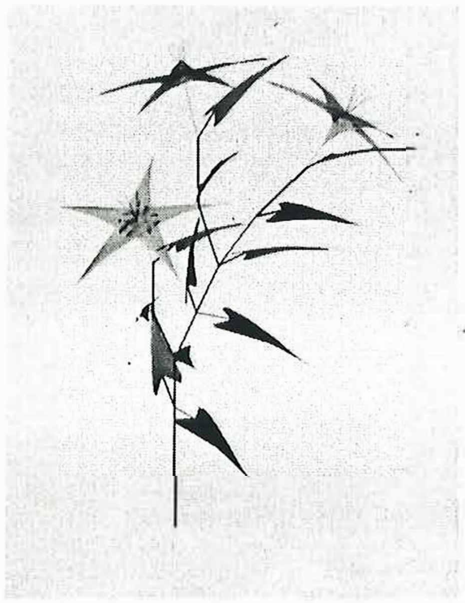
Fig. 10. A plant with leaves and flowers modeled using Bezier patches.