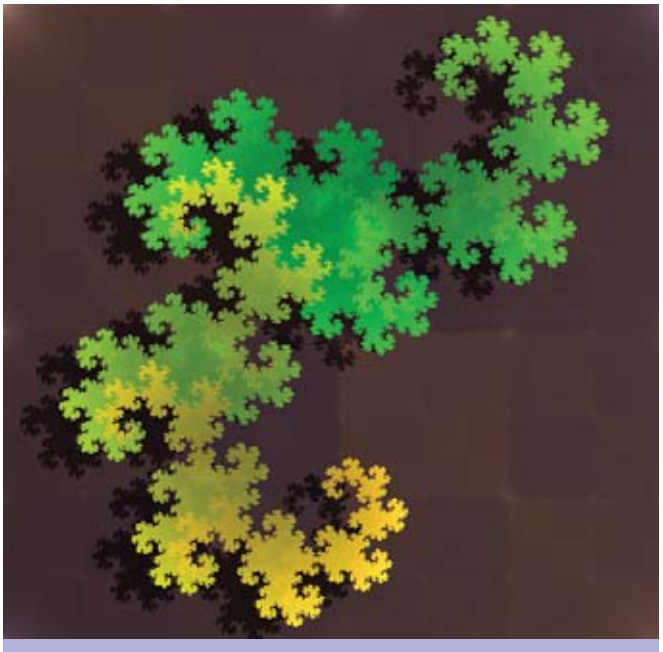
Viewpoint MODELING

By **GERGELY VASS**

Formal Modeling Grammars

The sheer complexity of trees, plants, or cities makes it impossible to model those structures manually for computer graphics applications. To reproduce the rich structure of such formations, procedural techniques are used, most of which are based on a set of very simple rewriting rules. These systems (such as the L-system) are based on formal grammars, a fundamental concept first introduced in linguistics and now studied in a branch of mathematics, the formal language theory.



Above is an image showing iterations of the construction of the Serpinksi triangle using L-system rules.

Formal Languages

Formal and natural languages (such as English) generally have two main aspects: syntax and semantics. The first is what the language "looks like," in so far as what are valid and what are invalid words or sentences; the latter refers to the meaning of various utterances. Formal language theory is only concerned with the syntax of a language and completely ignores meaning. The precise description of a language is called the grammar (or formation rules).

When thinking about the uses of grammars, our first idea usually is the recognition of error—for instance, not spelling a word properly or using wrong punctuation. However, grammars are also regarded as a means to generate valid strings of a language. When learning a foreign language, we not only memorize basic words but also a set of rules on how to inflect them or form sentences with them. The rules of grammar help us to create valid utterances of which we may never have even heard.

Gergely Vass is a software developer in the Image Science Team of Autodesk Media and Entertainment, where his research focus includes image processing and computer vision. Located in Budapest, Hungary, Vass can be reached at gergely.vass@autodesk.com.

A formal generative grammar is essentially a set of rules that we can apply to strings (sequences of symbols) to transform them. Triggering an applicable rule one after the other, we can generate any valid combination of words in the language. Generative grammar originates in the work of an American linguist, Noam Chomsky, beginning in the late 1950s. It is considered to be one of the most significant contributions to the field of linguistics made in the 20th century.

But why would it be significant to other fields of science, like computer graphics? While in linguistics the alphabet is the set of letters or set of words of a particular language, in other applications of formal grammars we can define an arbitrary alphabet: a set of algae cells, commands of a programming language, or 3D modeling operations. Using such symbols as the building blocks of the language, we have a powerful tool to express very complex structures and patterns. Formal grammars are used to detect erroneous HTML (or XML) code of Web pages, identify syntax problems in computer programs, or to simulate biological formations, such as realistic neural cells.

Non-linguistic Application

As mentioned earlier, a grammar consists of a set of re-write rules. The synthesis of valid strings begins with only a single start symbol,

and then we successively apply the rules (any number of times, in any order) to transform this string. By definition, the corresponding language consists of all the strings that can be generated in this manner.

Let's consider a simple example of a "sandwich grammar," where we attempt to generate a proper breakfast. Our symbols are not letters, but ingredients. Let the start symbol be [sandwich], and the production rules (the grammar) be:

1. *[sandwich]* ➡ *bread, [inside], bread*
2. *[inside]* ➡ *salad, [meat], cheese*
3. *[meat]* ➡ *ham*
4. *[meat]* ➡ *sausage*
5. *[meat]* ➡ *[meat], [meat]*

I indicated the so-called non-terminal symbols by [ ], which are abstract elements—to be replaced at some point with a "real" symbol—expressing structure in the language. The production of the breakfast in this case should start with rule one, and then with rule two.

[sandwich] ➡ *bread, [inside], bread* ➡ *bread, salad, [meat], cheese, bread*

As this example is a non-deterministic grammar, we can generate different sandwiches. Choosing a certain order of generation rules we can produce:

bread, salad, ham, cheese, bread
bread, salad, ham, sausage, cheese, bread
bread, salad, sausage, sausage, sausage, sausage, cheese, bread

It would be easy to prove, as well, that vegetarian sandwiches are not part of the language (sorry) and the top and the bottom of a sandwich is always bread. My intention for using this simple example is to illustrate that a grammar of basic re-writing rules can express extremely complex formations.

L-system

In 1968, not much after Chomsky published his concepts of formal grammars, Aristid Lindenmayer (a Hungarian biologist) introduced a slightly modified production mechanism. He attempted to model the growth patterns of multicellular organisms, like algae, and created the Lindenmayer system. In the L-system (as it is mostly referred to), multiple rules are applied simultaneously in each iteration step. This parallel string-rewriting mechanism of the production procedure makes it possible to realistically describe the development of branching structures. The resulting strings can be interpreted geometrically and visualized using computer graphics techniques to create realistic renderings of the modeled plants, trees, or other complex structures.

This CG plant with tiny hairs was generated with an L-system.

An L-system, just like our sandwich grammar, is also defined by a set of symbols and production rules. For computer graphics applications, these symbols are mostly geometrical primitives, and the generated strings represent buildings, cities, trees, or flowers. It is often desirable to reproduce the natural randomness in the growing process, such as leaves or branches, thus non-deterministic rules are used. By assigning probability percentages to the individual rules, we create a stochastic L-system. To avoid artificial regularity of the variations, which may be striking when synthesized and real plants are displayed side by side, the L-systems randomize the rules and their interpretation at the same time. The first technique affects the topology of the plant, while the latter—randomizing the stem lengths and branching angles—varies the geometric aspects.

The computing power of modern game consoles and PCs make it possible to use huge and detailed virtual environments in computer games. Creating these assets manually takes a lot of effort, a growing obstacle for game development studios. To address this issue, L-systems often are used to generate random yet controllable virtual objects. Building and city generation has been one of the first applications of the Lindenmayer system. By analyzing the statistical properties of the architecture of well-known and unique buildings or city layouts, new variations resembling the original can be produced.

The L-system is a great example of how innovations of a very distant field, like linguistics, can be utilized in computer graphics. It should not be surprising, though. Capturing and reproducing the complexity of nature is a fundamental task in all areas of science. And there is still great room for improvement. ■