International Journal of Machine Intelligence, ISSN: 0975–2927, Volume 1, Issue 2, 2009, pp- 34-37

Intelligent approach of modeling self similar plants and trees using parallel string rewriting techniques

Bana Bihari Mohanty[1]*, Saroja Nanda Mishra[2] and Srikanta Pattanayak[3]
[1]*Women's Polytechnic, Dhenkanal, Orissa, India
[2]IGIT, Saranga, Dhenkanal, Orissa, India
[3]IIMT, Bhubaneswar, Orissa, India

Abstract- Plants and Trees are such natural objects which exhibit the property of self similarity. There are many methods developed in computer graphics to generate these self similar objects. Efforts are still going on to formulate new techniques to generate such objects with minimal efforts and time. Parallel string rewriting method has been used to develop fractal formalism in computer graphics. This string rewriting technique can be represented in graphical form through turtle graphics. In this paper it has been investigated and experimented to generate objects like plants and trees using parallel string rewriting formalism.
Key words- Graphical formalism, string rewriting grammar, self similar object, Plant modeling, parallel grammar

Introduction

While the shape and structures of many objects like plants and trees often appear irregular and chaotic, they also exhibit a high degree of self-similarity. Self-similarity suggests a recursive or iterated approach to the modeling of specific form of presentation. As such, constructs such as fractal geometry and Lindenmayer systems seem prime candidates for the generation of such self similar objects. Simply saying, these self similar objects are geometric patterns that exhibit self-similarity on all scales, at any level of magnification. In this paper we have shown that objects such as plants and trees with property of self similarity can be better modeled using parallel string rewriting principle as this involves recursive approach of generating object specification. While implementing L-system string rewriting method to generate different graphical objects; there are multiple variables or production rules used [1, 6]. It has been investigated and experimented by us those self similar objects can be generated using context free grammar through single and multiple production rules in a much simplified manner. In this paper we have shown that it is possible to generate self similar graphical objects like plants and trees using parallel string rewriting grammar.

Problem of Research

A plant or Tree can be described as an arrangement of functional modules such as stem, trunks, leaves etc. For individual plant their modules share common traits, which are otherwise called self similarity. In computer graphics visual models are often described using scene graphs. These graphs consist of various primitives like line, triangle, cylinder etc. and also their transformations. The transformations define the arrangements by displacements and rotations. The theory of formal languages deal with grammars; consisting of an alphabet of basic symbols and rules for describing the synthesis of words from these symbols. In this paper it is investigated and explored as to how formal language and its grammar can be used to model the plant and trees having the property of self similarity.

String Rewriting

A. Context sensitive & Context free grammar

The basic functionality of an L-Systems model consists of an initial axiom and a set of production rules which are iteratively applied to that axiom. The first iteration starts off with the axiom as the current string, and then during any subsequent iteration, the current string is examined for single characters which have associated production rules. If such a rule exists, the string which corresponds to that letter, as defined by the set of rules, is substituted into the string. Each successive iterations produces a longer and more complicated string of characters, which can then be read in as a set of sequential drawing commands to produce an abstract view of the object. In order to model the plants and trees which occur in nature, several different kinds of modeling methods have been developed. Context-free rule based systems are the simplest and most common type of implementation available. Context-sensitive rules, on the other hand, create models whose segments can be affected by their surrounding neighbors [5]. This means that the way a particular element is drawn is affected by its proximal environment. This allows for the correct modeling of objects whose components are affected by their predecessors. In this paper we shall consider the context free grammar.

B. String rewriting using L-system

Basic definitions

Lindenmayer systems are a particular type of symbolic dynamical system with the added feature of a geometrical interpretation of the evolution of the system. The limiting geometry of even very simple systems can be extraordinary fractals. The components of an *L*-system are as follows:

Alphabet:

The *alphabet* is a finite set *V* of formal symbols, usually taken to be letters *a*, *b*, *c*, etc., or possibly some other characters.

Axiom:

The *axiom* also called the *initiator* is a string w of symbols from *V*. The set of strings also called *words* from *V* is denoted *V**. Given V= {a,b,c}, some examples of words are *aabca*, *caab*, *b*, *bbc*, etc. The *length* of a word *w* is the number of symbols in the word.

Production:

A *production* also called *rewriting rule* is a mapping of a symbol α є V to a word w є V*. This will be labelled and written with notation:

P: α → w

We allow as possible productions mappings of α to the *empty word*, denoted φ, or to α itself. If a symbol α є V does not have an explicitly given production, we assume it is mapped to itself by default. In that case, α is a *constant* of the *L*-system.

C. Geometric interpretation

In computer science terms, an L-System is a context-free, recursive, text substitution scheme, followed by geometric interpretation. A simple L-System starts with a seed, let's say the letter F, and has one rule to replace the existing seed. A simple replacement rule might be:
F-F-FF+F-F-F

This rule would be applied many times to produce a series of strings of increasing complexity. Examples will be given below after we talk more about the meaning of the text. It can treat the long text strings resulting from multiple replacements as commands to build objects. In two dimensions, the interpretation is:

- Start at x=0, y=0, facing along the x-axis.
- Capital F means "draw a line of length=1 in the direction you are now facing".
- + means turn left.
- -means turn right.

Let's look at the result of applying the rule in the first paragraph 3 times. The string length grows *fast*. The three output strings are shown below, followed by the resulting geometric interpretation, assuming that left and right turns are each 90 degrees.

1st Resulting string becomes
F-F-FF+F-F-F

2nd Resulting string becomes
F-F-FF+F-F-F-F-F-FF+F-F-F-F-F-FF+F-F-FF-F-
FF+F-F-F+F-F-FF+F-F-F-F-F-FF+F-F-F-F-F-
FF+F-F-F

3rd Resulting string becomes
F-F-FF+F-F-F-F-F-FF+F-F-F-F-F-FF+F-F-FF-F-
FF+F-F-F+F-F-FF+F-F-F-F-F-FF+F-F-F-F-F-
FF+F-F-F-F-F-FF+F-F-F-F-F-FF+F-F-F-F-F-
FF+F-F-FF-F-FF+F-F-F+F-F-FF+F-F-F-F-F-
FF+F-F-F-F-F-FF+F-F-F-F-F-FF+F-F-F-F-F-
FF+F-F-F-F-F-FF+F-F-FF-F-FF+F-F-F+F-F-
FF+F-F-F-F-F-FF+F-F-F-F-F-FF+F-F-FF-F-FF+F-
F-F-F-F-FF+F-F-F-F-F-FF+F-F-FF-F-FF+F-F-
F+F-F-FF+F-F-F-F-F-FF+F-F-F-F-F-FF+F-F-F+F-
F-FF+F-F-F-F-F-FF+F-F-F-F-F-FF+F-F-FF-F-
FF+F-F-F+F-F-FF+F-F-F-F-F-FF+F-F-F-F-F-
FF+F-F-F-F-F-FF+F-F-F-F-F-FF+F-F-F-F-F-
FF+F-F-FF-F-FF+F-F-F+F-F-FF+F-F-F-F-F-
FF+F-F-F-F-F-FF+F-F-F-F-F-FF+F-F-F-F-F-
FF+F-F-F-F-F-FF+F-F-FF-F-FF+F-F-F+F-F-
FF+F-F-F-F-F-FF+F-F-F-F-F-FF+F-F-F

Here are the images resulting from following the geometric instructions for each string. Repetitions 4 and 5 have not been included, otherwise the strings would have filled pages. It can be seen that the application of this simple rule makes very complex structures which are constructed of repeated, modular units, much like a plants.

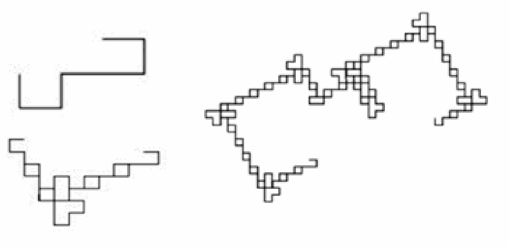

Fig 1: Geometric interpretation of string writing

Plant and Tree Structures

The plant kingdom is dominated by branching structures. For modeling purposes, a mathematical description of tree-like shapes and methods for generating them is required. To begin searching for such descriptions, we start in the realm of graph theory. In graph theory, a rooted tree has edges that are directed and labeled. The edge sequences form paths from an initial node, called the root or base, to terminal or leaf nodes. In a botanical context, these edges are referred to as branch segments. A segment that is followed by at least one more segment in at least one path is called an internode. A terminal segment is called an apex. An axial tree is a special type of rooted tree with the botanically motivated notion of branch axis. At each node of an axial tree, at most one outgoing straight segment is distinguished. All remaining edges are called lateral or side segments. A sequence of segments is called an *axis* if the first segment in the sequence originates at the root of the tree or as a lateral segment at some node, each subsequent segment is a straight segment, and the last segment is not followed by any straight segment in the tree. Together with all its descendants, an axis constitutes a branch, which is itself an axial (sub) tree. Within an axial tree, axes and branches are ordered. The axis originating at the root node of the tree has order zero. An axis originating as a lateral segment of an *n*-order parent axis has order *n*+1. The order of a branch is equal to the order of its lowest-order or main axis. In order to model the branching structures of trees, an L-system can be developed that directly operates on an axial tree. A rewriting rule, or tree production, replaces an existing edge in the tree with a successor axial tree. An example of tree production is depicted in figure 2.

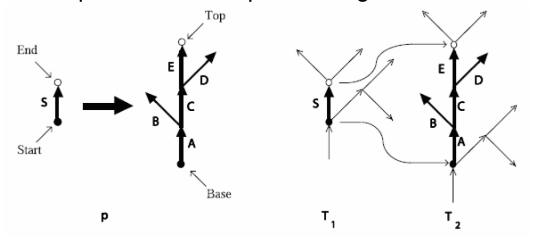

Fig 2: Edge rewriting in a tree using string rewriting

An L-system G describing tree generation is composed of:

V – the alphabet, consisting of a set of edge labels

ω - the axiom, some initial axial tree (may be a single edge, or *trunk*)

P – the productions, a set of tree productions, which replace edges with axial (sub)trees

Provided the L-system G, an axial tree T_2 is derived from a tree T_1 ($T_1 \rightarrow T_2$) if T_2 is obtained from T_1 by simultaneously replacing each edge in T_1 by its successor according to the set of tree productions, P. Furthermore, a general tree T is generated by G in a derivation of length *n* if there exists a sequence of trees $T_0, T_1, T_2, \ldots, T_n$, such that $T_0 = \omega$, $T_n = T$, and $T_0 \rightarrow T_1 \rightarrow T_2 \rightarrow \ldots \rightarrow T_n$.

Plant Modeling Process

Context free string rewriting method can be used to model the plant through its geometric interpretation. This is done by turtle graphics. The axiom and the productions are to be so defined so that there will be realistic looking objects that occur in nature and in particular the branching structure of plants. In this present method of experimentation, one of the important characteristics is that only a small amount of information is required to represent very complex objects. So while the bushes in a tree contain many thousands of lines they can be described in a database by only a few bytes of data, the actual bushes are only grown when required for visual presentation. Using suitably designed axiom string, production rules and the implementing algorithm it is possible to create a model of a particular class of plant. The parallel rewrite system used to model plant growth is a system of formal rules to be applied to a string of characters. The process starts with a 'seed', a string to represent the start state of the System. 'Transforms' are then applied to the string. This process involves replacing set sub strings with new string in accordance to the 'Transform Rules'. The characters of the string belong to the alphabet of the System. They may be drawing commands (to be plotted), describing the shape of the fractal, or characters to aid the growth of the fractal, which are ignored by the plotting system. The following are the alphabets used in modeling of plants in our algorithm. The Algorithm has been implemented using mat lab constructs.

- F move forward and draw a line
- F move forward without drawing a line
- + rotate clockwise a specific angle
- rotate anti-clockwise a specific angle
- [save the current position
-] return to the last saved position

For each tree to be modeled, we also need a set of transform rules, a seed and a angle to apply for rotation commands. Consider a sample case of generating a tree with the following string rewriting technique.

Axiom:F

Angle=30^0

F= F[-FF]F[+FF]F

This rule would be applied many times to generate a tree structure as follows.

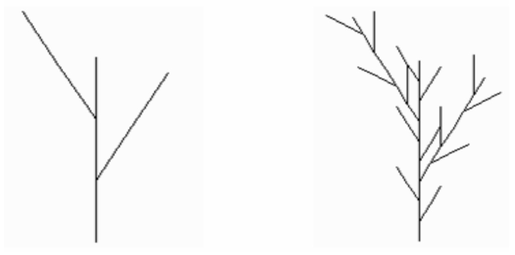

Fig 3: Example of realization of tree

The different plants generated using the techniques through our algorithm are shown below in Figure 4.

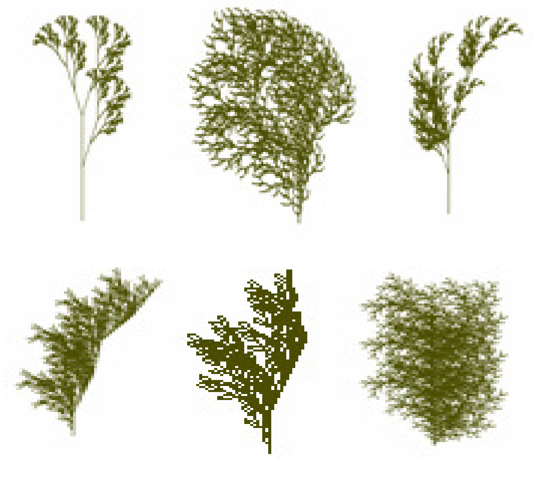

Fig 4: Trees generated through string rewriting

Conclusion

The parallel string rewriting grammar and its use in research and education seems promising. The parallel string rewriting system can be used to generate tree like pictures. The string grammar contains symbols which are used to map into two dimensional figures. The case of self similar objects bears special importance because the whole object is replication of similar objects which involves recursive procedure. Modeling of self similar objects becomes easier through the use of parallel string rewriting principles. To add to this, production rule based system makes it simpler, flexible and easier to generate varieties of self similar objects with modifications in the production rules and axioms. As seen from the experiments made in this exercise, parallel string rewriting grammar can be used for generation of 2D self similar objects like plants and trees easily.

References

[1] Prusinkiewicz P. and Lindenmayer A. (1990) *The Algorithmic Beauty of Plants. Springer, Berlin*.

[2] Lindenmayer A. (1968) *Mathematical models for cellular interactions in development, parts I-II. Journal of Theoretical Biology 1*8 : 280-315.

[3] Zamir M. (2001) *The Journal of General Physiology*, 118, Number 3.

[4] Prusinkiewicz, P. (1984) *Proceedings of Graphics interface* 86, 247-253.

[5] Herman G. and Rozenberg G. (1975) *Developmental systems and languages. North-Holland: Amsterdam*

[6] Prusinkiewicz P. (1986) *Proceedings of Graphics Interface '86 — Vision Interface '86*, 247–253.

[7] Smith A.R. Plants, Fractals, Formal Languages, computer Graphics 18, Nr 3, pp1-10

[8] Mandelbrot B.B. The Fractal geometry of Nature, W.H Freeman, Sanfrancisco

[9] Prusinkiewicz P., Karwowski R., Měch R. and Hanan J. (2000) *Lecture Notes in Computer Science* 1779, 457–464.

[10] Prusinkiewicz P., Mündermann L., Karwowski R. and Lane B. (2001) *In Proceedings of ACM SIGGRAPH 2001*, 289–300.