International Journal of Information & Computation Technology.
ISSN 0974-2255 Volume 2, Number 1 (2012), pp. 1-12

http://www. ripublication.com

Novel Approach of Modeling Self Similar Objects using Parallel String Rewriting Methods through Combined L-System Techniques

[1]Bana Bihari Mohanty, [2]Sasmita Mishra, [3]Saroja Nanda Mishra and [4]Srikanta Pattanayak

[1]Govt. Polytechnic, Dhenkanal, Orissa, India
[2]IGIT, Saranga, Dhenkanal, Orissa, India
[3]IGIT, Saranga, Dhenkanal, Orissa, India
[4]IIMT, Bhubaneswar, Orissa, India
E-mail: bbm_wpd@rediffmail.com

Abstract

Many natural objects appearing in reality, 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. These string rewriting methods are based on L-system techniques which are ordinarily focused on individual axioms and production rules meant for specific graphical image of objects. In this paper it has been investigated and a novel approach have been evolved to generate a new string rewriting method which uses combined L-system techniques, to generate these self similar objects

Keywords: graphical formalism, string rewriting, self similar objects, fractals, parallel grammar, combined L-system

Introduction

Computer graphics based software is good at creating representations of man-made objects using primitives such as lines, rectangles, polygons, and curves in 2D or boxes and surfaces in 3D. These geometric primitives and usual tools for manipulating them typically prove inadequate when it comes to representing most objects found in nature

such as clouds, trees, waves, lightening etc. There has been considerable interest recently in fractal geometry as we find that many processes in the world can be accurately described using this theory. The computer graphics industry is rapidly incorporating these techniques to generate beautiful images as well as realistic natural looking structures. 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 seems prime candidate for the generation of such self similar objects. These self similar objects are geometric patterns that exhibit self-similarity on all scales, at any level of magnification[1][9]. In this paper we have shown that such self similar fractal objects can be better modeled using parallel string rewriting principle as this involves recursive approach of generating object specification[12]. While implementing L-system string rewriting method to generate different graphical objects, there are axioms and production rules used focused to generate specific objects. It has been investigated and experimented by us that self similar objects can be generated using context free grammar through combined L-system using different axioms and production rules in a simplified manner.

Fractal Graphics Formalism and String Rewriting

A Fractal object can be described as an arrangement of functional modules where the modules share common traits which is 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[18]. The formal language and its grammar can be used to model the fractal objects like plant and trees having the property of self similarity using turtle graphics.

Background of L-System string rewriting

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[1][18]. 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 $\alpha \in V$ to a word $w \in V^*$. This will be labeled and written with notation:

$$P: \alpha \rightarrow w$$

We allow as possible productions mappings of α to the empty word, denoted φ, or to α itself. If a symbol $\alpha \in V$ 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.

Context free L-system String

Normally the L-system string is context free which means that the production rule only considers the individual symbol without considering the neighbours. Let us consider the following case:

$V = \{a, b\}$
$W = a$
$P1: a \rightarrow ab$
$P2: b \rightarrow ba$

The sequence of generations runs as follows:

g_0 a
g_1: ab
g_2: abba
g_3: abbabaab
g_4: abbabaabbaababba

The number of organisms clearly doubles each generation, so that $|g_n| = 2^n$. To find out the global processes that emerges out is that, generation g_n does appear at the beginning ofg_{n+1}. If we look at what follows, we eventually conclude that it resembles g_n, except that each *a* has been replaced by a *b* and each *b* has been replaced by an a. For any word *w* in V*, we define R(w) to be the mirror image word where we replace a by *b* and *b* by a. Then the global process is

$$g_{n+1} = g_n R(g_n)$$

This L-system produces an infinite sequence g_∞ of a's and *b*'s that looks like

g_∞ = abbabaabbaabbabaabbabaabbaabbabaab…..

Production rule based rewriting

If there is exactly one production for each symbol, then the L-system is called deterministic and the sequence of generations g_n is uniquely defined as a sequence of elements of V*. If there is more than one production for a given symbol, say a → w1 and a → w2. This is called a stochastic L-system. In the present work, we will consider only deterministic L-systems which is popularly called a deterministic context-free L-system or D0L-system.

The study of generation of formal subsets of V* by means of processes similar to that underlying L-systems originated in formal language theory, advanced by Chomsky as a mathematical way for discussing the formation and evolution of natural languages. For this reason, any subset S of V* is called a language. L-systems languages are examples of a much broader class known as recursively enumerable language in theoretical computer science.

In the production rule based string rewriting the principle is that, there shall be single or multiple Production rules to generate the desired object or figure. In multiple production rule based system, the object can be generated using multiple production rules except the constants. This on one hand makes it easier to formulate the axiom and production rules for generation of the objects, but on the other hand makes it complex and complicated in implementation because of involvement of so many parameters and substitutions. Single production rule makes it simpler and easy to implement in programming constructs. In the present work we have tried to generate varieties of objects which are having the concept of self similarity by parallel string rewriting using both single and multiple production rules. The modeler for such string rewriting was written using Matlab programming.

Geometric interpretation through Turtle graphics

Turtle graphics is a system used for translating a sequence of symbols into the motions of an automaton or turtle on a graphics display[17][15]. This turned out to be an ideal system for giving a geometrical interpretation to the dynamics of L-systems. The basic system is as follows. We fix a step length *d* to be the distance covered by the turtle in one step. We also set δ to be a given angle. We make the following definitions.

F: Draw forward one step in current heading
+: Turn heading by δ counterclockwise
-: Turn heading by δ clockwise

As a first example of the use of these symbols, we introduce the Koch curve L-system:

V = {F, +, -}
W= F
P1: F → F+F--F+F

Without specific productions mentioned for some symbols (in this case + and -), we assume they are constants in the sense that the implied production rule is a→a. Generation 0 is just a straight line; we will assume the heading at the start is along the positive horizontal axis. We take the angle δ to be 60^0. Geometrically, this production means replace a straight line segment *F* by the following arrangement of four line segments.

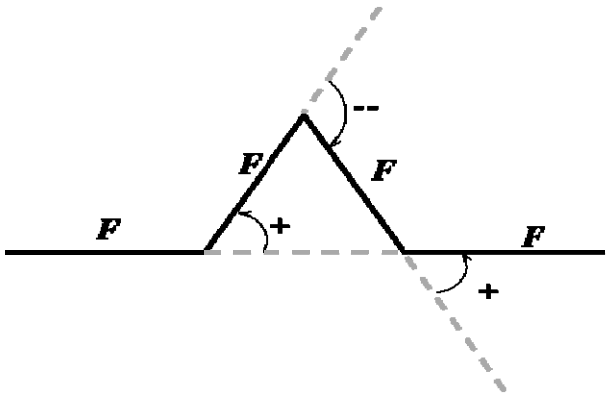


Figure: The production for the Koch curve

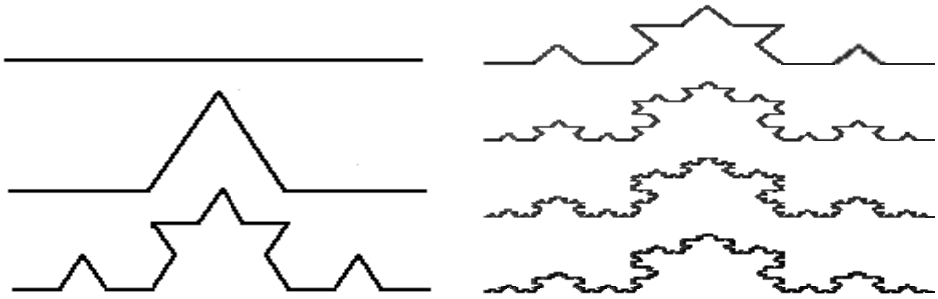


Figure: Koch curve generations

Modeling Process

In the modeling process, we have made use of bracketed L-Systems, in which, line drawing is performed by reading and executing drawing commands (represented as single characters) from the string generated during the iterative character replacement stage. The characters '[' and ']' are use in their production rules to push and pop current position (states) to and from a stack. Utilizing these characters the modeler form structures which branch off of a parent structure without losing the location from which they branch. Popping this state off the stack at the end of the branch resumes the drawing of the parent segment by using this state to set the pen's current drawing properties such as location, line length and current angle. At any particular instant of time, the turtle position or state is determined by the coordinates (x, y). When the turtle moves following the symbol 'F', the new destination coordinate (x', y') will be calculated as $x' = x + d\cos\alpha$ and $y' = y + d\sin\alpha$, where d is the distance or length that the turtle shall move and α is the angle of turtle. When the turtle encounters the '+' the new state of the turtle shall be at the same coordinate but the angle shall be rotated as clockwise δ. The new angle shall be α+ δ. Similarly When the turtle encounters the '-' the new state of the turtle shall be at the same coordinate but the angle shall be rotated as anticlockwise δ. The new angle shall be α- δ.

The Matlab programming constructs have been used to develop the modeler which has less programming complexity. The Axiom, production rule, no. of iteration,

angle, length of movement are taken as input parameters. We have observed that as against other programming language constructs wherein there is requirements of large number of parameters and complicated process of calculation and plotting the drawing, Matlab appears to be a better choice for developing a modeler.

The following are the alphabets used in modeling process..

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

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.

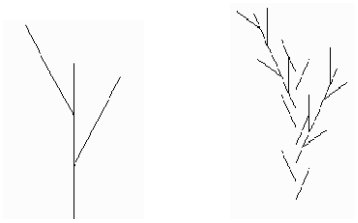


Figure: Example of realization of tree. Proposed Approach of Combined L-system.

One can combine any L-System for any generated fractal object with that of another L-System for any other generated fractal object to formulate a new combined L-System. For example, let L1 represents the L-System for an object and L2 represents the L-System for another object. The possible new combined L-systems which may be formulated are as follows.

Case I: Prefix Combination
Case II: Postfix Combination

The proposed Algorithm can work under the following assumptions and preconditions.

1. Individual L-system L1 and L2 must generate some definite objects.
2. L1 and L2 must not be same
3. Axioms of L1 and L2 shall be used for combination to generate the axiom.
4. Production Rules of both L1 and L2 shall be used for the new system

Algorithm

The proposed algorithm is suggested with the following steps, which shall be executed recursively. First individual L-systems with specific characteristics to be developed using any number of variables. These L-systems shall be tested so as to generate required object using appropriate Axiom and Production Rule. Next the axioms of both the systems to be combined in predetermined order of prefix or postfix to form new combined L-system. After combination, the newly formed L-System may be tested with any turtle graphic L-System implemented software to generate the new class of fractals. The proposed algorithm and all these steps are discussed with few examples below.

Step-1 Formulate base fractal in L-system with Axiom and Production Rule.
Step-2 Test the effectiveness of Axiom and Production Rules of base fractals.
Step-3 Formulate new fractal in L-system by combining the Axioms, production rules and other parameters of base fractals in any predetermined order.
Step-4 Expand recursively the combined L-system with appropriate parallel grammar substitution for the specified iteration.
Step-5 Generate the image of the new L-system using appropriate Modeler.

The Algorithm for the combined L-System have been tested with the software developed to produce various new types of fractals.

Experimentation

To experiment with the proposed algorithm, implementation was done with the following base fractals.

L1: axiom = 'F+F-[FG+FG]-F[F-F]';
Production 'F' = 'FF+[F+F+F]-[F-F-F]';
alpha = 23;

L2: axiom = 'G-[GF]+G+G-G-[G+G]-G+G-G';
Production 'G' = 'F+[F+F-[F-GF]]';
alpha = 23;

CL1: Combined L-system of Prefix order.
CL2: Combined L-system of Postfix order.

Figure: Images after 1st iteration

Image of L1 Image of L2 Image of CL1 Image of CL2

Figure: Images after 2nd iteration

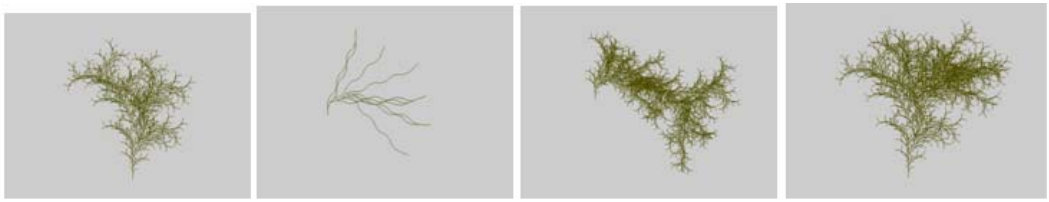


Image of L1 Image of L2 Image of CL1 Image of CL2

Figure: Images after 3rd iteration

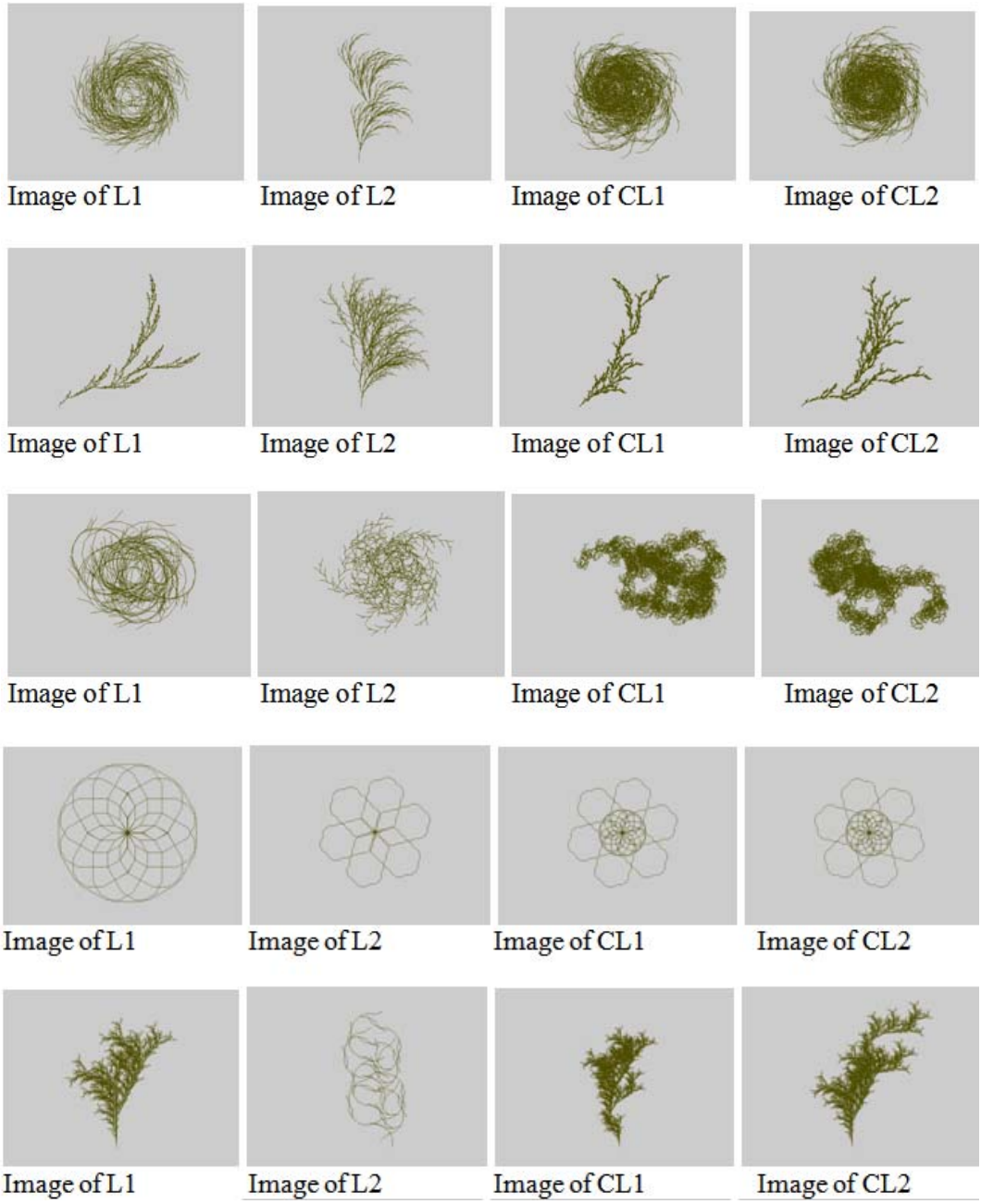
Other Experimental Results

The Experimentation was further performed with different other combined fractals. The Axioms, Production, Angle of rotation and iterations for 8 different cases are listed below in a table. The corresponding images are also given below.

Table: Axiom, Production, angle and iterations used in experimentations

Case	L1	L2
1.	axiom = 'F+F-[FG+FG]-F[F-F]'; Production 'F' = 'FF+[F+F+F]-[F-F-F]'; alpha = 23; iteration=3	axiom = 'G-[GF]+G+G-G-[G+G]-G+G-G'; Production 'G' = 'F+[F+F-[F-GF]]'; alpha = 23; iteration=3
2.	axiom = 'F+F-[FG+FG]-F[F-F]'; Production 'F' ='F+FF+[F+F]-[F+F-F]' alpha = 23; iteration=3	axiom = 'G-[GF]+G+G-G-[G+G]-G+G-G' Production 'G' = 'FG+F-[F+GG[FG+GF]-F-G]'; alpha = 23; iteration=3
3.	axiom ='FG+[GF+G+FG]-G-GF+G' Production 'F' = 'F+GF+GG+FG+[FG-FF] alpha = 23; iteration=3	axiom = 'F+F-[FG+FG]-FF+F'; Production 'G'='G+[FG-GG]+F+GG[FG+GG]-F-G' alpha = 23; iteration=3
4.	axiom = 'F+F'; Production 'F' = '-F+F+F+F'; alpha = 30; iteration=4	axiom = 'G-G-G'; Production 'G' = 'G+G-G+G'; alpha = 30; iteration=4
5.	axiom = 'F+[F]'; Production 'F' = 'FF-[F+F+F]+[F-F-F]'; alpha = 25;Iteration=4	axiom = 'G-[G]'; Production 'G = 'F-G+G+F+[F+F-F]'; alpha = 25;Iteration=4
6.	axiom = 'F-F-F'; Production 'F' = 'F+F-F+F'; alpha = 25;Iteration=4	axiom = 'G-G+G-G-G+G-G-G+G'; Production 'G' = 'G+F-F-G-F-F'; alpha = 30;Iteration=4

7.	axiom = 'F+G'; Production 'F' = '[-F+F+F]+[F-F-F]' alpha = 30;Iteration=4	axiom = 'G+G+F+F'; Production 'G' = '[F+F]-[G+F]'; alpha = 30;Iteration=4
8.	axiom = 'F+F+G'; Production 'F' = '-F+F+F+F' alpha = 30;Iteration=4	axiom = 'G+G'; Production 'G' = 'F+F-F+F'; alpha = 30; Iteration=4



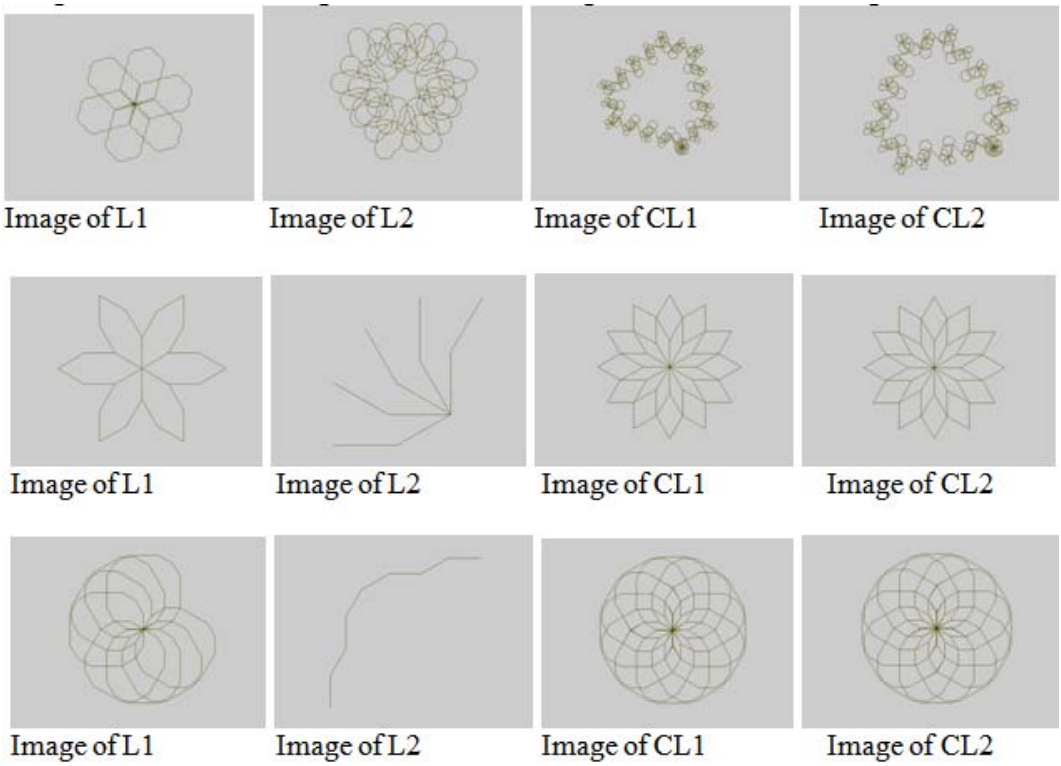


Figure: Images of Case 1 to 8 in sequential order

During the process of experimentation, besides the generations of images, the characteristics of the L-systems were observed. The number of rewriting substitutions and the length of final string of all the L-Systems, both base fractal as well as resulting fractal L-systems are listed in the following Table. Here K indicates the number of rewriting substitutions and L indicates the length of final string. The values are arranged as per the cases exhibited above. From the above images and the following Table it is observed as:

1. Number of rewriting substitution in both CL1 and CL2 are same.
2. Length of resulting final string in both CL1 and CL2 are same.
3. Images of both CL1 and CL2 are similar.
4. Ratio of length to substitution in both CL1 and CL2 <= mean ratio of L1 and L2, which indicates that performance of combination yields better result in comparison to individual l-system.
5. While it may not be possible to obtain the desired image, even after increasing the number of iteration of base fractal but still it may be possible to have it with combined L-system in less iterations.

Table: rewriting substitutions and length of final string.

Case	L1		L2		CL1		CL2	
	K	L	K	L	K	L	K	L
1.	511	8705	210	3594	1616	27514	1616	27514
2.	637	14669	430	9054	3388	75334	3388	75334
3.	172	2943	222	5344	3419	72956	3419	72956
4.	170	1193	255	1535	425	2728	425	2728
5.	1170	19895	30	425	2150	36470	2150	36470
6.	255	1535	135	1367	1650	10462	1650	10462
7.	259	3888	8	87	1248	18670	1248	18670
8.	42	299	2	15	105	740	105	740

Conclusion

Geometric patterns showing the property of self similarity are seen in day to day life. It is natural to look for a mathematical way to describe and model such patterns or objects. L-Systems involve string rewriting techniques which were initially used in study of formal languages. It has been shown in a variety of studies that this technique can be applied in different circumstances to produce abstract models of biological figures. The tools used in this technique, though, are still rudimentary on some platforms, leaving areas for improvement in both design and functionality. 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. While implementing L-system string rewriting to generate different graphical objects, there are single L-system consisting of multiple variables or production rules used. It has been investigated and experimented by us that self similar objects and graphical objects can be generated using L-system string rewriting through combined L-system techniques in a much simplified manner. In this paper we have shown that combined L-system techniques can be used to generate different fractal self similar graphical objects using parallel string rewriting methods.

References

[1] Prusinkiewicz, P. & 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 Biology18:280-315.

[3] M. Zamir, Arterial Branching within the Confines of Fractal L-System Formalism, The Journal of General Physiology, Volume 118, Number 3, September 1, 2001
[4] Prusinkiewicz, P.Graphical applications of L-systems, Proceedings of Graphics interface 86, pp 247-253, 1984
[5] Smith, A.R, Plants, Fractals, Formal Languages, computer Graphics 18, Nr 3, pp1-10
[6] Mandelbrot, B.B., The Fractal geometry of Nature, W.H Freeman, Sanfrancisco
[7] FOLEY, D. 1996. Computer Graphics Principles and Practice Second Edition. Addison-Wesley: Reading, Massachusetts
[8] HERMAN, G. and ROZENBERG, G. 1975 Developmental systems and languages. North-Holland: Amsterdam
[9] PRUSINKIEWICZ, P. 1986. Graphical applications of L-systems. Proceedings of Graphics Interface '86 — Vision Interface '86, pp. 247–253
[10] PRUSINKIEWICZ, P., LINDENMAYER, A. and HANAN, J. 1988. Developmental Models of Herbaceous Plants. Computer Graphics 22(4), pp. 141-150
[11] SZILARD, A. L. and QUINTON, R. E. 1979. An interpretation for D0L systems by computer graphics. The Science Terrapin 4, pp. 8–13
[12] MOHANTY, B.B, MISHRA, S.N, and PATTANAYAK, S, 2010 Developing Modelor for Generating self Similar Objects using Context Free Parrallel String Rewriting. International Journal of Comptational IntelligenceResearch Vol.6, pp. 13–20
[13] MOHANTY, B.B, MISHRA, S.N and PATTANAYAK, S, 2009 Intelligent Approach of Modeling self similar plants and trees using Parrallel String Rewriting techniques. International Journal of Machine Intelligence Vol.1, pp. 34–37
[14] MOHANTY, B.B, MISHRA, S.N and PATTANAYAK, S Generating Natural objects by applying Fractal Graphics techniques, National Conference on CTIT, SIET, Dhenkanal, Orissa, India. 13-14 Feb. 2009
[15] MOHANTY, B.B, MISHRA, S.N and PATTANAYAK, S Modeling Self similar Natural objects in 2D using string rewriting methods, National Conference on Recent Advances in Communication Technology, NIT, Rourkela, India, 30-jan-1st Feb 2009
[16] MOHANTY, B.B, MISHRA, S.N and PATTANAYAK, S Generating Natural objects by applying Fractal Graphics techniques through Multi View Image System, International Conference on Electronic Systems, NIT, Rourkela, India, 7-9 Jan. 2011
[17] MOHANTY, B.B, MISHRA, S.N and PATTANAYAK, S Modeling of 3D Self similar objects using Parallel string rewriting methods and Development of a 3D Modeler, International Journal of Software Engineering Vol.2, pp. 59–66
[18] H.ABELSON AND A.A.DISESA, TURTLE GEOMETRY, MIT PRESS 1982.
[19] K.J FALCONER, Fractal Geometry: Mathematical Foundations and Applications, John Willey & Sons, 1997.
[20] Barnsley, M. 1988. Fractals Everywhere, Academic press.