

An Intuitive Approach for Generating a Class of Fractal Images using L-System String

*Ramakanta Bhoi¹ Supriya Lenka², Sarojananda Mishra³,
¹BPUT, Rourkela, Odisha

^{2,3} Department of Computer Sc., Engg. & Applications, Indira Gandhi Institute of Technology, Sarang,
Dhenkanal, Odisha (India-759146)
Corresponding Author: Ramakanta Bhoi

Abstract: This paper presents an intuitive method of generating a class of fractals associating the axioms and productions of arbitrary fractal images represented in L-system, with that of various number series represented in L-System. The turtle graphic interpretation has been used and infinite types of fractals have been generated.

Keywords: fractals, L-System, number series

Date of Submission: 25-09-2017

Date of acceptance: 05-10-2017

I. Introduction

Various types of fractals are easily represented by L-Systems and generated by means of a given graphic interpretation [1]. However, one must distinguish clearly between the L-System itself and the graphic interpretation that generates the fractal. The possible combinations of axiom and rules used in various fractal images with that of axiom and rules for recursive mathematical formula in L-system formats have been tested with a turtle interpreted software L-System Editor 1.2.0.0, which is free software available, for generating a new hybrid class of fractals. Simple alphabets like A, B, C...Etc., have been used as variables required for representing the mathematical sequences represented in L-System as non-graphic symbol has no impact on turtle interpretations. The fractals are generated by Iterated Function System (IFS) technique. We apply the rules to generate the fractals [1]. The different types of fractals are generated for the same data, because of the great sensitivity to the initial condition in presence of variables.

II. Background

A fractal is a natural phenomenon or a mathematical set that exhibits a repeating pattern that displays at every scale [2]. It is also known as expanding symmetry or evolving symmetry. If the replication is exactly the same at every scale, it is called a self-similar pattern. An example of this is the Menger Sponge. Fractals can also be nearly the same at different levels. This latter pattern is illustrated in the magnifications of the Mandelbrot set. Fractals also include the idea of a detailed pattern that repeats itself. Fractals are different from other geometric figures because of the way in which they scale. Doubling the edge lengths of a polygon multiplies its area by four, which is two (the ratio of the new to the old side length) rose to the power of two (the dimension of the space the polygon resides in). Likewise, if the radius of a sphere is doubled, its volume scales by eight, which is two (the ratio of the new to the old radius) to the power of three (the dimension that the sphere resides in). But if a fractal's one-dimensional lengths are all doubled, the spatial content of the fractal scales by a power that is not necessarily an integer. This power is called the fractal dimension of the fractal, and it usually exceeds the fractal's topological dimension. As mathematical equations, fractals are usually nowhere differentiable. An infinite fractal curve can be conceived of as winding through space differently from an ordinary line, still being a 1-dimensional line yet having a fractal dimension indicating it also resembles a surface. The mathematical roots of the idea of fractals have been traced throughout the years as a formal path of published works, starting in the 17th century with notions of recursion, then moving through increasingly rigorous mathematical treatment of the concept to the study of continuous but not differentiable functions in the 19th century, and on to the coining of the word fractal in the 20th century with a subsequent burgeoning of interest in fractals and computer-based modeling in the 21st century. The term "fractal" was first used by mathematician Benoît Mandelbrot in 1975 [3],[6]-[8]. Mandelbrot based it on the Latin *fractus* meaning "broken" or "fractured", and used it to extend the concept of theoretical fractional dimensions to geometric patterns in nature. There is some disagreement amongst authorities about how the concept of a fractal should be formally defined. Mandelbrot himself summarized it as "beautiful, damn hard, and increasingly useful. That's fractals." The general consensus is that theoretical fractals are infinitely self-similar, iterated, and detailed

mathematical constructs having fractal dimensions, of which many examples have been formulated and studied in great depth. Fractals are not limited to geometric patterns, but can also describe processes in time. Fractal patterns with various degrees of self-similarity have been rendered or studied in images, structures and sounds and found in nature, technology, art, and law.

There are various methods proposed by many researchers to generate fractal images. L-System (Lindenmayer System) is one of the methods which was put forward in 1968 by Lindenmayer as a method of describing plants when he studied the evolution and structure of plant morphology, but in terms of pattern combination this paper emphasizes on generating fractal images by implementing the Hybrid algorithm using the combination of even number series, odd number series and n- number of series in L-System [4],[9]-[12].

The organization of this paper is given below. The paper starts with introductory concept regarding the fractal image generation. Section 2 briefs about the detail background as obtained from various literatures. Section 3 elaborates the concept of fractal image generation using L-System. Section 4 focuses on the new proposed algorithm with result analysis with visualization of images as generated. Section 5 concludes with future scope.

III. Fractal Image Using L-System

L-system is a parallel rewriting system based on symbols, whose central concept is rewriting. In general, rewriting is a technique for defining complex objects by successively replacing parts of a simple initial object using a set of rewriting rules or productions [5].

Formal definition of L-system and its operation are given below:

Let V denote an alphabet, V^* the set of all words over V . A string L-system is an ordered triplet $G = \langle V, \omega, P \rangle$ where $\omega \in V^*$ is a nonempty word called the axiom and $P \subset V \times V^*$ is a finite set of productions. A production $(a, x) \in P$ is written as $a \rightarrow x$. The letter 'a' and the word x 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 $x \in V^*$ such that $a \rightarrow x$. If no production is explicitly specified for a given predecessor $a \rightarrow V$, the identity production $a \rightarrow a$ is assumed to belong to the set of productions P.

According to the definition above it is known that L-system gets a character string consisting of specific characters. To let this string represent certain images, to it is need to do some appropriate interpretations for it. The common method is called turtle interpretation, whose basic idea is given below. A state of the turtle is defined as a triplet (x, y, α) , where the coordinates (x, y) represent the turtle's position, and the angle α , called the turtle's heading, is interpreted as the direction in which turtle is facing.

Given the step size d and the angle increment δ , the turtle can respond to commands represented by the following symbols:

F: Move forward a step of length d. The state of the turtle changes to (x', y', α) , where $x' = x + d \cos \alpha$ and $y' = y + d \sin \alpha$. A line segment between points (x, y) and (x', y') is drawn.

+: Turn left by angle δ . The next state of the turtle is $(x, y, \alpha + \delta)$. The positive orientation of angles is counter-clockwise.

-: Turn right by angle δ . The next state of the turtle is $(x, y, \alpha - \delta)$. The positive orientation of angles is clockwise.

[: Push the current state of the turtle onto a pushdown stack.

]: Pop a state from the stack and make it the current state of the turtle.

IV. Proposed Algorithm

4.1 Preliminaries and Assumptions

The complete algorithm is based upon some intuitive approach of representing simple mathematical number series in combination with arbitrary turtle drawing strings[1]. Few examples of such number series are mentioned below for better clarification.

4.2 Representing number sequence in L-System

This section illustrates the representation of mathematical series in L-System [1].

Example –1: To generate Fibonacci sequence

Consider the simple L-System grammar that is defined as follows...

Variables: A, B

Constants: none

Axiom : A

Rules : A \rightarrow B

B \rightarrow AB

This L-System produces the following sequence of strings...

- Stage 0: A
- Stage 1: B
- Stage 2: AB
- Stage 3: BAB
- Stage 4 : ABBAB
- Stage 5 : BABABBAB
- Stage 6 : ABBABBABABBAB
- Stage 7 : BABABBABABBABABBAB

If we count the length of each string, we obtain the well-known Fibonacci sequence of numbers:

1 1 2 3 5 8 13 21 34

Production rules for Fibonacci series is:

- Axiom = A
- A = FFF++FFF++FFF++FFF++B Replaced by A->C
- B = FFF++FFF--FFF++FFF--FFF++FFFFF--FFFFF++FFFA Replaced by B-> D
- C= AFFF++FFF++FFF++FFF++B Replaced by C->AC

String Calculation

- Axiom A-----1
- >C----1
- >AC---2
- >AAC—3
- .
- .
- .

This series gives Fibonacci series.

Example 2: To generate positive integers from 1 to n

The equivalent L-System may be as per the following:

Variables: A, B

Constants: any lowercase letters or digits, e.g., a, b, c... or 1, 2, 3.... etc.

Axiom: A

Rules: A => B, B=>Ba (where a is any constant)

This L-system produces the following sequence of strings ...

- Stage 0: A
- Stage 1: B
- Stage 2: Ba
- Stage 3: Baa
- Stage 4: Baaa
- Stage 5: Baaaa
- Stage 6: Baaaaa
- Stage 7: Baaaaaa and so on.

If we count the length of strings generated from stage 1 onwards, we can get the positive numbers from 1 to infinity.

Alternatively if we take axiom A then subsequent substitution can generate 1 to n number series as given below.

Axiom A

- A=F+B replaced by **A → aB**
- B=F+F+C replaced by **B → bC**
- C=F+F+F+D replaced by **C → cD**
- D=F+F+F+FC replaced by **D → dC**

String calculation

- Axiom A -----1
- A -> aB----2

abC---3
 abcD--4
 abcdC--5
 .
 .
 .

At every stage if we count number of variables in each substitution we can get 1 to n number series.

Example 3: To generate n numbers of positive odd integers.

The equivalent L-System may be represented as follows.

Variables: A, B

Constants: any lowercase letters or digits, e.g., a or 1,2,3.. etc

Axiom: A

Rules: $A \Rightarrow B$, $B \Rightarrow Baa$ (where a is any constant)

This L-system produces the following sequence of strings .

Stage 0: A

Stage 1: B

Stage 2: Baa

Stage 3: Baaaa

Stage 4: Baaaaaa

Stage 5: Baaaaaaaa

Stage 6: Baaaaaaaaaa

Stage 7: Baaaaaaaaaaaa and so on.

If we count the length of strings generated from stage 1 onwards we can get the positive odd numbers from 1 to infinity. Alternatively like in Example-1 we can get this odd number series as mentioned below.

Axiom A

$A = F+B$ replaced by $A \rightarrow abB$

$B = F+F+C$ replaced by $B \rightarrow abc$

$C = F+F+F+D$ replaced by $C \rightarrow abD$

$D = F+F+F+F+FC$ replaced by $D \rightarrow abC$

String calculation

Axiom A-----1

A->abB-----3

ababC-----5

abababD----7

ababababC—9

.
 .
 .

It is of odd number series.

Example 4: To generate positive even integers.

The equivalent L-System grammar may be as per the following.

Variables : A,B

Constants : any lowercase letters or digits, e.g., a or 1,2,3.. etc

Start : A

Rules : $A \Rightarrow Ba$, $B \Rightarrow Baa$ (where a is any constant)

This L-system produces the following sequence of strings ...

Stage 0 : A
 Stage 1 : Ba
 Stage 2 :Baaa
 Stage 3 :Baaaaa
 Stage 4 :Baaaaaaa
 Stage 5 :Baaaaaaaaa
 Stage 6 :Baaaaaaaaaaa
 Stage 7 :Baaaaaaaaaaaaa and so on.

If we count the length of strings generated from stage 1 onwards, we can get the positive even numbers from 1 to infinity.

Similarly we can various numbers of number series using L-System grammar as per following

Axiom AB
 A= F++B replaced by $A \rightarrow aB$
 B=F++F++Creplaced by $B \rightarrow bC$
 C=F+++F+++F+++F+++Dreplaced by $C \rightarrow bcD$
 D=F++++F++++F++++F++++FDreplaced by $D \rightarrow deD$
 F=F++F++F++F++F++F++F

String calculation
 Axiom AB-----2
 aBbC-----4
 aBbbcD-----6
 aBbbcdD-----8

.
 .
 .
 It is of 2n series.

Axiom ABC
 A=++B replaced by $A \rightarrow aB$
 B=F++F++C replaced by $B \rightarrow abaC$
 C=F+++F+++F+++F+++F replaced by $C \rightarrow ababD$

String calculation
 Axiom ABC-----3
 aBabaC-----6
 aBabababD-----9

.
 .
 .
 It is of 3n series.

4.3 Assumptions

As the symbol 'F' is considered as the only graphical symbol in turtle graphic, the L-System involving non-graphic symbols like A,B,C etc in its axiom as well productions are not graphically interpreted. Any fractal curve represented in L-System using the symbol 'F' can be easily visualized by manual method or by any standard turtle graphic interpreted software for visualization. Graphical interpretation for mathematical sequence involving the symbols other than the symbol 'F' is simply omitted in Turtle graphics.

4.4 The Algorithm

The proposed algorithm is represented below in Table No.1 for information.

Table No.1: The Algorithm

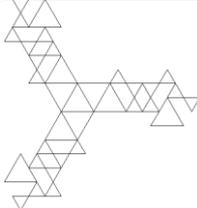
Step-1	Formulation of random arbitrary L-System string
Step-2	Formulation of any number series represented in L-System
Step-3	Formulation of new hybrid L-System by combining the axiom and rules of fractals as generated in Step-1 and Step-2 by variable positioning
Step-4	Repetition of Step-1 through Step-6 for arbitrary angles
Step-5	Compiling and Running the new L-System
Step-6	Storing the image

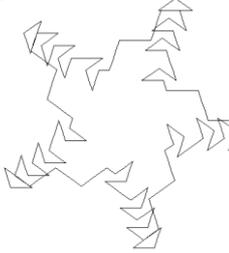
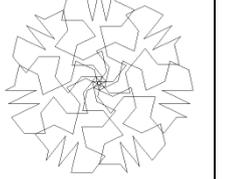
4.5 Results with Visualization

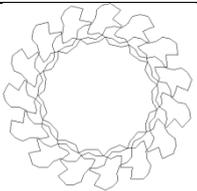
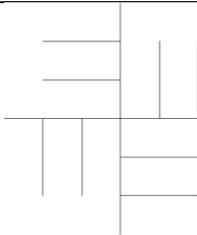
The newly formulated strings with variables have been compiled and run using LSysedit free software and the visualizations have been observed and few generated images are mentioned in Table No.2 to Table No.5. By using this method 3D figures are also generated which are mentioned in Table No.6.

Table No.2: Generated images using Fibonacci series

Turtle Graphics Command	Generated Images using variable	Generated Images without using variable
Dirs = 7 Axiom = A #Iterations = 16 A = FFF++FFF++F FF++FFF++B B = FFF++FFF-- FFF++FFF-- FFF++FFFFF-- FFFFF++FFFA C = AFFF++FFF++ FFF++FFF++B		
Dirs = 8 Axiom = A #Iterations = 16 A = FFF++FFF++F FF++FFF++B B = FFF++FFF-- FFF++FFF-- FFF++FFFFF-- FFFFF++FFFA C = AFFF++FFF++ FFF++FFF++B		
Dirs = 9 Axiom = A #Iterations = 18 A = FFF++FFF++F FF++FFF++B B = FFF++FFF-- FFF++FFF--		

<p>FFF++FFFFF-- FFFFF++FFFA C= AFFF++FFF++ FFF++FFF++B</p>		
<p>Dirs = 9 Axiom = A #Iterations = 18 A = ---- FFF++++FFF--- - FFF++++FFF+ +B B = FF+FF++FF- FF-- FF+FF++FF- FF-- FF+FF++FF- FFF-- FF+FF++FF- FA C = A---- FFF++++FFF--- - FFF++++FFF+ +B</p>		
<p>Dirs = 3 Axiom = A #Iterations = 18 A = ---- FFF++++FFF--- - FFF++++FFF+ +B B = FF+FF++FF- FF-- FF+FF++FF- FF-- FF+FF++FF- FFF-- FF+FF++FF- FA C = A---- FFF++++FFF--- - FFF++++FFF+ +B</p>		

<p>Dirs = 5 Axiom = A #Iterations = 18 A = ---- FFF++++FFF--- - FFF++++FFF+ +B B = FF+FF++FF- FF-- FF+FF++FF- FF-- FF+FF++FF- FFF-- FF+FF++FF- FA C= A---- FFF++++FFF--- - FFF++++FFF+ +B</p>		
<p>Dirs = 6 Axiom = A #Iterations = 18 A = ---- FFF++++FFF--- - FFF++++FFF+ +B B = FF+FF++FF- FF-- FF+FF++FF- FF-- FF+FF++FF- FFF-- FF+FF++FF- FA C= A---- FFF++++FFF--- - FFF++++FFF+ +B</p>		
<p>Dirs = 7 Axiom = A #Iterations = 18 A = ---- FFF++++FFF--- - FFF++++FFF+ +B B = FF+FF++FF- FF-- FF+FF++FF-</p>		

<p>FF-- FF+FF++FF- FFF-- FF+FF++FF- FA C= A--- FFF++++FFF--- - FFF++++FFF+ +B</p>		
<p>Dirs = 15 Axiom = A #Iterations = 32 A = --- FFF++++FFF--- - FFF++++FFF+ +B B = FF+FF++FF- FF-- FF+FF++FF- FF-- FF+FF++FF- FFF-- FF+FF++FF- FA C= A--- FFF++++FFF--- - FFF++++FFF+ +B</p>		
<p>Dirs = 4 Axiom = A #Iterations = 18 A = -++F-- FFF++- F++FFF---F-- FFF++- F++FFF++B B = FF+FF++FF- FF-- FF+FF++FF- FF-- FF+FF++FF- FFF-- FF+FF++FF- FA C= A--F-- FFF++- F++FFF---F-- FFF++- F++FFF++B</p>		

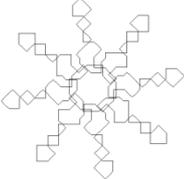
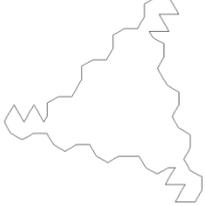
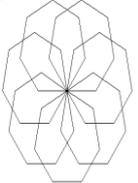
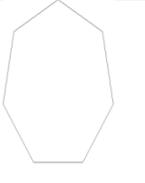
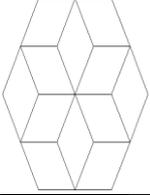
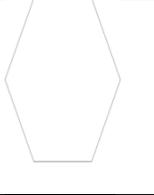
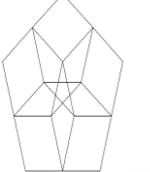
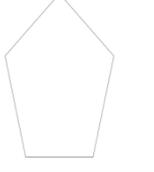
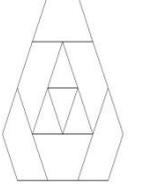
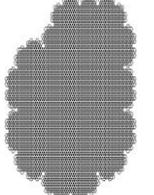
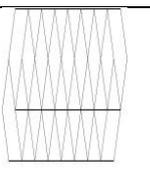
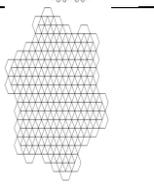
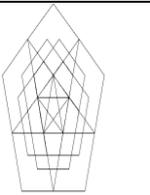
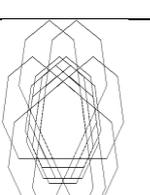
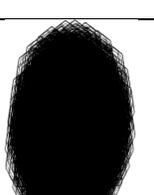
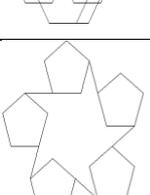
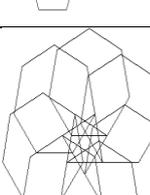
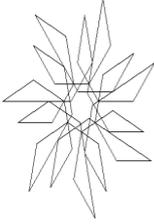
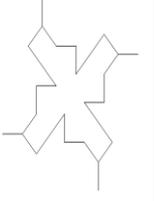
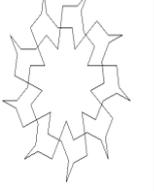
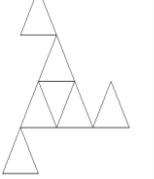
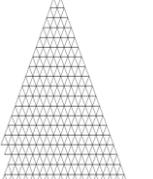
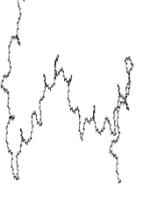
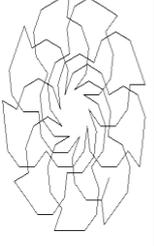
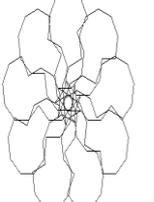
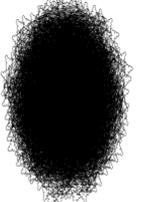
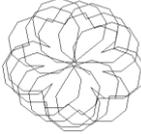
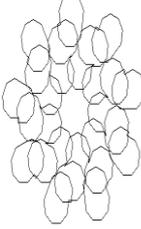
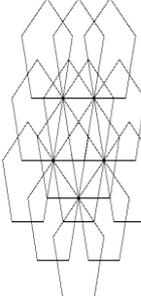
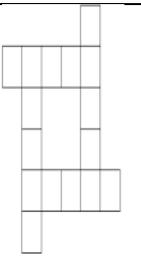
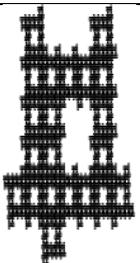
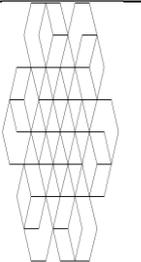
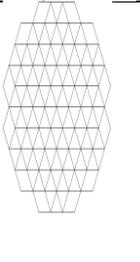
<p>Dirs = 8 Axiom = A #Iterations = 18 A = --+++F-- FFF++-- F++FFF--++F-- FFF++-- F++FFF++B B = FF+FF++FF- FF-- FF+FF++FF- FF-- FF+FF++FF- FFF-- FF+FF++FF- FA C= A--++F-- FFF++-- F++FFF--++F-- FFF++-- F++FFF++B</p>		
<p>Dirs = 12 Axiom = A #Iterations = 18 A = ---- FFF++++FFF-- - FFF++++FFF+ +B B = FF+FF++FF- FF-- FF+FF++FF- FF-- FF+FF++FF- FFF-- FF+FF++FF- FA C= A---- FFF++++FFF-- - FFF++++FFF+ +B</p>		

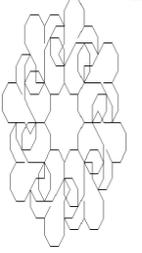
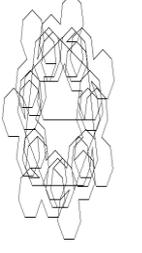
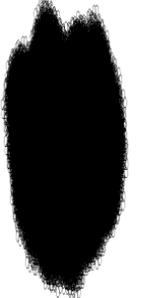
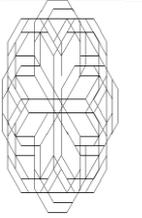
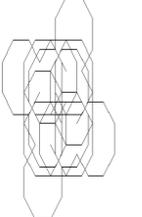
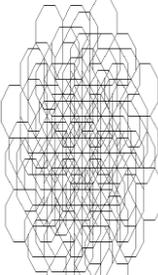
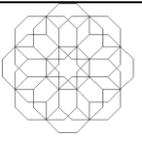
Table No. 3: Generated images using 1 to n number and odd number series

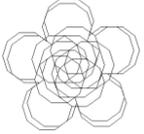
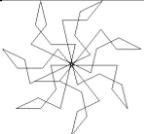
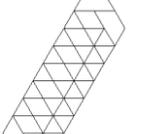
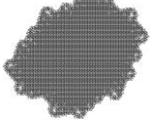
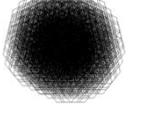
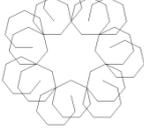
Turtle Graphics Command	Generated images using variables	Generated Images without using variables
<p>Dirs = 7 Axiom = A #Iterations = 16 A = F+B B = F+F+C C = F+F+F+D D = F+F+F+FC</p>		

<p> Dirs = 5 Axiom = A #Iterations = 17 A = F+B B = F+F+C C = F+F+F+D D = F+F+F+FC </p>		
<p> Dirs = 6 Axiom = A #Iterations = 17 A = F+B B = F+F+C C = F+F+F+D D = F+F+F+F+FC </p>		
<p> Dirs = 6 Axiom = A #Iterations = 17 A = F+B B = F+F+F+FC C = F+F+FD D = F+F+F+F+F+FC </p>		
<p> Dirs = 6 Axiom = A #Iterations = 17 A = F+B B = F+F+F+FC C = F+F+F+FD D = F+F+F+F+F+F+FC </p>		
<p> Dirs = 5 Axiom = A #Iterations = 17 A = F+F+FB B = F+F+F+FC C = F+F+F+F+FD D = F+F+F+F+F+F+FC </p>		
<p> Dirs = 7 Axiom = A #Iterations = 17 A = F+F+FB B = F+F+F+FC C = F+F+F+F+FD D = F+F+F+F+F+F+F+FC </p>		
<p> Dirs = 5 Axiom = A #Iterations = 17 A = F+++B B = --F+C C = ++FD D = F+F+F+F+F+F+F+C </p>		
<p> Dirs = 7 Axiom = A #Iterations = 17 A = F+++B B = --F+C C = ++FD D = F+F+F+F+F+F+F+C </p>		

<p>Dirs = 7 Axiom = A #Iterations = 17 A = F+++B B = --F+C C = ++FD D = F+F+F---- F+F+F+F+C</p>		
<p>Dirs = 8 Axiom = A #Iterations = 17 A = ---F+++B B = --F+C C = ++FD D = F--F+F----F+F+F-- F+C</p>		
<p>Dirs = 9 Axiom = A #Iterations = 20 A = ---F+++B B = --F+C C = ++FD D = F--F+F----F+F+F-- -F+C</p>		
<p>Dirs = 3 Axiom = A #Iterations = 17 A = ---F+++B B = --F+C C = ++FD D = F--F+F----F+F+F-- F+C</p>		
<p>Dirs = 9 Axiom = A #Iterations = 17 A = ---F+++B B = --F+C C = ++FD D = F-- F+F+F+F+F+F+F---- F+F+F--F+C</p>		
<p>Dirs = 9 Axiom = A #Iterations = 20 A = ---F+++B B = --F+C C = ++FD D = F--F+F+F---- F+F+F+F+F--F+C</p>		
<p>Dirs = 9 Axiom = A #Iterations = 20 A = ---F+++B B = --F+C C = ++FD D = F+F+F+F-- F+F+F+F----F+F+F+F-- F+C</p>		
<p>Dirs = 9 Axiom = A #Iterations = 20 A = ---+B B = ---+C C = +F+F+D D = F+F+F+F-- F+F+F+F----F+F+F+F-- F+C</p>		

<p>Dirs = 9 Axiom = A #Iterations = 15 A = ---+B B = ---+C C = +F+F+F+FD D = F+F+F+F+F-- F+F+F+F+F---- F+F+F+F+F--F+C</p>		
<p>Dirs = 9 Axiom = A #Iterations = 20 A = ---+B B = ---+C C = +F+F+F+FD D = F+F+F+F+F-- F+F+F+F+F---- F+F+F+F+F-- F+F+F+F+F+C</p>		
<p>Dirs = 7 Axiom = A #Iterations = 17 A = ---+B B = ---+C C = +F+F+F+FD D = F+F+F+F+F-- F+F+F+F+F---- F+F+F+F+F-- F+F+F+F+F+C</p>		
<p>Dirs = 5 Axiom = A #Iterations = 17 A = ---+B B = ---+C C = +F+F+F+FD D = F+F+F+F+F-- F+F+F+F+F---- F+F+F+F+F-- F+F+F+F+F+C</p>		
<p>Dirs = 4 Axiom = A #Iterations = 17 A = ---+B B = ---+C C = +F+F+F+FD D = F+F+F+F+F-- F+F+F+F+F---- F+F+F+F+F-- F+F+F+F+F+C</p>		
<p>Dirs = 6 Axiom = A #Iterations = 17 A = ---+B B = ---+C C = +F+F+F+FD D = F+F+F+F+F-- F+F+F+F+F---- F+F+F+F+F-- F+F+F+F+F+C</p>		

<p>Dirs = 8 Axiom = A #Iterations = 18 A = ---+B B = ---+C C = +F+F+F+F+FD D = F+F+F+F+F+F-- F+F+F+F+F+F---- F+F+F+F+F+F-- F+F+F+F+F+F+C</p>		
<p>Dirs = 7 Axiom = A #Iterations = 17 A = --F+F+F+B B = -F+F+F+C C = +F+F+F+F+F+F+ F+F+F+FD D = F+F+F+F+F+F-- F+F+F+F+F+F---- F+F+F+F+F+F-- F+F+F+F+F+F+C</p>		
<p>Dirs = 7 Axiom = A #Iterations = 16 A = --F+F+F+B B = &F&F&F--+C C = +F+F+F+F+F+F+ F+F+F+FD D = &F&F&F&F+F+F+F +F+F--F+F+F+F+F+F-- -F+F+F+F+F+F-- F+F+F+F+F+F+C</p>		
<p>Dirs = 8 Axiom = A #Iterations = 16 A = ---+B B = ---+C C = -F+F+F+F+F+F+F +F+FD D = F+F+F+F+F+F-- F+F+F+F+F+F---- F+F+F+F+F+F-- F+F+F+F+F+F+C</p>		
<p>Dirs = 8 Axiom = A #Iterations = 16 A = ---+B B = ---+C C = -F+F+F+F+F+F+F +F+FD D = F+F+F+F+F+F-- F+F+F+F+F+F---- F+F+F+F+F+F-- F+F+F+F+F+F+C</p>		
<p>Dirs = 8 Axiom = A #Iterations = 18 A = ---+B B = ---+C C = -F+F+F+F+F+F+F +F+FD D = F+F+F+F+F+F-- F+F+F+F+F+F---- F+F+F+F+F+F-- F+F+F+F+F+F+C</p>		

<p> Dirs = 10 Axiom = A #Iterations = 18 A = ---B B = --+C C=F+F+F+F+F+F+F+F +F+FD D=F+F+F+F+F+F+F-- F+F+F+F+F+F+F---- F+F+F+F+F+F+F-- F+F+F+F+F+F+C </p>		
<p> Dirs = 8 Axiom = A #Iterations = 19 A = F---B+F+F B=F+F-+C+F+F+ C=F+F+F+F+F+F+F+F +F+FD D=F+F+F+F+F+F+F-- F+F+F+F+F+F+F---- F+F+F+F+F+F+F-- F+F+F+F+F+F+C </p>		
<p> Dirs = 7 Axiom = A #Iterations = 17 A = F+++B B = --F+C C= ++FD D=F--F+F----F+F+F-- F+C </p>		
<p> Dirs = 10 Axiom = A #Iterations = 17 A = F+F+FB B = F+F+FC C= F+F+F+F+FD D=F+F+F+F+F+F+F+F +FC </p>		
<p> Dirs = 6 Axiom = A #Iterations = 17 A = F+F+FB B = F+F+F+FC C= F+F+F+F+FD D=F+F+F+F+F+F+F+F +FC </p>		
<p> Dirs = 6 Axiom = A #Iterations = 8 A = *---B B = "F+-----+C C= &F+F+F+F----- +D D= &F+----- +&F+F+FC </p>		
<p> Dirs = 7 Axiom = A #Iterations = 16 A = *---B B = "*-----+C C= *F+F+F+F----- +D D=&F+F+F+F+F+F+--- ---F+F+F+F+F+C </p>		
<p> Dirs = 7 Axiom = A #Iterations = 16 A = ^---B B = "^-----+C C= [^F+F+F+F]----- +D D= [^F+F+F+F+F+F]----- F+F+F+F+F+C </p>		

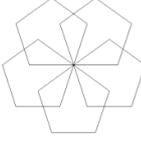
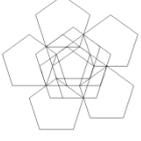
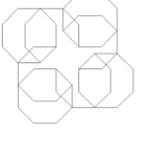
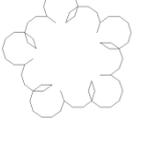
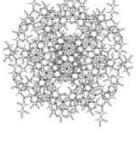
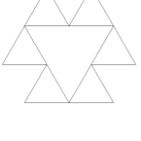
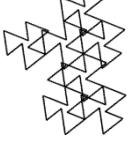
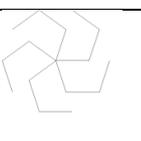
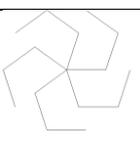
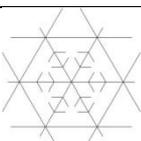
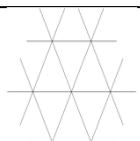
<p>Dirs = 5 Axiom = A #Iterations = 16 A = ^----B B = "-----C C = [^F+F+F+F]----- +D D= [^F+F+F+F+F]----- [^F+F+F+F+F+C]</p>		
<p>Dirs = 5 Axiom = A #Iterations = 12 A = {^----B} B = {"-----C} C = {^F+F+F+F]----- +D D= {^F+F+F+F+F]----- {^F+F+F+F+F+C}</p>		
<p>Dirs = 8 Axiom = A #Iterations = 10 A = {^----B} B = {"-----C} C = {^F+F+F+F]----- +D D= {^F+F+F+F+F]----- {^F+F+F+F+F+C}</p>		
<p>Dirs = 10 Axiom = A #Iterations = 12 A = {^----B} B = {"-----C} C = {^F+F+F+F]----- +D D= {^F+F+F+F+F]----- {^F+F+F+F+F+C}</p>		
<p>Dirs = 3 Axiom = A #Iterations = 8 A = ---+B B = --+C C = F+F+F+F+F+F+ F+F+F+FD D= +F+F+F+F+F+F-- F+F+F+F+F+F +F----F+F+F+F+F+F+F- -F+F+F+F+F+F+C</p>		
<p>Dirs = 5 Axiom = &ABF+F #Iterations = 3 A = [&F+F+B^)+///+ [&F+ F+B^)+\ \ \ + [&F+F+B^] B= [&F^)+\ \ \ + [&F^)+// /+ [&F^] F= +///[&F+F+F^)\ \ \ +</p>		
<p>Dirs = 3 Axiom = FFFA #Iterations = 6 A = " [& FFB] +///+ [&*FFB] +///+ [&*FFB] B = ! [&*FFB] +///+ [&*FF] +///+ [&*FF] F = {FFFB}</p>		

Table No.4:Generated imagesusing 2n series

Turtle Graphics Command	Generated Images using variable	Generated images without using variable
Dirs = 7 Axiom = ABCD #Iterations = 10 A = F++B B = F++F++C C=F+++F+++F+++F+++D D=F++++F++++F++++F++++FD F=F++F++F++F++F++F++F		

Table No. 5:Generated images using 3n series

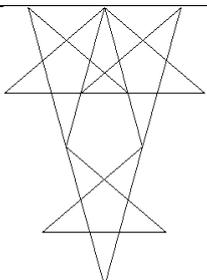
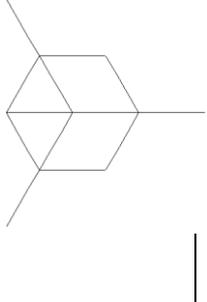
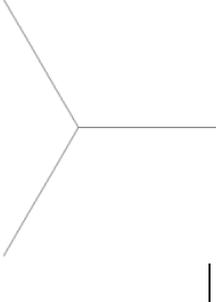
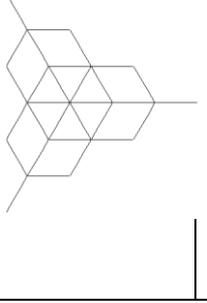
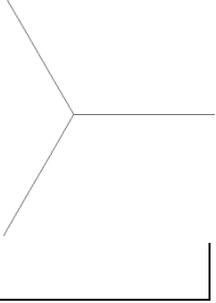
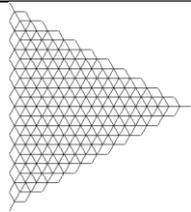
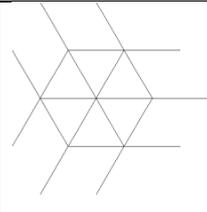
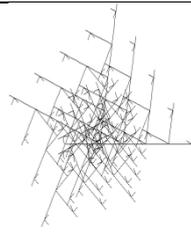
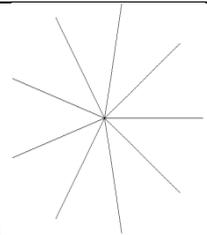
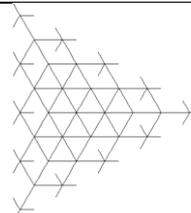
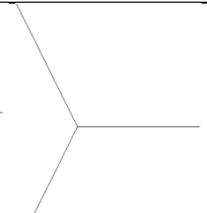
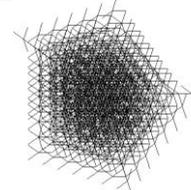
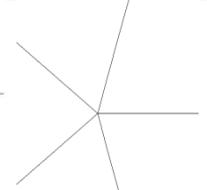
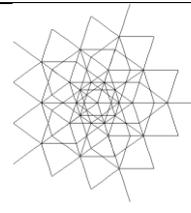
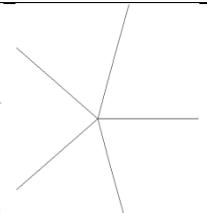
Turtle Graphics Command	Generated Images using variable	Generated images without using variable
Dirs = 5 Axiom = ABC #Iterations = 3 A = ++B B = F++F++C C=F++F++F ++F++F		

Table No.6: Generated 3D images using 1 to n number series

Turtle Graphics Command	Generated Images using variable	Generated Images without using variable
Dirs = 3 Axiom = FFFA #Iterations = 2 A = "[&FFFA] +////+ [&FFFA]+ ////+ [&FFFA]		
Dirs = 3 Axiom = FFFA #Iterations = 2 A = "[&FFFA] +////+ [&FFFA]+ ////+ [&FFFA]		

<p> Dirs = 3 Axiom = FFFA #Iterations = 11 A = +"[&FFFA]+ ///+[&FFFA] +/// +[&FFFA]+ A=+" [B]+ ///+[B] +/// +[B]+ B= +&FFFA+ </p>		
<p> Dirs = 9 Axiom = FFFA #Iterations = 10 A = "[&FFFB] +/// B= "[&FA] +/// [&FA]+ /// [&FA] </p>		
<p> Dirs = 6 Axiom = FFFA #Iterations = 10 A = "[&FFFB] +/// B= "[&FFFA] +/// [&FFFA]+ /// + [&FFFA] </p>		
<p> Dirs = 5 Axiom = FFFA #Iterations = 8 A = "[&FFFA] +/// [&FFFA]+ /// + [&FFFA] </p>		
<p> Dirs = 10 Axiom = FFFA #Iterations = 3 A = "[&FFFB] +/// [&FFFB]+ </p>		

<pre> ////+ [&FFFB] + ////+ [&FFFC]+//// + [&FFFC] B = " [&FFFC] +////+ [&FFFC]+ ////+ [&FFFC]+ ////+ [&FFFC]+//// + [&FFFC] C = " [&FFFD] +////+ [&FFFD]+ ////+ [&FFFD]+//// + [&FFFC]+//// + [&FFFC] D = " [&FFFC] +////+ [&FFFC]+ ////+ [&FFFC]+//// + [&FFFC]+//// + [&FFFC] </pre>		
---	--	--

4.6. Result Interpretations and Analysis

L-systems can be used to generate fractals by applying a turtle interpretation to the symbols of the grammar. These same grammars can also be applied to generate many number 1 series. The purpose of this paper is to combine these two applications of L-systems to generate new fractals by applying a turtle interpretation to the grammars that generate interesting mathematical series to build new fractals.

From the tables (Table No.2 to Table No.6) it is observed that variables have definite impact in generating of new fractal patterns. These generated figures cannot be generated using only drawing symbol 'F'. Although detail characteristics of these newly generated fractal figures are beyond the scope of this paper but it is sure that variables used are forming a loop for generating beautiful patterns which are converging after few iterations.

All these generated figures are fractal by nature and they adhere to characteristics of fractal images.

It is observed that out of all mathematical sequences considered, 1 to n sequence would always yield most space-filling fractals for the same number of iterations.

V. Concluding Remarks And Future Scope

The approach suggested in this paper for generating hybrid fractals could form the basis for modeling various types of growth phenomena found in nature. A biological structure appears to be constructed in part from a Fibonacci blueprint [56]. Although very few arbitrary fractals have been tested in this paper hybrid fractal implemented software will definitely solve this complexity. Relationship of the fractal dimension [48], of the generated hybrid fractal to that of the base arbitrary fractal needs to be explored. In subsequent growing process of these fractals are directly proportional to the strings associated with non-graphic symbols. Other number series like Fibonacci series and the like can be tested with arbitrary strings and classical fractals like Koch curve, Sierpinski triangle, etc., may be taken as base string instead of arbitrary L-system string. Few converged fractal images can be thoroughly studied for practical applications of these images like Koch curve and Sierpinski triangle. Standard tools can be developed for these types of fractal images for further research. More research work is necessary towards colour fractal images using non-graphic variables used in turtle graphics.

References:

- [1]. Mishra S.N., et al; L-Systems Fractals: Elsevier Publications, 2007, Netherland.
- [2]. <https://en.wikipedia.org/wiki/Fractal>
- [3]. Mishra S.N.; Bisoi,A.K and .Mishra,J; Growing a class of fractals based on the combination of classical fractals and recursive mathematical series in L-Systems: Machine Graphics & Vision, Poland, Vol.13, No.3, 2004.
- [4]. Mohanty, B B, Mishra, Sasmita , Mishra, S.N.; Novel Approach of Modeling Self Similar Objects using Parallel String Rewriting Methods through Combined L-System Techniques: pp.1-12, International Journal of Information & Computation Technology (IJICT) Volume 2, Number 1 (2012).
- [5]. Lindenmayer, A, Mathematical models for cellular interaction in development. : J. Theoret. Biology, 18:280—315, 1968.
- [6]. Barnsley, Michael F.; and Rising, Hawley; *Fractals Everywhere*. Boston: Academic Press Professional, 1993. ISBN 0-12-079061-0.
- [7]. Mandelbrot, Benoit B.; *The Fractal Geometry of Nature*. New York: W. H. Freeman and Co., 1982. ISBN 0-7167-1186-9.
- [8]. Peitgen, Heinz-Otto; and Saupe, Dietmar; eds.; *The Science of Fractal Images*. New York: Springer-Verlag, 1988. ISBN 0-387-96608-0.
- [9]. Prusinkiewicz,P; Lindenmayer,A ; *The Algorithmic Beauty of Plants*.
- [10]. Rozenberg,Grzegorz; Salomaa,Arto; *Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology* :ISBN 978-3-540-55320-5.
- [11]. Yang, Tingjun;Huang, Zhengge;Lin,Xingsheng, Chen, ,Jianjun and Ni, Jun;A *Parallel Algorithm for Binary-Tree-Based String Rewriting in L-systems*: IEEE 2007.
- [12]. Sun, Bowen;Jian, Litao;Sun, Boling, and Jiang ,Shengtao; *Research of Plant Growth Model Based on the Combination of L-system and Sketch*.:978-0-7695-3398-8/08 2008 IEEE.

About Authors:

	<p>Mr.Ramakanta Bhoi is an educationist having qualifications MCA and M. Tech. He is doing his research work since 2008.His area of interest is networking. He is having 10 years of teaching experience in teaching.</p>
	<p>Ms.SupriyaLenka is currently pursuing her MTech degree at Indira Gandhi Institute of Technology,Sarang, Dhenkanal,Odisha. She is doing her MTechdissertation under the able guidance of Prof. Sarojananda Mishra. Her research area is fractal image generation.</p>
	<p>Dr. Sarojananda Mishra is currently working as Professor & Head of the department of Computer Science, Engg. & Applications at Indira Gandhi Institute of Technology (IGIT), Sarang, Dhenkanal, Odisha, India. He has published more than 120 papers in International Journals and National Journals of repute. His research area focuses on fractal graphics, fractal geometry and internet data analysis. He has more than 25 years of teaching and research experiences. Many students obtained PhD degree and are continuing their PhD and MTech research work under his guidance.</p>

Ramakanta Bhoi An Intuitive Approach for Generating a Class of Fractal Images using L-System String.” International Journal of Engineering Inventions (IJEN), vol. 6, no. 10, 2017, pp. 18-37.