A New Vision of Fractal Geometry with Triangulation Algorithm

Yasser M. Abd El-Latif, Fatma S. Abousaleh, and Daoud S. S.

Abstract—L-system is a tool commonly used for modeling and simulating the growth of fractal plants. The aim of this paper is to join some problems of the computational geometry with the fractal geometry by using the L-system technique to generate fractal plant in 3D. L-system constructs the fractal structure by applying rewriting rules sequentially and this technique depends on recursion process with large number of iterations to get different shapes of 3D fractal plants. Instead, it was reiterated a specific number of iterations up to three iterations. The vertices generated from the last stage of the L-system rewriting process are used as input to the triangulation algorithm to construct the triangulation shape of these vertices. The resulting shapes can be used as covers for the architectural objects and in different computer graphics fields. The paper presents a gallery of triangulation forms which application in architecture creates an alternative for domes and other traditional types of roofs.

Keywords—Computational geometry, Fractal geometry, L-system, Triangulation.

I. INTRODUCTION

NOWADAYS, a lot of researches are directed to generate 2D and 3D fractal objects. The Euclidean-geometry methods are adequate for describing manufactured objects; those that have smooth surfaces and regular shapes. But natural objects, such as mountains, clouds and plants, have irregular or fragmented features, and Euclidean methods do not realistically model these objects. Natural objects can be realistically described with fractal-geometry methods, where procedures rather than equations are used to model objects [1]. Fractal geometry is one of the youngest theories of contemporary mathematics, which developed thanks to the advance of computer technology [2]. A fractal is generally "a rough or fragmented geometric shape that can be subdivided into parts, each of which is (at least approximately) a reduced-size copy of the whole," a property called self-similarity. The term was suggested by Benoit Mandelbrot in 1975 and was derived from the Latin fractus meaning "broken" or "fractured" [3]. The most common techniques used for generating fractals are iterated function system (IFS) where a fractal object is generated by applying a specified transformation function to point within a region of space and L-system (Lindenmayer system) which represents the fractal structure of plants using grammatical expressions. Starting with an initial structure, L-system constructs the fractal structure by applying rewriting rules sequentially [8].

Computational geometry is the study of efficient algorithms to solve geometric problems, such as: given \( N \) points in the plane or in the space, how the smallest convex hull that encloses all the points is computed, given \( N \) points in a plane, what is fastest way to find the nearest neighbor of a point, given \( N \) straight lines, how to find the lines which intersect with each other [4] and the triangulation problem given \( N \) points in the plane how they can be joined by nonintersecting straight line segments so that every region internal to the convex hull is a triangle. It is noticed that Computational geometry and computer graphics both consider geometric phenomena as they relate to computing. Computational geometry provides a theoretical foundation involving the study of algorithms and data structures for doing geometric computations. Computer graphics concerns the practical development of the software, hardware and algorithms necessary to create graphics (i.e. to display geometry) on the computer screen [10]. In this study, it was shown how computer graphics and computational geometry interact.

The paper is organized as follows. Section II, reviews some of the prior researches in generating fractal objects and their relations to computational geometry. Section III, gives a brief summary of the L-system in two dimensions with some examples. Section IV, it was explained how L-systems can be extended to three dimensions, and a proposed algorithm was introduced with simple analysis to its complexity. Some experimental results with discussions are given in sections V. Finally, in section VI, a conclusion and some directions for future work were given.

II. PRIOR RESEARCH

One of the basic methods of generating fractal objects is Lindenmayer system called L-system for short [2]. L-Systems were invented by Aristid Lindenmayer in the 1968 [5]. Lindenmayer noticed that complex biological plants and structures had recursive patterns and could be compactly represented through simple grammars (strings of text), called L-Systems. Lindenmayer published a book, The Algorithmic Beauty of Plants [6], where he displays the raw power of these simple L-Systems by producing beautifully magnificent plants and structures [7]. The framework of L-system consists of an initial structure and rewriting rules (or generating rules). The essence of development is parallel replacement using the
rewriting rules. Starting from the initial structure, L-system replaces each part of the current structure by applying the rule sequentially [8]. Along with the development of the theory, the L-system has been enriched with geometric aspects and has become a universal tool not only for modeling plants, but also for creating fractals which shape is a function of time.

Another idea, besides the rule of rewriting, of L-system is a method of notation the fractal's structure, based on graphical interpretation of a string of characters. This method is known in literature under the name of the turtle graphics, whose creator was Seymour Papert. The main idea of the turtle graphics is graphical interpretation of a string of characters in the form of commands given to a specially trained turtle. The method's tool is the turtle's tail which draws straight lines on the plane according to the received commands [2].

Computing The Delaunay triangulation is one of the most famous computational geometry problems. The Delaunay triangulation was invented in 1934 by, and named after, the Russian mathematician Boris Nikolaevich Delaunay (1890-1980). It has a lot of applications in science and computer graphics. It is often used in the graphic representation of geometrically irregularly distributed data—thinks weather maps or altitude maps. Its 3D-variant is important in creating virtual worlds for video games, among many other things [11].

The Delaunay triangulation involves creating from the sample points a set of non-overlapping triangularly bounded facets, the vertices of the triangles are the input sample points. There are a number of triangulation algorithms that may be advocated, the more popular algorithms are the radial sweep method and the Watson algorithm which implement Delaunay triangulation. The Delaunay triangulation is closely related to the Voronoi diagram also known as the Voronoi diagram [12].

A short list of commands by means of which the simplest fractals using the turtle graphics method can be created as follows [2]:
- F Move forward a step of constant length >0 draw a line segment from the previous to the new position.
- f Move forward a step of constant length >0 without drawing a line.
- + Turn left by constant angel $\delta$.
- – Turn right by constant angel $\delta$.

Axiom $\omega : F$

Rewriting rule $P : F \rightarrow F[+F][F[-F]F]$ (1)

Parameters $+ : \delta = 30^\circ$

Fig. 1 L-system generating a fractal resembling a weed

IV. OUR PROPOSED ALGORITHM

Turtle interpretation of L-systems can be extended to three dimensions. The key concept is to represent the current orientation of the turtle in space by three vectors $H$, $L$, $U$, ...
indicating the turtle’s heading, the direction to the left, and the
direction up. These vectors have unit length, are perpendicular
to each other, and satisfy the equation $H \times L = U$. Fig. 2
illustrates these vectors in three dimensions. Rotations of the
turtle are then expressed by the equation
\[
[H' \ L' \ U'] = [H \ L \ U]R
\]
where $R$ is a $3 \times 3$ rotation matrix. Specifically, rotations by
angle $\alpha$ about vectors $U$, $L$ and $H$ are represented by the
matrices [6]:
\[
R_U(\alpha) = \begin{bmatrix}
\cos \alpha & \sin \alpha & 0 \\
-\sin \alpha & \cos \alpha & 0 \\
0 & 0 & 1
\end{bmatrix}
\]
\[
R_L(\alpha) = \begin{bmatrix}
\cos \alpha & 0 & -\sin \alpha \\
0 & 1 & 0 \\
\sin \alpha & 0 & \cos \alpha
\end{bmatrix}
\]
\[
R_H(\alpha) = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \alpha & -\sin \alpha \\
0 & \sin \alpha & \cos \alpha
\end{bmatrix}
\]
Fig. 2 Controlling the turtle in three dimensions

It should be noted that the commands given to the turtle
correspond to the orientation connected with the local co-ordinate
system in which the turtle moves and which moves along with
the turtle. For the needs of this study it has been assumed that
initially, the turtle’s head is turned upwards, thus the command
$F$ means movement upwards, and commands $\&$ or $^\wedge$ mean an
inclination by angle $\alpha$ [2]. The algorithm can be described as
follows:
**Input:** the parameters of 3D L-system (axiom, rewriting rules,
number of iterations and orientation angels).
**Output:** the 3D fractal plant (obtained from the given L-
system and the 3D triangulation algorithm of the vertices
obtained from the last stage of the rewriting process).
**begin**
\[
\begin{align*}
&n = \text{number of iterations} \\
&k = \text{length of the initial axiom} \\
&\text{for } i = 1 \text{ to } n \text{ begin}
\end{align*}
\]
**end**
for $j = 1$ to $k$
replace the character with index $j$ in the axiom by the
character corresponding to the rewriting rule.
$k = \text{length of the axiom result from the previous step}$
for $i = 1$ to $k$
\[
\begin{align*}
&c = \text{character with index } i \text{ in the axiom} \\
&\text{switch } (c) \\
&\quad \text{case } 'F' \\
&\quad \text{move forward a step of constant length } a>0 \\
&\quad \text{draw a line segment from the previous to the } \\
&\quad \text{new position} \\
&\quad \text{case } 'F' \\
&\quad \text{move forward a step of constant length } a>0 \\
&\quad \text{without drawing a line} \\
&\quad \text{case } 'L' \\
&\quad \text{Turn left by angle } \alpha, \text{ using rotation matrix } \\
&\quad R_U(\alpha) \\
&\quad \text{case } 'R' \\
&\quad \text{Turn right by angle } \alpha, \text{ using rotation matrix } \\
&\quad R_U(-\alpha) \\
&\quad \text{case } 'G' \\
&\quad \text{pitch up by angel } \alpha, \text{ using rotation matrix } \\
&\quad R_L(\alpha) \\
&\quad \text{case } 'O' \\
&\quad \text{pitch down by angel } \alpha, \text{ using rotation matrix } \\
&\quad R_L(-\alpha) \\
&\quad \text{case } 'L' \\
&\quad \text{pitch up by angel } \alpha, \text{ using rotation matrix } \\
&\quad R_H(\alpha) \\
&\quad \text{case } 'R' \\
&\quad \text{pitch down by angel } \alpha, \text{ using rotation matrix } \\
&\quad R_H(-\alpha) \\
&\quad \text{case } 'G' \\
&\quad \text{roll left by angel } \alpha, \text{ using rotation matrix } \\
&\quad R_U(\alpha) \\
&\quad \text{case } 'O' \\
&\quad \text{roll right by angel } \alpha, \text{ using rotation matrix } \\
&\quad R_U(-\alpha) \\
&\quad \text{case } 'G' \\
&\quad \text{roll left by angel } \alpha, \text{ using rotation matrix } \\
&\quad R_L(\alpha) \\
&\quad \text{case } 'O' \\
&\quad \text{roll right by angel } \alpha, \text{ using rotation matrix } \\
&\quad R_L(-\alpha) \\
&\quad \text{case } 'L' \\
&\quad \text{pitch up by angel } \beta, \text{ using rotation matrix } \\
&\quad R_L(\beta) \\
&\quad \text{case } 'O' \\
&\quad \text{pitch down by angel } \beta, \text{ using rotation matrix } \\
&\quad R_L(-\beta) \\
&\quad \text{case } 'L' \\
&\quad \text{roll left by angel } \beta, \text{ using rotation matrix } \\
&\quad R_U(\beta) \\
&\quad \text{case } 'O' \\
&\quad \text{roll right by angel } \beta, \text{ using rotation matrix } \\
&\quad R_U(-\beta) \\
&\quad \text{case } 'G' \\
&\quad \text{pitch up by angel } \gamma, \text{ using rotation matrix } \\
&\quad R_H(\gamma) \\
&\quad \text{case } 'O' \\
&\quad \text{pitch down by angel } \gamma, \text{ using rotation matrix } \\
&\quad R_H(-\gamma) \\
&\quad \text{case } 'L' \\
&\quad \text{roll left by angel } \gamma, \text{ using rotation matrix } \\
&\quad R_U(\gamma) \\
&\quad \text{case } 'O' \\
&\quad \text{roll right by angel } \gamma, \text{ using rotation matrix } \\
&\quad R_U(-\gamma)
\end{align*}
\]
case 'l'
    meeting this command causes the turtle's current state is remembered
end

case 'r'
    meeting this command makes the turtle come back to the state before the symbol [ 
otherwise
    return (error)
end

A [m, 3] = m by 3 array of the m vertices obtained from the last stage of the rewriting process.
Apply The Delaunay triangulation algorithm to the vertices in A.

end

The analysis of the time complexity of the proposed algorithm is given as follows. The key of the algorithm is to
construct the connectivity of the output vertices correctly and to accomplish this task efficiently. For any given example let k
is the length of the longest rule in the example. Then at most O(kn) is needed to calculate the final string of the axiom
where n is the number of iterations, O(kn) to draw the fractal plant according to the resulting final string of the axiom. Now
since, the Delaunay triangulation algorithm needs O(n log n) in time in the worst case [13], then
O(kn log kn) in time is needed to compute the triangulation of the vertices obtained from the last stage of the rewriting
process. Hence the total run time of our algorithm is O(kn) + O(kn log kn) ≈ O(kn log kn).

V. EXPERIMENTAL RESULTS AND DISCUSSIONS

Based on the proposed algorithm, the following examples illustrate the successive steps of the algorithm:

Example 1
ω : F(a)
P : F(a) → F(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][ F(0.5a)]
& ^ = 50°
\ / = 50°
n = 3

It is noticed that depending on the accepted formula of the L-System, examples of different triangulation shapes can be
obtained. Selected examples are presented in Table 1. Results are shown in Figures 4-15

Example 2
ω : F(a)
P : F(a) → F(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][ F(0.5a)]
& ^ = 30°
\ / = 30°
n = 3

Example 3
ω : F(a)
P : F(a) → F(a)[^F(1/3a)][&F(1/3a)][/ F(1/3a)][ F(1/3a)]
& ^ = 70°
\ / = 70°
n = 3
Example 4
\[ \omega \quad : \quad \text{X} \]
\[ P : F(a) \rightarrow B(0.5a) [ F(0.5a) ] [ & F(0.5a) ] [ / F(0.5a) ] [ / F(0.5a) ] [ F(a) ] \]
\[ \text{X} \rightarrow A(0.5a) [ * F(a) ] [ & F(a) ] [ / F(a) ] [ F(a) ] \]
& \^ = 30^\circ
\]
/ = 30^\circ
\]
n = 3

Example 5
\[ \omega \quad : \quad \text{X} \]
\[ P : F(a) \rightarrow B(0.5a) [ F(0.5a) ] [ & F(0.5a) ] [ / F(0.5a) ] [ / F(0.5a) ] [ F(a) ] \]
\[ S \rightarrow B(0.5a) [ * F(1/3a) ] [ & F(1/3a) ] [ / F(1/3a) ] [ F(1/3a) ] \]
\[ \text{X} \rightarrow A(0.5a) [ * S ] [ & S ] [ / F(a) ] [ F(a) ] \]
& \^ = 30^\circ
\]
/ = 30^\circ
\]
n = 3

Example 6
\[ \omega \quad : \quad \text{X} \]
\[ P : F(a) \rightarrow B(0.5a) [ F(0.5a) ] [ & F(0.5a) ] [ / F(0.5a) ] [ F(0.5a) ] \]
\[ S \rightarrow C(1/3a) [ * F(0.5a) ] [ & F(0.5a) ] [ / F(0.5a) ] [ F(0.5a) ] \]
\[ \text{X} \rightarrow A(a) [ * S ] [ & S ] [ / F(a) ] [ F(a) ] \]
& \^ = 30^\circ
\]
/ = 30^\circ
\]
n = 3

Example 7
\[ \omega \quad : \quad [ F(a) ][ B(a) ] \]
\[ P : F(a) \rightarrow F(a) [ F(0.5a) ] [ & F(0.5a) ] [ / F(0.5a) ] [ F(0.5a) ] \]
\[ B(a) \rightarrow B(a) [ F(0.5a) ] [ & F(0.5a) ] [ / F(0.5a) ] [ F(0.5a) ] \]
& \^ = 30^\circ
\]
/ = 30^\circ
\]
n = 3
Example 8
\[ \omega : [F(a)][B(a)] [C(a)][D(a)] \]
\[ P : F(a) \rightarrow F(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)] \]
\[ B(a) \rightarrow B(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)] \]
\[ C(a) \rightarrow C(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)] \]
\[ D(a) \rightarrow D(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)] \]
& \ = 30^\circ \\
/ \ = 30^\circ \\
n = 3

Example 9
\[ \omega : F(a)\] 
\[ P : F(a) \rightarrow F(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)] \]
\[ G(0.5a) \rightarrow G(0.5a)*B(0.25a) ] [%B(0.25a)] \]
\[ B(0.25a) \rightarrow B(0.25a)[S(1/8a)][S(1/8a)]\# S(1/8a)] \]
& \ = 60^\circ \\
/ \ = 60^\circ \\
* \ = 600 \\
* \ = 60\^o \\
S \ = 20\^o \\
! \ = 20\^o \\
n = 3

Example 10
\[ \omega : B(2a)[@F(a)][F(a)][@F(a)] \]
\[ P : F(a) \rightarrow F(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)] \]
& \ = 30^\circ , / \ = 30^\circ \\
@ \ = 50^\circ , ! \ = 50^\circ \\
n = 2

Example 11
\[ \omega : F(a)\] 
\[ P : F(a) \rightarrow F(a)[^F(1/3a)][&F(1/3a)][/ F(1/3a)][\ F(1/3a)] \]
& \ = 120^\circ , / \ = 120^\circ , n = 2

Fig. 9 (a) The fractal plant generated from the L-System of Example 7 renders as lines 
(b) Covering the resulting fractal plant with triangulation 
(c) The resulting triangulation shape

Fig. 10 (a) The fractal plant generated from the L-System of Example 8 renders as lines 
(b) Covering the resulting fractal plant with triangulation 
(c) The resulting triangulation shape

Fig. 11 (a) The fractal plant generated from the L-System of Example 9 renders as lines 
(b) Covering the resulting fractal plant with triangulation 
(c) The resulting triangulation shape

Fig. 12 (a) The fractal plant generated from the L-System of Example 10 renders as lines 
(b) Covering the resulting fractal plant with triangulation 
(c) The resulting triangulation shape

Fig. 13 (a) The fractal plant generated from the L-System of Example 11 renders as lines 
(b) Covering the resulting fractal plant with triangulation 
(c) The resulting triangulation shape
Example 12
\[ \omega : F(a) \]
\[ P : F(0.5a)[F(0.5a)]/[F(0.5a)] \]
\[ \wedge = 120^\circ \]
\[ n = 2 \]

Fig. 14 (a) The fractal plant generated from the L-System of
Example 12 renders as lines
(b) Covering the resulting fractal plant with triangulation
(c) The resulting triangulation shape

Example 13
\[ \omega : F(a) \]
\[ P : F(0.5a)[F(0.5a)]/[F(0.5a)] \]
\[ \wedge = 30^\circ \]
\[ n = 3 \]

Fig. 15 (a) The fractal plant generated from the L-System of
Example 13 renders as lines
(b) Covering the resulting fractal plant with triangulation
(c) The resulting triangulation shape

VI. CONCLUSION AND FUTURE WORK

L-system is a tool commonly used for modeling and simulating the growth of plants [8]. In this paper, a tree modeling system based on L-system was proposed that allows the user to control the overall appearance, the depth of recursions, the length of branches and the rotation angles (for each iteration). Also, the triangulation shape can be controlled according to the resulting tree model. It is shown that changes in initial conditions when reiterated could cause big changes in the resulting triangulation shape. As shown in Example 1 and Example 13 change in the rotation angel causes big changes in the resulting triangulation shape.

Fractals have been used with varying success in a wide range of scientific fields such as medicine, biological systems, astrophysics, computer and video games and recently in architecture. The proposed algorithm was built on the symmetric construction of the tree branches. It is hoped in the future work that the construction of tree branches becomes randomly.

REFERENCES

TABLE I
RESULTS OF FIGURES 4-15

<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Faces</th>
<th>Vertices</th>
<th>Edges</th>
<th>Run Time in Sec.</th>
<th>Number of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>110</td>
<td>64</td>
<td>172</td>
<td>3.1040</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>212</td>
<td>125</td>
<td>335</td>
<td>6.5700</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>244</td>
<td>125</td>
<td>367</td>
<td>7.7510</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>98</td>
<td>64</td>
<td>160</td>
<td>3.8060</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>110</td>
<td>64</td>
<td>172</td>
<td>3.9360</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>102</td>
<td>64</td>
<td>164</td>
<td>3.8160</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>222</td>
<td>128</td>
<td>348</td>
<td>5.5280</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>474</td>
<td>256</td>
<td>728</td>
<td>7.8810</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>106</td>
<td>64</td>
<td>168</td>
<td>3.5050</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>102</td>
<td>64</td>
<td>164</td>
<td>4.2860</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>36</td>
<td>25</td>
<td>59</td>
<td>3.0950</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>16</td>
<td>36</td>
<td>2.0930</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>98</td>
<td>64</td>
<td>160</td>
<td>3.6350</td>
<td>3</td>
</tr>
</tbody>
</table>