International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:2, No:8, 2008

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

Y. M. Abd El-Latif, Phd Computer Science, Faculty of Science, Ain Shams University, Cairo, Egypt. (e-mail: Y.AbdEllatif@Gmail.com).

F. S.Abousaleh,, was with Zagazig University, Cairo, Egypt. She has a scholarship with Faculty of Science, Ain Shams University (e-mail: fatma_said6@yahoo.com).

Daoud S. S., Faculty of Science, Ain Shams University, Cairo, Egypt. (e-mail: sameh_daoud2003@yahoo.com).

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]. 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 geometrically to the Direchlet tesselation also known as the Voronoi diagram [12]. The Voronoi diagram of a set $S = \{p_1, p_{2, \dots}, p_n\}$ of points in the plane, called sites, is a partitioning of the plane into n convex regions, one per site. Each Voronoi region V_i contains all points in the plane closer to p_i than to any other site. The straight line dual of the Voronoi diagram, obtained by adding a line segment between each pair of sites of S whose Voronoi regions share an edge, is called the Delaunay triangulation. Given the Voronoi diagram of a set of sites, $V(S)$, the Delaunay triangulation of those sites, D(S), can be obtained in $O(n)$ time and vice versa [13].

One of the new research based on L-System made by Piotr Furmanek in which he present a modification of L-system to generate 3D fractal plant which could be applied as the supporting construction for polyhedral covers of architectural objects [2]. “Fractal geometry and its applications in the field of construction” is another new research. This research project aims to translate virtual fractal models into physically built architectural objects. The considered fractal models are based on iterative algorithms, which were developed at the LIRIS for the creation of virtual images. The physical objects will aim for an application in the field of construction on the scale of architecture and design objects. Fractal objects will be designed and built as bearing shell structures, irregular three dimensional polygonal structures [9].

III. L-SYSTEMS

The basic idea of the L-system is a rule of rewriting, also called a rule of replacing. Its operation can be presented in terms of intuition by means of the following example [2]:

Consider a string built of elements ‘a’ and ‘b’, which can occur many times in the string. Each element is associated with one rewriting rule P. The notation P: $a \rightarrow b$ means that element ‘a’ is replaced with element ‘b’, and the notation P: $a \rightarrow ab$ means that element ‘b’ is replaced with two element string ‘ab’. The rewriting process starts from distinguished string called the axiom ω. Assumptions constructed in this way make it possible to generate the following example string of elements [6].

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 δ.
– Turn right by constant angel δ.
[Meeting this command causes that the turtle's current state is remembered.
] Meeting this command makes the turtle come back to the state before the symbol [.
[…] Between these symbols occur commands defining the construction of the branches.

These notations can be illustrated by the following simple example. Fig. 1 displays the L-system processes, which are created as using the rewriting rule below:

Axiom $\omega : F$
Rewriting rule $P : F \rightarrow F[+F]F[-F]F$ (1)
Parameters $+ - : \delta = 30^{o}$

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×L = U. Fig. 2 illustrates these vectors in three dimensions. Rotations of the turtle are then expressed by the equation

$$\left[\vec{H}' \quad \vec{L}' \quad \vec{U}'\right] = \left[\vec{H} \quad \vec{L} \quad \vec{U}\right] R \tag{2}$$

where R is a 3×3 rotation matrix. Specifically, rotations by angle α about vectors U, L and H are represented by the matrices [6]:

$$\mathbf{R_U}(\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R_L}(\alpha) = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{3}$$

$$\mathbf{R_H}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

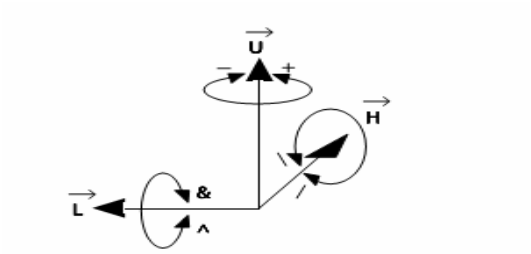


Fig. 2 Controlling the turtle in three dimensions

It should be noted that the commands given to the turtle concern 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 ^ mean an inclination by angle α [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

n = number of iterations
k = length of the initial axiom
for *i* = 1 to *n*
begin
for *j* = 1 to *k*
replace the character with index *j* in the axiom by the corresponding rewriting rule.
k = length of the axiom result from the previous step
end
for *i* = 1 to *k*
begin
c = character with index *i* in the axiom
switch (*c*)
case 'F'
move forward a step of constant length *a*>0 draw a line segment from the previous to the new position
case 'f'
move forward a step of constant length *a*>0 without drawing a line
case '+'
Turn left by angle α, using rotation matrix $R_U(\alpha)$
case '-'
Turn right by angle α, using rotation matrix $R_U(-\alpha)$
case '&'
pitch up by angel α, using rotation matrix $R_L(\alpha)$
case '^'
pitch down by angel α, using rotation matrix $R_L(-\alpha)$
case '\'
roll left by angel α, using rotation matrix $R_H(\alpha)$
case '/'
roll right by angel α, using rotation matrix $R_H(-\alpha)$
case '*'
pitch up by angel β, using rotation matrix $R_L(\beta)$
case '%'
pitch down by angel β, using rotation matrix $R_L(-\beta)$
case '~'
roll left by angel β, using rotation matrix $R_H(\beta)$
case '?'
roll right by angel β, using rotation matrix $R_H(-\beta)$
case '$'
pitch up by angel γ, using rotation matrix $R_L(\gamma)$
case '@'
pitch down by angel γ, using rotation matrix $R_L(-\gamma)$
case '!'
roll left by angel γ, using rotation matrix $R_H(\gamma)$
case '#'
roll right by angel γ, using rotation matrix $R_H(-\gamma)$

case '['
meeting this command causes the turtle's current state is remembered
case ']'
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

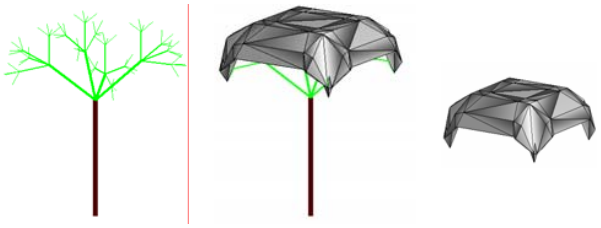
The analysis of the time complexity of the proposed algorithm is given as follows. The key of the algorithm is to construct the connectivity of the output vertices correctly and to accomplish this task efficiently. For any given example let *k* is the length of the longest rule in the example. Then at most $O(k^n)$ is needed to calculate the final string of the axiom where *n* is the number of iterations, $O(k^n)$ 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 $O(k^n \log k^n)$ 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 $O(k^n)$ + $O(k^n)$ + $O(k^n \log k^n) \approx O(k^n \log k^n)$.

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

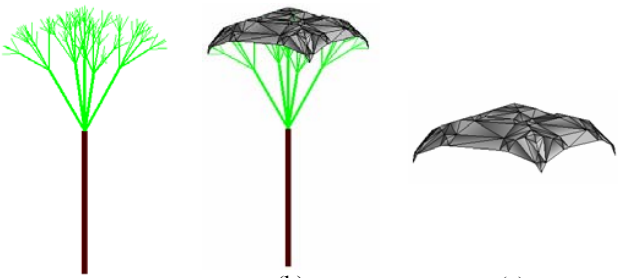
It is noticed that depending on the accepted formula of the L-System, examples of different triangulation shapes can be obtained. Selected examples are presented in Table 1. Results are shown in Figures 4-15



(a) (b) (c)

Fig. 3 (a) The fractal plant generated from the L-System of Example 1 renders as lines.
(b)Covering the resulting fractal plant with triangulation shape.
(c) The resulting triangulation shape

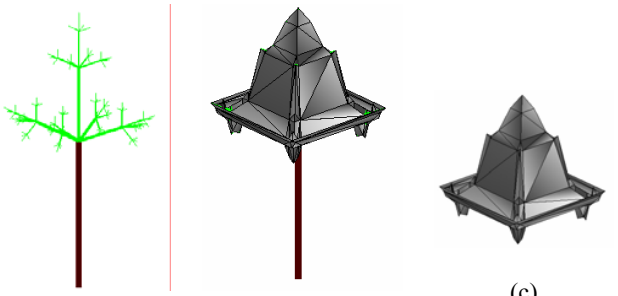
Example 2
ω : F(a)
P : F(a) ⟶ F(a)[F(0.5a)][^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)]
& ^ = 30°
\ / = 30°
n = 3



(a) (b) (c)

Fig. 4 (a) The fractal plant generated from the L-System of Example 2 renders as lines.
(b) Covering the resulting fractal plant with triangulation shape.
(c) The resulting triangulation shape

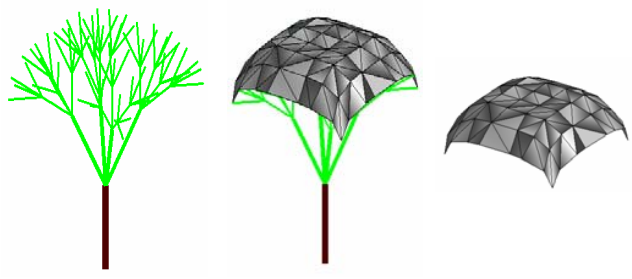
Example 3
ω : F(a)
P : F(a) ⟶ F(a)[F(0.5a)][^F(1/3a)][&F(1/3a)][/ F(1/3a)][\ F(1/3a)]
& ^ = 70°
\ / = 70°
n = 3



(a) (b) (c)

Fig. 5 (a) The fractal plant generated from the L-System of Example 3 renders as lines
(b) Covering the resulting fractal plant with triangulation
(c) The resulting triangulation shape

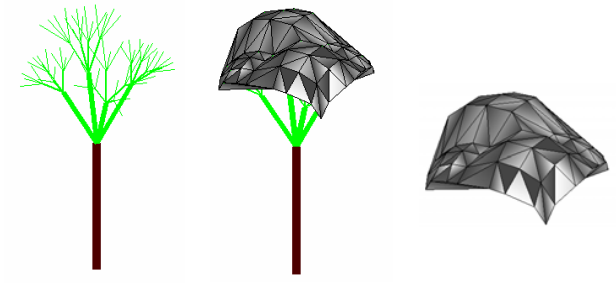
Example 4
ω : X
P : F(a) → B(0.5a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)]
X → A(0.5a)[^F(a)][&F(a)][/ F(a)][\ F(a)]
& ^ = 30°
\ / = 30°
n = 3



(a) (b) (c)

Fig. 6 (a) The fractal plant generated from the L-System of Example 4 renders as lines
(b) Covering the resulting fractal plant with Triangulation
(c) The resulting triangulation shape

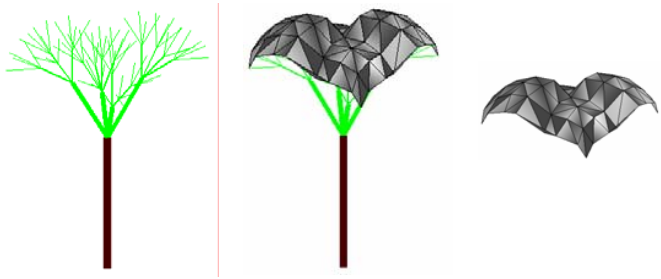
Example 5
ω : X
P : F(a) → B(0.5a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)]
S → B(0.5a) [^F(1/3a)][&F(1/3a)][/ F(1/3a)][\ F(1/3a)]
X → A(a) [^S] [&S][/ F(a)][\ F(a)]
& ^ = 30°
\ / = 30°
n = 3



(a) (b) (c)

Fig. 7 (a) The fractal plant generated from the L-System of Example 5 renders as lines
(b) Covering the resulting fractal plant with triangulation
(c) The resulting triangulation shape

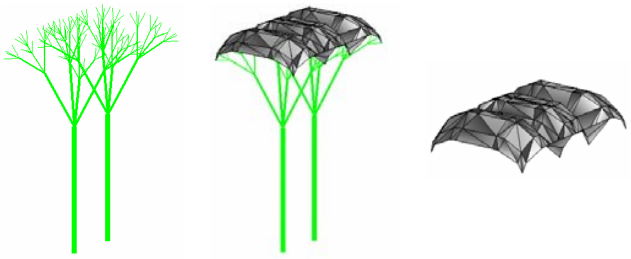
Example 6
ω : X
P : F(a) → B(0.5a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\F(0.5a)]
S → C(1/3a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)]
X → A(a)[^S][&S][/ F(a)][\ F(a)]
& ^ = 30o
\ / = 30o
n = 3



(a) (b) (c)

Fig. 8 (a) The fractal plant generated from the L-System of Example 6 renders as lines
(b) Covering the resulting fractal plant with triangulation
(c) The resulting triangulation shape

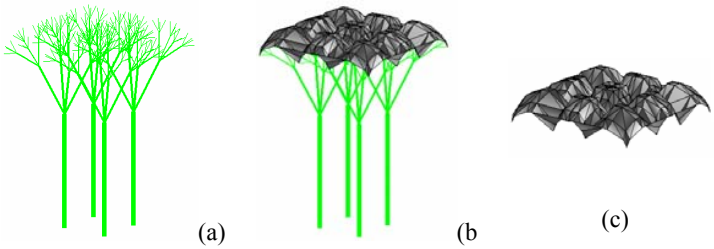
Example 7
ω : [F(a)][B(a)]
P : F(a) → F(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)]
B(a) → B(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)]
& ^ = 30°
\ / = 30°
n = 3



(a) (b) (c)

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

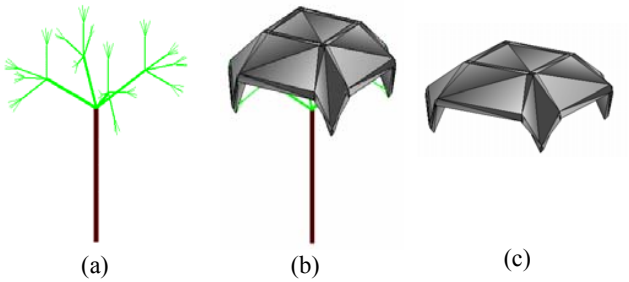
Example 8
ω : [F(a)][B(a)] [C(a)][D(a)]
P : F(a) → F(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)]
B(a) → B(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)]
C(a) → C(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)]
D(a) → D(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)]
& ^ = 30°
\ / = 30°
n = 3



(a) (b) (c)

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

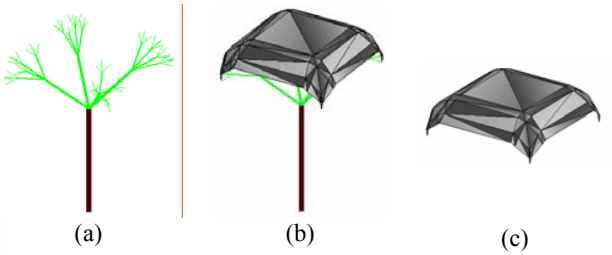
Example 9
ω : F(a)
P : F(a) → F(a)[^G(0.5a)][&G(0.5a)][/ G(0.5a)][\ G(0.5a)]
G(0.5a) → G(0.5a)[*B(0.25a)] [%B(0.25a)] [~B(0.25a)][?B(0.25a)]
B(0.25a) → B(0.25a)[$S(1/8a)][@S(1/8a)][! S(1/8a)][# S(1/8a)]
& ^ = 60°
\ / = 60°
* % = 600
~ ? = 60°
$ @ = 20°
! # = 20°
n = 3



(a) (b) (c)

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

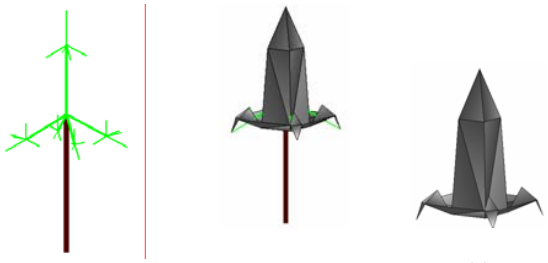
Example 10
ω : B(2a)[@F(a)][$F(a)][!F(a)][#F(a)]
P : F(a) → F(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)]
& ^ = 30° , \ / = 30°
@ $ =50° , ! # =50°
n = 2



(a) (b) (c)

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

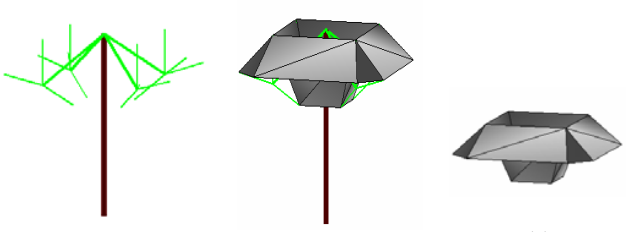
Example 11
ω : F(a)
P : F(a) → F(a)[F(0.5a)][^F(1/3a)][&F(1/3a)][/ F(1/3a)][\ F(1/3a)]
& ^ =120° , \ / = 120° , n = 2



(a) (b) (c)

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

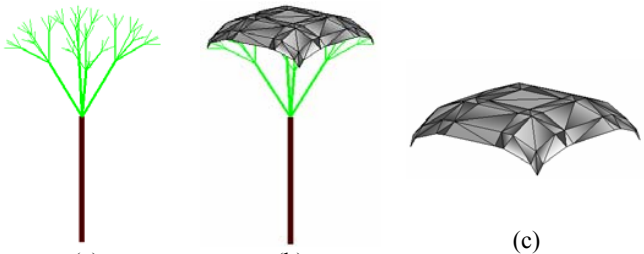
Example 12
ω : F(a)
P : F(a)⟶ F(a)[^F(0.5a)][&F(0.5a)][/ F(0.5a)][\ F(0.5a)]
& ^ =120°
\ / = 120°
n = 2



(a) (b) (c)

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



(a) (b) (c)

Fig. 15 (a) The fractal plant generated from the L-System of Example13 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 angels(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 Example1 and Example13 change in the rotation angel causes big changes in the resulting triangulation shape.

TABLE I
Results of Figures 4-15

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

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

[1] D. Hearn and M.P. Baker, Computer Graphics. C version, 2nd ed, CA: A Viacom Company Upper Saddle River, New Jersey 07458, 1997, pp. 362-363.
[2] P. Furmanek, "POLYHEDRAL COVERS BASED ON L-SYSTEM FRACTAL CONSTRUCTION," The Journal of Polish Society for Geometry and Engineering Graphics, vol. 14, 2004, pp. 40-47.
[3] From Wikipedia and the free encyclopedia, "Fractal". Online: *http://en.wikipedia.org/wiki/Fractal*
[4] S. Sarkar, "Introduction to Computation Geometry," Spring 2008. Online: http://figment.csee.usf.edu/~sarkar/ComputationalGeometry/
[5] A. Lindenmayer, "Mathematical models for cellular interaction in development," parts I and II, Journal of Theoretical Biology, vol. 18, 1968, pp.280-315.
[6] P. Prusinkiewicz and A. Lindenmayer, The algorithmic beauty of plants. New York: Springer-Verlag, 1990, ch. 1.
[7] B. D. Farkas, N. Romanyshyn, and P. Clary, "L- Systems Construction Kit". Online: http://l3d.cs.colorado.edu/~ctg/classes/ttt2005/projectreports/Lsystemreport2.pdf
[8] T. Ijiri, S. Owada, and T. Igarashi, "The Sketch L-System: Global Control of Tree Modeling Using Free-form Strokes," Smart Graphics (2006), pp. 138-146.
[9] P. Buser, E. Tosan, and Y. Weinand, "Fractal Geometry and its applications in the field of construction," summer 2005. Online:http://fractals-ibois.epfl.ch/wiki/images/105c06_project_plan.pdf
[10] D. P. Dobkin, "Computational Geometry and Computer Graphics," in Proc. *IEEE*, vol. 80, Issue 9, Sep. 1992, pp. 1400 – 1411.
[11] S. Priester, "Delaunay Triangles," July 19, 2005. Online:http://www.codeguru.com/Cpp/data/mfc_database/misc/article.php/c8901/
[12] P. Bourke, "Triangulate Efficient Triangulation Algorithm Suitable for Terrain Modelling" or "An Algorithm for Interpolating Irregularly-Spaced Data with Applications in Terrain Modelling," presented at the 1989 Pan Pacific Computer Conference, Beijing, China.
[13] G. Leach, "Improving Worst-Case Optimal Delaunay Triangulation Algorithms," In 4th Canadian Conference on Computational Geometry, Canada, June 15, 1992.