芸術科学会論文誌 Vol.2 No.1 pp.61-70

Automatic Generation of 3-D Linear Fractal Shapes with Quadratic Map Basins Animation

Hussein KARAM
Ain Shams University
Faculty of Science
Math. & Computer Sc. Dept.
Abbassia, Cairo, Egypt

Masayuki NAKAJIMA
Graduate School of Information Sc. & Eng.,
Tokyo Institute of Technology
2-12-1 O-okayama, Meguro-ku, Tokyo,
152-8552 Japan

Abstract

Quadratic map basins is a method used to generate the images of Julia and Mandelbrot sets and has been applied to visualize the attractors of iterated function systems (IFS). Three dimensional extension of such mapping with animation is an aspiring goal and a challenging task. In this paper an extension algorithm of quadratic map basin for classifying points in the complement of a 3-D linear fractal shapes are discussed. Such extension produces fractal surfaces that exhibit self-similarity and suggest smooth evolution under animation. The proposed technique has fast computations and simple representation whereby a computer can automatically select parameters and generate a large collection of aesthetically appealing 3-D fractal patterns. The simple representation and the speed of computation of our algorithm make 3-D fractal patterns reliable for interactive machine.

1 Introduction

Several methods for modeling and realistic visualization of complex forms found in nature have been developed in the history of computer graphics. These methods are more or less related to Mandelbrot's fractal geometry of nature [15]. Fractal methods are quite popular in the modeling of natural phenomena in computer graphics ranging from random fractal models of terrain [15], to deterministic botanical models such as L-systems [17], iterated function systems (IFS), recurrent iterated function systems (RIFS) and language iterated function systems [1, 5, 3, 4, 16]. On the other hand, methods for classifying points in the complement of a fractal shapes were originally developed in the complex plane C based on the escape-time algorithm [19, 12]. The escape-time visualization method was extended from Julia and Mandelbrot sets to more higher order [21, 22], and several methods for classifying divergent points in the complement of a fractal shapes to generate fractal patterns in two dimension were described in [18, 10, 8, 24]. These methods plotting simple discrete level sets by counting the number of function applications required to transform a point outside a large circle. Although these methods generate an aesthetically appealing 2-D fractal patterns, an extension algorithm for visualizing points in the complement of the 3-D fractal patterns still aspiring goals, challenging tasks and harder to be described. In this paper we propose an extension algorithm for rendering the IFS escape-time behavior in three dimension without inverting the IFS transformations which is computationally expensive, therefore reducing the time of complexity. The proposed algorithm has two basic steps. The first is based on the way of approximating the fractal attractor and the second is for creating numerical data that characterize the space outside the attractor. The advantages of the proposed method are that it is neither explicitly define regions, nor follows all point iteration possibilities. Moreover, it is simple to implement and its speed of computation allows visible surface rendering to be performed simply and efficiently. In addition makes rendering 3-D fractal patterns reliable for interactive machines.

The paper is organized as follows. Section 2, surveys the key principles and basic notions of escape-time visualization method, iterated function systems

IFS, and recurrent iterated function systems $RIFS$. The dynamics of the escape-time classifications of quadratic fractals, IFS and RIFS are explained in section 3. In section 4, the proposed algorithm for extending the IFS escape-time from 2-D to 3-D linear fractals is explained. Fractal rendering technique is introduced in section 5. Some experimental results with discussion are given in section 6. Finally, conclusion and direction for future work are discussed in section 7.

2 Background and Basic Notions

The common ways to generate fractals is through iterated function systems IFS, recurrent iterated function system RIFS and quadratic map basins based on escape-time algorithm. The mathematical theory of both IFS and RIFS are based on the contraction mapping theorem which is described in details in [2, 6, 13]. Mathematically, the IFS is usually defined as a pair $\{X; T_n, n = 1, 2, \cdots, N\}$, where X is a complete metric space and each T_n are affine contractions, that is

$$T_i(x) = C_i x + B_i \tag{1}$$

where C_i is a square matrix with n rows and B_i is a vector with n elements. By a theorem of Hutchinson[13], there exist for each IFS a single compact non empty set $\hat{A}$, called its attractor which is the union of images of itself under the IFS maps, that is, $\hat{A} = \bigcup_{i=1}^{N} T_i(\hat{A})$. On the other hand, recurrent modeling is the process of partitioning an object into components and representing each component using smaller copies of itself and/or other components [2, 3]. The determination of which components are used is controlled by a graph. It consists of an IFS $W = \{T_i\}_{i=1}^{N}$ as well as a control graph G consisting of N vertices corresponding to the IFS maps and directed edges denoted by the ordered pair $< i, j >$. The notation $< i, j > \in G$ indicates that graph G contains a directed edge from vertex i to vertex j, and implies that transformation T_j may be applied directly after transformation T_i. Figure 1, shows an example of a directed graph with $N = 3$ and numbers $p_{ij} \geq 0$ which represents the probabilities of transfer among the vertices.

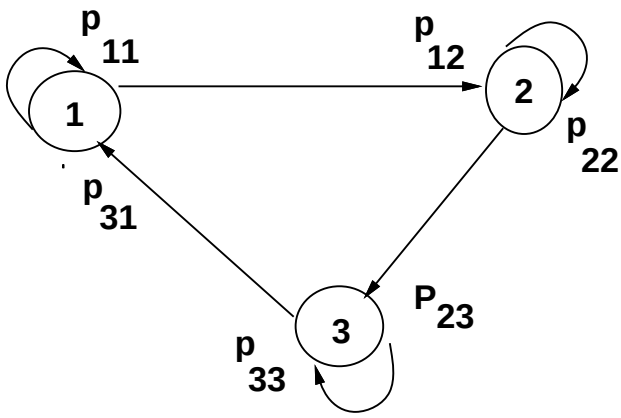
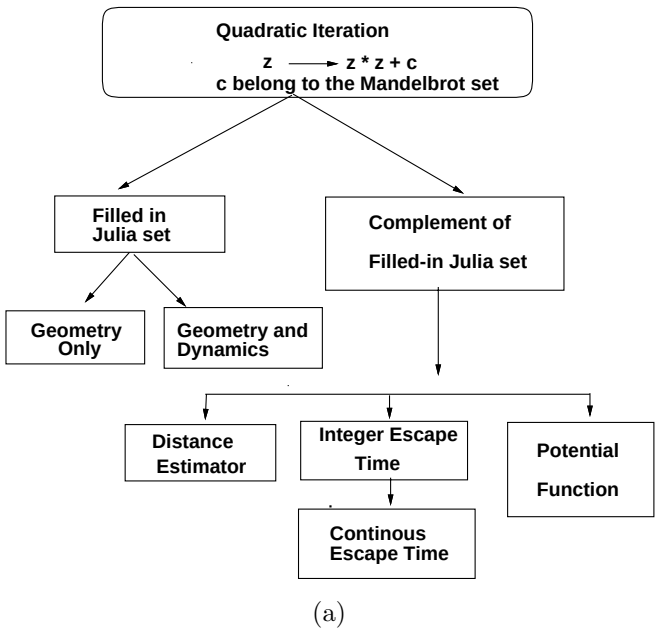
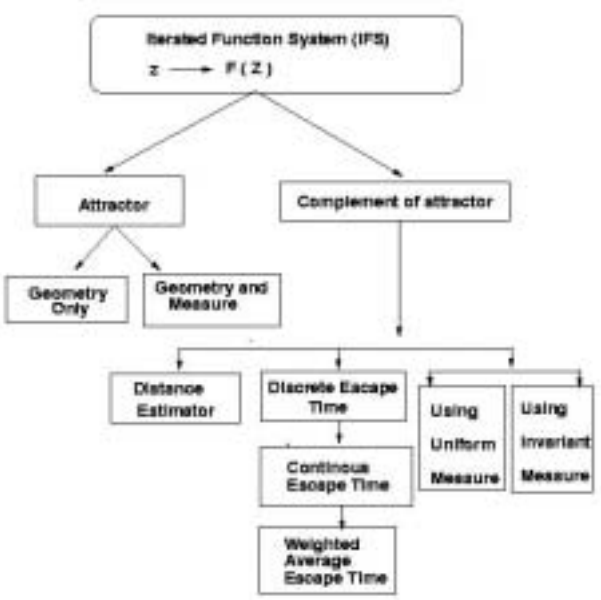


Figure 1. RIFS Directed Graph Example.

Quadratic map basins based on escape-time algorithm was originally developed as a method for visualizing the dynamics of complex quadratic fractals and has been extended to linear fractals modeled by IFS and RIFS. An overview of the 2-D rendering methods classifications of the dynamics for the quadratic iteration maps and its corresponding IFS escape-time are shown in figure $2(a) - (b)$.



(a)



(b)

Figure 2. 2-D rendering methods classifications in (*a*) Quadratic maps and (*b*) IFS maps.

3 Escape-time Classifications

Escape-time method has two forms: the first one is called discrete form and the other is called continuous form[10, 24]. The discrete form of the escape-time classification counts the number of iterations for a point to iterate outside the infinity circle. The continuous form smoothly interpolates the area between the discrete escape-time boundaries based on the location of the point that escapes the infinity circle. This section briefly explains the different escape-time forms for representing the 2-D fractal patterns using the IFS and their corresponding methods of the quadratic iteration function. Then, our extension algorithm of the IFS escape-time computation from 2-D to 3-D fractal patterns will be discussed in the next section.

3.1 Discrete Escape-time

Definition 3.1 (Quadratic Discrete Escape-time) *Given the function $F_c(z) = z^2 + c$, a disk D_R centered at the origin and of radius R sufficient such that the filled-in Julia set $K_c \subset D_R$, then the discrete escape-time $DT : C \to Z$ is given by*

$$DT(z) = min\{t : ||F_c^t(z)|| \geq R\} \quad (2)$$
$$= \begin{cases} 1 + DT(F_c(z)) & if\ ||z|| < R \\ 0 & otherwise \end{cases}$$

3.2 Discrete Escape-time for IFS and RIFS

The concept of the escape-time function can be extended for IFS in which all transformations are invertible. In order to define the escape-time function for iterated function systems, let us review the analogous concept for Julia sets. The Julia set for the quadratic mapping Equation(1), can be viewed as the attractor $\hat{A}$ defined by the nonlinear IFS:

$$T_1(z) = \sqrt{z - c}, \quad T_2(z) = -\sqrt{z - c} \quad (3)$$

Equation (8), illustrate that inversion of an IFS is required to compute its escape-time classification.

Definition 3.2 (IFS Discrete Escape-time) *Given an IFS $W = \{T_1, T_2, \cdots, T_N\}$ with attractor $\hat{A}$, let D_R be a disk of radius R centered at the origin sufficiently large such that,*

$$W(D_R) \subset D_R, \quad (4)$$

then the IFS discrete escape-time is given by the function $DT : R^2 \to R$ which is defined recurrently as:

$$DT(x) = \begin{cases} 1 + MAX & if\ x \in D_R \\ 0 & otherwise \end{cases}$$

Where,

$$MAX = \max_{i=1,2,..,N} DT(T_i^{-1}(x), D_R)$$

The discrete escape-time function is the maximum number of inverse transformations $T_i^{-1} \in W^{-1}$ necessary to iterate x to a point outside D_R.

Definition 3.3 (RIFS Discrete Escape-time) *Given an RIFS W, G with attractor $\hat{A}$, Let $R > 0$ satisfy condition (10). Then the discrete escape-time is given by $DT(x) = \max_{i=0,..,N} DT_i(x)$ where the functions $DT_i : R^2 \to R$ are defined as follows*

$$DT_i(x) = \begin{cases} D & \text{if } x \in D_R \\ 0 & otherwise \end{cases}$$

Where,

$$D = 1 + \max_{<j,i> \in G} DT_j(T_i^{-1}(x))$$

3.3 Continuous Escape-time Function

A continuous escape-time function requires a means to interpolate smoothly between the boundaries of the discrete escape-time level sets. Computation of the continuous escape-time consists of computing the discrete escape-time and determining the residual of the point just before it escapes the infinity circle.

Definition 3.4 (The Residual Function) *Given an (R)IFS $W = \{T_1, T_2, \cdots, T_N\}$ and the radius R of the infinity circle, interpolation between the boundaries of the discrete escape-time level sets is given by the residual function Res_i defined as*

$$Res_i(x) = \frac{\log(||x||^2/R^2)}{\log(||x||^2/||T_i^{-1}(x)||^2)} \tag{5}$$

for each map T_i^{-1} of the inverted IFS. We define the residue of an IFS as:

$$res(x) = \max_{i=1,\cdots,N} Res_i(x) \tag{6}$$

The residue components and the resulting maximum for IFS of the Sierpinski triangle are shown in figure 3. The left three images plot the $res_i()$ function for $i = 1, 2, 3$ respectively. The image on the right plots the resulting $res()$ function.

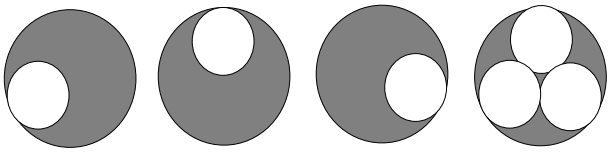


Figure 3. Residual function for an IFS

Definition 3.5 (Continuous Escape-time) *Let $W = \{T_1, T_2, \cdots, T_N\}$ be an IFS with invertible transformations and attractor $\hat{A}$. Let $R > 0$ satisfy condition (9). The continuous escape-time $CT : R^2 \to R$ is defined as follows*

$$CT(x) = \begin{cases} 1 + MAXX & Cond1 \\ res(x) & Cond2 \\ 0 & otherwise \end{cases}$$

Where,

$$MAXX = max_{i=1,\cdots,N} CT(T_i^{-1}(x))$$
$$Cond1 = \text{if } x \in D_R, T_i^{-1}(x) \in D_R(\exists i)$$
$$Cond2 = \text{if } x \in D_R, T_i^{-1}(x) \notin D_R(\forall i$$

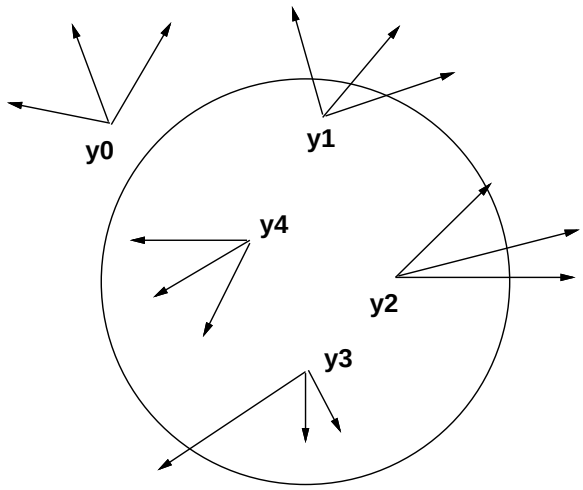


Figure 4. Continuous escape-time function

The diagram shown in Figure 4, illustrates the continuous escape-time for five points $y_0, \cdots, y_4$ with invertible transformations $IFS = \{T_1, T_2, T_3\}$. In case of a point y_0, $CT(y_0) = 0$, because the point satisfies $||y_0|| \geq R$, R is the radius of the circle. For a point y_1 we have $||y_1|| < R$ but all pre-images of y_1 have norm greater than R thus, $0 < CT(y_1) < 1$. The Point y_2 has two pre-images outside the disk and one on the boundary hence, $CT(y_2) = 1$. For points, y_3 and y_4 we have $1 < CT(y_3) < 2$ and $CT(y_4) > 2$.

4 Our Proposed Approach

This section introduces a new extension algorithm of the quadratic map basins for rendering the IFS escape-time behavior in three dimension without inverting the IFS transformations that is usually required and is computationally expensive. The original computation of the IFS escape-time in 2-D as given by definition 3.1 maps pixels into classified regions such that the proper transformation would be applied to any point based on the region containing the point. Its dynamics associates different colors to separate intervals of distance values based on the region containing the point. The question which may arises is how we can measure the dynamics of the IFS escape-time in 3-D. In this paper we use a concept called height field to satisfy such purpose. Therefore, instead of varying colors, we

vary the height of the points, thus representing the distance value as a height field. The concept of height field is motivated by a research on measure theory [26, 4, 13], but because the terminology of height field is convenient in the domain of computer graphics we replace the terminology of measures. The reason for choosing the measure theory is based on the fact that linear fractals are often attractors for the IFS and occur as the supports of probability measures associated with functional equations. The height field is given as follows: Based on measure theory concept [26, 13], we defined a canonical projection $\sigma : R^2 \times R \rightarrow R^2$, where, $\sigma(x, z) = x$. Thus, if $B \subset R^2 \times R$ be a height field such that $(x, z_1) \in B$ implies there exists no other point $(x, z_2) \in B$ with $z_1 \neq z_2$. The height field evaluates like a function $B : R^2 \rightarrow R$ defined as follows:

$$B(x) = \begin{cases} z & \text{if } (x, z) \in B \\ 0 & otherwise \end{cases} \tag{7}$$

Furthermore, define the maximum of two height fields B_1, B_2 as follows:

$$max(B_1, B_2) = \begin{cases} \{(x, z) : x \in \sigma(B_1) \bigcup \sigma(B_2), \\ \quad z = max(B_1(x), B_2(x))\} \end{cases} \tag{8}$$

The maximum operator given by equation (8) performs the the same way that a z-buffer [7] maximizes the z-component of geometry to produce correct visible surface classifications. In terms of measure theory, the set B(x) is the measure [26]. The height field definition given by equations (7) and (8) measures and display the dynamics of the escape-time behavior by storing prior escape times in the complement of the fractal attractor to avoid inverting the IFS transformation given in definition 3.1 which is required to compute its escape time classification, and their recomputation for later calculations. Unlike the definition 3.1 which maps pixels into classified regions in 2-D which is not sufficient for further extension. Based on equations 7 and 8 and by analogous to the block coding IFS techniques of Jacquin [14], our idea is to map classified regions into pixels. In order to satisfy this condition we have to map the disk D_R instead of the inverse-mapping of the given point x given by definition (3.1). Therefore, we'll introduce a parallel (equivalent) definition to the definition (3.1) called 3-D direct escape-time.

Definition 4.1 (3-D Direct Escape-time) *Given W, $\hat{A}$ and D_R as in definition 3.1. Then the direct escape-time can be given by $DT(x, I)$, where I is the unity transformation $I(x) = x \forall x$. Now $DT(x, T)$ operates on both a point $x \in R^2$ and a homogeneous 3×3 transformation matrix T and is defined as*

$$DT(x, T) = \begin{cases} 1 + MAXXX & \text{if } x \in T(D_R) \\ 0 & otherwise \end{cases}$$

where,

$$MAXXX = max_{i=1,\cdots,N} DT(x, T \circ T_i)$$

The notation "$T \circ T_i$" refers to a transformation composition and indicates that transformation T may be applied directly after transformation T_i. Definitions (3.1) and (4.1) are equivalent because of three categories. First, definition (4.1) shows that alternatively one can map the Disk D_R instead of inverse mapping the point x. Second, computing a point x in the image of a region $x \in T(D_R)$ is equivalent to computing the inverse image of a point x in the original region $T^{-1}(x) \in D_R$. Third, definition 4.1 doesn't use any IFS inversion, and directly worked by mapping classified regions into pixels which will be useful for further extensions with the height fields. Consequently, given an IFS ($W = \{T_1, \cdots, T_N\}$) with attractor $\hat{A}$, and D_R be a disk satisfies condition (4). Then define a new three dimensional IFS, $\grave{W} = \{\grave{T}_i\}_{i=1}^N$ constructed from T_i with

$$T_i = \begin{pmatrix} a_i & b_i & e_i \\ c_i & d_i & f_i \\ 0 & 0 & 1 \end{pmatrix}$$

by the following homogeneous transformation matrices:

$$\grave{T}_i = \begin{pmatrix} a_i & b_i & 0 & e_i \\ c_i & d_i & 0 & f_i \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$\grave{T}_i$ operates as T_i in the first two columns, but translates by one in the third. By this extension, the dynamics of the direct escape-time method can be measured with the height fields by the following function:

$DT : R^n \rightarrow R$
$DT(x) = max_{n=1,\cdots,\infty} B_n(x)$

Where B_n is the sequence of height fields defined as:

$$B_0 = \{(x, z) : x \in D_R, z = res(x)\},$$
$$B_n = max_{i=1,\cdots,N}\grave{T}_i(B_{n-1}) \tag{9}$$

This sequence of height fields B_n refines the IFS attractor as n increases. For example, B_0 contains the points whose continuous escape-time is in $[0, 1]$ and in general, B_n contains the points whose continuous escape falls between n and $n+1$ inclusive. The diagram shown in Figure 5 demonstrates an example for two height field levels with different iteration. The height field behaviors maximizes the N individual escape-time component in the same way that a z-buffer maximizes the z component of geometry to produce correct visible surface classification.

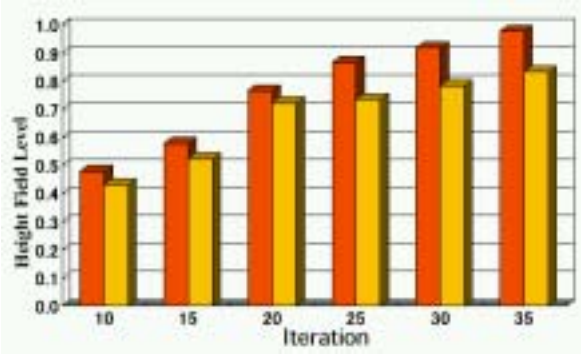


Figure 5. Height field level example illustration

5 Fractal Rendering Technique

One fascinating aspects of fractals generated by quadratic maps is the beauty of their graphical representations. This section describes the rendering technique of the proposed IFS escape-time extension algorithm of the quadratic map basins in three dimension. From its inception, the method of ray tracing has always been the technique of choice when ultimate realism of computer generated images[25, 20, 9]. Based on the principle of ray tracing, a similar use of it for rendering the IFS escape-time behavior in 3-D will be given based on a previously discussed height field. The basic operation in the ray tracing scheme is the computation of the first intersection of a given ray in a three dimensional scene with the set of all objects contained in the scene. Therefore, the similarity between the ray tracing operation and the suggested height field operation is that, the visual image produced by a height field is a function of the viewpoint, the nature of the light source, the albedo of the surface and other parameters [23, 27]. The function of the height field is to store prior escape times in the complement of the fractal attractor to avoid inverting the IFS transformation given in definition 3.1 which is required to compute its escape time classification and their re-computation for later calculations. In our case we render the IFS escape-time behaviour in 3-D by approximating its attractor say $\hat{A}$, with a set of small spheres covering such attractor. In addition, the use of hierarchical bounding volumes lends itself very naturally to the attractor of the IFS. This occur by evolving only a piece of the fractal surface at a time using a subdivision tree and enclosing each part of the surface with some bounding volume or extent E. The extent E of an object is the region of space occupied by the object. Such tree is constructed by recursively subdividing a previously discussed height field which is the escape trajectory of a given point up to some predefined maximum length say m. Since the dynamics of escape-time method is based on the principle of counting the number of iterations necessary to force a given point z outside a large disk of radius R. Then assign this number as a color to the given point z. Figure 6 shows such dynamics with different cases for iterating a given point z with magnitude $|z|$ in a complex plane C for the unit circle.

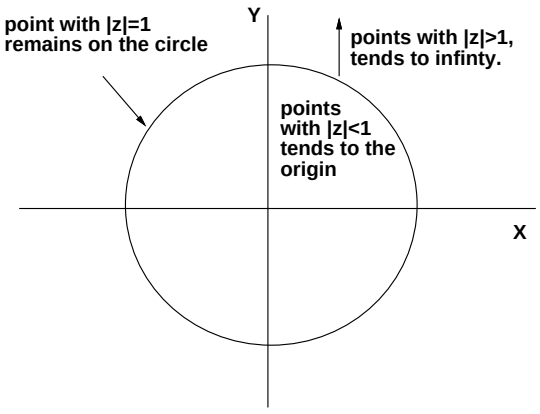


Figure 6. A unit circle with orbits classifications.

Based on such principle, the color of a given point is assigned to the norm of the maximum height field calculated from equation (8) according to the number of subdivisions tree at different levels that forces the point to escape outside a disk of radius R. In order to

differentiate between the escaping points and the non-escaping one, we associate to the non-escaping points some specific color. The process of our algorithm can be described as: follows:

Step1 (Initialization)
• Select some suitable attractor domain D for drawing.
• Set the transformation functions T_i
• Assign the value parameters "a-f" of each T_i.
Step2 (Loop)
• For all pixels (x,y) in the domain D do
Step3 (Iteration)
• Calculate the residual function of the given pixels of each T_i.
• Calculate the height field levels function of each T_i.
• Approximate the fractal attractor with the height field previously described.
Step4 (Fractal Rendering)
• Calculate the maximum value of height fields ($Equ.(8)$) of a given point (x,y).
• Render the approximated fractal attractor.
• Assign the appropiate color of a given point.

6 Experimental Results and Discussion

A critical review of previous algorithms for computing approximations to both quadratic and IFS fractal maps in 2-D has provided new insights into how more general and efficient algorithms may be designed specifically for fractal patterns. In this paper we proposed an extension algorithm for approximating the fractal attaractors of the IFS escape-time in 3-D. Our experimental results demonstrate some of the varied shapes that our proposed algorithm can model. Figure 7, presents the output of our algorithm on the Sierpinski triangle in which the height field B_0 is computed using the residual function, whereas B_1 is the maximum of $\grave{T}_1(B_0)$, $\grave{T}_2(B_0)$ and $\grave{T}_3(B_0)$. Figure 8, shows a 3-D animation of the sierpinski triangle. Figure $9(a)-(d)$, demonstrate a 3-D animation of some of Sierpinski hyperbolic platonic solids. Figure 10, illustrates an animation of the self-similar Menger Sponge fractal which is a three dimensional analogs of Sierpinski triangle with different levels. Since one fascinating aspects of fractals is the beauty of their graphical representations and provides a basis for modeling the infinite detail found in nature. Figure 11 shows one results satisfied such aspects. The resulting image resembles tree and it is a three dimensional analogs of Sierpinski triangle with about 131072 equivelant primitives and is renderd in aproximately 51 : 14 seconds. Table (1) shows the rendering running time (execution time) for the experimental results given in this paper, where L denotes the maximum iteration level.

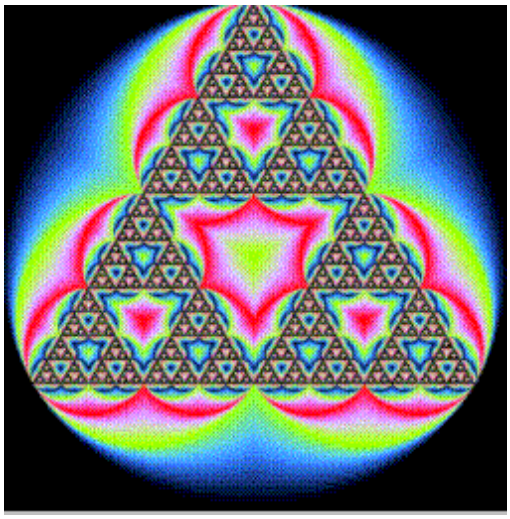


Figure 7. Sierpinski Triangle with height fields

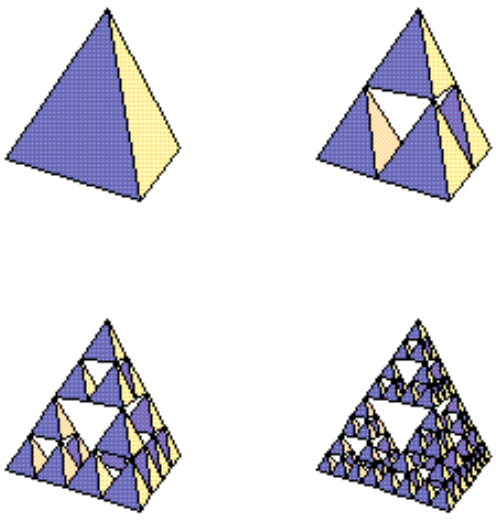
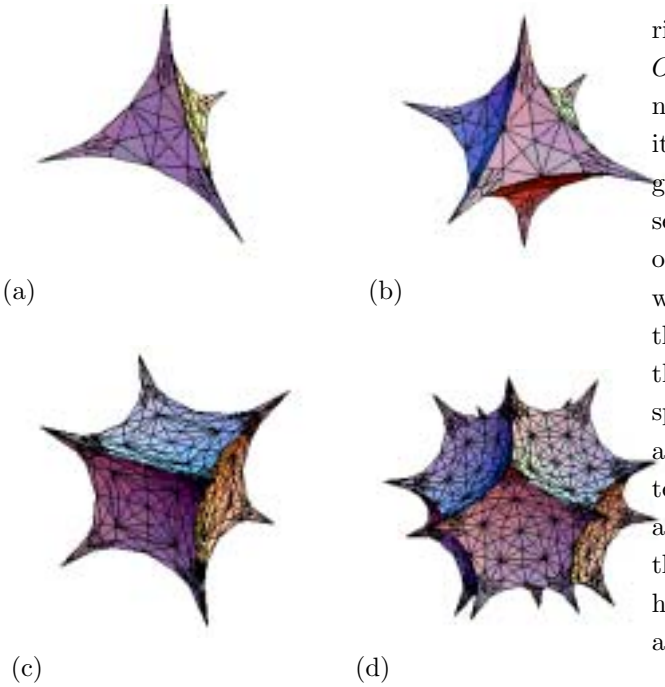


Figure 8. 3-D Sierpinski Triangle Animation



(a) (b)

(c) (d)

Figure 9. 3-D Sierpinski hyperbolic animation

(a) Level =1 (b) Level =2 (c) Level =3

Figure 10. Menger sponge animation.

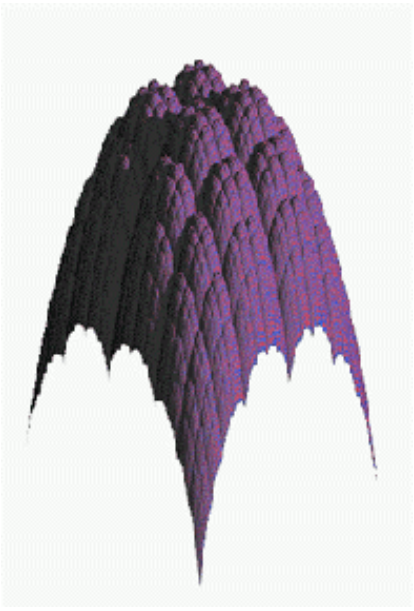


Figure 11. Tree fractal pattern.

In terms of the complexity time, our proposed algorithm has $O(log(n))$ execution time compared with $O(n)$ execution time for other proposed grid techniques [10, 24], where n is the maximum number of iteration level. The proof of such complexity time is given by analyzing the algorithm pseudcode given in section 5. In such case the time complexity is based on the starting selected point in the Domain D, the way of applying to this point the tree of sequences of the transformation T_i and the order of constructing the tree of transformations. In our case such order is speciefied as a height-field depth-first order in which all transformation sequences with iteration "n" is extended to its maximum depth before other sequences are performed. Figure 12 illustrates such tree, where the numbers $\{1, 2, \cdots, 12\}$ refer to the order of the height field computation of points in the tree of images.

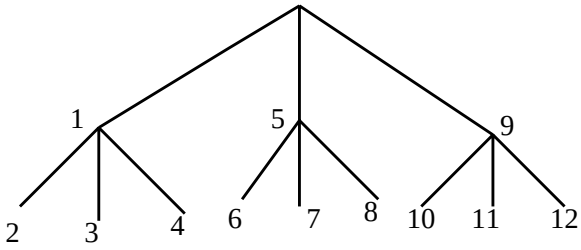


Figure 12. Depth-first tree order illustration.

Consequently, in the algorithm pseudcode specified by "Step2(Loop)" all the internal nodes of a subdivision tree are at levels less than "k" for some k. Then it is obvious that there can be at most 2^k external nodes in the tree. Hence,

$$n \leq 2^k \tag{10}$$

By taking the logarithum "log" to both sides of equation (10), we get

$$l \geq log(n) \tag{11}$$

In equation (11), the parameter $l = k\log(2)$ is constant, therefore complete the proof.

Table (2), illustrates the complexity rendering time comparison between our algorithm and other proposed

techniques [10, 24, 22] for some fractal patterns. From the given results one can see that our algorithm gives satisfied 3-D fractal shape rendering results in a reasonable time. Therefore, makes the rendering process of the 3-D fractals reliable for interactive machine. We list in tables 3, 4, and 5, the IFS code values of the parameters "a-f" for the transformation T_i given in equation 1 and shown in the experimental results.

Table (1), Rendering Execution time Illustrations

Shape	L	RET
Menger Sponge	1	01.117 Sec.
Menger Sponge	2	10.233 Sec.
Menger Sponge	3	26.654 Sec.
Sierpinski Triangle	6	42.467 Sec.
Sierpinski Tetrahedron	3	53.2133 Sec.
Sierpinski Hyperbolic Octahedron	3	59.721 Sec.
Sierpinski Hyperbolic Cube	3	63.874 Sec.
Sierpinski Hyperbolic Dodecahedron	3	66.132 Sec.
Tree	7	51.14 Sec.

Table (2), fractal rendering complexity time.

Shape	n	Grid	Our algorithm
Menger Sponge	3	48.27	28.19
Sierpinski triangle	4	51.39	30.51
Sierpinski Tetrahedron	3	53.84	34.56

Table (3). Sierpinski triangle IFS codes.

T	a	b	c	d	e	f
1	0.500	0.0	0.0	0.500	0.0	0.0
2	0.500	0.0	0.0	0.500	0.500	0.0
3	0.500	0.0	0.0	0.500	0.250	0.433

Table (4). Menger Sponge IFS codes.

T	a	b	c	d	e	f
1	0.0	0.577	-0.577	0.0	0.095	0.589
2	0.0	0.577	-0.577	0.0	0.441	0.789
3	0.0	0.577	-0.577	0.0	0.095	0.989

Table (5). Sierpinski Tetrahedron IFS codes

T	a	b	c	d	e	f
1	0.0	-0.50	0.50	0.0	0.50	0.0
2	0.0	0.50	-0.50	0.0	0.50	0.50
3	0.50	0.0	0.0	0.50	0.25	0.50

7 Conclusion and Future Work

Many interesting fractal patterns using quadratic maps and IFS are naturally visualized by plots on the complex plane, while a 3-D extension algorithm are still aspiring goals and challenging tasks. This paper propose an extension algorithm for rendering the IFS escape-time behavior in three dimension without inverting the IFS transformations which is computationally expensive, therefore reducing the complexity time. The proposed algorithm has has fast computations and simple representation and due to its speed of computation makes rendering rendering 3-D fractal patterns reliable for interactive machines. Many interesting problems regarding the quadratic map basins for fractal remain open. In our future work we would like to extend the proposed method to handle cases where it doesn't contain the infinity circle. In such case it may be possible to gain further understanding about the dynamics associated with fractal patterns and would greatly increase the efficiency of computation and discovering more realistic general fractal shapes.

References

[1] Alastair N., "IFS and interactive image synthesis", *Computer Graphics Forum,* Vol. 9, pp. 127-137, 1990.

[2] Barnsely, M. F., "Fractals everywhere", Academic Press, 1993.

[3] Barnsely, M.F., "Recurrent iterated function systems", Constructive Approximation Vol. 1, No. 1, 1989.

[4] Barnsely, M.F., and Demko S.G., "Iterated function systems and the global construction of Fractals", Proceedings of the Royal Society of London Series A 399, pp. 243-275,1985.

[5] Canright,D., "Estimating the spatial extent of attractors of iterated function systems", Computers and Graphics Vol.18, No.2, pp. 231-238,1989.

[6] Demko S., L. Hodges, and B. Naylor, " Construction of Fractal Objects with Iterated Function Systems", Computer Graphics, Vol. 19, No. 3, pp 271-278, SIGGRAPH'85 Proceedings.

[7] Foly D., Van Dam, Feiner S., and Hughes J., "Computer Graphics : Principles and Practice" Addison-Wesley, 1990.

[8] Hart, J. C. “The object instancing paradigm for linear fractal modeling”, Graphics interface ’92 proceedings, pp. 224-231,1992.

[9] Hart, J. C., and Sandin, “Ray tracing deterministic 3-D fractals”, Computer Graphics, Vol.23, No.3, pp. 289-297, 1989.

[10] Hepting, D., P. Prusinkiewicz and D. Saupe, “Rendering methods for iterated function systems”, Fractal in the fundamental and applied sciences, New York, 1991.

[11] Hussein Karam, and Masayuki Nakajima, ”Towards Realistic Modeling and Rendering of 3-D Escape-Time Deterministic Fractal Shape”, 7^{th} IEEE International Conference on Virtual Systems and Multimedia, VSMM’01, U.S.A., pp. 565-574, 2001.

[12] Hussein Karam and M. Nakajima “A novel 3-D Visualization algorithm for rendering fractal patterns”, The international Workshop on Advanced Image technology, IWAIT’2002, Taiwan, pp. 49-56, 2002.

[13] Hutchinson, J.E., “Fractals and self-similarity”, Indiana Unversity Journal of Math. Vol. 30, No. 5, pp. 713-747, 1981.

[14] Jacquin, A.E., “Image Coding based on a fractal theory of iterative contractive image transformations”, IEEE Transformations on image processing Vol. 1, No.1, pp. 18-30, 1992.

[15] Mandelbrot B.B., “The fractal geometry of nature”, W. H. Freeman, New York, 1982.

[16] Prusinkiewicz, P. and Hammel M., “Escape-time visualization for language-restricted iterated function systems”, Graphics interface ’92 proceedings, pp. 224-231,1992.

[17] Prusinkiewicz P., and Hanan J., L-systems : from formalism to programming languages. *In Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics and Developmental Biology. Eds. G. Rozenberg and A. Salomaa. Springer-Verlag, Berlin, pp. 193-212, 1992.*

[18] Prusinkiewicz, P. and Sandness G. “Koch curves as attractors and repellers”, IEEE Computer Graphics and applications Vol. 8, No.6,pp.26-40,1988.

[19] Sprott J.C. “Automatic generation of general quadratic map Basins”, Computer and Graphics, Vol.19, No.2, pp 309-313,1995.

[20] Yagel, R, Cohen, D. and A. Kaufman, “Discrete ray tracing”, IEEE Computer Graphics and Applications, Vol. 12, No. 5, pp. 19-28, 1992.

[21] G. Uday, and C. Virendra, “Fractals from $z \leftarrow z^{\alpha} + c$ in the complex plane”, Computer and Graphics, Vol. 15, No. 3, pp. 441-449, 1991.

[22] K. Albert, “An abstract Mandelbrot algorithm for $Z^n + c$”, Fractals, Vol. 6, No. 1, pp.1-10, 1998.

[23] P. Lindstrom, D. Koller, W. Ribarsky, F. Hodges and N. Faust , “Real-Time, Continuous Level of Detail Rendering of Height Fields” , Proc. of SIGGRAPH’96, pp. 109-118, 1996.

[24] M. Monro, and F. Dudbridge “Rendering Algorithms for Deterministic Fractals”, IEEE Computer Graphics and Applications, January 1995.

[25] L. Kay, and T. Kajiya “Ray tracing complex scenes”, Proc. of SIGGRAPH’86, Vol. 20, No. 4, pp. 269-278, 1996.

[26] L. Joseph, Measure Theory, New York: Springer-Verlag, 1994.

[27] Q. Zheng and R. Chellappa, “Estimation of illumination direction, albedo, and shape from shading”, IEEE Transaction PAMI, 13(7), pp. 680-702, 1991.