VOL. 13, NO. 4, FEBRUARY 2018 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

DEVELOPING FRACTALS USING ITERATED FUNCTION SYSTEMS

Bulusu Rama[1] and Jibitesh Mishra[2]
[1]Department of Computer Science and Engineering, M L R Institute of Technology, Hyderabad, India
[2]Department of Computer Science and Applications, College of Engineering and Technology, BPUT, Bhubaneswar, India
Email: bulusurama1967@gmail.com

ABSTRACT

The geometric modeling of fractal objects is a difficult process. An important class of complex objects in nature is trees, plants, clouds, mountains, etc. These objects cannot be satisfactorily described or acceptable in quality or quantity using conventional geometry. Diverse techniques are at present being investigated for modeling these complex objects. The development of a new approach to computing fractals is being taken up by us, known as Iterated Function Systems. Any set of linear maps (affine transformations) and an associated set of probabilities determines an Iterated Function System (IFS). IFS description forms, through a set of simple geometric transformations, a basic set of tools for interactive image construction. This paper presents the role of IFS in geometric modeling of fractal objects.

Keywords: fractals, IFS, mapping transform, iteration.

INTRODUCTION

The word fractal was coined by Benoit Mandelbrot in 1975 and was derived from the Latin word "fractus" which means "broken" or "fractional" [12]. Fractals generated dynamically are interestingly-styled algebraic fractals, e.g. Mandelbrot Set & Julia Set. In the arena of computer graphics, researchers at all times strive to try everywhere to construct geometric model of objects. Adding gadgetry, graphics provides distinct ways to construct man-made objects e.g. buildings, plants etc. Fully widely-aged arithmetical polynomials are reachable to model such type of objects. These well push off precise polynomials fundamentally generate suave geometry. Because of non-smooth and highly irregular nature of fractals, common polynomial methods require more specification information. As mentioned above, the initiation of fractal was claimed by IBM mathematician Benoit Mandelbrot. He came to the conclusion that traditional geometry was not enough to define the ordering of unpretentious objects which are extravagant such as mountains, clouds, coastlines and trees The non-Euclidean geometry or fractal geometry deals on every side irregular and fragmented patterns.

The general things to create a fractal are: a set of transformations (IFS), a base from which iteration starts, and a condensation set (possibly the empty set). IFS provide a very concise representation, productive and effective computation, and a lesser amount of user specifications. The entire information for generation of an IFS structure, known as an "attractor", is available in the definition of a few transformation functions, typically affine transformations described by simple linear equations. These transform 2D or 3D objects by a composition of translation, scaling and rotation, retaining parallel lines. A set of contraction mappings, basically a set of probabilities acting on a space X constitutes an IFS. The use of either of the two standard methods to generate an attractor is uncomplicated and easy to do. On the other hand, the inverse problem to identify the IFS for a particular attractor is somewhat cumbersome. Fractal image construction using IFS begins with original image with some successive transformations applied over the image. The result of IFS is an attractor which is a fix point. An image is a contraction mapping of this fix point. The mapping of the corresponding fractal onto itself done by an IFS is a collection of smaller self-similar copies. The paper introduces and reviews the uses of Iterated Function Systems, for image/object generation.

MATHEMATICAL BASIS OF IFS

A hyperbolic IFS is made up of a complete space (X, d) together with a finite set of contraction mappings w_n : X->X, with respective contractility factors s_n, for n = 1,2, ..., N. This IFS can be expressed by the notation as {X ; w_n, n = 1, 2,…, N}and its contractility factor as s = max{s_n : n = 1, 2,…., N}.

An iterated function system with probabilities [8] comprises of IFS {X; *w*1, *w*2,…., *w*N} along with an ordered set of numbers {*p*1, *p*2,…,*p*N}, such that *p*1 + *p*2 + *p*3+ …. + *p*N = 1 and *pi* > 0 for *i* = 1, 2,…., *N*. The probability pi is associated with the transformation wi. The nomenclature IFS with probabilities may be used as an abbreviation. The full notation for such an IFS is given by[2].{X; *w*1, *w*2,…., *w*N ; *p*1, *p*2,…., *p*N}.The probabilities are in relation with the measure theory of IFS attractors, and are used in the computation of images of the attractor of an IFS attractor using the random iteration algorithm, and also in the use of the multiple reduction photocopy machine algorithm, as applied to grayscale images [2]. They do not have a role with the multiple reduction photocopy machine algorithm when it is applied to binary images [2].

The fixed point $A \in H(X)$ described in the IFS Theorem is called the attractor of the IFS. Let (X, *d*) be a metric space and let $C \in H(X)$. Define a transformation *w*0: H(X) -> H(X) by *w*0 (*B*) = *C* for all $B \in H(X)$. Then the condensation transformation is given by w0 and the associated condensation set is given by C.

Let a hyperbolic IFS be defined by {X; *w*1, *w*2,…., *w*N} with contractivity factor $0 \leq s < 1$ and a condensation transformation be given by *w*0: H(X) -

>H(X). Then {X; *w*0, *w*1,…., *w*N} is called a (hyperbolic) IFS with condensation.

STEPS FOR CREATING FRACTALS USING IFS

Generation of a fractal using iterated function [13] system involves the following steps:

a) Establish a set of transformations.
b) Draw any initial pattern on the plan.
c) Apply transformations on the initial patterns which are defined in the first level.
d) Again apply transformation on the new image which is the combination of initial pattern and pattern after applying transformations.
e) Repeat step 4 again and again, this step 4 can be repeated infinite number of times.

IMPLEMENTATION OF IFS

The deterministic algorithm

At first start with an image and apply some affine transformation on each subset of this image. Then find out next image which should be complete subset of R^2 space where image lie. After applying affine transformation iteratively, a sequence of image will be generated which should converge at some point. This point will be the limit point. This limit point is nothing but an image. Then apply some contractive mapping to get an image from other image. This process of generating fractal requires large amount of memory, since each iteration generates some image and to store the generated image by affine transformation needs large amount of memory.

Let {X; *w*1, *w*2,…., *w*N} be a hyperbolic IFS. Choose a compact set A_0 C R^2. Then compute successively $An = W \circ n(A)$ according to

$An+1 = Un \in N\ wj\ (An)$ for $n = 1, 2,\ldots$.

Thus a sequence {*An*: *n* = 0, 1, 2, 3,….} C H(X) is constructed.

Then according to the IFS theorem the sequence {*An*} converges to the attractor of the IFS in the Hausdorff metric.

The random iteration algorithm

The random approach is different from the deterministic approach in that, in this approach, the initial set is a singleton point. Here, one of the defining affine transformations is used to calculate the next level, at each level of iteration, which will also be a singleton point. This differentiates the random approach from the deterministic approach. The affine transformation is randomly selected and applied at each level. Points are plotted, except for the early ones, and are discarded after being used to calculate the next value. The random algorithm has the advantage of avoiding large computer memory, and is well suitable for the small computers on which one point at a time can be calculated and display on a screen. The drawback of it is that it takes thousands of dots to produce an image in this way that does not appear too sketchy.

Let {X; *w*1, *w*2,…., *w*N ; *p*1, *p*2,…., *p*N} be an IFS with probabilities. Choose $x_0 \in X$ and then choose recursively, independently [2], $x_n \in \{w1(x_n\text{-}1), w2(x_n\text{-}1),\ldots., wN(x_n\text{-}1)\}$ for $n = 1, 2, 3,\ldots.$, where the probability of the event $x_n = wi(x_n\text{-}1)$ is *pi*. Thus, construct a sequence $\{x_n: n = 0, 1, 2, 3,\ldots.\}$ C X

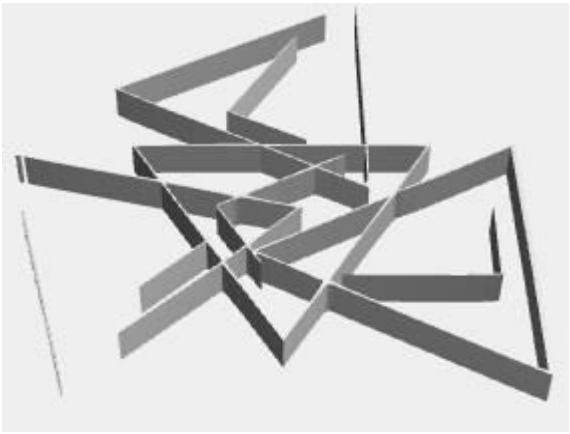

Figure-1. Design of IFS under deterministic fractal generation algorithm.

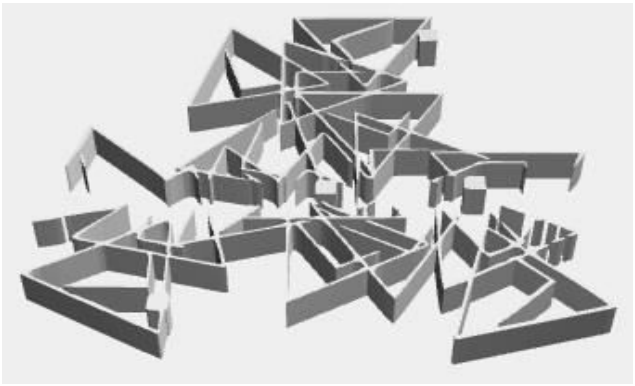

Figure-2. Fractal generated after 2 iterations with four fix points.

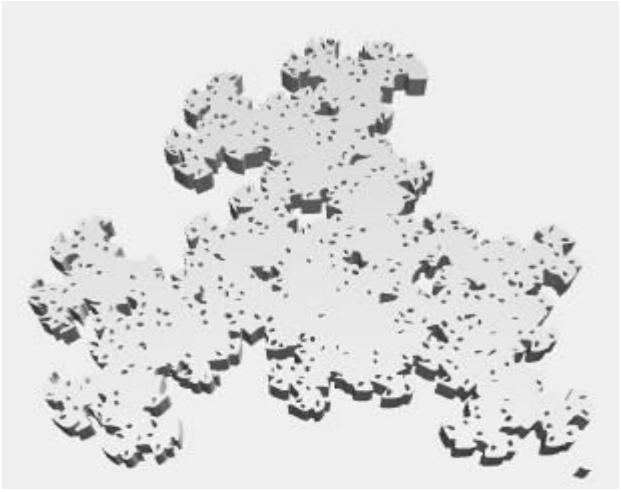

Figure-3. Fractal generated after 5 iterations with four fix points.

(Fractal image complexity depends upon the number of iterations).

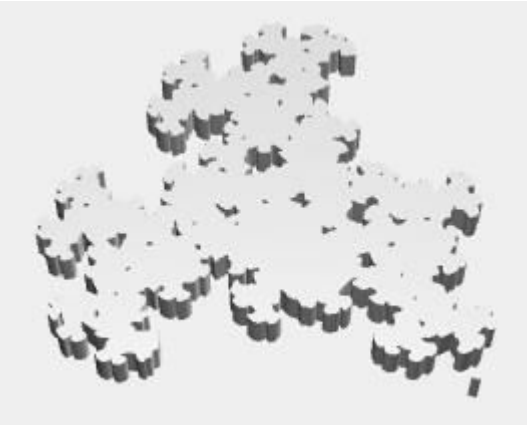

Figure-4. Attractor of IFS after applying 18 iterations.

Figure-5. Fractal image after applying random fractal generation algorithm with 20 iterations.

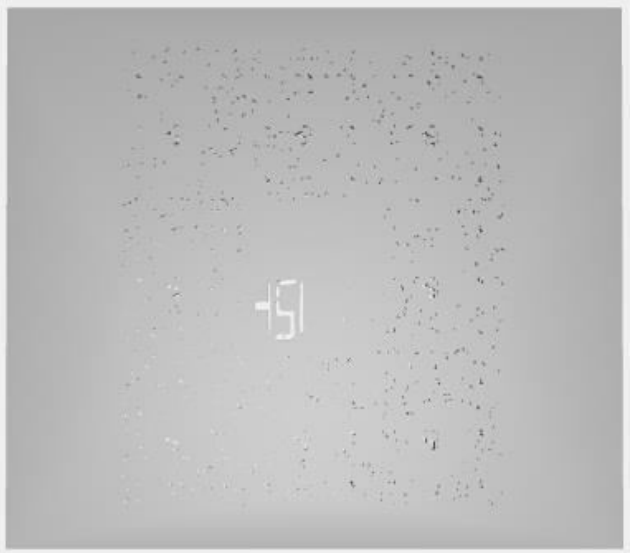

Figure-6. Fractal image after applying random fractal generation algorithm with 50 iterations.

QUANTIFIABLE INFORMATION ABOUT FRACTAL IMAGES

The pictures obtained after applying the random iteration algorithm are shown in Figure-1. The time taken and number of iterations to generate each picture differs from one another. The amount of contraction or expansion of the original image is given by the scale factor of each image. The points x ∈ X are multiplied by the affine transformation Wi, after which the scaling is applied. Deterministic algorithm takes less iteration but more time to generate picture whereas the random iteration algorithm requires lots of iteration to generate same picture.

Figures 7(a) to 7(d) show the fractal images [22] after the application of certain number of iterations on each image shown: (a) IFSX fractal (b) Devoured fractal (c) Face 2 Face fractal (d) Triangular Terifoil fractal.

Figure-7a. Fractal image after applying some number of iterations for the IFSX fractal.

Figure-7b. Fractal image after applying some number of iterations for the devoured fractal.

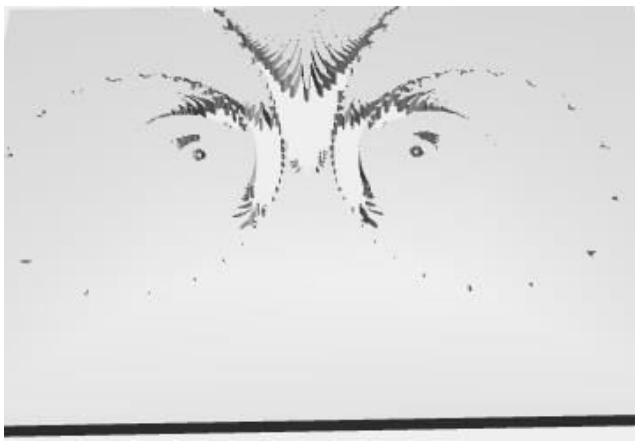

Figure-7c. Fractal image after applying some number of iterations for face 2 face fractal.

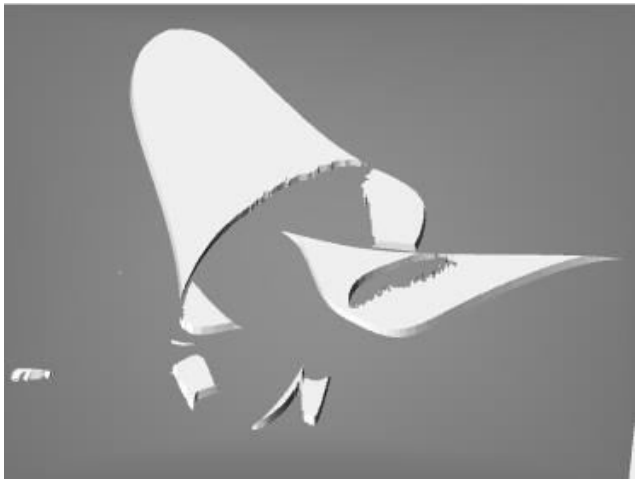

Figure-7d. Fractal image after applying some number of iterations for triangular terifoil fractal.

The following quantitative information [22] as depicted in Table-1 below has been obtained about different types of fractal images shown in Figures 7(a) to 7(d).

Table-1. Quantitative information about the different types of fractals in figures 7(a) to 7(d).

Fractal	Number of Iterations	Resolution	Tine taken to generate the picture in secs
IFSX	205	640x480	25.132
DEVOURED	140	640x480	37.465
FACE2FACE	580	640x480	0.821
TRIANGULAR TERIFOIL	55	640x480	11.929

AFFINE TRANSFORMATION

A set of 2- dimensional affine map serves as a simplest example of an IFS. 2D affine mapping comprises of a single point. This single point is called fixed point. Applying IFS to the fix point may give rise to either stable (attractor) or unstable (repelling). Unstable fix point results in an unbounded system because of the larger and larger growth of this fix point on successive application of IFS on this point. Affine transformations are linear transformations. A combination of rotation, translations, dilations and shears produces an affine transformation. Affine transformation has the property that it does not preserve angles or length. Two or more successive transformations can be applied on the image using an affine transformation.

A transformation w: **R**2 -> **R**2 of the form

$$w(x,y) = (ax + by + e, cx + dy + f) \quad (1)$$

where a, b, c, d, e, and f are real numbers, is called a (two-dimensional) affine transformation.
This can be equivalently expressed as:

$$T\begin{pmatrix}x\\y\end{pmatrix} = s\begin{pmatrix}cos\theta & -sin\theta\\ sin\theta & cos\theta\end{pmatrix} + \begin{pmatrix}e\\f\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} \quad (2)$$

In an IFS consisting of affine contractions {S1, S2, S3 … Sn,} on $\mathbb{R}^n$, the attractor F is called a self-affine set. A contractive transformation in fractal geometry is that which moves pair of two points closer together. A formal definition can be: a transform W is said to be contractive if for any two points P1 and P2, the distance

$$d(W(P1), W(P(2)) < Sd(P1,P2) \quad (3)$$

where s ∈ (0, 1), and is called the contractive factor.

The property that when contractive affine transforms are repeatedly applied holds that they converge to a point which remains fixed upon further iterations.

Figure-8. Fractal Image after applying some number of iterations for Triangular Terifoil Fractal.

FRACTAL GENERATION BY USING DETERMINISTIC ALGORITHM

We get the output an attractor which is a fractal image after successively applying IFS over the initial image. The final image after the application of IFS is called self-similar set because it is a union of a number of smaller self-similar copies of itself.

$$T_1([\begin{matrix}x1\\x2\end{matrix}]) = [\begin{matrix}1/2 & 0\\0 & 1/2\end{matrix}][\begin{matrix}x1\\x2\end{matrix}][\begin{matrix}0\\0\end{matrix}] \quad (4)$$

$$T_1([\begin{matrix}x1\\x2\end{matrix}]) = [\begin{matrix}1/2 & 0\\0 & 1/2\end{matrix}][\begin{matrix}x1\\x2\end{matrix}][\begin{matrix}1/2\\0\end{matrix}] \quad (5)$$

$$T_1([\begin{matrix}x1\\x2\end{matrix}]) = [\begin{matrix}1/2 & 0\\0 & 1/2\end{matrix}][\begin{matrix}x1\\x2\end{matrix}][\begin{matrix}1/4\\\sqrt{3}/4\end{matrix}] \quad (6)$$

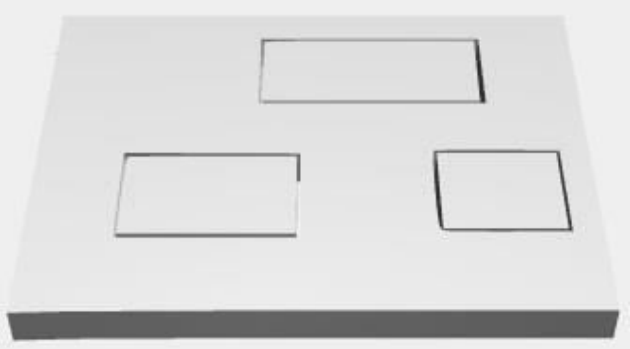

Figure-9. Collage for constructing triangle.

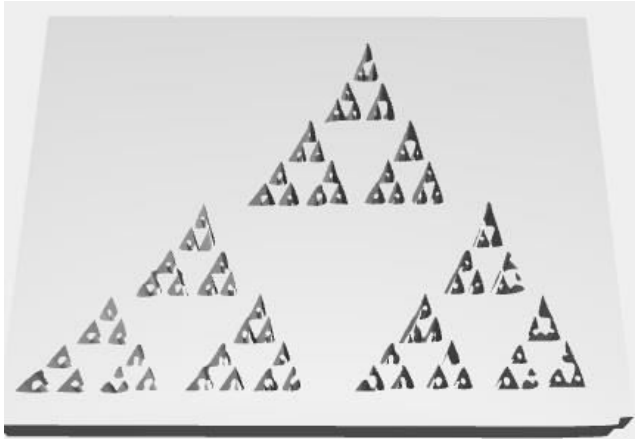

Figure-10. Image of fractal after applying successive IFS.

THE CHAOS GAME ALGORITHM USING IFS

Consider an IFS containing M affine transformation T1, T2, T3,…., TM, and each affine transformation in IFS associated with probabilities p1, p2, ….., pN, respectively. Apply the chaos game algorithm to the affine transformation with probability. Then select one affine transform in the IFS according to its probability and apply it to initial point (x0, y0) to get a new point (x1, y1). Again apply another transform with another probability. We get a new point (x2, y2) generated. Repeat this process to result in a long sequence of points:

(x0, y0) , (x1, y1) , (x2, y2) , (x3, y3) , . . .

By this we state an essential result of the IFS theory that this sequence of points will converge, with 100% probability, to the attractor of the IFS [7].

GENERATION OF ALGEBRAIC FRACTALS USING IFS

Construction of mandelbrot set and julia set

Dynamically constructed fractal systems are of algebraic in nature. Two famous fractals, the Mandelbrot Set and the Julia Set are popular examples of algebraic fractals (fractals of the complex plane) [2]. These fractals are generated by the iterated function $f(z) = z_n^2 + c$, where z and c are complex numbers [6]. The primary difference between the Julia set and the Mandelbrot set is the manner in which the function is iterated. The Mandelbrot set iterates as per $z = z_n^2 + c$ with z always starting at 0 and varying the c value. The Julia set iterates as per $z = z_n^2 + c$, where c is constant and z is variable. In other words, the Mandelbrot set is in the parameter space or the c-plane, while the Julia set is in the dynamical space, i.e., the z-plane. The method of generation of the above two fractals [6] is given as in the following steps:

a) For a given maximum number of iterations denoted by the constant maxit, plot the Set defined by the (min, max) range of values of each point the axes (Xmin, Xmax), (Ymin, Ymax).
b) This is done by first generating a grid based on the

line spacing values for each Xz and Yz. The line spacing is calculated using the distance of each such point based on (Xmin, Xmax) & (Ymin, Ymax), set to lower and upper limits from the origin of z, whose initial value is always 0. For each co-ordinate axis involved, we get a line spacing value, e.g., Xlinespacing, Ylinespacing etc.

c) Then the actual Mandelbrot Set/Julia set is plotted by selecting all points in the grid obtained from the step above, and checking if it qualifies as a candidate for being part of a Mandelbrot Set/Julia Set. If the difference between these line spacing values is 0, then the corresponding point belongs to the Mandelbrot Set/Julia Set being generated. And the union of all such points gives the domain of the resulting Mandelbrot Set/Julia Set.

d) The next step is to plot the actual Mandelbrot Set/ Julia Set based on the set of points obtained from step 3 above. This is done by iterating the above steps based on the max number of iterations (this is now an input variable) and the (Xmin, Xmax), (Ymin, Ymax) ranges, till all the iterations are complete. However, a super limit for the maximum number of iterations being input is fixed at value greater than maxit, to ensure that the generated Mandelbrot Set/Julia Set doesn't blow up into an infinite space on the screen.

e) The program runs as a Windows console application with the 3D images of the Mandelbrot Set/Julia Set being rendered as a colour-mapped image projected onto the 3D plane, using MATHLAB 3D Image Rendering software.

The following Figures 11-14 show the results obtained by taking as input the 2D images of the Mandelbrot and Julia sets and the outputs of the same based on 100 iterations rendered in 3D.

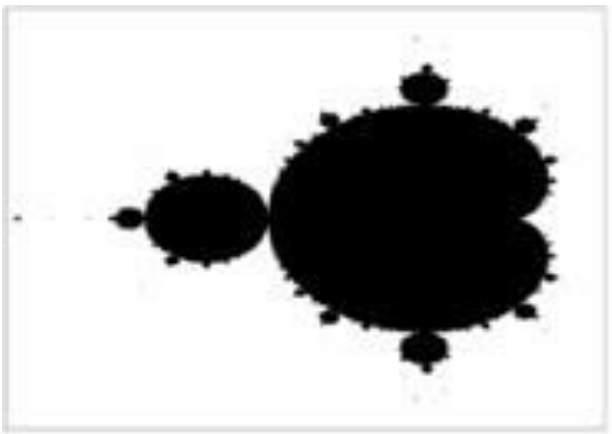

Figure-11. 2D Mandelbrot set.

Figure-12. 3D Mandelbrot set generated using 100 iterations.

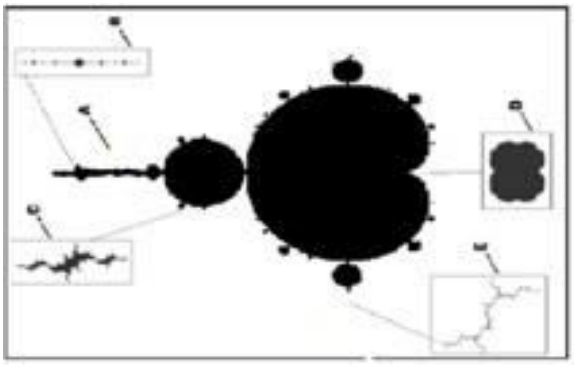

Figure-13. 2D Julia set.

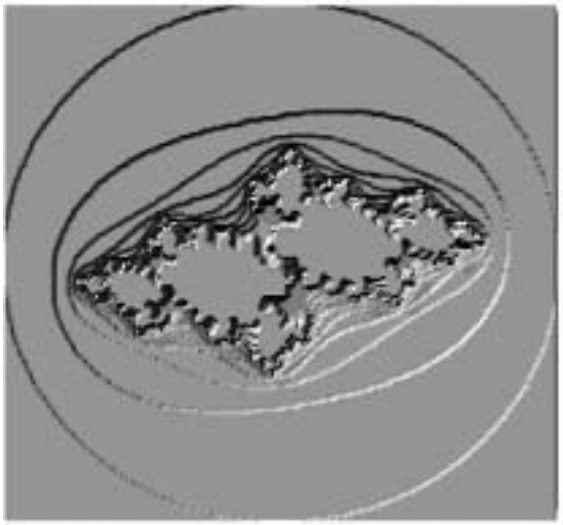

Figure-14. 3D Julia set generated using 100 iterations.

3D OBJECT CONSTRUCTION USING 3D IFS

The IFS parameters to generate a fractal image are already specified by the formula itself except a few parameters which still remain to be mentioned. The other parameters are [15]:

- Iteration counter - It specifies the number of points that will be iterated.

- Color mode - It specifies the color to paint the calculated point using IFS.

- ***Transformation*:** Gives color to each point in accordance with the transformation, which leads to it.
- ***Probability*:** Gives color to each point in accordance with the probability, i.e. according to the previous random number.
- ***Measure*:** Gives color to each pixel in the window in accordance with how often a point hits the pixel.

The affine matrix required for 3D fractal image is as follows [15].

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The generation of 2D fractal in 3D environment involves simply putting j=k=l=m=c=g=0 and taking remaining values a, b, d, e, f according to the 2D affine transformation.

Figure-15. Construction of 3D fractal using IFS.

CONCLUSIONS

IFS can be used to generate fractal images. For this any initial set of points is taken and successive iterated function system applied over the initial set of points a finite number of times. As a result, an attracter is generated. This paper briefs about fundamentals of Iterated function system and its application in geometric modeling of complex objects. In this paper quantitative information like number of iterations and time taken to generate the fractal image obtained by applying IFS on fractals like IFSX, DEVOURED, FACE2FACE and TRIANGULAR TERIFOIL is shown. The time taken as well as quality of fractal image relies on the number of iterations under IFS. Two algorithms - the deterministic algorithm and the random algorithm are at hand to generate fractal images. Fractals generated by deterministic algorithm take less time when compared with random generation algorithm. A fixed point after applying IFS is an attractor which can be stable or unstable. Unstable fix point results in an unbounded system, the reason being the unstable fix point growing larger and larger on successive IFS application on this point. The generation of 3D shapes requires the definition of specific number of iterations, transformations (translation, rotation, reflection, scaling) of additive shapes over the basic model. The number of iterations is a factor upon which the 2D and 3D object complexity is dependent.

REFERENCES

[1] Construction of fractal objects with iterated function systems, San Francisco July 22-26.

[2] Sewell MV. 1994. Fractal Image Compression, Birkbeck College, University of London.

[3] Bernsley MF and Hurd LP. 1993. Fractal Image Compression, AK peters Ltd. Massachusetts.

[4] Literature Number BPRA065: October 1997. An Introduction to Fractal Image Compression. Texas Instruments, Europe.

[5] Literature Number BPRA065: October 1997. An Introduction to Fractal Image Compression. Texas Instruments, Europe.

[6] Rama B. and Mishra J. doi>10.1145/1947940.1947990. 2011. Generation of 3D fractal images for Mandelbrot Set. ACM International Conference Proceedings Series, Proceedings of the 2011 International Conference on Communication, Computing & Security. 235-238.

[7] Stewart. 1988. A review of the science of fractal images. Nature. 336-289.

[8] Guojun L. 2001. Fractal-Based Image and Video Compression. The Transform and Data Compression Handbook, CRC Press LLC. Boca Raton.

[9] Norton A. 1982. Generation and display of geometric fractals in 3D. Computer Graphics. 16(1): 61-67.

[10] Kriska JB. Construction of Fractal Objects with Iterated Function Systems. MATH 547.001.

[11] Literature Number: BPRA065: October 1997. An Introduction to Fractal Image Compression. Texas Instruments, Europe.

[12] Mandelbrot BB. 1982. The Fractal Geometry of Nature. W. H. Freeman and Company.

[13] Groeller E. and Wegenkiltl H. 1996. Interactive design of non-linear functions for iterated function systems. Proc. 4th Intl. Conf. Comp. Graph. and Visual. WSCG'96, Plzen: 93-102.

[14] Barnsley MF, Elton JH, and Hardin DP. 1989. Recurrent iterated function systems. Constructive Approximation. 5: 3-31. http://dspace.thapar.edu:8080/dspace /bitstream/123456789/426/1/m91536.pdf.

[15] Yokoya N, Yamamoto K, and Funakubo N. 1989. Fractal-based analysis and interpolation of 3D natural surface shapes and their application to terrain modeling. Comp. Vis. Graph. Image Proc. 46: 289-302.

[16] Hart JC. and De Fanti TA. 1991. Efficient anti-aliased rendering of 3-D linear fractals.

[17] Proc. SIGGRAPH'91. Computer Graphics. 25(4): 91-100.

[18] Wyvill GB and McPheeters C. 1987. Solid texturing of soft objects. IEEE Comp. Graph. and Appl. 7(12): 20-26.

[19] Krzysztof G. May 2008. Fractals. Tech. Rep.

[20] Peitgen HO. *et al*. 1988. The Science of Fractal Images. Springer-Verlag, New York, Berlin.

[21] Tan L. 1990. Similarity between the Mandelbrot Set and Julia Sets. Communications in Math. Phys. 134: 587-617.

[22] Garg. A, Negi. A, Agrawal. A, Latwal. B. 2014. Geometric Modeling Of Complex Objects Using Iterated Function Systems. Intl. Jour. of Scientific & Technology Research. 3(1): 1-8