# THE RANDOM ITERATION ALGORITHM

Daniela Alexandra CRIŞAN, PhD
Justina Lavinia STĂNICĂ, PhD Candidate

Romanian – American University, Bucharest, Romania
dacrisan@yahoo.com, lavinia.stanica@gmail.com

**ABSTRACT**
In the last decades, many researchers concerned their attention on fractals properties of objects. Fractals can be use to describe natural shapes so their applications are various in many fields such as informatics, economics, engineering, medical studies. In this paper we present a way to describe fractal, using the Iterated Function System (IFS). We present the random iterated algorithm implemented in the C++ programming language used to generate selfsimilar fractals.
**Keywords:** fractal, IFS, random iteration algorithm

## 1. INTRODUCTION

Around the year 1970, the French mathematician Benoit Mandelbrot introduced the concept of Fractals in order to describe some dynamic systems. Although Mandelbrot named these features as fractals, other mathematicians have studied those forms years before: Cantor, Sierpinski and Koch were attracted by the strange properties of these forms around the beginning of the XIX century.  In 1918, the German mathematician introduced the fractional dimension.
Mandelbrot defined a fractal as a form with self-similarity: a form composed by copies transformed of him. Also, as one looks more deeply or more closely at a fractal, its inner parts have a similar design to that of the whole object.

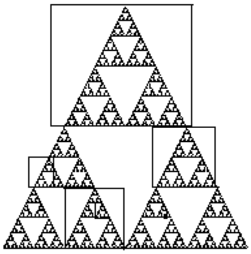
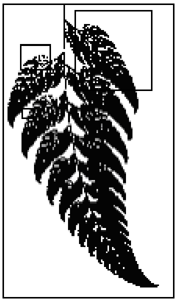
(a)  (b) 

Fig. 1. Two famous fractal: (a)Sierpinski Triangle and (b) Barnsley Fern, composed by copies of the whole object.

Later he proposed another definition: a fractal is a form with a fractional dimension. In Euclidian terms a form can be one-dimensional (lines), two-dimensional (filled figures such as squares, trapezoids, and circles), and three-dimensional (filled objects such as cubes and spheres) objects. Fractals have dimensions that are in between these three integer dimensions.

## 2. ITERATED FUNCTION SYSTEM (IFS)

An Iterated Function System (IFS) is a finite collection of contractions $F_i$: $X \rightarrow X$ defined on a metric space X. Each extends to a mapping (different but denoted by the same letter) $F_i$: $H(X) \rightarrow H(X)$, where H(X) is the space whose points are nonempty compact subsets of X. When endowed with the Hausdorff metric, H(X) is complete if X is. In addition, contractions $F_iX \rightarrow X$ remain contractions as mappings of H(X). Together, the $F_i$ define another contraction F: $H(X) \rightarrow H(X)$, by the following formula: for every $A \in H(X)$, $F(A) = \cup F_i(A)$. M. Barnsley demonstrated that since we are working in a complete metric space, F has a fixed point (set!) $A_F$, so that $F(A_F) = A_F$, and this point can be reached by successive approximations from any starting location. This result is known as the College Theorem. Fixed points of IFSs are called attractors or invariant sets.

Two problems arise: the direct and the inverse one. The direct problem is to find the fixed point of a given IFS and is solved by what is known as the *deterministic* algorithm. Start with a set $A_0 \in H(X)$ and compute successively $A_k = \cup F_i(A_{k-1}) = F(A_{k-1})$, $k > 1$.

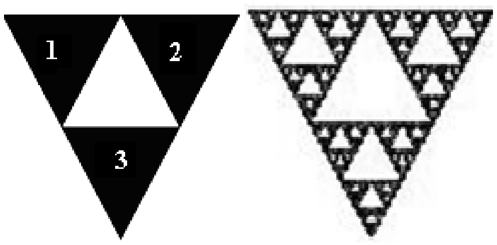
The sequence $\{A_k)$ converges to the fixed point $A_F$ of $\{F_i\}$ as $k \to \infty$.
The inverse problem is: for a given set $A \in H(X)$, find an IFS $\{F_i\}$ that has A as its fixed point. The mathematics of the second, *random iteration* algorithm is more complex but implementation is more straightforward. Assign positive frequencies $p_i$ to the mappings $F_i$. Start with an arbitrary point $x_0 \in X$. At every step k+1, select $x_{k+1}$ from the set $\{F_i(x_k)\}$. $F_j(x_k)$ is selected with the probability $p_j / \sum p_i$. The sequence $\{x_k\}$ converges to $A_F$. In practical terms, to depict an approximation of $A_F$ on a computer, the points of the sequence are displayed starting with a reasonably large index. The numbers $\{p_i\}$ have no effect on the fixed point $A_F$ but influence significantly the rendering of its approximations. The inverse problem is the base of the fractal compression: it is more efficient to store into computer the parameters defining the IFS that the whole image.

**3. IFS AND FRACTALS**

Fractal can be represented using iterated function system (IFS). Considering that a fractal is a composed by copies of himself, each transformation in the IFS describes the affine transformation characterizing every copy.
An affine transformation is a function:
*$f(x, y) = (a*x+b*y+e, c*x+d*y+f)$,*
*$x, y \in X$*
with the parameters *a, b, c, d, e, f* .
It can be written also as:

$$f(x, y) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}.$$

Consider the famous selfsimilar fractal the Sierpinski Triangle composed by tree copies of himself:



$$f_1(x,y)=\begin{bmatrix}1/2 & 0\\ 0 & 1/2\end{bmatrix}*\begin{pmatrix}x\\ y\end{pmatrix}+\begin{bmatrix}0\\ 0\end{bmatrix},$$

$$f_2(x,y)=\begin{bmatrix}1/2 & 0\\ 0 & 1/2\end{bmatrix}*\begin{pmatrix}x\\ y\end{pmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix},$$

$$f_3(x,y)=\begin{bmatrix}1/2 & 0\\ 0 & 1/2\end{bmatrix}*\begin{pmatrix}x\\ y\end{pmatrix}+\begin{bmatrix}1/2\\ 1/2\sqrt{3}\end{bmatrix}$$

Fig. 2. The Sierpinski Triangle and the associated IFS

Two other interesting results are the Barnsley's Fern and the Heighway dragon presented below.

$$f_1(x,y)=\begin{bmatrix}0 & 0\\ 0 & 0.15\end{bmatrix}*\begin{pmatrix}x\\ y\end{pmatrix}+\begin{bmatrix}0\\ 0\end{bmatrix},$$

$$f_2(x,y)=\begin{bmatrix}0.2 & -0.25\\ 0.2 & 0.2\end{bmatrix}*\begin{pmatrix}x\\ y\end{pmatrix}+\begin{bmatrix}0\\ 1.5\end{bmatrix},$$

$$f_3(x,y)=\begin{bmatrix}-0.15 & 0.3\\ 0.25 & 0.25\end{bmatrix}*\begin{pmatrix}x\\ y\end{pmatrix}+\begin{bmatrix}0\\ 0.45\end{bmatrix},$$

$$f_4(x,y)=\begin{bmatrix}0.85 & 0.05\\ -0.05 & 0.85\end{bmatrix}*\begin{pmatrix}x\\ y\end{pmatrix}+\begin{bmatrix}0\\ 1.5\end{bmatrix}$$

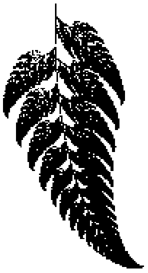


Fig. 3. The selfsimilar fractal Barnsley's Fern and the associated IFS

$$f_1(x,y)=\begin{bmatrix}\sqrt{3}/2 & 0\\ 0 & 1/2\end{bmatrix}*\begin{pmatrix}x\\ y\end{pmatrix}+\begin{bmatrix}0\\ 0\end{bmatrix},$$

$$f_2(x,y)=\begin{bmatrix}1/2 & 0\\ 0 & 1/2\end{bmatrix}*\begin{pmatrix}x\\ y\end{pmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix},$$

$$f_3(x,y)=\begin{bmatrix}1/2 & 0\\ 0 & 1/2\end{bmatrix}*\begin{pmatrix}x\\ y\end{pmatrix}+\begin{bmatrix}1/2\\ 1/2\sqrt{3}\end{bmatrix}$$

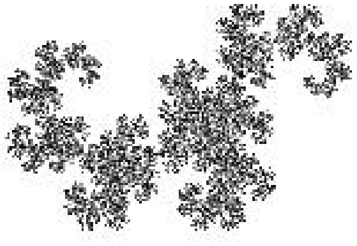


Fig. 4. The Heighway dragon and the associated IFS

## 4. THE RANDOM ITERATION ALGORITHM

The random iteration algorithm is a form of the deterministic algorithm. Knowing the IFS we can generate the fractal associated to it, using the direct problem and the college theorem discussed above. Starting with a single point, we can transform it using the IFS-transformations for thousands times and finally we obtain the fractal we want.

The method may be illustrated with the Yuval Fisher's special copyrighter which receives as entry an arbitrary image (may be a point) and applies to it the set of affine transformation, generating a new image. The image obtained is transmitted, using a feedback process, on the entry of the copyrighter and the process is repeated for several times. For example, consider that the transformations are those which describe the Sierpinski Triangle. If we test the Yuval Fisher copyrighter for different initial images we can observe that the final image is the same, so it not depends on the initial image but is defined by the affine transforms applied to it:

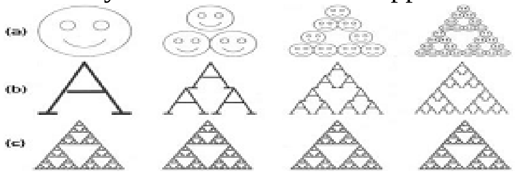


Fig. 5. The Yuval Fisher copyrighter

The algorithm is:

*P1. Set (x,y) the starting point - may be (0,0);*

*P2. Let (x1, y1) be the point obtained by applying an arbitrary transformation in the IFS; repeat the second step with (x1,y1) as initial point*

In the section below we implemented the algorithm in C++ programming language; consider that PutPixel(x,y) is a function which display on screen a point with coordinates (x,y).

```
void IFS(double p[][6], int n)
//n=numar transformari
//p[k][0]=r
//p[k][1]=s
//p[k][2]= α
//p[k][3]=β
//p[k][4]=e
//p[k][5]=f
{
        double a, b, c, d, e, f;
        int x = 0., y = 0.;//starting point
        int k;//current transformation
        int xx;
        srand((unsigned)time(NULL));
        for( long i=0;i<10000;i++)
        {
k=rand()%n; //an arbitrary transformation
a= p[k][0] *cos(p[k][2]*PI/180);
b = -p[k][1]*sin(p[k][3]*PI/180);
c =p[k][0]*sin(p[k][2]*PI/180);
d = p[k][1]*cos(p[k][3]*PI/180);
e = p[k][4];
f = p[k][5];
xx = a*x + b * y + e;
y = c*x+d*y+f;
x = xx;
PutPixel(x, y);
        }
}
```

## CONCLUSIONS

In 1900's decades, fractals were regarded as "strange images". Representing a fractal image (although the name "fractal" was imagined much later, around 1970) was a difficult task; these forms were studied only for mathematical reasons. In 1981 John
Hutchinson defines the Iterated Function System theory in his work „Fractals and selfsimilarity" (*Hutchinson, [10])*. Later, in 1988 the American researcher Michael Barnsley succide to demonstrate the collage theorem in his works „Fractals everywhere" *(Barnsley, [1])*.
The collage theorem yield to the random iteration algorithm, those implementation in C++ language is presented above and which is very ingeniously illustrated by Yuval Fisher's special copyrighter. It is one of the most used methods of generating a selfsimilar fractals such are Sierpinski Triangle, Barnsley's Fern and Heighway dragon exemplified above.
As this paper shows, the random iteration algorithm is very simple, it is easy to implement in any programming language and the results it generates are very spectacular and may be used for any graphical goal, not only for mathematical resons!

## BIBLIOGRAPHY

[1] Barnsley M., "Fractals Everywhere, 2nd ed.", Academic Press, Boston, 1993

[2] Barnsley, M., S.G. Demko, "Iterated Function Systems and the Global Construction of Fractals" Proc.Roy.Soc.London, Ser.A 399, 1985, 243-275

[3] Buchnicek M., M. Nezadal, O. Zmeskal, „Numeric calculation of fractal dimension", Nostradamus 2000, Prediction Conference, 2000;

[4] Chen S., J.M. Keller and R.M. Crownover, "On the calculation of fractal features from images", IEEE PAMI, 1998

[5] Crişan D., "Fractal spectrum of gray-level images", Cercetare ştiinţifică. Sesiunea cadrelor didactice, Universul Juridic, Bucureşti, 2005

[6] Dobrescu R.(ed), C. Vasilescu (ed), "Interdisciplinary Applications Of Fractal And Chaos Theory", Academia Română, Bucureşti, 2004

[7] Y., "Fractal image compression with quadtree", Fractal Compression Theory and Application to digital images, Springer Verlag, New York, 1994

[8] Fisher Y., E.W. Jacobs, R. D. Boss, "Fractal image compression using iterated transformes", NOSC Techical Report, Naval Ocean System Center, San Diego, 1995

[9] Harris J. W., H. Stocker, "Hausdorff Dimension", "Scaling Invariance and Self-Similarity", "Construction of Self-Similar Objects.", cap. 4.11.1-4.11.3, Handbook of Mathematics and Computational Science, Springer-Verlag, New York, 1998, pg. 113-135

[10] Hutchinson J., "Fractals and Self-Similarity.", Indiana Univ. Journal Mathematics, 35, 1981, pg. 713-747

[11] Jaquin A. E., "Image coding based on a fractal theory of iterated contractive image transformation", IEEE Trans. On Image Processing, vol I, No 1, January 1992

[12] Lauwerier H., "Fractals: Endlessly ŞRepeated Geometric Figures" Princeton, NJ: Princeton University Press, 1991, pg. 29-31

[13] Mandelbrot B. „The fractal geometry of nature", Freeman, New York, 1983