AN EFFICIENT FRACTAL MODELLER
(Modification of the Iterated Function Systems)

Slawomir S. Nikiel;
Technical University of Zielona Gora; Department of Robotics and Software Engineering
ul. Podgorna 50; 65-246 Zielona Gora; Poland
e-mail: snikiel@irio.wsi.zgora.pl

ABSTRACT: Research and development of fractal techniques for modelling and control over intricate artificial and natural phenomena fall in the scope of this paper. The principal objective is for close to real-time display of fractal structures. Iterated Function Systems rendering algorithms can be easily adapted to parallel processing. Modification of the IFS representation allows great speedup of the rendering process. Since the method relies on classical geometrical transformations, intricate structures such as clouds, plants, etc. can be plugged in already existing graphical editors and virtual environments. Possible applications of the method include: travel & flight simulation, architectural design (CAD systems), art and home entertainment.

KEYWORDS: Computer Graphics, CAD, Fractals, Iterated Function Systems, Virtual Reality

1. INTRODUCTION

Computer graphics moves toward the highly detailed images, giving us real time impressions similar to natural scenes [1, 3, 11, 14, 15]. There are many attempts to reach the ultimate graphics presentation. Generally, we expect images that represent highly intricate structures, offer high quality in different viewing scales, are efficiently rendered, have limited memory consumption and can be interactively manipulated. The chaos and the fractal theories seem to give us a tool fulfilling these requirements. However, typical fractals are either very time consuming or very beautiful but far from natural objects [1, 5, 6, 7, 12, 13]. The IFS theory gives the best control over the rendered fractals [1, 2, 4, 8, 9]. The original Barnsley technique encodes fractal structure as a set of maps - geometrical transformations. The gray scale is originally described by the associated probabilities. In this paper I present the Modified Iterated Function Systems where the gray scale transform is encoded together with the shape. There are several advantages of such a modification. Multicolour objects can be described by one set of IFS functions. Colours are determined by code not by probabilities, this drastically reduces the rendering process. A single deterministic framework based on the presented technique can reach a seemingly unlimited range of objects. The present approach contrasts with the usage of random recursive algorithms (used to produce terrain models) and stochastic procedures used to produce clouds and textures [11, 12], in all such cases the final product depends upon the precise random number sequence called during computation. The Modified IFS have the feature that small changes in the parameter values in the code yield only small changes in the resulting image. This is important for system independence, interactive usage and animation. Furthermore, images vary consistently with respect to changes of viewing window and resolution. Images can be efficiently generated to a very high resolution, or equivalently viewed within a small window, without reducing to blocks of solid colour.

2. FORMALISM EMPLOYED

A two-dimensional IFS consists of a set of N affine contractive transformations, N an integer, denoted by $\{w1, w2, ..., wN\}$ where w: $R^2 \rightarrow R^2$ is defined by

$$w\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} a \times x + b \times y + e \\ c \times x + d \times y + f \end{bmatrix} \tag{1}$$

where a, b, c, d, e, f are real constants.

Each $w(x, y)$ is taking R^2 into R^2 together with a set of probabilities $\{p1, p2, ..., pN\}$ each $pi > 0$ and

$$\sum_{i=0}^{N} pi = 1.0 \tag{2}$$

Let si denote Lipshitz constant for wi for each i = 1, 2, ... , N. Then we say that IFS code is an IFS $\{wi, pi, i = 1, 2, ... , N\}$ if obeys the average contractivity condition:

$$s1^{p1} \times s2^{p2} \times ... \times sN^{pN} < 1.0 \tag{3}$$

By theorem of Barnsley and Elton [1, 2, 4, 8, 9] there is a unique associated geometrical object, a subset of R^2 called the attractor of the IFS (Fig.1). There is also a unique associated measure. This measure can be thought of as a distribution of infinitely fine sand of total mass one, lying upon the attractor. The underlying model consists of the attractor together with the measure. The structure of an attractor is controlled by the affine maps $\{w1, w2, ..., wN\}$ in the IFS code. The measure is governed by the probabilities $\{p1, p2, ..., pN\}$ and in classical representation is displayed as a grayscale transorm. Algorithms given by Barnsley [1, 2, 3] compute images with the usage of random iteration. In effect of a random walk in R^2 the attractor is generated from the IFS code. The measure for the pixels is obtained from the relative frequencies with which the different pixels are visited. The result is a rendered image.

A simple example in two dimensions would be the effect of affine transformations $w1$, $w2$ (Fig. 1). $w2$ is a scaled down, translated copy of a rectangle (0, 0 / 1, 1), while $w1$ is rotated, skewed and translated one.

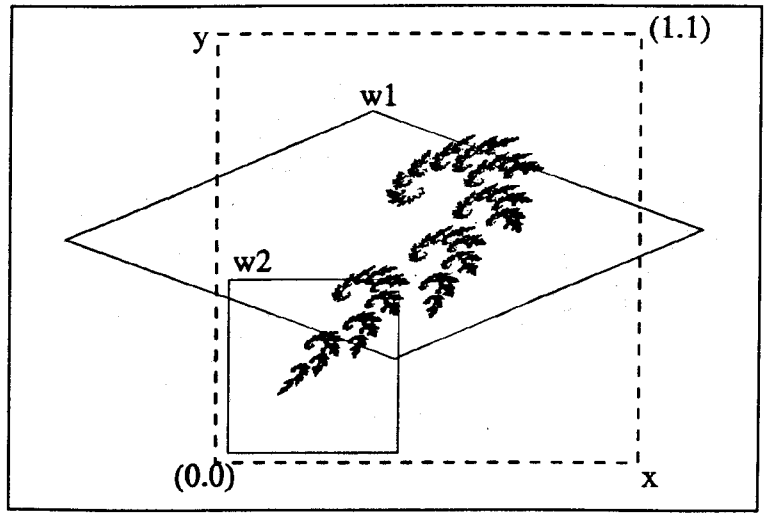


Figure 1. Sample of affine transformations resulting in an IFS attractor.

The Collage Theorem [1, 2, 3, 4] provides means for calculating IFS transformations {$w1$, $w2$, ..., wN}. Generally, the better the target image is covered by its smaller copies, the closer visually is the original and the rendered set. The overlapping due to number of iterations is preferred to be avoided.

3. MODIFICATION OF THE IFSs

Let modify the w transform (1):

$$w\begin{bmatrix} x \\ y \\ c \end{bmatrix} = \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ c \end{bmatrix} + \begin{bmatrix} b1 \\ b2 \\ b3 \end{bmatrix} \tag{4}$$

Now, we can observe that the colour does not depend on the set of probabilities (2). Associated probabilities are chosen according to the volume of each code wi, with total sum 1.0. Larger areas demand larger probabilities [1, 2, 3, 4, 8].

The IFS can be modified in such a way that upper dimensions (for instance representing brightness (gray scale transform), hue and saturation) are added to an IFS code [8]. For the purposes of this paper we restrict our attention to a three dimensional IFS, where the brightness may be treated as the third dimension (6) In that case, the brightness is evaluated by the following function :

$$w(c) = a31 \times x + a32 \times y + a33 \times c + b3 \tag{5}$$

Considering a modified IFS code (4) we have a number of operators determining the shape and the colour of an MIFS attractor: $a11$, $a12$, $a21$, $a22$ determine the rotation, skew and scaling; $b1$, $b2$ determine the translation; $a33$ defines self-similarity of the gray scale transform; $b3$ is the basic value for the module brightness; $a31$, $a32$ determine local shading (in the horizontal and vertical direction, respectively); $a13$ - when different from 0.0 show influence of the brightness value on the horizontal wi geometry; $a23$ - as $a13$ but in the vertical direction. Parameters $a31$, $a32$, $a33$, $a13$, $a23$, $b3$ with assigned value 0.0 have no influence on the attractor. Above listed operators are determined for the simplest model of a MIFS. If we consider more complex MIFS structures (e.g., $w(x, y, H, S, B)$ or $w(x, y, R, G, B)$) it will result in a number of new operators available, e.g., operators determining influence of hue on the saturation process.

4. PROPERTIES

The model presented in the preceding section has several interesting properties:
- a seemingly unlimited range of objects can be generated;
- thanks to the Collage Theorem we can model graphically most of natural phenomena;
- images can be generated to a very high resolution;
- objects are fully scaleable, transformable and their shape is under control;
- small changes in the values in the code result in small changes in rendered images;
- the model provides deterministic colour control;
- rendering is fast and depends mostly on the target image resolution;
- rendering time is almost invariable considering binary, gray scale and full colour;
- higher dimensions, while applied to MIFS, enable better control of the object;
- the method can be relatively easy developed to 3 dimensional objects;
- parallel and matrix processing will increase efficiency of the rendering process;

Additional important feature is the high compression rate (considering object data/image data), for instance: if we have a MIFS object described by 4 codes each {x, y, z, h, s, b, t}; each parameter is encoded in 4 bytes. Then it needs 4×7×8×4=896, say 1kB to store. Then,

A. for the Hi-res SVGA screen has 1024 × 1024 pixels, each RGB (3 bytes). It needs 3MB of memory to store. Compression rate: 3000 to 1.

B. for the A4 page with 200 DPI resolution needs approximately 36MB of memory storage. Compression rate: 36000 to 1

C. Say, that the average MIFS object consists of 20 codes [4, 9], then it needs approximately 5kB. Typical CD-ROM usually offers 650 MB of memory. Hence, we can save up to 650 000 000 / 5 000 = 130 000 objects on one optical disc.

4. THE RENDERING ALGORITHM

The algorithm presented in the paper (Appendix A) is a developed version of the ideas given by Barnsley [1, 2, 3]. It starts from a MIFS code $\{wi, pi, i = 1, 2, \ldots, N\}$ for wi descibed as (4) together with a specified viewing window V and resolution L - horizontal and M - vertical. In effect of a random walk in R^3 the gray scale attractor is generated from the MIFS code. An initial point $(x0, y0, c0)$ needs to be fixed. We choose its parameters where we suppose the point $(x0, y0, c0)$ belongs to the image. A total number of iterations *num* large compared to $L \times M$ also needs to be specified.

This process can be easily applied to parallel processing architectures. It has been shown [10] that classical IFS rendering algorithm can be mapped to the hypercube architecture. Also, it has been observed that the larger the dimension of the cube, the better performance is achieved (almost linear speedup compared with the number of nodes/processors). In the approach presented in [10] the Host was devoted to the setup and to collection of statistic over the pixels (the gray scale evaluation) received from different nodes. In such a system the rate of 450.000 pixels in 150 ms for 16 nodes (DEC ALPHA 3000/500 workstation with iPSC/2 Hypercube simulator) has been achieved [10]. The Modified IFS can decrease the process of rendering several orders of magnitude. Thus, when applied to parallel architecture, real-time imaging will be available.

5. RESULTS

The examples of images obtained with the MIFS technique (Appendix B) have been rendered by the algorithm (Appendix A). All the images have been rendered with a C++ compiler on an IBM 386DX, 40MHz, SVGA computer. Considering that, the computation time ranges usually from several seconds, to 5 minutes (800 × 600 - screen resolution). The images should be treated as basic samples, showing possibilities of the method.

6. CONCLUDING REMARKS

The major advantage of the ideas mentioned in the paper is that the MIFS objects can be easily designed and manipulated. Achievements in rendering techniques, especially parallel processing will bring the method to the real-time displaying and interaction. The method, however, needs further investigations. Full colour three {x, y, z, hue, saturation, brightness, transparency} dimensional representation will be considered in the nearest future. This should result in a stand alone fractal design environment [9] or support the existing CAD and Virtual Reality systems [14, 15], extending their modelling capabilities with natural phenomena, e.g., plants, clouds, fur, etc..

REFERENCES

[1] Barnsley M.F.: Fractals Everywhere, Academic Press, Boston, 1988.

[2] Barnsley M.F.: Fractal Image Compression, AK Press, Wellesley, MA, 1993.

[3] Baumann R.: Grafik mit dem Home-Computer, Vogel-Buchverlag, Würzburg, 1984.

[4] Horn A.N.: IFSs and interactive image synthesis, Computer Graphics Forum, vol. 9; No 2. pp. 127-137, June 1990.

[5] Kudrewicz J.: Fraktale i Chaos, WNT, Warszawa, 1993.

[6] Mandelbrot B.: The Fractal Geometry of Nature, W.H. Freeman, N.Y., 1977.

[7] Mandelbrot B.: Fractals - a Geometry of Nature, New Scientist, No 1734, 1990, pp.38-40.

[8] Nikiel S.S.: Modified Iterated Function Systems, Report PREPRINTY No 22/23, ICT PWr, Wroclaw, 1993.

[9] Nikiel S.S. : A Graphic Design Environment Based on Modified IFS. Proc. of the Tenth International Symposium on Computer and Information Sciences, Turkey, 1995, pp. 597-604.

[10] Ozkasap. O, Kantarci. A. : Models for Paralell and Distributed Implementation of Fractals,Proc. of the Tenth International Symposium on Computer and Information Sciences, Turkey, 1995, pp.699-706

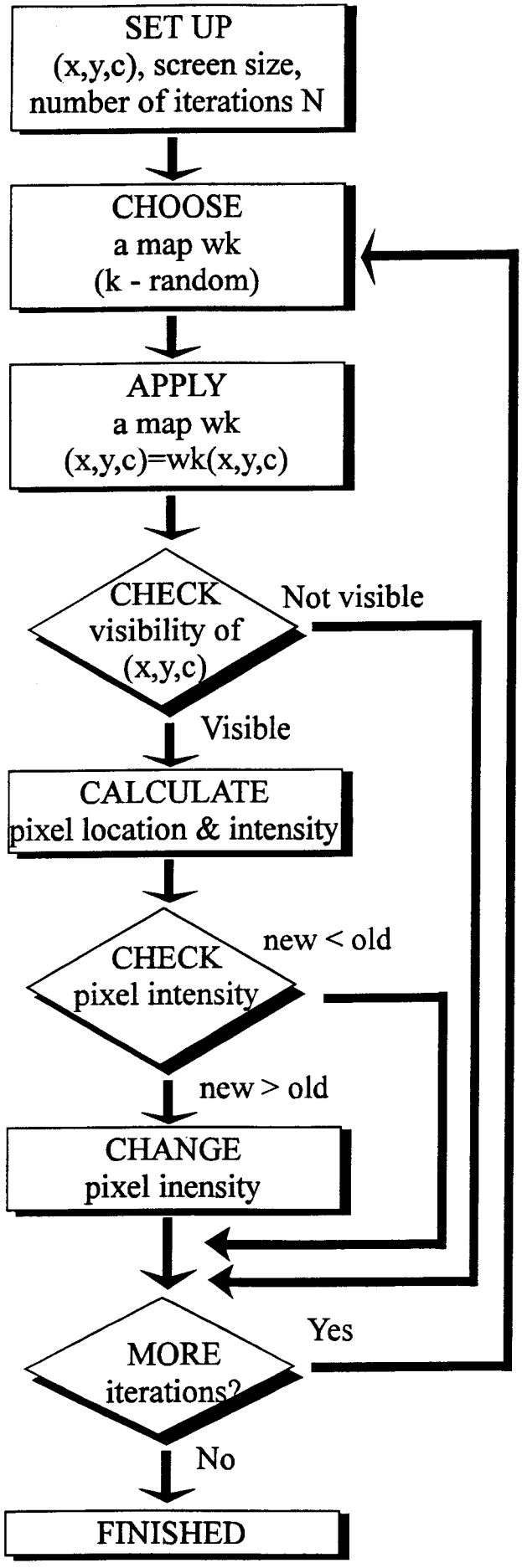
[11] Schachter B.J.: Computer Image Generation, Wiley, N.Y., 1983.

[12] The Science of Fractal Images, Springer Verlag, N.Y. 1988.

[13] Wang B. et al:The Fractal Nature of an Ecological Model, Computer Graphics Foum, Vol. 2, No 1, 1992, pp. 61-64.

[14] Wodaski R.: Virtual Reality Madness, Prentice Hall International, 1993

[15] Zabrodzki J.: Computer Graphics vs Virtual Reality, Proceedings of the 5th Microcomputer School on Computer Vision and Graphics, FORMAT, Wroclaw, 1994, pp. 171-182.



APPENDIX A

Algorithm **RenderMIFS**(x, y, P, num, xmin, xmax, ymin, ymax, cmin, cmax, L, M, Maxcolour, seed)

Arguments:
x, y, c-starting point of iteration
P[ ]-array of probabilities summing to 1
num-total number of iterations
xmin, xmax- low and high x-value of image window
ymin, ymax- low and high y-value of image window
cmin, cmax- low and high colour value
Maxcolour- number of colours to be used
L, M- image resolution in x- and y- direction
seed- seed value for random number generator

Variables:
colour, k, l, m, n - integer
sum - real number

Globals:
Arand- rand( ) returns values between 0 and Arand

Functions:
srand( )- initialization of random numbers
rand( )- random number generator
int(x)- integer part of argument x
w(x, y, c, k)- returns image of (x, y, c) under wk
putpixel(x, y, c)- puts a pixel of colour c in (x, y)
getpixel(x,y)- returns colour of the pixel (x, y)

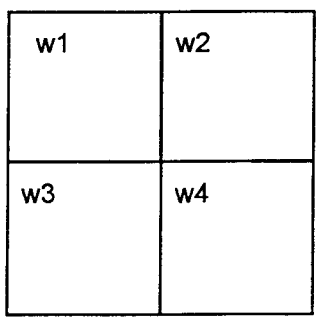
```
BEGIN
srand (seed);
FOR  n:=1 TO num DO
        r:=rand( )/Arand;
        sum:=P[1];
        k:=1;
        WHILE (sum<r) DO /* choose a map wk */
              k:=k+1;
              sum:=sum+P[k];
        END WHILE
        (x, y, c):= w(x, y, c, k); /*apply the map wk*/
        IF(x>xminANDx<xmax AND y>ymin
        AND y<ymax AND c>cmin AND c<cmax)
        /* check visibility of the rendered point */
        THEN BEGIN /* previewcalculated pixel */
              l:= int(L×(x-xmin)/(xmax-xmin));
              m:= int(M×(y-ymin)/(ymax-ymin));
              colour:= int(Maxcolour×(c-cmin)/
                    (cmax-cmin));
              IF (getpixel(l, m)<colour)
              /* check pixel intensity */
              THEN putpixel (l, m, colour)
              END IF
        END
END FOR
END
```

APPENDIX B

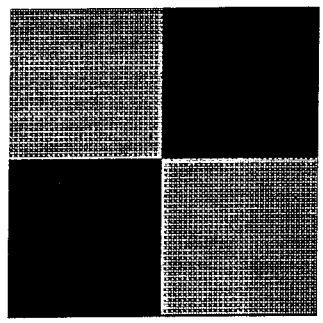
For the MIFS representation (4), the images presented below have sepecified values:
$-1.0 < ai, bi < 1.0$; All $a11 = a21 = 0.5$; $a12 = a13 = a22 = a23 = 0.0$

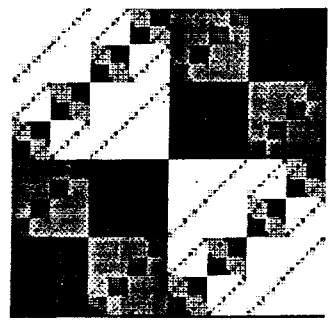
	$b1$	$b2$
$w1$	0.0	0.5
$w2$	0.5	0.5
$w3$	0.0	0.0
$w4$	0.5	0.0

Position of the maps (O) and associated attractors (A, B) are presented below together with the parameters:



O


A


B

A: All $a31 = a32 = a33 = 0.0$; for $w1$, $w4$ $b3 = 0.2$; for $w2$, $w3$ $b3 = 0.0$;

B: All $a31 = a32 = 0.0$, $a33 = 0.9$; for $w1$, $w4$ $b3 = 0.1$; for $w2$, $w3$ $b3 = -0.05$;

Images obtained for different values of a grayscale transorm plus self similarity are presented below: