An Interactive Application for Modeling Two-Dimensional IFS Fractals

Elena Hadzieva
University of Information Science and Technology "St. Paul the Apostle"
Partizanska bb, 6000 Ohrid, Macedonia
Email: elena.hadzieva@uist.edu.mk

Jovan Petkoski
University of Information Science and Technology "St. Paul the Apostle"
Partizanska bb, 6000 Ohrid, Macedonia
Email: jovan.petkoski@cse.uiest.edu.mk

Abstract

The paper presents a novel interactive application created in the C# programming language under the Microsoft Visual Studio environment, called Fractal Engine. The application enables generating and interactive modeling of two dimensional IFS fractals in real time, modifying different parameters in the IFS code, making animations by gradually changing one or more coefficients in the IFS code, coloring fractals, saving files in different image formats for later post-processing or creating more complex images. The modeling is enabled by assigning barycentric coordinates of the points that form the fractal image, with respect to three non-collinear points defined by the user. The application allows finding vertexes of a kind of triangle with minimal area that contains the fractal, which has particular advantages in modeling.

I. INTRODUCTION

Fractals are very convenient structures for representing natural objects. Collage theorem offers a possibility for obtaining an approximate model of a natural shape of interest, but in general if one needs (slight) modification of the model, there the problems begin. Namely, after application of the collage theorem and simple calculations, the IFS code (with or without probabilities) of the model can be obtained, but slight changes of the parameters in the IFS code might result in unpredictable changes both in the shape and location of the fractal model. There are some cases where experienced researchers can controllably change the coefficients in the particular IFS code to obtain the intended transformation (8). We emphasize here that we speak about experienced researchers and specific IFS codes (in this case the codes of tree-like fractals).

In the paper we introduce a novel application which enables interactive affine transformations of any two-dimensional IFS fractal, as well as several other options related to visualization of IFS attractors. The key point of our application is the representation of the points that constitute fractal image by barycentric coordinates with respect to a predefined three pointwise independent points. This work is extension of the previous work of one of the authors, [13].

The remainder of this paper is arranged as follows. Section 2 contains related articles that describe different approaches in transforming fractals and well explain the importance and advantages of fractal models and transformations. The basic definitions and theoretical development of the application are specified in Section 3. Section 4 provides description of the application and contains examples that illustrate the practical value of the application. The conclusions and future work are given in Section 5.

mds
January 11, 2007

II. RELATED WORK

In [8] the authors make animation of fractal objects through interactive changes in the IFS code of the fractal. The same authors in [6] make weaving effects in metamorphic animation of tree-like fractals. In the last paper they show both local control of the fractal tree (by moving its branches) as well as global control (by making weaving effect.) Again this is done by sensible choice of the coefficients. In both cases, if the change of the coefficients in the IFS code is not carefully done, it might yield a complete loss of the form of the fractal and possibly in its location. Similar work is shown in [7], where the fractals are transformed by arbitrary affine transformations. After determining the coordinates of the points that form the fractal image, the authors modify the IFS code according to the desired location and generate the new fractal from the modified IFS code.

Barnsley and his collaborators in [4, 5, 6] with the aid of the code space and sections of the coding map, define a transformation between fractals and present the importance of transformation of fractals in digital imaging. To understand and to apply these transformations, one needs a big understanding of some sensitive parts of fractal theory (code space, coding maps, sections of coding maps, point-fibred IFS, mask, etc).
Kocić and Simoncelli in [15], [17] for the first time use barycentric coordinates to model fractals, by means of new type of IFS, which they called Affine Invariant IFS (AIFS). Also, for the first time Kocić and his collaborators mention minimal simplex as the best option for controlling affine transformations of fractals (see [1], [13]).

Our approach for transforming fractals is simpler, since it rather uses the IFS code of the fractal instead of the AIFS code and it relies on relatively basic theory. Moreover, the IFS code of the fractal attractor is not changed at all during the transformation (like in [8], [9], [7]); the only thing that is changing is the location of the three pointwise independent points. Also, our application is more advanced, because the transformation is visible and done in real time by clicking and dragging the three predefined points that form the affine basis. The application provides possibility for animation of objects by changing any coefficient in the code, in any reasonable interval, for any reasonable step-size. All features of application will be discussed in more details in Section IV.

III. MATHEMATICAL BACKGROUND OF THE APPLICATION

The theoretical development of the application is based on more or less commonplace terms for the fractal geometers, but essentially, very simple to be understood by any mathematician or engineer. That is the big advantage of our application.

A. Definitions from Barycentric Calculus

The following definitions are mainly taken from [19], but they can be also found in other references, like [12] or [10].

Definition 1: A set $S$ of $N$ points, $S = \{A_1, A_2, \ldots, A_N\}$ in $\mathbb{R}^n$, $n \geq 2$, is called pointwise independent set (and the points $A_1, A_2, \ldots, A_N$ are called pointwise independent) if the $N-1$ vectors $-A_1 + A_k$, $k = 2, 3, \ldots, N$, are linearly independent.

Definition 2: Let $S = \{A_1, A_2, \ldots, A_N\}$ be a pointwise independent set of $N$ points in $\mathbb{R}^n$. Then, the real numbers $a_1, a_2, \ldots, a_N$ satisfying $\sum_{i=1}^{N} a_i = 1$ are called barycentric coordinates of a point $P \in \mathbb{R}^n$ with respect to the set $S$, if

$$P = \sum_{i=1}^{N} a_i A_i. \quad (1)$$

The set of all points with nonnegative barycentric coordinates with respect to the set $S$ is called the convex span of $S$.

Definition 3: The convex span of the pointwise independent set $S = \{A_1, A_2, \ldots, A_N\}$ of $N \geq 2$ points of $\mathbb{R}^n$, is called $(N-1)$-dimensional simplex, or $(N-1)$-simplex and denoted $A_1A_2 \ldots A_N$.

Any set of two distinct points, for any $n$, is pointwise independent set in $\mathbb{R}^n$. The convex span of the two points is 1-simplex. Any set of three non-collinear points $A, B$ and $C$, for $n \geq 2$ is a pointwise independent set in $\mathbb{R}^n$ and the convex span of the points is 2-simplex, which is actually the triangle ABC together with its interior.

B. Definitions from Fractal Geometry

The second part of the theoretical background is related to fractals - their definition and construction ([2], [15], [11]). We will focus on two dimensional fractals, although the fractals can be defined on any complete metric space.

Definition 4: The transformation $w : \mathbb{R}^2 \to \mathbb{R}^2$ defined by

$$w(x) = A \cdot x + b, \quad (2)$$

where A and b are two-dimensional real matrix and vector, respectively, is called a (two-dimensional) affine transformation.

Definition 5: The finite set of affine contractive transformations, $\{w_1, w_2, \ldots, w_n\}$ with respective contractivity factors $s_i$, $i = 1, 2, \ldots, n$, together with the Euclidean metric space $(\mathbb{R}^2, d_E)$ is called (contractive) iterated function system (IFS) and denoted $\{\mathbb{R}^2; w_1, w_2, \ldots, w_n\}$. The number $s = \max\{s_i, i = 1, 2, \ldots, n\}$ is called contractivity factor of the given IFS.

Let $(\mathcal{H}(\mathbb{R}^2), h(d_E))$ denote the space of nonempty compact subsets of $\mathbb{R}^2$, endowed with the Hausdorff metric $h(d_E)$, derived from the Euclidean metric $d_E$. It can be shown that $(\mathcal{H}(\mathbb{R}^2), h(d_E))$ is a complete metric space. It can be also shown that if the IFS $\{\mathbb{R}^2; w_1, w_2, \ldots, w_n\}$ is contractive with contractivity factor $s$, then the operator $W : \mathcal{H}(\mathbb{R}^2) \to \mathcal{H}(\mathbb{R}^2)$, defined by

$$W(B) = \bigcup_{i=1}^{n} w_i(B),$$

is also contractive with the same contractivity factor, $s$. There exist a unique fixed point $A$ of $W$ (which is insured by the fixed point theorem) which obeys $W(A) = A = \lim_{n \to \infty} W(B)$, for any $B \in \mathcal{H}(\mathbb{R}^2)$. A is called the attractor of the IFS $\{\mathbb{R}^2; w_1, w_2, \ldots, w_n\}$. (Note that there exists a more general theorem that establishes weaker conditions under which the attractor of an IFS can be obtained - the IFS need not be contractive, see [3].)
The attractor of an IFS can have smooth structure, but taking into account the reason for establishing the theory of IFS - setting theoretical framework for examining fractal sets, we will focus only on fractal attractors. Also, generating and modeling simple, smooth objects is much more covered topic.

The two main algorithms for computing fractals from iterated function systems are deterministic algorithm and random iteration algorithm. Here we have a reason to compute fractals with the second one, because it rather deals with iterating points then iterating sets. We will assign barycentric coordinates to every point that constitutes the image of the fractal.

The short description of the random iteration algorithm follows. Besides the iterated function system \( \{ \mathbb{R}^2; w_1, w_2, \ldots, w_n \} \), in order to run this algorithm one needs a set of probabilities \( \{ p_1, p_2, \ldots, p_n \} \) that add to 1 (22). Each probability \( p_i \) is associated with the affine transformation \( w_i \). Having the IFS with probabilities defined, the initial point \( x_0 \) is chosen arbitrarily, and then recursively and independently, the sequence \( \{ x_n, n \in \mathbb{N} \} \) is generated, such that the event \( x_n = w_i(x_{n-1}) \) has the probability \( p_i \). Under certain conditions (more information could be found in (2), (3)), the sequence \( \{ x_n \}_{n=0}^{\infty} \) converges to the attractor of the given IFS.

C. The Theoretical Basis of Fractal Transformations

Three is the maximal number of pointwise independent points in \( \mathbb{R}^2 \), which is the reason why we deal with 2-simplex, i.e. triangle. Put in another way, three pointwise independent points form affine basis of the affine space \( \mathbb{R}^2 \). Given three pointwise independent points \( A, B, C \in \mathbb{R}^2 \) with respective rectangular coordinates \( (a_1, a_2), (b_1, b_2), (c_1, c_2) \) and arbitrary point \( M \in \mathbb{R}^2 \) with rectangular coordinates \( (x, y) \), according to (1) they are connected with the relation

\[
\begin{bmatrix}
    x \\
    y
\end{bmatrix} = a \begin{bmatrix}
    a_1 \\
    a_2
\end{bmatrix} + b \begin{bmatrix}
    b_1 \\
    b_2
\end{bmatrix} + c \begin{bmatrix}
    c_1 \\
    c_2
\end{bmatrix},
\]

(3)

where \( (a, b, c) \) are barycentric coordinates of \( M \) with respect to the points \( A, B, C \). The relation (3) represents conversion from barycentric to rectangular coordinates, while the unique solution of the system (3) in \( a \) and \( b \) and the relation \( c = 1 - a - b \) represent the conversion from rectangular to barycentric with respect to \( A, B, C \).

Let the simplex \( ABC \) be transformed into another simplex \( A'B'C' \). According to the fundamental theorem of affine transformations, there exists a unique affine transformation \( f \) of type \( \mathbb{R}^2 \rightarrow \mathbb{R}^2 \), mapping one simplex to the other. If the point \( Q \) has the same barycentric coordinates \( (a, b, c) \), but now with respect to \( A', B', C' \), it can be shown that exactly \( Q = f(M) \) (14). Therefore, when barycentric coordinates with respect to the simplex \( ABC \) are assigned to each point of the fractal image and kept unchanged, any affine transformation of the simplex will result in the same affine transformation of the fractal image.

After theoretical examination of the problem of transforming fractals, we decided to develop software application based on the simple and promising theory. The application, that we called Fractal Engine is described in the next section.

IV. THE APPLICATION - DESCRIPTION AND EXAMPLES

Fractal Engine is a user interactive fractal generation and rendering application written in the C# programming language under the Microsoft Visual Studio environment. Its GUI is user-friendly, consisting of several intuitive windows with a neatly organized layout, where all options and features could be selected or set up. In general, it provides really fast and high quality rendering results with real-time performance, despite all calculative capabilities fractals require. It was developed and tested on Intel Core i7-4500U machine with 8GB of RAM.

It produces the image of the fractal attractors by random iteration algorithm, with equal or distinct probabilities. The main application window is shown in Figure 1. The user can plot three distinct pinpointed by the mouse clicks. The points will be automatically connected with lines in a triangle \( ABC \), that could be called control triangle. The coordinates of the three points are displayed in the right side of the window. Now, a fractal attractor defined with the mappings in the coefficients window (Figure 2) could be produced by pressing the GENERATE button. The points that represent the fractal attractor are then related to the points \( A, B, C \), which makes the fractal ready to follow the affine transformation of the three points. The user performs the affine transformation by sliding the points, one at a time.

The fractal presented in most of the figures, called Peitgen tree, is an attractor of the following IFS: \( \{ \mathbb{R}^2; w_1, w_2, w_3, w_4 \} \), where the mappings \( w_i : \mathbf{x} \mapsto A_i \mathbf{x} + b_i, \ i = 1, 2, 3, 4, \) are defined with the matrices on the left list box and vectors on the middle list box on Figure 2.

Each transformation given in the coefficients window (Figure 2) could be additionally modified with a right click, deleted, copied or completed with probabilities (Figure 3). Invalid inputs like characters or special symbols are forbidden with the use of regular expressions (programming technique), preventing the program from crashing. There is also an option for automatic calculation of equal probabilities and a list of embedded templates of known IFS codes (Figure 4).

The button MINIMAL ABC (Figure 1) gives the right-angle isosceles triangle with its catheti parallel to the axes, that contains the fractal image. Such minimal simplex would produce better control since the whole attractor is in the convex span of the vertices. The visual transformation of a generated Peitgen tree fractal with the use of an arbitrary triangle and a minimal triangle are given on Figures 5 and 6, respectively. Part a) of both Figures 5 and 6 shows how and where primarily the fractal was
Fig. 1. The main application window.

Fig. 2. The coefficient window.

Fig. 3. Editing a particular transformation.

Fig. 4. List of included IFS code templates. Each item could be used with a right click.
generated and the position of the triangle vertices. Parts b), c) and d) of the figures show how the fractal is visually transformed by clicking and dragging the vertices.

Fig. 5. Modeling fractal by means of arbitrary triangle.

Fig. 6. Modeling fractal by means of a special triangle with minimal area that contains the fractal.

Figure 7 depicts the window where all rendering features could be set up in real time. The background, vertices and sides color could be changed (by selecting a precise color when the color selection window pops up, including a transparent color), a grid with certain cell size and color could be enabled behind the fractal and the fractal color could be modified into a solid color or color from image (each fractal pixel is colored with the location corresponding color from an user-defined picture). Several rendering results are displayed in Figure 8. In part a) of Figure 8 the fractal is generated with solid green color, in part b) the fractal is generated with solid color and green grid with cell size 15, in part c) the fractal is generated with solid color, vertices and sides colors are modified, while in part d) the fractal is rendered with the color from image algorithm explained earlier.

Fig. 7. A window where all rendering features could be set up in real time.

A window form with all the animation settings is shown in Figure 9. There is a repeater option for regenerating the same fractal several times/frames with a certain pause in milliseconds between each frame for observing the randomness. Also, there is an animator option where the user could set definite number of increment/decrement commands for any coefficient for a particular number of frames. Using zero-starting numbering, in the above case (Figure 9), the command a{0, 0, +0.1] means that
the element with index 0 in the 0th line from the matrix $A_1$ (in our case, the specified coefficient is 0.195) would be incremented by +0.1 each frame in the next 15 frames (i.e. the specified coefficient will take values $0.195, 0.295, \ldots, 1.595$). The result is shown in Figure 10. The whole animation concept is made possible with the use of multithreading programming technique. Animated peitgen tree (available in the templates list) with the command $a\{0, 0, +0.1\}$ after 15 frames is shown in Figure 10, part f). Part a) of the same figure shows the original tree with the minimal triangle, while parts b), c), d), e), f) show each third frame.

We created a real-time easy to use interactive application for interactive transforming, animating, coloring two-dimensional IFS fractals, and saving them in different formats for easier post-processing. The theoretical basis of affine transformation of the fractals lies in barycentric calculus, or more precisely, in the representation of the points that constitute the fractal image by barycentric coordinates with respect to given three pointwise independent points $A, B, C$. On such way, any affine transformation of the points will be followed by the same affine transformation of the fractal. The points $A, B, C$ and therefore the fractal, are affinely transformed visually, by clicking and dragging the points, which means that any IFS coded image can be subject to any affine transformation. The very important thing here is that the IFS code is not changed at all.

Several areas and aspects need to be subject to further modification and improvement, like the ability to save parameter files, allowing the user to easily return to previously created images for later exploration; larger choice of color rendering algorithms; use of filters and other image manipulation techniques; exporting an entire animation file; interactive continuous zoom and more methods for fractal transformation and control, like controlling with more than three points, transforming only part of the fractal or developing an application for modeling 3D fractals.
Fig. 10. Animated Peitgen tree.

REFERENCES


