Математички Билтен
50 (LXXVI) 1 (2026), 1–16

UDC: [TBD]
DOI: [TBD]

INTERACTIVE AND PREDICTABLE MODELLING OF NATURAL FORMS GENERATED BY ITERATED FUNCTION SYSTEMS

ELENA HADZIEVA [1]

Abstract. Natural objects are often poorly described by classical Euclidean geometry. Fractals, characterised by fragmented and highly irregular structures, provide an effective mathematical framework for representing natural forms. Fractals are commonly generated using iterated function systems (IFS), defined as sets of contractive mappings on a complete metric space, which guarantee the existence of a unique fixed-point (attractor), typically exhibiting fractal properties. In this work, we present a method for affine transformations of fractals, implemented in an interactive software application. The proposed approach enables intuitive and predictable design by directly manipulating predefined control points via click-and-drag operations. The effectiveness of the method and functionalities embedded in the software application are demonstrated through a case study of nature-like fractals, including a maple tree, a maple leaf, a seahorse, and the Barnsley fern. This case study also provides a review of selected articles on the modelling of fractals in which the author has been involved.

1. INTRODUCTION

Natural forms encountered in biology and the environment — such as plants, animals, and geological structures — exhibit a high degree of irregularity and structural complexity. Classical Euclidean geometry, while effective for describing regular and smooth objects, is often inadequate for modelling such complex forms. This limitation has motivated the development of a new mathematical branch in the 1970s, so-called fractal geometry. The main constructs of fractal geometry, fractals, are characterised by i) irregularity (the fractals are usually continuous but nowhere differentiable), ii) self-similarity (invariance under changes of scale), and iii) non-integer dimensions (defined for example as fractal, box-counting, or Hausdorf-Besicovitch dimension), properties that make fractals

2010 *Mathematics Subject Classification.* Primary: 28A80, Secondary: 65D18, 92C00.

Key words and phrases. Iterated Function System, Natural Fractal, Software Application, Interactive Modelling.

convenient to approximate natural objects more realistically than traditional geometric models. The term fractal was first introduced by Mandelbrot in 1975 in his French monograph *Les Objets Fractals: Forme, Hasard et Dimension*. The word is derived from the Latin word fractus, meaning "broken" or "fragmented". Although 1975 is often regarded as the beginning of fractal geometry, a more systematic approach to this field with many definitions, properties, fundamental concepts, tools, examples, and counterexamples, was presented later in Mandelbrot's pivotal book [21]. Fractals can be generated using a variety of mathematical approaches. In this paper, we restrict our attention to fractals generated by iterated function systems (IFS), which are introduced in the following section. The use of such systems as a tool for constructing fractals was first proposed by Hutchinson [18], although not under the name iterated function system. The term itself was later introduced by Barnsley and Demko [8]. Subsequent works, including [2, 12, 24], further developed the theoretical foundations of IFS and explored a wide range of applications. Currently, many software tools for generating IFS fractals are available. However, to the best of our knowledge, none of them, except Fractal Engine, are simultaneously interactive, predictable, and real-time. Most of them generate static fractals or allow modifications that lack interactivity or predictability. The Fractal Engine software application, developed in 2015-2017 ([14]–[17]), to the best of our knowledge, for the first time enabled interactive, real-time, and predictable affine transformations of fractals. While the literature extensively addresses the generation of two-dimensional IFS fractals (see, e.g., [2, 12, 18, 24], comparatively less attention has been devoted to their transformation. Barnsley and his collaborators in [3, 4, 5] define fractal transformations from one fractal attractor to another, whose IFSs have a common code space. Chang [10] proposes a method for obtaining affinely transformed 2D fractals by modifying the IFS code and applying affine transformations in their homogeneous form. Based on the theorem stated in [2], which claims that the attractor depends continuously on the coefficients of the IFS mappings, Darmanto [11] varies a coefficient in an IFS to produce a weaving effect in a tree fractal. However, such approaches are generally based on trial-and-error experimentation and do not provide interactive or fully predictable control. Predictable, but not interactive, transformation of IFS fractals are reported in [19, 20], while interactive, but non-predictable, transformation of Julia fractal sets, generated by the escape-time algorithm, is described in [1]. The main contribution of this paper is the interactive and predictable modelling of natural fractal forms generated by iterated function systems, with the aim of supporting further applications in biomimetics and regenerative design. At the same time, the paper reviews selected previous work by the author on the modelling of fractals and relates the presented approach to these earlier studies. The Fractal Engine application, which provides the modelling tool, is revisited in Section 3 and applied to the modelling of natural-like fractals. Section 2 briefly reviews the mathematical background of iterated function systems, barycentric (affine) coordinates, and affine transformations. Finally, Section 4 concludes the paper and outlines directions for future research.

2. Theoretical Background

2.1. **Foundations from Fractal Geometry.** Let (X, d) be a complete metric space and let $S = \{w_1, w_2, \ldots, w_N\}$ be a finite family of contraction mappings $w_i : X \to X$, each satisfying

$$d(w_i(\mathbf{x}), w_i(\mathbf{y})) \leq s_i\, d(\mathbf{x}, \mathbf{y}), \quad \text{for all } \mathbf{x}, \mathbf{y} \in X,$$

where $0 \leq s_i < 1$.

The pair (X, S) is called an *iterated function system (IFS)*. In some references, such systems are referred to as contractive or hyperbolic IFS; however, in this paper, contractivity is incorporated directly into the definition.

Associated with the metric space (X, d) is the metric space $(\mathcal{H}(X), d_H)$, where $\mathcal{H}(X) = \{B \subset X, B \neq \emptyset, B \text{ is compact}\}$ is endowed with the Hausdorff metric induced by d, defined by

$$d_H(A, B) = \max\left\{ \max_{\mathbf{a}\in A} \min_{\mathbf{b}\in B} d(\mathbf{a}, \mathbf{b}),\ \max_{\mathbf{b}\in B} \min_{\mathbf{a}\in A} d(\mathbf{a}, \mathbf{b}) \right\},$$

for any non-empty compact sets $A, B \subset X$.

Associated with an IFS S is the *Hutchinson operator* $W_S : \mathcal{H}(X) \to \mathcal{H}(X)$, defined by

$$W_S(B) = \bigcup_{i=1}^{N} w_i(B).$$

The operator W_S is a contraction on $(\mathcal{H}(X), d_H)$ (see [2]). Consequently, by the Banach fixed-point theorem, there exists a unique non-empty compact set $F \subset X$ that is a fixed point of W_S, that is,

$$F = W_S(F) = \bigcup_{i=1}^{N} w_i(F).$$

The fixed point F is called the *attractor* of the IFS S. It typically exhibits fractal properties and represents the limit of the iterative process, independently of the initial compact set. More precisely,

$$F = \lim_{j\to\infty} W_S^{(j)}(B), \text{for any } B \in \mathcal{H}(X) \tag{2.1}$$

where $W_S^{(j)}$ denotes the j-th forward iterate of W_S.

One of the main algorithms for generating fractals, the so-called deterministic algorithm, is based exactly on equation (2.1). Starting from an arbitrary initial compact set A_0, repeated application of W_S produces a sequence of sets that converges to the attractor with increasing precision as the number of iterations grows.

The second main algorithm that enables creating fractal sets is the random iteration algorithm (RIA). Exactly RIA is used in the *Fractal Engine* software application. This algorithm offers a stochastic method for generating an IFS attractor. Starting from an arbitrary initial point $\mathbf{x}_0 \in X$, at each iteration a mapping w_i

is selected according to a prescribed probability distribution, where w_i is chosen with probability p_i such that $\sum_{i=1}^{N} p_i = 1$. This process generates a sequence

$$\mathbf{x}_{k+1} = w_i(\mathbf{x}_k), \tag{2.2}$$

which, under suitable conditions, converges to the attractor A with probability one [2, 4]. When plotted, the points of the sequence (2.2) typically reveal the intricate fractal structure of F[1].

In practical applications, the mappings w_i are often chosen as affine transformations in $\mathbb{R}^2$ or $\mathbb{R}^3$, of the form $w(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$, where A is a 2×2 or 3×3 matrix and $\mathbf{b}$ is a vector in $\mathbb{R}^2$ or $\mathbb{R}^3$, respectively.

The choice of the random iteration algorithm in *Fractal Engine* is motivated by the fact that it represents a fractal as a discrete set of points. Each point can be described both by its Cartesian coordinates and by its barycentric (affine) coordinates, with respect to a chosen affine basis. The latter representation plays a crucial role in enabling interactive and predictable transformations, as explained in the next subsection.

2.2. **Foundations from Affine Geometry.** Given the complete metric space $X = \mathbb{R}^m$, and its affine basis $\mathcal{A} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{m+1}\}$. The Cartesian coordinates of $\mathbf{a}_i$ are denoted by $[a_{i1} \;\; a_{i2} \ldots \; a_{im}]^T$, $i = 1, 2, \ldots, m+1$. For any point $\mathbf{p} \in \mathbb{R}^m$, the relation between its Cartesian coordinates $\mathbf{p} = [p_1 \;\; p_2 \ldots p_m]^T$ and its barycentric coordinates with respect to $\mathcal{A}$, denoted by $\mathbf{p}_{\mathcal{A}} = [p_{\mathcal{A}1} \;\; p_{\mathcal{A}2} \ldots p_{\mathcal{A},m+1}]^T$, is expressed by

$$\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{bmatrix} = \sum_{i=1}^{m+1} p_{\mathcal{A}i} \begin{bmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{im} \end{bmatrix}, \quad \sum_{i=1}^{m+1} p_{\mathcal{A}i} = 1, \tag{2.3}$$

which is equivalent to

$$\begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = A \cdot \mathbf{p}_{\mathcal{A}}, \quad \text{where } A = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \ldots & \mathbf{a}_{m+1} \\ 1 & 1 & \ldots & 1 \end{bmatrix}. \tag{2.4}$$

Consider now an affine transformation

$$w : \mathbb{R}^m \to \mathbb{R}^m, \quad w(\mathbf{x}) = B \cdot \mathbf{x} + \mathbf{b} \tag{2.5}$$

where B is an $m \times m$ matrix and $\mathbf{b}$ is an $m \times 1$ vector.

A fundamental property of affine transformations is that barycentric coordinates are preserved, meaning that the barycentric coordinates of $w(\mathbf{p})$ with respect to the transformed basis $w(\mathcal{A})$ coincide with those of $\mathbf{p}$ with respect to $\mathcal{A}$. This key property enables *predictability* in the Fractal Engine application: When an affine transformation is applied to the points defining the affine basis $\mathcal{A}$ (called *control points* in Fractal Engine), all points of the fractal consistently and almost

[1]Usually, the first 10-50 points are not plotted, because the randomly chosen initial point needs a few iterations to settle down on the image. 10000 attractor points are enough to depict the fractal with all of its details.

immediately (with a delay of a few milliseconds), follow the same transformation.

The formula (2.3) (or equivalently, (2.4)), and the aforementioned fundamental property provide the theoretical foundation for the *interactivity* of the software. Namely, when the user performs an affine transformation by clicking and dragging the control points, the barycentric coordinates remain constant, while the new screen positions of the fractal points are computed using (2.3) and plotted in milliseconds on the screen.

The theoretical basis of the affine transformation of IFS attractors and further details on the Fractal Engine application are given in [13]–[17]. In particular, while [14]–[17] focus primarily on the practical modelling and implementation aspects, [13] provides the theoretical proof that the affine image of an IFS attractor is itself the attractor of a precisely determined IFS (see Section 3 in [13]).

In the remainder of this paper, we restrict our attention to $X = \mathbb{R}^2$ endowed with the Euclidean metric d_E, which allows for a clear and intuitive representation of fractal sets. The following simple example illustrates the application of formulas (2.3)–(2.5) and the fundamental property.

Example. Consider the point $\mathbf{p} = (5, 5)$ and the affine basis $\mathcal{A} = \{\mathbf{a}_1,\ \mathbf{a_2},\ \mathbf{a}_3\}$, where $\mathbf{a}_1 = (1, 1)$, $\mathbf{a}_2 = (2, 4)$ and $\mathbf{a}_3 = (1, 5)$. To determine the barycentric coordinates of $\mathbf{p}$ w.r.t. $\mathcal{A}$, denoted by $\mathbf{p}_{\mathcal{A}} = [p_{\mathcal{A}1}\ \ p_{\mathcal{A}2}\ \ p_{\mathcal{A}3}]^T$, we solve the linear system given by the equation (2.4),

$$\begin{bmatrix} 5 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 4 & 5 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_{\mathcal{A}1} \\ p_{\mathcal{A}2} \\ p_{\mathcal{A}3} \end{bmatrix}.$$

Its solution is $\mathbf{p}_{\mathcal{A}} = (-1, 4, -2)$. Applying the affine transformation $w(\mathbf{x}) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \mathbf{x} + \begin{bmatrix} 0 \\ -2 \end{bmatrix}$ to the affine basis yields the transformed basis

$$w(\mathcal{A}) = \{w(\mathbf{a}_1),\ w(\mathbf{a_2}),\ w(\mathbf{a}_3)\} = \{(-1, -1),\ (-2,\ 2),\ (-1, 3)\}.$$

By the fundamental property, the barycentric coordinates of $w(\mathbf{p})$ w.r.t. $w(\mathcal{A})$ are the same as those of $\mathbf{p}$ w.r.t. $\mathcal{A}$, i.e. $w(\mathbf{p})_{w(\mathcal{A})} = (-1, 4, -2)$. Substituting these values into equation (2.4) produces the Cartesian coordinates of $w(\mathbf{p})$:

$$\begin{bmatrix} w(\mathbf{p})_1 \\ w(\mathbf{p})_2 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & -2 & -1 \\ -1 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 4 \\ -2 \end{bmatrix} = \begin{bmatrix} -5 \\ 3 \\ 1 \end{bmatrix}.$$

This result is consistent with the direct computation

$$w(\mathbf{p}) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 5 \end{bmatrix} + \begin{bmatrix} 0 \\ -2 \end{bmatrix} = \begin{bmatrix} -5 \\ 3 \end{bmatrix}.$$

This simple example illustrates the core idea underlying the Fractal Engine: instead of regenerating the entire fractal after applying an affine transformation, the transformed fractal can be obtained efficiently by recomputing the Cartesian

coordinates of its points with respect to the affinely transformed basis. This approach significantly reduces both computational time and memory usage ([16, 17]).

3. Fractal Engine – A Software Tool for Generating 2D IFS Fractals and Their Interactive and Predictable Modelling in Real Time

As mentioned earlier, the Fractal Engine application was developed in the period 2015–2018 ([14]–[17]). It was implemented in C# using Microsoft Visual Studio and tested on a system equipped with an Intel Core i7-4500U processor and 8 GB of RAM—hardware representative of typical consumer-grade laptops of that time, yet sufficient to support real-time fractal generation. The application is user-interactive and designed for fractal generation and rendering, featuring a user-friendly graphical interface with intuitively organised windows that allow easy selection, configuration, and modification of parameters. Despite the computational demands typically associated with fractals, the system delivers fast, high-quality rendering with real-time performance, enabling continuous interaction and immediate visual feedback essential for exploratory and design-oriented applications.

Fractal generation is based on the random iteration algorithm, while fractal modelling and manipulation rely on the theoretical framework presented in Section 2. Prior to generating a two-dimensional IFS fractal, the user defines three noncolinear control points A, B, and C, by pinpointing them on the main application window (see Figure 1). These three points form a triangle - an affine basis of $\mathbb{R}^2$. Each point of the fractal image is then assigned barycentric coordinates with respect to this basis, and these coordinates remain invariant throughout the modelling process. Any affine transformation of the form (2.5) applied to the control points—performed through simple click-and-drag interactions, one point at a time — induces the same affine transformation of the entire fractal image in real time. This approach is not only interactive but also allows users to anticipate the effect of a modification before it is applied, thereby achieving a predictable modelling experience.

A time complexity analysis was conducted in [16], and it was established that the running time of the algorithm is linearly bounded by the number of points composing the fractal. Experimental evaluations showed that both fractal generation and subsequent affine transformations are executed almost instantaneously, with execution times not exceeding 6 ms (on the AMD Ryzen 3 5425U with Radeon Graphics processor with 16 GB of RAM). Based on these results, the Fractal Engine application is considered an efficient solution for real-time fractal generation and affine manipulation.

Fractal Engine includes a wide range of tools for both fractal generation and interactive modelling:

- Manually defining a custom IFS code or selecting one from a library of predefined templates;

- Modifying individual parameters within the IFS code to explore variations of the fractal attractor;
- Adding IFS codes to the internal library, thereby extending the collection of predefined fractals;
- Gradually varying one or more IFS parameters to observe continuous transformations of the attractor, enabling animation-like effects;
- Customising visual appearance by colouring the fractal, the background, or the control triangle;
- Applying precise affine transformations (defined by formula) as well as selected nonlinear transformations;
- Saving generated fractals in various image formats for further post-processing;
- Exporting and importing IFS parameter files;
- Performing fast basic transformations using hotkeys;
- Automatically computing a "minimal" enclosing triangle[2] and rectangle[3] for the fractal attractor for improved geometric control.

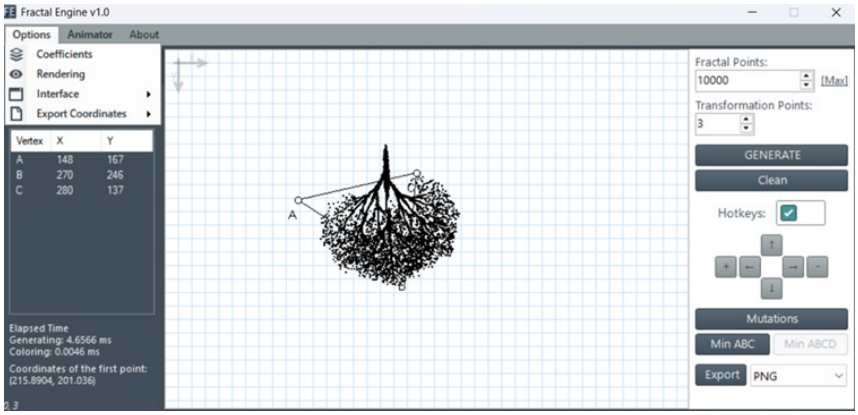


FIGURE 1. The main application window – fractal generation module. In the lower left corner, the execution time is noted, 4.6566*ms*. The number of points that compose the fractal is limited to 10000, while the number of controlling points (affine basis points) is 3. The precise location of the controlling points is given in the left middle part of this window.

We will illustrate the stated modules and functionalities through representative screenshots and examples. The natural-like fractals illustrated on the figures are given below with their IFS codes.

[2]This "minimal" triangle is actually a rectangular isosceles triangle with minimal area, whose catheti are parallel to the coordinate axes.

[3]This "minimal" rectangle is the minimal by area rectangle, whose sides are parallel to the axes.

a) *Maple tree*, defined by $S = \{\mathbb{R}^2; w_1,\ w_2, w_3, w_4\}$, where

$$w_1\left([x\ \ y]^T\right) = \begin{bmatrix} -0.04 & 0 \\ -0.19 & -0.47 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -0.12 \\ 0.30 \end{bmatrix};$$

$$w_2\left([x\ \ y]^T\right) = \begin{bmatrix} 0.65 & 0 \\ 0 & 0.56 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.06 \\ 1.56 \end{bmatrix};$$

$$w_3\left([x\ \ y]^T\right) = \begin{bmatrix} 0.41 & 0.46 \\ -0.39 & 0.61 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.46 \\ 0.40 \end{bmatrix};$$

$$w_4\left([x\ \ y]^T\right) = \begin{bmatrix} 0.52 & -0.35 \\ 0.25 & 0.74 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -0.48 \\ 0.38 \end{bmatrix};$$

and equal probabilities, $p_1 = p_2 = p_3 = p_4 = 0.25$.

b) *Barnsley fern*, defined by $S = \{\mathbb{R}^2; w_1,\ w_2, w_3, w_4\}$, where

$$w_1\left([x\ \ y]^T\right) = \begin{bmatrix} -0.03 & 0.31 \\ 0.35 & 0.01 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.24 \\ -2.20 \end{bmatrix};$$

$$w_2\left([x\ \ y]^T\right) = \begin{bmatrix} -0.03 & 0.01 \\ -0.31 & 0.40 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -0.83 \\ -2.36 \end{bmatrix};$$

$$w_3\left([x\ \ y]^T\right) = \begin{bmatrix} 0.78 & 0.08 \\ -0.10 & 0.78 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.06 \\ 0.73 \end{bmatrix};$$

$$w_4\left([x\ \ y]^T\right) = \begin{bmatrix} 0.08 & -0.26 \\ 0.34 & 0.15 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -1.72 \\ -2.25 \end{bmatrix};$$

and equal probabilities, $p_1 = p_2 = p_3 = p_4 = 0.25$.

c) *Seahorse*, defined by $S = \{\mathbb{R}^2; w_1,\ w_2\}$, where

$$w_1\left([x\ \ y]^T\right) = \begin{bmatrix} 0.824074 & 0.281482 \\ -0.2123469 & 0.864198 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -1.88229 \\ -0.110607 \end{bmatrix};$$

$$w_2\left([x\ \ y]^T\right) = \begin{bmatrix} 0.088272 & 0.520988 \\ -0.463889 & -0.377778 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.78536 \\ 8.095795 \end{bmatrix};$$

and equal probabilities, $p_1 = p_2 = 0.5$.

d) *Maple leaf*, defined by $S = \{\mathbb{R}^2; w_1,\ w_2, w_3, w_4\}$, where

$$w_1\left([x\ \ y]^T\right) = \begin{bmatrix} 0.14 & 0.01 \\ 0 & 0.51 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -0.08 \\ -1.31 \end{bmatrix};$$

$$w_2\left([x\ \ y]^T\right) = \begin{bmatrix} 0.43 & 0.52 \\ -0.45 & 0.50 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1.49 \\ -0.75 \end{bmatrix};$$

$$w_3\left([x\ \ y]^T\right) = \begin{bmatrix} 0.45 & -0.49 \\ 0.47 & 0.47 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -1.62 \\ -0.74 \end{bmatrix};$$

$$w_4\left([x\ \ y]^T\right) = \begin{bmatrix} 0.49 & 0 \\ 0 & 0.51 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.02 \\ 1.62 \end{bmatrix};$$

and the corresponding probabilities are $p_1 = 0.01$, $p_2 = p_3 = 0.35$, $p_4 = 0.29$.

The featured examples are excellent representatives of fractals that closely approximate natural forms, which can be later applied in biomimetics, regenerative design, or nature-based solutions in architecture design. Unfortunately, there are few such IFS code examples available (commonly of leaves and trees), despite the existence of a highly precise theorem—the so-called Collage Theorem ([2, 6]) — that enables finding IFS codes for any given image. This theorem has a rigorous mathematical proof but remains underutilized in practice, likely due to inherent limitations in practical application. Since fractal geometry has proven particularly effective in describing structures observed "everywhere" ([2], [9], [21], [25]), working on creation of IFS codes will be of particular interest of designers.

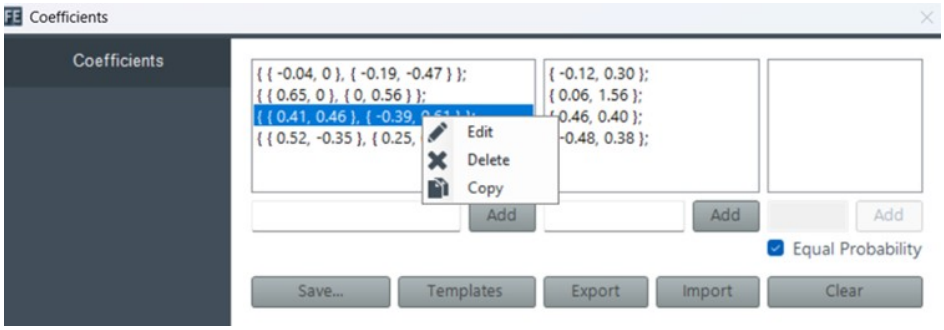


FIGURE 2. The coefficient window – the user can edit, delete or copy a row; export or import an IFS code or use the list of templates. The user can also save an IFS code in the list of templates.

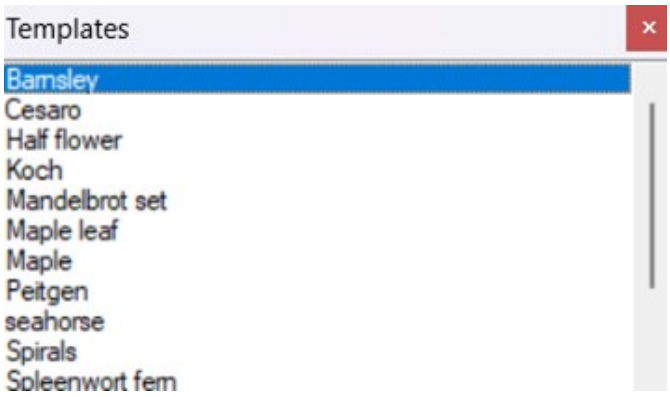


FIGURE 3. List of fractal templates.

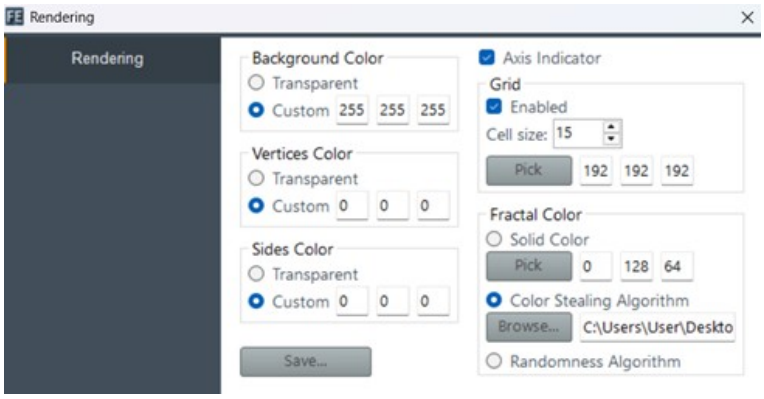


FIGURE 4. The rendering window. There are possibilities for colouring the background, the vertices and sides of the controlling polygon, the grid, and the fractal (three different colouring options for the fractal).

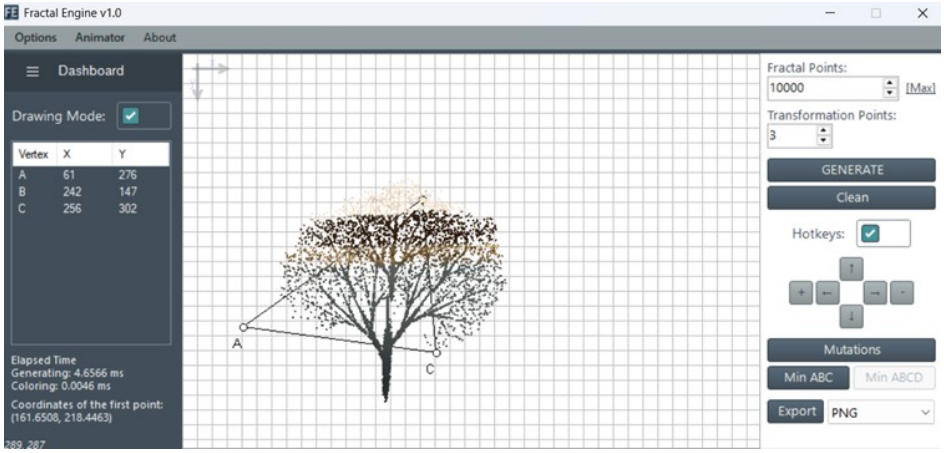


FIGURE 5. The Peitgen tree fractal, modelled by Fractal Engine interactive possibilities and coloured by the colour stealing algorithm (the fractal points are coloured by "stealing" the colours from the selected image shown on the bottom right corner in Figure 4. The modelling is performed by clicking and dragging the controlling points, which enables predictable and interactive modelling of the fractal. The new locations of the controlling points are shown in the left-middle part.

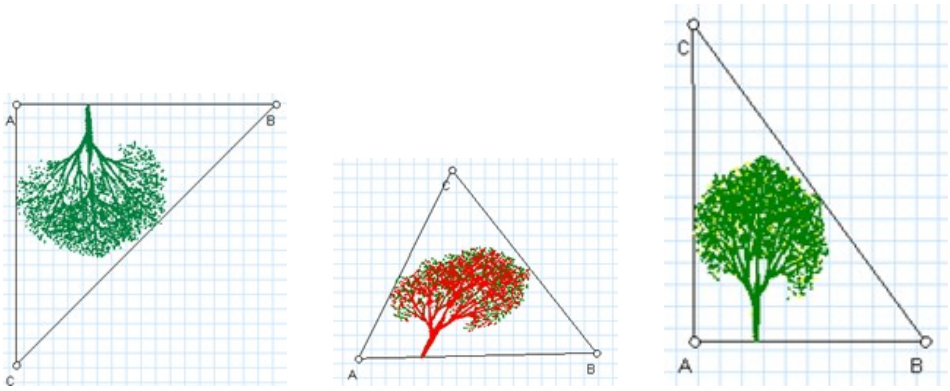


FIGURE 6. Controlling by "minimal" triangle that contains the fractal ensures that the modifying transformations will be contractive and images under these transformations will remain in the convex hull of the triangle. Not only a triangle, but also a "minimal' rectangle can be computed and visualised by Fractal Engine. The possibility for randomly picking the colour of the fractal points (the bottom-right option in the rendering window, see Figure 4, is shown in the middle and right images.

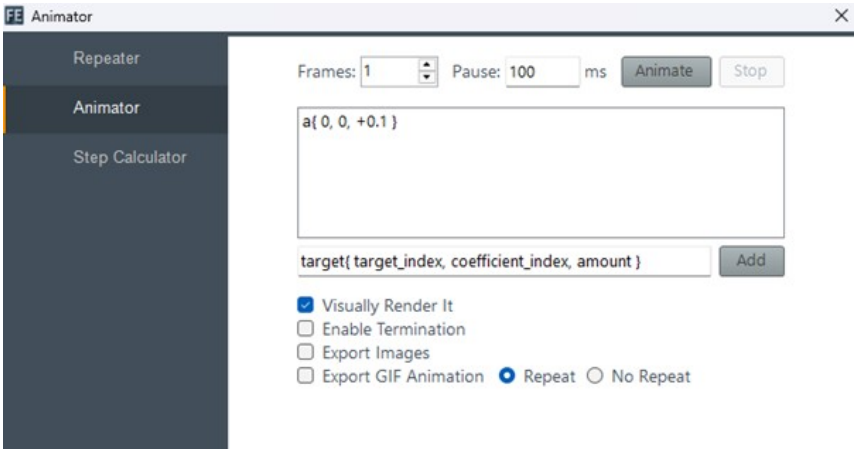


FIGURE 7. Animator window includes three options: 1) Repeater – enables repeating the fractal generations for observing the randomness and convergence of the RIA; 2) Animator, enables: visual rendering of fractals obtained by predetermined gradual change of coefficient(s), number of frames and length of the break between consecutive frames, termination of the process, exporting images and GIF animation; and 3) Step calculator (presented in Figure 8).

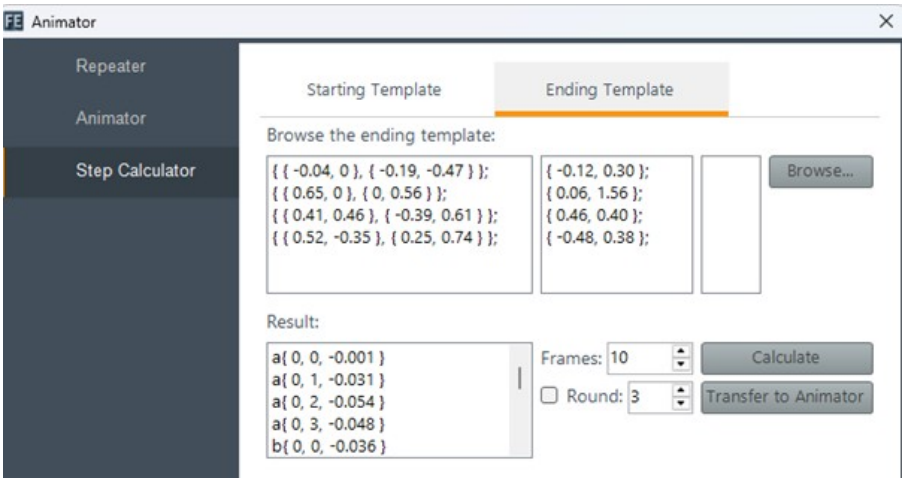


FIGURE 8. Step calculator – enables blending between two fractal sets and exporting as images or animation.

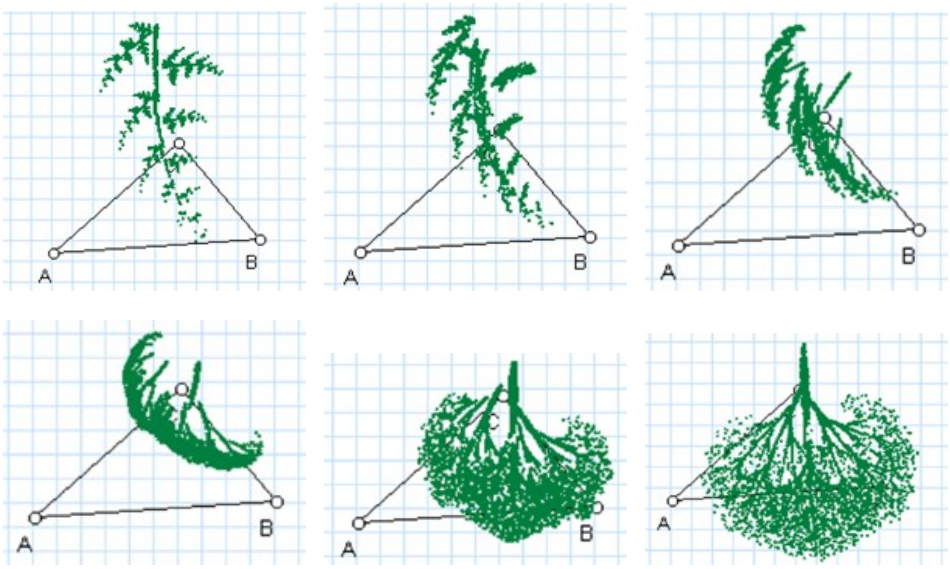


FIGURE 9. The result of blending between Barnsley fern and Maple tree fractals. The blending is done with 10 frames. We are showing here the first, fourth, sixth, ninth, and tenth iterations.

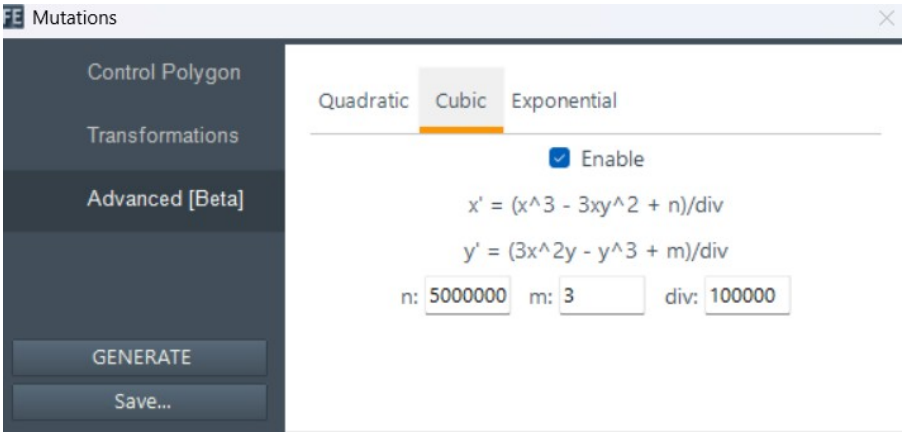


FIGURE 10. Mutation window. Enables precisely defined transformations: translation, scaling, reflection, sharing, rotation, as well as advanced transformations: quadratic, cubic and exponential.

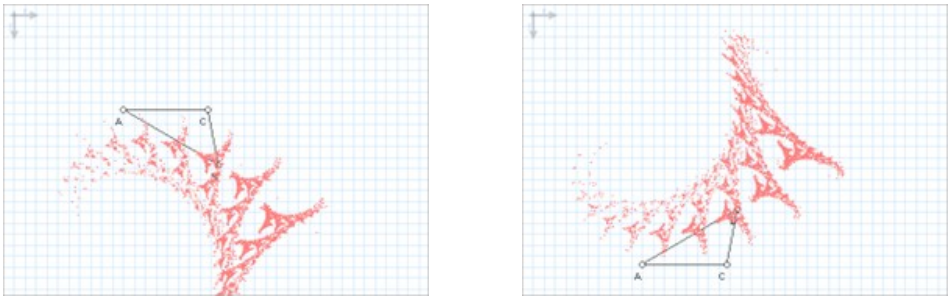


FIGURE 11. Precisely defined reflection about a line parallel with the x-axis by means of "Transformation" functionality, which can be observed in Figure 10.

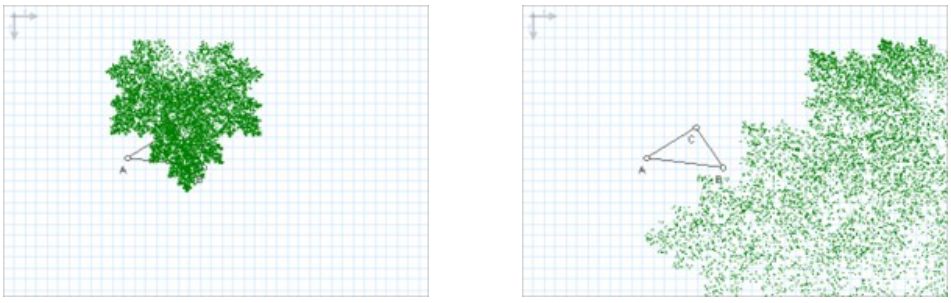


FIGURE 12. Maple leaf on the left and its cubic transformation on the right as defined in Figure 10.

4. Conclusions and future work

The Fractal Engine software application enables generating fractals (by random iteration algorithm) and their interactive and predictable modelling. Instead of modifying the IFS's matrices and translation vectors, the user interacts with the fractal through predefined control points that define the affine basis. When the affine basis is modified (by clicking and dragging one point at a time or by formula), the entire fractal follows the transformation. This interaction mechanism enables intuitive, predictable transformations of the fractal attractor, while preserving the mathematical consistency of the underlying IFS. The software also provides functionalities such as defining new and modifying existing IFS codes, applying colouring schemes, blending two fractals, exploring the effect of gradual changes in coefficient(s) on the fractal, and exporting generated images and animations for further processing. Several examples of naturally inspired fractals (tree, leaf, fern, seahorse) are presented to demonstrate the approach's flexibility and effectiveness, as well as its suitability for exploratory modelling. Future work may focus on improving existing functionalities and extending the capabilities of the proposed framework in several directions: improving the colouring scheme, implementing filters and other image manipulation techniques, enabling interactive continuous zoom, and adding more methods for fractal local and global transformation and control. An extension toward three-dimensional fractals would be of great importance. The possibility of integration with an existing web application for generating and modelling 3D IFS fractals, so-called Fractal Transformer (https://srumi9.github.io/FractalTransformer/), presented in [26], should also be considered. Last but not least, given the current lack of IFS codes capable of generating realistic natural-like fractals, the integration of artificial intelligence techniques may offer a promising direction for constructing such codes, based on the Collage Theorem approach. The development of these codes remains an important challenge for future research, where human expertise could be effectively supported by AI-driven methods. In contrast, the modelling of fractals appears to require primarily further refinement and integration of existing approaches, rather than fundamentally new techniques.

Acknowledgements

I thank Jovan Petkovski for several discussions on Fractal Engine software.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work, the author used Grammarly and ChatGPT to correct grammatical and syntactical errors, improving the study's correctness and readability. After using these tools/services, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

REFERENCES

[1] Banisch, S., Sbert, M., *Fast visualisation and interactive design of deterministic fractals*, in: CAe, Eurographics Association, Lisbon, Portugal, 2008, 17–24.

[2] Barnsley, M. F., *Fractals everywhere*, Academic Press, Inc., 1988.

[3] Barnsley, M. F., *Transformations between fractals*, Progress in Probability, 61, 2009, 227–250.

[4] Barnsley, M. F., Harding, B., *Fractal transformations in 2 and 3 dimensions*, arXiv:1308.6648v1, 2013.

[5] Barnsley, M. F., Harding, B., Igudesman, K., *How to transform and filter images using iterated function systems*, SIAM Journal on Imaging Sciences, 4 (4), 2011, 1001–1028.

[6] Barnsley, M. F., Sloan, D. A., *A better way to compress images*, Byte Magazine, 1988, 215–223.

[7] Barnsley, M. F., Vince, A., *The chaos game on a general iterated function system*, Ergodic Theory and Dynamical Systems, 31 (4), 2011, 1073–1079.

[8] Barnsley, M. F., Demko, S., *Iterated function systems and the global construction of fractals*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 399 (1817), 1985, 243–275.

[9] Borda-de-Água, L., *A brief review on fractals, their applications and implications*, New Design Ideas, 4 (2), 2020, 69–84.

[10] Chang, H. T., *Arbitrary affine transformations and their composition effects for two-dimensional fractal sets*, Image and Vision Computing, 22 (13), 2004, 1117–1127.

[11] Darmanto, T., Suwardi, I. S., Munir, R., *Animation model of multi-object in fractal form based on partitioned-random iteration algorithm*, Procedia Technology, 11, 2013, 93–98.

[12] Falconer, K. J., *Fractal geometry: Mathematical foundations and applications*, John Wiley & Sons Ltd, 1990.

[13] Hadzieva, E., *Exact solution of an inverse problem for affine images of IFS attractors*, Fixed Point Theory Algorithms Sci Eng 2026, 16 (2026). https://doi.org/10.1186/s13663-026-00834-8.

[14] Hadzieva, E., Petkoski, J., *A novel software application for interactive affine transformations of fractals*, International Journal of Mathematical Models and Methods in Applied Sciences, 10, 2016, 134–142.

[15] Hadzieva, E., Shuminoska, M., *Real-time tool for affine transformations of two-dimensional IFS fractals*, Journal of Electrical Engineering, 4 (3), 2016, 150–155.

[16] Hadzieva, E., Grupcev, V., Petkoski, J., *Extension and software analysis of an interactive application for transforming fractals*, European Journal for Information Science and Technology, 2017.

[17] Hadzieva, E., Petkoski, J., *An interactive application for modeling two-dimensional IFS fractals*, in: Proceedings of the 2015 International Conference on Mathematics and Computers in Sciences and Industry, Sliema, Malta, (MCSI), IEEE, 2015, 49–54.

[18] Hutchinson, J. E., *Fractals and self-similarity*, Indiana University Mathematics Journal, 30 (5), 1981.

[19] Kocić, L. M., Simoncelli, A. C., *AIFS: A tool for modeling fractal images*, in: Complexity and Evolution in the Living World, SICC III International Conference, Rome, Italy, Rome, 1998.

[20] Kocić, L. M., Simoncelli, A. C., *Shape predictable IFS representations*, Emergent Nature, 2002, 435–436.

[21] Mandelbrot, B. B., *The fractal geometry of nature*, W. H. Freeman, 1982.

[22] Nikiel, S., *Iterated function systems for real-time image synthesis*, Springer, 2007.

[23] Peitgen, H.-O., Jürgens, H., Saupe, D., *Fractals for the classroom: Part one – Introduction to fractals and chaos*, Springer, 1992a.

[24] Peitgen, H.-O., Jürgens, H., Saupe, D., *Chaos and fractals: New frontiers of science*, Springer, 1992b.

[25] Sala, N., *Fractal geometry and superformula to model natural shapes*, International Journal of Research and Reviews in Applied Sciences, 16 (4), 2013, 78–92.

[26] Shuminoska, M., Hadzieva, E., Celakoska, E., *A novel web application for generating and modeling 3D fractals*, Egyptian Computer Science Journal, 42 (1), 2018, 43–56.

[27] Voss, R. F., *Fractals in nature: From characterization to simulation*, in: H.-O. Peitgen, D. Saupe (eds.), *The science of fractal images*, Springer, 1988.

ELENA HADZIEVA
UNIVERSITY OF INFORMATION SCIENCE AND TECHNOLOGY "ST. PAUL THE APOSTLE",
FACULTY OF INFORMATION SYSTEMS, VISUALIZATION, MULTIMEDIA AND ANIMATION,
PARTIZANSKA BB, OHRID, N. MACEDONIA
Email address: elena.hadzieva@uist.edu.mk

Received 06.05.2026
Revised 24.09.2026
Accepted 06.10.2026