A novel software application for interactive affine transformations of fractals

Elena Hadzieva†, Jovan Petkoski

***Abstract*—**The paper presents a novel interactive application created in the C# programming language under the Microsoft Visual Studio environment: its introduction, theoretical grounds and application. The application allows generating two-dimensional fractals defined by iterated function systems and performing their arbitrary affine transformation. Most importantly, this transformations are easily controllable and visible, done by dragging and dropping three predefined non-collinear points, exploiting the full potential of high quality fractal rendering in real-time. The affine transformations are enabled by assigning barycentric coordinates to the points that represent the image of the fractal attractor with respect to the three above mentioned points and thus relating the fractal to the points. Besides this main purpose of our application, many other features are offered: modifying specific parameters in the IFS code, gradually changing one or more parameters in the IFS code to follow the continuous transformation of the fractal attractor as a form of animation, coloring fractals, saving files in different image formats for post-processing, exporting and importing IFS parameter files and performing fast basic transformations to the fractals with the hotkeys option. Also, the application allows finding vertices of a kind of triangle with minimal area that contains the fractal, which form an affine basis. The practical use of such affine basis in modeling particular fractal is that the fractal is in the convex hull of the last affine basis, which makes the modeling more governable.

***Index Terms*—**iterated function system, fractal, affine transformations, barycentric coordinates, interactive application.

I. Introduction

Fractals are very convenient geometrical 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 or transformation 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 elements in the IFS code might result in unpredictable changes in the shape of the fractal model. When the IFS is hyperbolic, continuous change of the fractal attractor, by continuous change of the coefficients in the IFS code is guaranteed ([2], p.111-117), but this not always yields to predictable modification or transformation of the fractal attractor. Obtaining the desired transformation of the fractal often includes many trials end errors. Later we give an example where the continuous change of one of the coefficients in the IFS code will lead to completely new form of the fractal attractor (it is shown on Figure 11). However, 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 whose main feature is enabling interactive affine transformations of any two-dimensional IFS fractal, but it also offers other options related to visualization and manipulation of two-dimensional IFS attractors. The key point and main novelty in our application is the representation of the points that constitute the fractal image by barycentric coordinates with respect to a predefined three pointwise independent points. This work is extension of the previous work of the authors, [16], [15].

The remainder of this paper is arranged as follows. Section II 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 III. Section IV provides description of the application and contains examples that illustrate its practical value. The conclusions and future work are given in Section V.

II. Related work

Fractals inspired many scientists, both theoreticians and practitioners, as well as artists, to examine, analyze and apply fractals in many fields ([2], [12], [18], [24]). The interest for finding optimal way for different transformations and modifications of fractals is motivated by their definition (will be explained in details in the subsection III-B), that says that fractals are *fixed* points of particular operator. Below we discuss about few articles that cover the transformations of fractals.

In [9] the authors make animation of fractal objects through interactive changes in the IFS code of the fractal. The same authors in [8] 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.

† Author to whom all correspondence should be sent.

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

Jovan Petkoski is a student at the Faculty of Computer Science and Engineering, University of Information Science and Technology "St. Paul the Apostle", Partizanska bb, 6000 Ohrid, Macedonia. Email: jovan.petkoski@cse.uist.edu.mk

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]–[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 [19], [20] 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, giving good theoretical (see [1]) and practical support ([21]) of the idea.

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 [7], [8], [9]); there is no new generation of the transformed fractal. The only thing that is changing is the location of the three pointwise independent points, which is followed by the appropriate change of the location of the points that represent the fractal. Also, our application is more advanced, because the transformation is visible and done in real time by clicking and dragging the three predefined points (one at a time). The fractal immediately follows the affine transformation defined by old and new position of the displaced vertex. The application also provides possibility for animation of objects by changing any coefficient in the code, in any reasonable interval, for any reasonable step-size (here, reasonable refers to interval or step-size for which the recursive sequence that forms the fractal image, converges). All features of the 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 [22], but they can be also found in other references, like [13] 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 $\overrightarrow{A_1A_k}$, $k = 2, 3, \ldots, N$, 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

$$P = \sum_{i=1}^{N} a_i A_i. \tag{1}$$

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

Remark: The equation (1) might be considered as a vector equation. Namely, if the points are given with their rectangular coordinates, $P = (p_1, p_2)$ and $A_i = (a_{i1}, a_{i2})$, for $i = 1, 2, \ldots, N$, then the equation (1) takes the form:

$$\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \sum_{i=1}^{N} a_i \begin{bmatrix} a_{i1} \\ a_{i2} \end{bmatrix} \tag{2}$$

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, is a pointwise independent set in $\mathbb{R}^n$, for $n \geq 2$, 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], [17], [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

$$w(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} + \mathbf{b}, \tag{3}$$

where $\mathbf{A}$ and $\mathbf{b}$ 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 $(\mathscr{H}(\mathbb{R}^2), h(d_E))$ 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:

$$h(d_E)(A, B) = \max\{d_E(A, B), d_E(B, A)\},$$

where

$$d_E(A, B) = \max_{a \in A} \min_{b \in B} d_E(a, b),$$

$$\text{for} \quad A, B \in \mathscr{H}(\mathbb{R}^2).$$

It can be shown that $(\mathscr{H}(\mathbb{R}^2), h(d_E))$ 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 $W : \mathscr{H}(\mathbb{R}^2) \to \mathscr{H}(\mathbb{R}^2)$, defined by

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

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 $W(A) = A = \lim_{n\to\infty} W^n(B)$, for any $B \in \mathscr{H}(\mathbb{R}^2)$. 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 (those are the attractors that have fine structure, with strict or approximate self-similarity, that are too irregular to be described by the tools of Euclidean geometry and usually their fractal dimension is greater then its topological dimension, see [**?**]).

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 $\{p_1, p_2, \ldots, p_n\}$, $p_1 + p_2 + \ldots + p_n = 1$ ([2]). Each probability p_i is associated with the affine transformation w_i. Having the **IFS with probabilities** defined, the initial point $\mathbf{x}_0$ is chosen arbitrarily, and then recursively and independently, the sequence $\{\mathbf{x}_n, n = 1, 2, 3, \ldots\}$ is generated, such that the event $\mathbf{x}_n = w_i(\mathbf{x}_{n-1})$ has the probability p_i. Under certain conditions (more information could be found in [2], [3]), the sequence $\{\mathbf{x}_n\}_{n=0}^{\infty}$ 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 $P \in \mathbb{R}^2$ with rectangular coordinates (x, y), according to (2) they are connected with the relation

$$\begin{bmatrix} x \\ y \end{bmatrix} = a \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + b \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + c \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \qquad (4)$$

where (a, b, c) are barycentric coordinates of P with respect to the points A, B, C. The relation (4) represents conversion from barycentric to rectangular coordinates. If we rearrange the equation (4), we will obtain the following form of a system in two unknowns a and b (we used the relation $c = 1 - a - b$):

$$\begin{aligned} (a_1 - c_1)a + (b_1 - c_1)b &= x - c_1 \\ (a_2 - c_2)a + (b_2 - c_2)b &= y - c_2\,. \end{aligned}$$

The last system has a unique solution because the determinant of its coefficient matrix is nonzero, due to the fact that A, B and C are non-collinear points. It is given with

$$\begin{aligned} a &= \frac{(x - c_1)(b_2 - c_2) - (b_1 - c_1)(y - c_2)}{(a_1 - c_1)(b_2 - c_2) - (b_1 - c_1)(a_2 - c_2)} \\ b &= \frac{(y - c_2)(a_1 - c_1) - (x - c_1)(a_2 - c_2)}{(a_1 - c_1)(b_2 - c_2) - (b_1 - c_1)(a_2 - c_2)}, \end{aligned}$$

and together with the relation $c = 1 - a - b$ represents the conversion from rectangular to barycentric coordinates 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 (3) mapping one simplex to the other. If the point Q has barycentric coordinates (a, b, c), with respect to the points A', B', C', the same as the barycentric coordinates of P with respect to the points A, B, C, it can be shown that exactly $Q = f(P)$ ([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. Fractal Engine - 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 8GBs 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 points 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, by their barycentric coordinates, 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

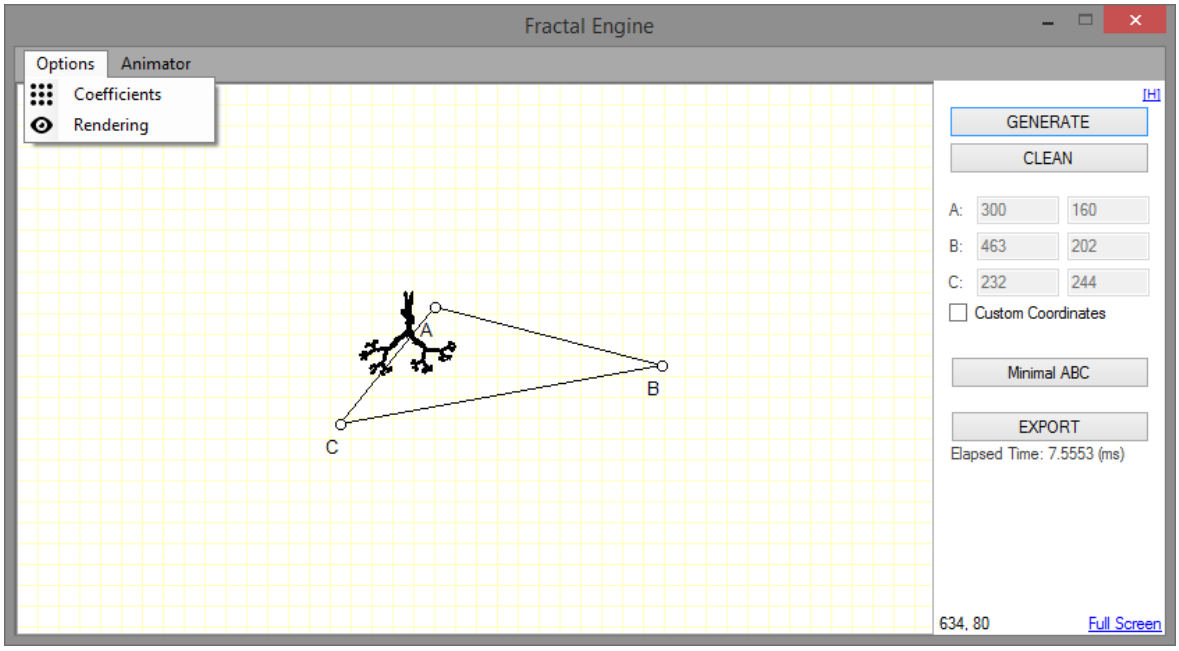

Fig. 1. The main application window.

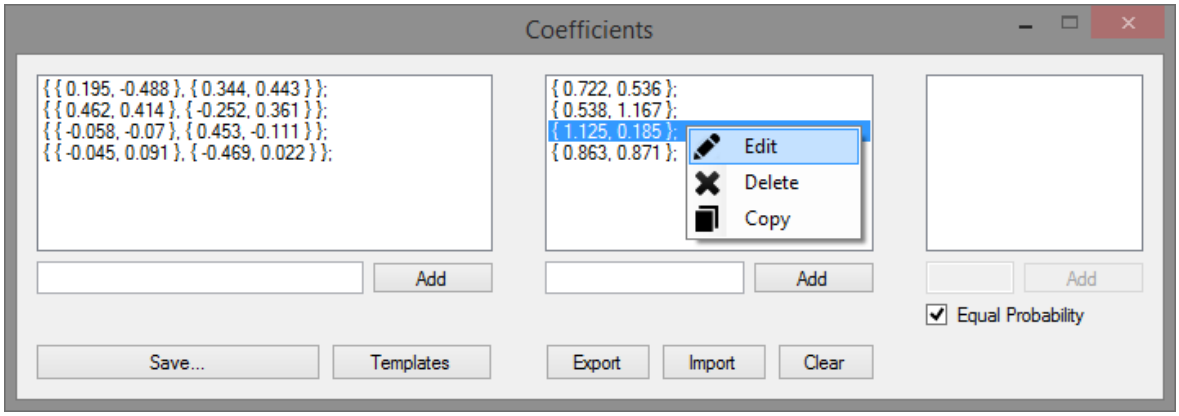

Fig. 2. The coefficient window.

at a time. And also, there is another feature accessible from the main window - switching the program to full screen mode and back.

We would like to note that there is another way of defining the control triangle - with the custom coordinates window, see Figure 7 (this window appears when ticking the "custom coordinate" check box on the main window), where the coordinates of n points need to be manually specified. Also, the selection type must be indicated, depending on which points from the list would be used for constructing the control triangle (first three or random three points from the previously specified list).

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 $w_i : \mathbf{x} \mapsto \mathbf{A}_i\mathbf{x} + \mathbf{b}_i$, $i = 1, 2, 3, 4$, 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 (Figure 3), deleted, copied or completed with probabilities. 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, a list of embedded templates of known IFS codes (Figure 4) and a possibility to export/import all this coefficients data into/from a parameter file, allowing the user to easily return to previously created fractals for later exploration, or it could be helpful as a defensive mechanism against a program or system crash.

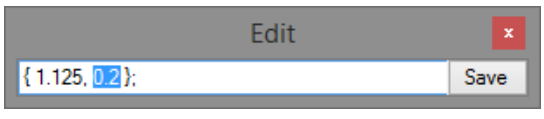

Fig. 3. Editing a particular transformation.

The button MINIMAL ABC (Figure 1) gives the right-angle isosceles triangle with its catheti parallel to the axes,

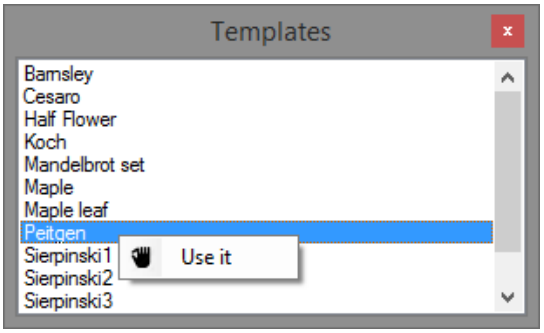

Fig. 4. List of included IFS code templates. Each item could be used with a right click.

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 transformations of a generated Peitgen tree fractal with the use of an arbitrary triangle and the "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 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.

The disadvantage of such free modeling is that if exactly defined affine transformation of the fractal is needed, it will be very difficult to achieve it by clicking and dragging. Because of this reason, we included "Basic transformation" button, which includes the following basic 2D affine transformations ([23]), that could affect the fractal, the triangle or both of them (see Figure 7):

- Translation, defined by
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix},$$
where T_x and T_y are x-axis and y-axis translations, respectively. In the window, the user only fills the values for T_x and T_y in the fields denoted by X and Y.
- Scaling relative to a point (x_p, y_p), defined by
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_p(1-S_x) \\ y_p(1-S_y) \end{bmatrix},$$
where S_x and S_y are x-axis and y-axis scaling factors, respectively. Here we added an option: taking centroid as a reference point.
- Reflection about a line parallel with the x-axis, $y = y_p$:
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 2y_p \end{bmatrix},$$
and reflection about a line parallel with the y-axis, $x = x_p$:
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 2x_p \\ 0 \end{bmatrix}.$$
In the field "Arbitrary line" the values x_p or y_p should be entered, depending whether the line is parallel to y- or x-axis.
- Shearing along the x-axis, defined by
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \tan\alpha \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix},$$
and shearing along the y-axis, defined by
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \tan\alpha & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix},$$
where α is the shear angle. It can be defined by the user in the appropriate text field.
- Rotation about the point (x_p, y_p), defined by
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_p(1-\cos\alpha) + y_p\sin\alpha \\ y_p(1-\cos\alpha) - x_p\sin\alpha \end{bmatrix},$$
where α is the angle of rotation. The user can define the point and angle of rotation. Also, he could select special points of rotation, origin and centroid.

Hotkeys is an additional feature included in the coordinates window (Figure 7) which assists the user to visually and interactively control all basic transformations using the numeric keypad of his keyboard. Actions like translation to any part of the screen, infinite zoom, shearing and rotation through a particular angle (clockwise or counterclockwise) around a particular point are now easy to perform and observe.

Figure 8 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 9. In part a) of Figure 9 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.

A window form with the animation settings is shown in Figure 10. 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 in the IFS code for a particular number of frames. Using zero-starting numbering, in the above case (Figure 10), the command $a\{0, 0, +0.1\}$ means that the element with index 0 in the 0th line from the first matrix given on Figure 2 (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 11. The whole animation concept is made possible with the use of multithreading programming technique. Animated peitgen tree with the command $a\{0, 0, +0.1\}$ after 15 frames is shown in Figure 11, part f).

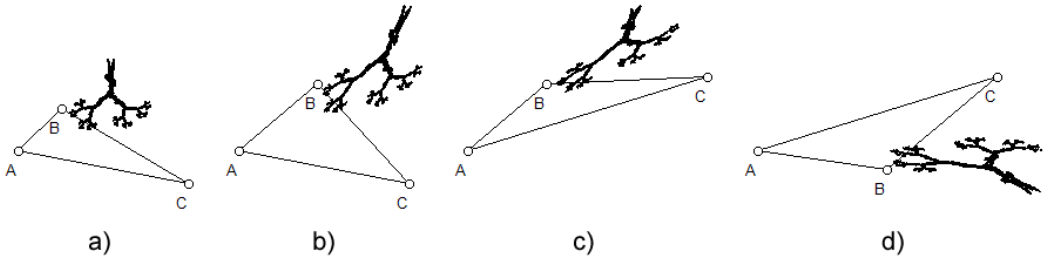

Fig. 5. Modeling fractal by means of arbitrary triangle. a) The fractal is generated after the on-click definition of the points A, B, C. The fractal follows the transformation of the triangle when the points B (b)), C (c)), and at the end B (d)) are moved.

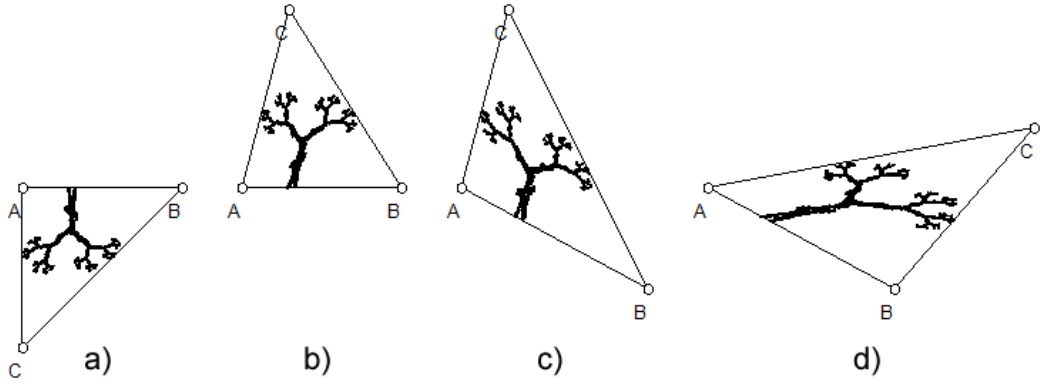

Fig. 6. Modeling fractal by means of a special triangle with minimal area that contains the fractal. a) The "minimal" triangle obtained after pressing the button MINIMAL ABC. The fractal follows the transformation of the triangle when the points C (b)), B (c)), and at the end C (d)) are moved.

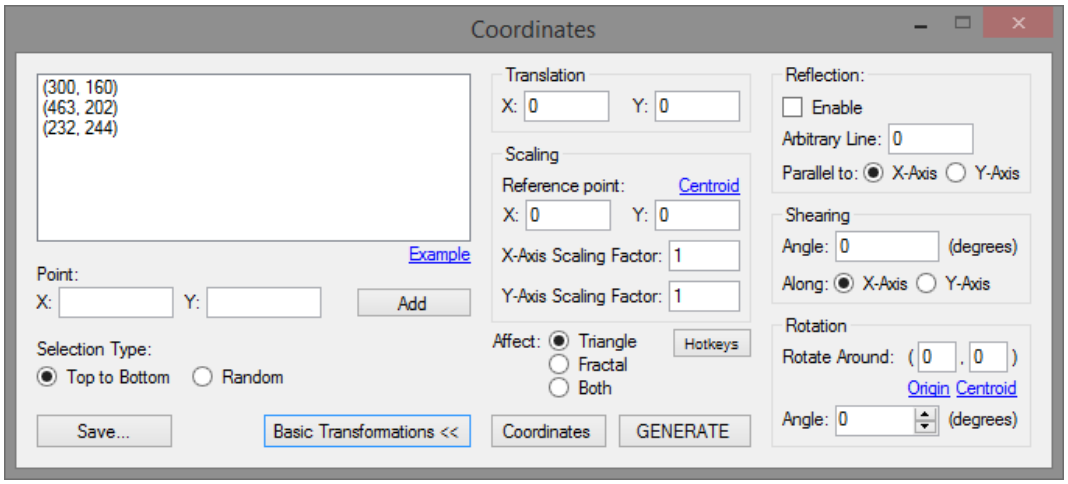

Fig. 7. The custom coordinates window.

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.

The amount of time needed by the application to visually display a defined fractal on the screen depends on the hardware configuration (primarily CPU and RAM) and this value is shown in the main window (calculated in milliseconds). A series of ten consecutive intervals for generating the same Peitgen tree on our previously stated system configuration is presented in Figure 12.

Additional data taken from Task Manager about the process itself is shown in Figure 13. The first line describes our application while it is idle and the second one while the affine transformations are performed by dragging the vertices. This modeling is done in real time and no lag appeared on our system. We could say that the application is relatively fast and requires relatively low memory.

V. Conclusion

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,

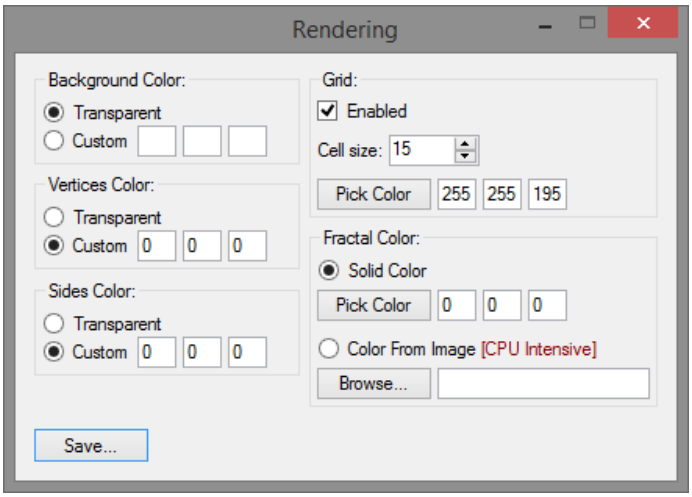

Fig. 8. A window where all rendering features could be set up in real time.

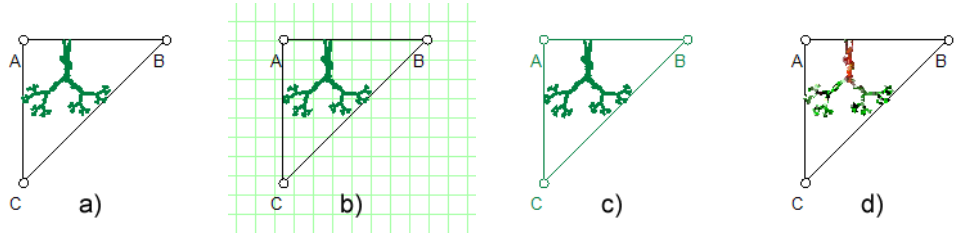

Fig. 9. Coloring possibilities.

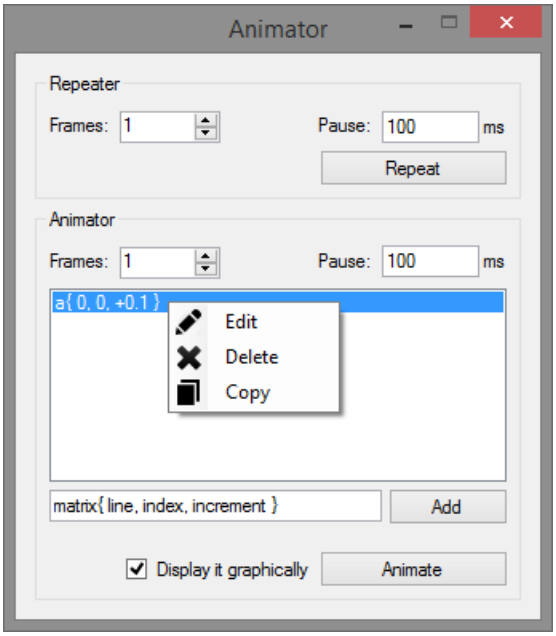

Fig. 10. Animation window.

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 affinly 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 including larger choice of color rendering algorithms; use of filters and other image manipulation techniques; exporting an entire animation file;

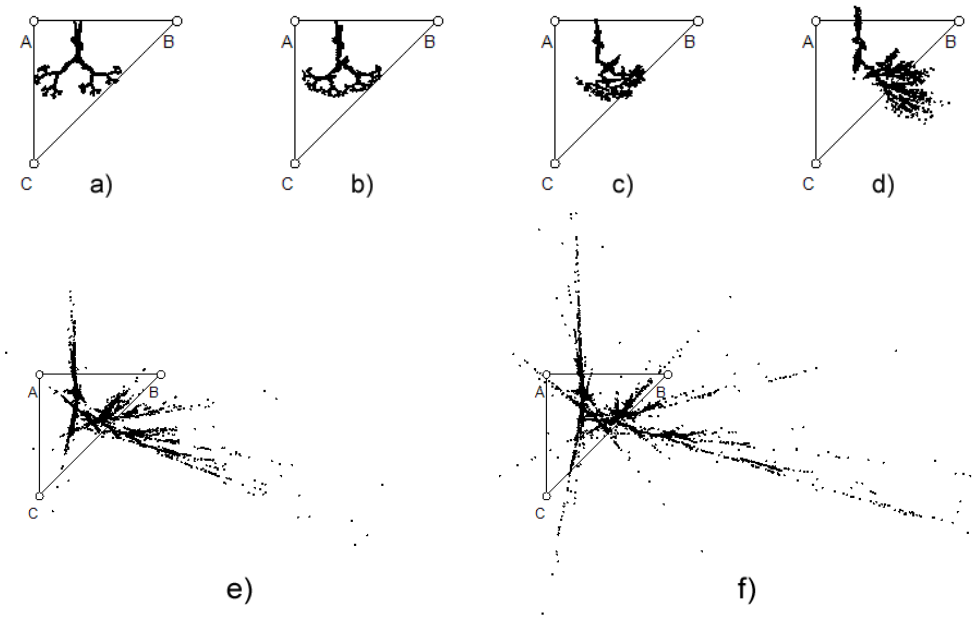

Fig. 11. Animated Peitgen tree.

```
11.5462 (ms)
08.6937 (ms)
08.8049 (ms)
13.0768 (ms)
09.5559 (ms)
12.1175 (ms)
08.5637 (ms)
07.8917 (ms)
08.7002 (ms)
06.5197 (ms)
Average: 09.5470 (ms)
```

Fig. 12. Estimated time for generating Peitgen tree 10 times.

Name	CPU	Working set (memory)	Memory (private working set)	Memory (shared working set)	Page faults	Handles	Threads
Fractal Engine.exe	00	25,552 K	9,612 K	15,940 K	111,831	209	4
Fractal Engine.exe	18	49,824 K	32,364 K	17,460 K	248,578	217	7

Fig. 13. Data taken from the Task Manager.

interactive continuous zoom and more methods for fractal transformation and control, like controlling with more than three points, transforming only part of the fractal (achieving local control), performing nonlinear transformations over the fractal or developing an application for modeling 3D fractals.

ACKNOWLEDGMENT

The authors would like to thank to Marija Shuminoska, Vladimir Grupcev and Angela Angeleska for their time and advice in the preparation of this paper.

REFERENCES

[1] E. Babace, Lj. M. Kocić. "Minimal Simplex for IFS Fractal Sets", *Lecture Notes in Computer Science* Volume 5434, pp.168-175, Eds.: S. Margenov, L. G. Vulkov, J. Wasniewski, Springer-Verlag Berlin Heidelberg, 2009.

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

[3] M. F. Barnsley, A Vince. "The Chaos Game on a General Iterated Function System", *Ergodic Theory Dynam. Systems* 31, no. 4, 1073-1079, 2011.

[4] M. F. Barnsley, B. Harding. "Fractal Transformations in 2 and 3 Dimensions", arXiv:1308.6648v1, 30 Aug 2013.

[5] M. F. Barnsley, T. Harding, K. Igudesman. "How to transform and Filter Images Using Iterated Functions Systems", *SIAM Journal of Imaging Sciences*, vol. 4, no. 4, pp 1001-1028, 2011.

[6] M. F. Barnsley. "Transformation between fractals" *Progress in Probability*, **61** 227-250, 2009.

[7] H. T. Chang. "Arbitrary affine transformation and their composition effects for two-dimensional fractal sets", *Image Vision Comput.* 22 (13) , 1117-1127, 2004.

[8] T. Darmanto, I. S. Suwardi, R. Munir. "Weaving Effects in Metamorphic Animation of Tree- like Fractal based on a Family of Multi-transitional Iterated Function System Code", presented at the International Conference on Computer, Control, Informatics and its Applications, 2013, Computer, DOI: 10.1109/IC3INA.2013.6819150, At Jakarta, Indonesia, Volume: I.

[9] T. Darmanto, I. S. Suwardi, R. Munir. "Animation Model of Multi-object in Fractal Form Based on Partitioned-random Iteration Algorithm", *Procedia Technology* 11, 93-98, Elsevier, 2013.

[10] T. D. DeRose. *Three-Dimensional Computer Graphics, A coordinate-Free Approach (manuscript)*, University of Washington, 1992.

[11] K. J. Falconer. *Fractal Geometry. Mathematical foundations and Applications*, John Wiley and Sons, 1990.

[12] D. Fang. "The Study of Terrain Simulation Based on Fractal", WSEAS Transactions on Computers, Issue 1, Vol. 9, January 2009.

[13] J. Gallier. *Geometric Methods and Applications For Computer Science and Engineering*, Springer - Verlag, TAM Vol.38, 2000.

[14] E. Hadzieva, Lj. M. Kocić. "On some automorphisms on affine spaces" (in Macedonian) *Matematichki bilten* 35(LXI), p. 51-59, Union of Mathematicians of Macedonia, 2011.

[15] E. Hadzieva, J. Petkoski. "An Interactive Application for Modeling Two-Dimensional IFS Fractals" *IEEE Proceedings of the 2nd International Conference Mathematics and Computers in Sciences and Industry - MSCI,* Sliema, Malta, August 17-19, 2015, to be published.

[16] E. Hadzieva, M. Shuminoska. "Real-Time Tool for Affine Transformations of Two-Dimensional Fractals Generated by Iterated Function Systems", *Proceedings of the 12th International ETAI conference*, September 24-27, 2015, Ohrid, Macedonia, ETAI 4-3.

[17] J. E. Hutchinson. "Fractals and self-similarity", *Indiana Univ. Math. J.* **30**, No. 5, 1981.

[18] R. Jovanovic, M. Tuba. "A Visual Analysis of Calculation - Paths of the Mandelbrot Set", *WSEAS Transactions on Computers*, Issue 7, Vol. 8, July 2009.

[19] Lj. M. Kocić, A. C. Simoncelli. "AIFS: a tool for modeling fractal images Complexity and Evolution in the Living World" *SICC IIIrd International Conference*, Rome, Italy, October 21-23, 1998.

[20] Lj. M. Kocić, A. C. Simoncelli. Lj. M. Kocić, A. C. Simoncelli. "Affine Invariant Iterated Function Systems", *SIMAI 98, Proc. IV Congresso Nationale Soc. Ital. Matem. Applicata e Industriale, Giardini Naxos (Messina), It.*, pp. 348-351, Soc. Ital. Mat. Appl. Industr., C.P. 385, Roma, 1998.

[21] Lj. M. Kocić, L. Stefanovska, E. Babace. "AIFS and the Minimal Simplex Problem", *Proceedings of The International Conference of Differential Geometry and Dynamical Systems*, 5-7.10.2007, Bucharest, Romania, pp.119-128, 2008.

[22] A. A. Ungar. *Barycentric Calculus in Euclidean and Hyperbolic Geometry. A comparative Introduction*, World Scientific, 2010.

[23] J. Vince. *Geometry for Computer Graphics, Formulae, Examples and Proofs*, Springer-Verlag London Limited, 2005.

[24] Z. Zhu, E. Dong. "Simulation of Sierpinski-type fractals and their geometric constructions in Matlab environment", *WSEAS Transactions on Mathematics*, Issue 10, Vol. 12, October 2013.