A Novel Web Application for Modeling 3D Fractals

Marija Shuminoska[1], Elena Hadzieva[2], Emilija Celakoska[3]

[1] University of Information Science and Technology "St. Paul the Apostle", Faculty of Computer Science and Engineering – Ohrid, Macedonia

[2] University of Information Science and Technology "St. Paul the Apostle", Faculty of Information Systems, Visualization, Multimedia and Animation, Department of Applied Mathematics – Ohrid, Macedonia

[3]Ss Cyril and Methodius University, Faculty of Mechanical Engineering, Department of Mathematics and Informatics – Skopje, Macedonia

marija.shuminoska@cse.uist.edu.mk, elena.hadzieva@uist.edu.mk, emilija.celakoska@mf.edu.mk

Abstract

This paper introduces a novel application for generating and interactively modeling three-dimensional fractals defined by iterated function system (IFS). The modeling is achieved by means of control points, just like Bezier curves and surfaces are controlled. Unlikely the common way for transforming 3D fractals used in some existing software applications, by setting parameters of the required affine transformations and regenerating fractals, in this application the modeling is free and visually intuitive. There are two ways of defining control points (which are vertices of a 3-simplex): the first one is by manually defining the vertices of the simplex and the second, by finding a kind of minimal 3-simplex that contains the fractal. By allocating barycentric coordinates to each fractal point with respect to the vertices, every repositioning of the vertices leads to a fractal modification visible in real-time. Furthermore, the application offers other interesting features, such as editing the fractal by changing the IFS coefficients, saving files in a specific image format, rotation animation and continuous transformation of the fractal by continuous change of the parameters of some basic affine transformation. The application is written in JavaScript and uses the Three.js library.

Keywords: *3D Fractal, Iterated Function System, Barycentric Coordinates, Simplex, Modeling, Web Application.*

1. Introduction

During the last few years, many engineers inspired by the benefits and beauty of fractals, developed software tools for generation of this specific family of shapes. The solution presented in this paper offers a novel web-based application for generation of three-dimensional fractals defined by an iterated function system (IFS) and for their interactive modeling. The IFS approach relies on a relatively basic theory and the implementation on a computer program is straightforward ([2, 14]). Furthermore, the modeling is performed by dragging and dropping four non-planar points, one at a time; any repositioning of a vertex defines a unique affine transformation that determines the transformation of the fractal. This novel feature, provides great controllability over the fractal image on a very convenient way – the transformations are visible in real time. There are two possibilities for controlling the fractal with the vertices of a simplex. The fractal might be controlled either by the so called *standard-like minimal simplex* (will be defined later) or by a manually added simplex defined by the user in an interactive manner. Besides these main characteristics, the presented application offers other noteworthy features, which will be discussed in more details in section 2.2.

This work is continuation of the work presented in [11-13] where the authors elaborate software application for modeling 2D IFS fractals. Throughout the literature, a lot of things are said about generating 2D fractals (like [2], [14], [17] and many more) and not so many about their transformation ([3], [7], [8], [9], [11-13]). Concerning the 3D fractals, there are very few things about their generation and almost nothing about their transformation. In [16] the authors propose an evolutionary system for automatic creation of 3D artistic fractals. The fractal shapes generated by a combination of an IFS and genetic algorithm, can be interactively modified by adjusting a set of parameters. In [4] a way of mathematical generation of volume data that represents fractals, including ones generated by IFS, is illustrated. For their representation and interactive exploration, the author uses already existing rendering software. The disadvantage is that during the visualization, the zooming requires a new calculation of the fractal volume. An approach for using kaleidoscopic IFS fractals in generating 3D complex buildings is presented in [15]. It provides an option for interactive modeling by selecting parameters that define the building. The results of the changes are instantly available. The authors in [18] introduce a means for generating 3D connected fractal solid. The fractal is then exported in a specific data structure such that the object can be modeled in a CAD platform. The data structure contains necessary information so that a physical object can be built and production of the fractal can be realized.

In the following lines we give some basic definitions of notions used in this paper.

Definition 1 ([5]). Let $\mathrm{A} = (a_1, a_2, a_3), \mathrm{B} = (b_1, b_2, b_3), \mathrm{C} = (c_1, c_2, c_3)$ and $\mathrm{D} = (d_1, d_2, d_3)$ be four non-planar points in $\mathbb{R}^3$. Then a point (x, y, z) in the space has the *barycentric coordinates* (r, s, t, u) with respect to ABCD if

$$\begin{aligned} x &= r \cdot a_1 + s \cdot b_1 + t \cdot c_1 + u \cdot d_1, \\ y &= r \cdot a_2 + s \cdot b_2 + t \cdot c_2 + u \cdot d_2, \\ z &= r \cdot a_3 + s \cdot b_3 + t \cdot c_3 + u \cdot d_3, \\ 1 &= r + s + t + u. \end{aligned} \tag{1}$$

Another way of expressing the formula (1) in terms of matrices is the following form

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \boldsymbol{M} \begin{pmatrix} r \\ s \\ t \\ u \end{pmatrix}, \quad where \quad \boldsymbol{M} = \begin{pmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \\ 1 & 1 & 1 & 1 \end{pmatrix}. \tag{2}$$

Since the vertices $\mathrm{A} = (a_1, a_2, a_3), \mathrm{B} = (b_1, b_2, b_3), \mathrm{C} = (c_1, c_2, c_3)$ and $\mathrm{D} = (d_1, d_2, d_3)$ are non-planar, the determinant of **M** is non-zero. Consequently, the matrix **M** is non-singular and the equation (2) can be reformulated as

$$\begin{pmatrix} r \\ s \\ t \\ u \end{pmatrix} = \boldsymbol{M}^{-1} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3}$$

The relations (2) and (3) represent conversions from barycentric coordinates relative to the affine base {A, B, C, D} to rectangular coordinates and vice versa.

Definition 2 ([5])**.** Every four non-planar points from $\mathbb{R}^n, n \geq 3$ together with their convex hull are called a *3-simplex* (*3D simplex*).

Note that the vertices of a 3-simplex in $\mathbb{R}^3$ are affine basis for this space. Here we will work only with 3-simpleces in $\mathbb{R}^3$.

Definition 3. The 3-simplex in $\mathbb{R}^3$ defined by the points (1,0,0), (0,1,0), (0,0,1) and (0,0,0) is called a *standard simplex*. The 3-simplex in $\mathbb{R}^3$ whose sides are parallel with the standard simplex will be called a *standard-like simplex*.

Definition 4. Given set $S \subset \mathrm{R}^3$. The standard-like simplex with minimal volume that contains the set *S* will be called a *minimal standard-like simplex for S*.

Transformations are essential part of computer graphics and they are generally used to manipulate models of objects by changing their position, shape or orientation. The most used transformations in computer graphics are affine transformations. The transformation of the form

$$\omega(\mathbf{x}) = \mathbf{Ax} + \mathbf{b}, \qquad (4)$$

where **A** is a $n \times n$ matrix and **b** is an *n*-dimensional vector, is called an *n-dimensional affine transformation* ([19]).

An important theorem related with the affine transformations should be mentioned in this section which is called the Fundamental Theorem of Affine Transformations.

Theorem 1. ([6]) *Given two ordered sets of four non-planar points from $\mathbb{R}^3$ each, there exists a unique affine transformation f mapping one set onto the other.*

To achieve our goal for interactive affine transformations of fractals, we use the random iteration algorithm (RIA) for fractal generation, which actually means that the fractal is a set of points ([2]). Relating this set to an affine basis of $\mathbb{R}^3$ with barycentric coordinates relative to that basis, will empower this set with the affine invariance property ([10, p.145-146]).

2. Review of the Available 3D Fractal Applications

The admiration of the fractal geometry of nature and the beauty of fractals themselves have inspired many engineers to develop software tools for generating, examining or analyzing fractals. The most popular applications, in the period when fractals started to take advantage of computers, were Fractal Studio, Astro Fractal, Fractint, FracZoom and 3DFract. However, these programs have not been upgraded in the last few years and therefore will not be taken into consideration in the review. All the selected applications in this section, incorporate the IFS approach as a method for generation of the fractals.

2.1. Mandelbulber ([20])

Mandelbulber is an experimental tool that offers options for translation and rotation of the fractal image. However, the changes are not visible in real time. Every time a particular parameter is modified or a transformation is applied, the user must click on the “render” button, after which the modified fractal is regenerated. Having the feature of combining fractals into infinite hybrid variations, makes this software different than the others. Mandelbulber is available for Windows, Linux and Mac operating systems.

2.2. IFS Builder 3D ([21])

With this software tool, transformations of the rendered fractal are possible by editing a text file written in a special interpretive language. The fractal can be modified by applying one of the basic transformations (translation, scaling, rotation and reflection), or few of them consecutively, to all points simultaneously. Once the text file is edited, the user needs to rebuild the program to make the changes visible on the image window. The transformation of the fractal is again non interactive and only the final transformation is shown on the screen. The software tool is available for Windows and Linux users.

2.3. Fractal Lab ([22])

Besides the generation of IFS fractals, the Fractal Lab offers their real-time transformation limited to rotation and few not well defined and unpredictable transformations. The last transformations yield to interesting fractal objects that might be used for exploration or in game and movie industry. Fractal Lab is a web based application and does not depend on a specific operating system, which is its advantage.

2.4. ChaosPro ([23])

ChaosPro contains several pre-made IFS fractals and fractals generated on many other ways (Escape Time, Quaternion, Layer Fractal, LSystem,…), but it also has an integrated formula editor so that the user can create their own fractal forms. Although the developer describes the software tool as offering a real-time transformation of fractals, that is the case only for rotation and translation. Other transformations for the IFS fractals are not really offered. The ChaosPro runs only on Windows operating system and comes with options with detailed theory explanation and documentation.

2.5. XenoDream ([24])

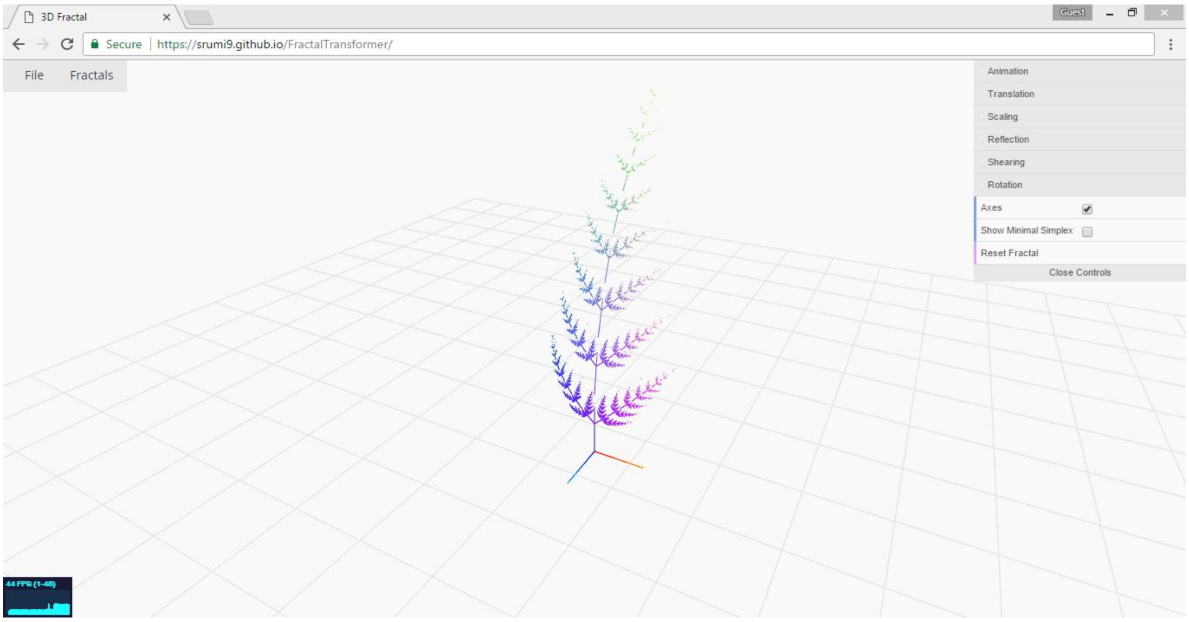
This software is worth mentioning since it offers a lot convenient possibilities for exploring and generating new fractals. However, it does not offer possibilities for transforming fractals which is the primary interest in this paper.

3. The application Fractal Transformer

3.1. User Interface Description

Fractal Transformer ([25]) is an interactive fractal generation and modeling web application, developed with the use of recent browser technologies. It is written in the JavaScript programming language and uses the Three.js library to take advantage of the WebGL API. The WebStorm integrated development environment (IDE) is used for its development.

The graphical user interface (GUI) is user-friendly and offers two menus with well-organized layouts contained within one page. Although simple, the GUI provides various options and features that can be selected or set up. The first fractal, which is rendered after the page content has finished loading, is the so called Barnsley Fern (Fig. 1). This is one of the most astonishing examples of IFS fractals, since it closely resembles nature but also lies in the same mathematical category of constructions as the Sierpinski gasket, the Koch curve and the Cantor set.



Axes ☑

Show Minimal Simplex □

Figure 1. The web application

The Barnsley fern is an attractor of the following IFS: $\{\mathbb{R}^3;\ \omega_1, \omega_2, \omega_3, \omega_4\}$, where the mappings $\omega_i : \mathbf{x} \longmapsto \mathbf{A}_i\mathbf{x} + \mathbf{b}_i\,,\ i = 1, 2, 3, 4,$ all of them equally probable to be applied, are defined by

$$\boldsymbol{A}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.18 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{b}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}; \qquad \boldsymbol{A}_2 = \begin{bmatrix} 0.2 & -0.2 & 0 \\ 0.2 & 0.2 & 0 \\ 0 & 0 & 0.3 \end{bmatrix}, \quad \boldsymbol{b}_2 = \begin{bmatrix} 0 \\ 0.8 \\ 0 \end{bmatrix};$$

$$\boldsymbol{A}_3 = \begin{bmatrix} -0.2 & 0.2 & 0 \\ 0.2 & 0.2 & 0 \\ 0 & 0 & 0.3 \end{bmatrix}, \quad \boldsymbol{b}_3 = \begin{bmatrix} 0 \\ 0.8 \\ 0 \end{bmatrix}; \qquad \boldsymbol{A}_4 = \begin{bmatrix} 0.85 & 0 & 0 \\ 0 & 0.85 & 0.1 \\ 0 & -0.1 & 0.85 \end{bmatrix}, \quad \boldsymbol{b}_4 = \begin{bmatrix} 0 \\ 1.6 \\ 0 \end{bmatrix}.$$

Besides the Barnsley Fern, this application offers several other fractal templates generated with the same random iterative process (Fig. 2, Fig. 3).

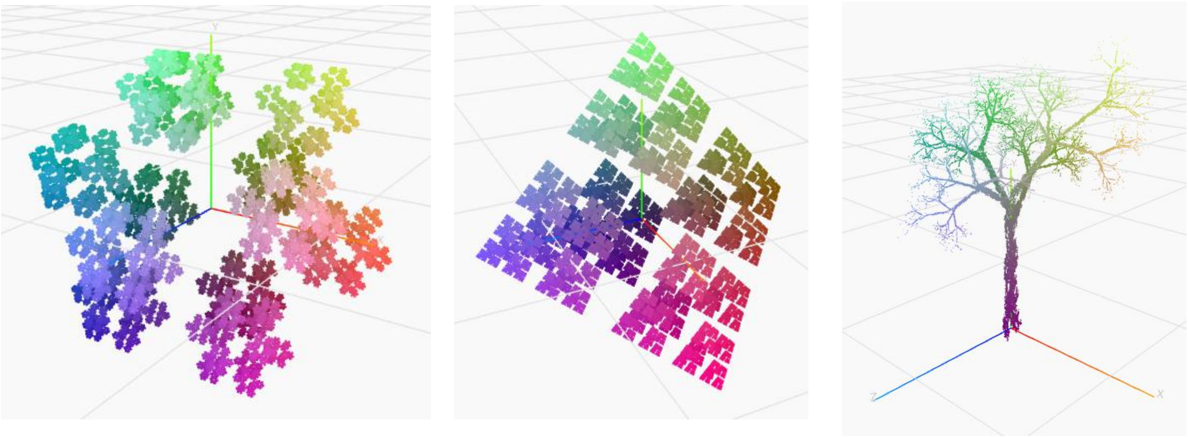


Figure 2. Dodecahedron Flake, Hexahedron and Elm Tree fractals

The whole scene, including the generated fractal are not fixed in the 3D space. Instead, the user can control the camera by rotating, zooming and panning it around the center of the scene (0,0,0). The camera actually behaves as a satellite in orbit around the center, allowing the user to move around with the aid of the mouse and keyboard in real time. This is a very useful feature of the application, especially because it allows observing the fractal from different perspectives as in a real 3D space.

The presented application has also a potential of transforming the fractal in several ways. The reviewed software tools and other available tools in this area, are capable of modifying the fractal by changing the value of some settings. Fractal Transformer goes further and offers a new approach for transformation by means of a control 3D simplex.

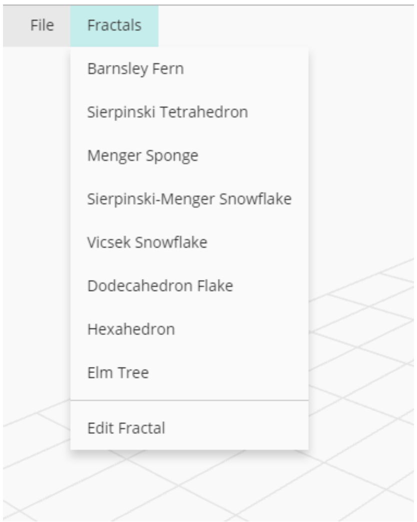


Figure 3. List of fractals from the top left menu

One of the controlling tetrahedrons available in the application is the standard-like minimal simplex of the fractal. Every time a new fractal is selected from the list, its minimal standard-like simplex is immediately calculated. The user can toggle the visibility of this tetrahedron with the 'Show Minimal Simplex' checkbox from the top-right menu. In order to keep the scene as clear as possible, the standard-like minimal simplex is hidden by default. The user can control the movement of the tetrahedron that defines a unique affine transformation (Theorem 1). Above being visually intuitive, the transformation of the fractal caused by the transformation of the simplex is precisely determined. The unique affine transformation defined by changing the position of a vertex of the control simplex (Theorem 1), is exactly the one that transforms the fractal as well. This is possible and mathematically justified by the affine invariance property of the barycentric coordinates ([10]). This is achieved by moving one of its four vertices using the mouse across the scene and placing them in a desirable location. When the mouse cursor is positioned over a vertex, its symbol is changed from default (an arrow) to a pointer (a hand with an outstretched index finger). Pressing and holding the left mouse button when the style is shown as a pointer, will enable movement of the particular vertex. During the repositioning of the mouse, its cursor is changed and remains as a 'move' symbol (two double-sided crossed arrows) until the left button is released. We would like to point out that the fractal image simultaneously and continuously follows every relocation of the vertices A, B, C or D. Since the fractal stays within the convex hull of the standard-like minimal simplex during each transformation, it can be controlled better compared to the manually added simplex (this simplex will be presented later in this section). Several transformations of the Barnsley fern fractal with the use of its standard-like minimal simplex are given in the Fig. 4.

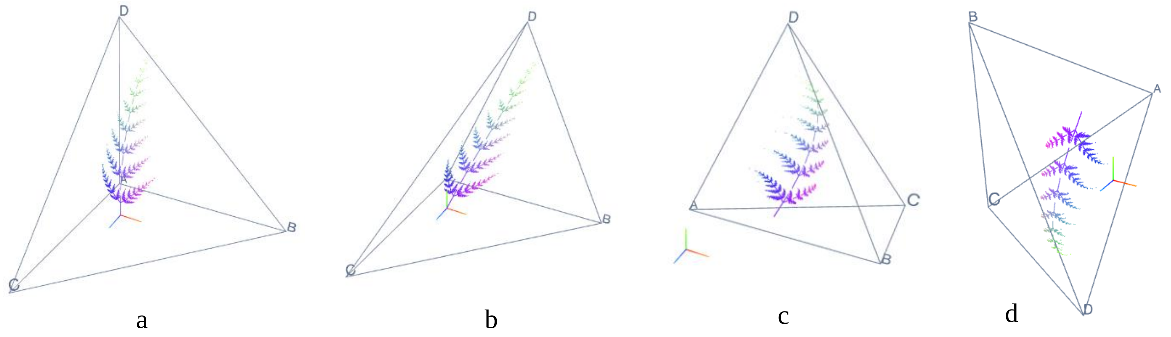
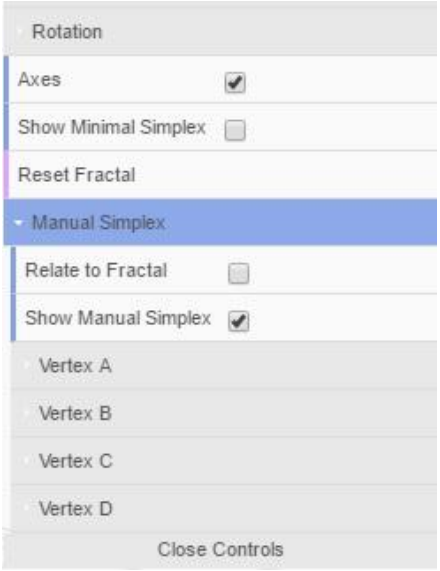


Figure 4. Transforming the Barnsley Fern by means of a standard-like minimal simplex. a. The minimal tetrahedron after selecting the fractal and checkbox 'Minimal Simplex' is constructed. When the simplex is transformed by moving its vertices (b. vertex D, c. vertex C, d. vertices D and A), the fractal follows the transformation.

There are situations when the user requires greater autonomy when it comes to determining the position of the tetrahedron's vertices. For that purpose, Fractal Transformer also gives an option to place the points of the simplex anywhere on the scene. Plotting three-dimensional points is as follows. First, the user needs to press and hold the Control key. As a result of this action, a blue semi-transparent plane that has its center at the origin of the 3D space (0, 0, 0) will be rendered on the scene. As the user rotates the scene, the plane changes its orientation such that it always faces the camera. Now, while the plane is visible the four vertices of the manually added tetrahedron can be added by left mouse click, one vertex at a time. Once the position of the fourth vertex is defined, the vertices will instantaneously be connected with edges. At the same time, a section with settings which are dedicated to the manually added simplex will appear in the right menu (Fig. 5).


Rotation

Axes ☑

Show Minimal Simplex □

Reset Fractal

Manual Simplex

Relate to Fractal □

Show Manual Simplex ☑

Vertex A

Vertex B

Vertex C

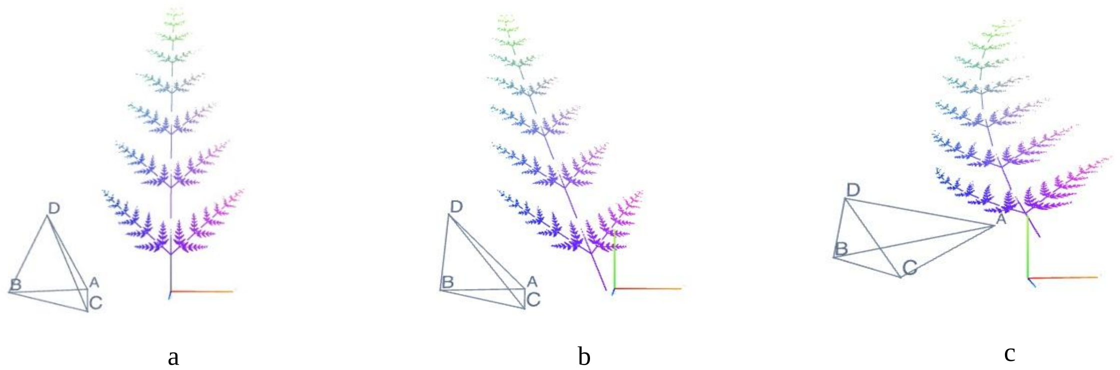
Vertex D

Close Controls

Figure 5. Menu section for the Manual Simplex

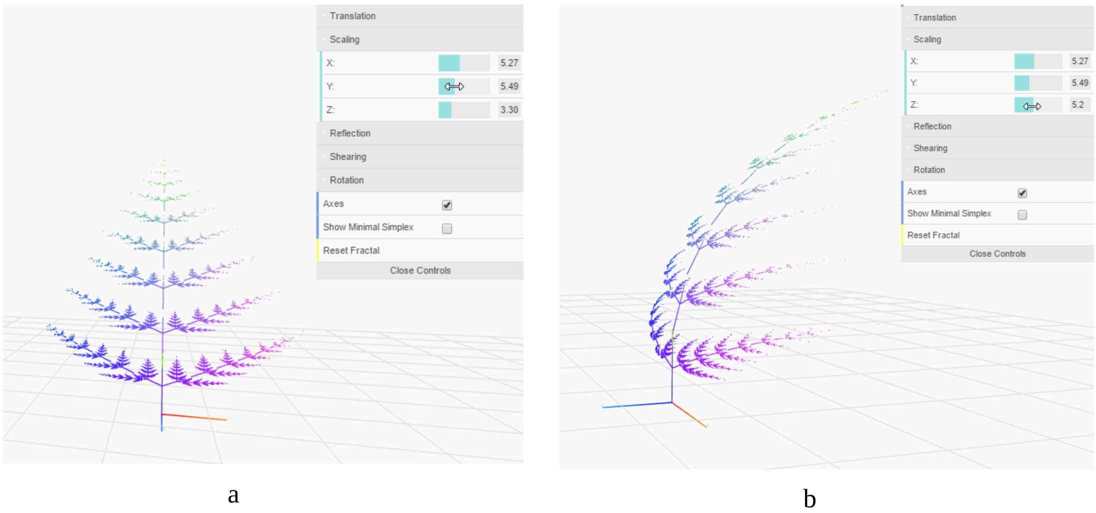
At this point, the manually added simplex is created but it is not related with the fractal image. As a result, the user can still modify the position of the vertices by the click and hold option of the left mouse button or by defining their coordinates in the ‘Manual Simplex’ section. When the tetrahedron is placed at a desired location, it can be connected with the rendered fractal by selecting the checkbox button ‘Relate to Fractal’ in the ‘Manual Simplex’ section. This action will not do any visible changes on the screen at that moment, but it will calculate the barycentric coordinates of each point belonging to the fractal with respect to the newly defined simplex. This enables the user to perform affine transformations over the attractor with the manually added simplex. The connection between the fractal image and manual simplex is based on the same relation (2) as for the minimal simplex. Again, each repositioning of the simplex vertices is followed by a fractal transformation visible in real time (Fig. 6).

Figure 6. Transforming the Barnsley Fern through a tetrahedron with manually plotted vertices. a.



The manual tetrahedron after plotting the forth vertex. When the simplex is transformed by moving its vertices (b. vertex D, c. vertex A), the fractal follows the transformation

Not only free modeling, but the application also offers defining precise transformations, like: translation for a vector **t**, rotation for an angle α around *x*-, *y*-, and *z*- axes, reflection, shearing w.r.t *x*-, *y*-, *z*- axes and scaling for factor *s* w.r.t. all axes (Fig. 7). None of the transformations described so far affects the IFS coefficients of the fractal attractor, meaning that the fractal is not generated over again.


Translation
Scaling
X: 5.27
Y: 5.49
Z: 3.30
Reflection
Shearing
Rotation
Axes ☑
Show Minimal Simplex □
Reset Fractal
Close Controls

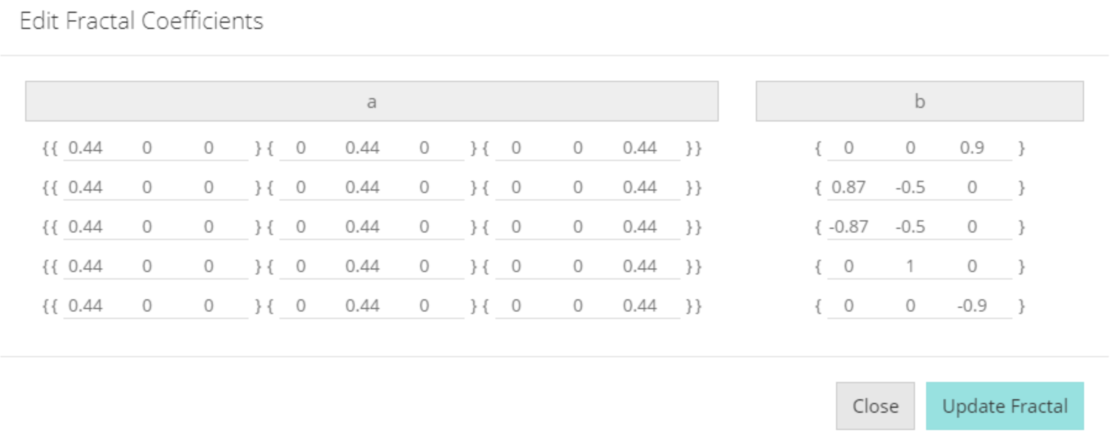
a

Translation
Scaling
X: 5.27
Y: 5.49
Z: 5.2
Reflection
Shearing
Rotation
Axes ☑
Show Minimal Simplex □
Reset Fractal
Close Controls

b

Figure 7. Scaling the Barnsley Fern fractal through the a. *y* and b. *z* axis with the transformation options available in the right menu.

Among the other things, Fractal Transformer allows the user to modify the fractal by altering its coefficients in the ’Edit Fractal Coefficients’ modal window shown on Fig. 8 and thus experimenting, making own fractals. When the user clicks the ‘Edit’ option of the ‘Fractals’ menu (Fig. 3), the modal window opens and it is populated with the coefficients of the currently rendered fractal. The content of the window is divided into two main parts, the left under the ‘a’ label for the **A** matrices and the right part is dedicated to the **b** vectors. The data fields accept only numerical values in order to avoid errors in the application’s code. Any changes made in the coefficients will affect the fractal by clicking the ‘Update Fractal’ button which is located in the bottom right corner of the modal window.



Edit Fractal Coefficients

a	b
{{ 0.44 0 0 }{ 0 0.44 0 }{ 0 0 0.44 }}	{ 0 0 0.9 }
{{ 0.44 0 0 }{ 0 0.44 0 }{ 0 0 0.44 }}	{ 0.87 -0.5 0 }
{{ 0.44 0 0 }{ 0 0.44 0 }{ 0 0 0.44 }}	{ -0.87 -0.5 0 }
{{ 0.44 0 0 }{ 0 0.44 0 }{ 0 0 0.44 }}	{ 0 1 0 }
{{ 0.44 0 0 }{ 0 0.44 0 }{ 0 0 0.44 }}	{ 0 0 -0.9 }

Figure 8. The Edit modal window containing the IFS coefficients of the Hexahedron fractal

The edit option can also be used to create new fractal attractors that are generated by up to 20 mappings. If the number of mappings is less than 20, the rest of the matrices **A** and vectors **b** need to be replaced with identity matrix and zero vector.

An interesting component of the application is the rotation animation option (Fig. 10). When the checkbox of this section is clicked and marked as selected, the currently rendered fractal image will rotate around the axes (the order of axes around which full rotation is performed is chosen randomly). To change the speed of rotation, the user can simply click and slide the mouse curser horizontally over the border between the green and gray area. As the speed value crosses the zero, from positive to negative and vice versa, the object changes its direction of rotation.

The ‘File’ menu located in the top left menu bar (Fig. 3) contains two listings - ‘Save as PNG’ and ‘Save as JPG’. Selecting one of the options will download the scene with the fractal in the user’s Download folder in png or jpeg file type, depending on the chosen option. The menus will not be included in the downloaded image.

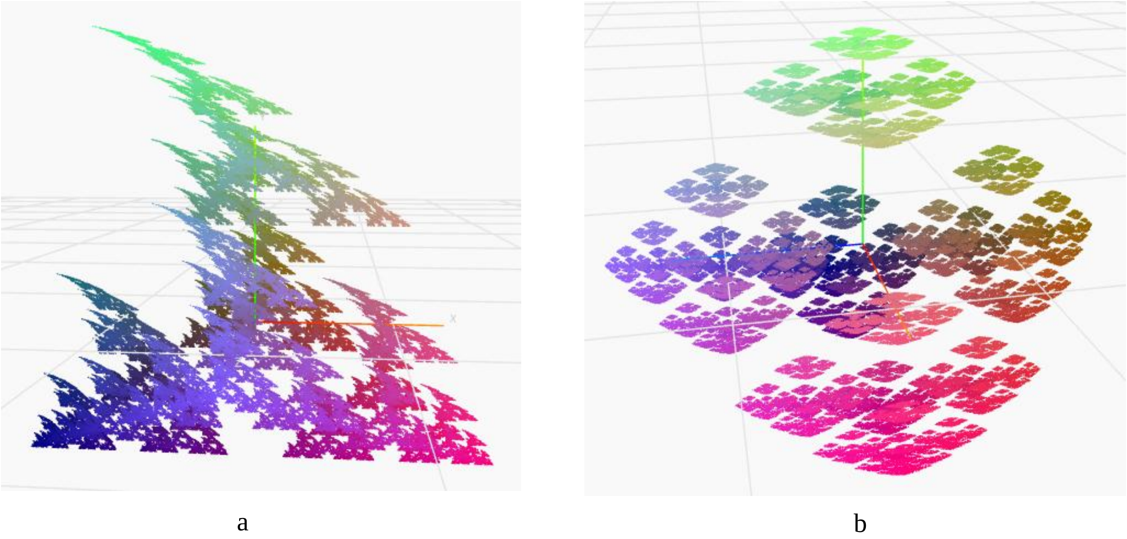
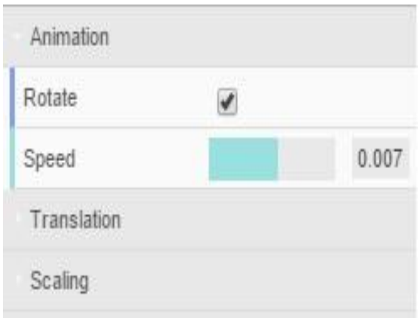


Figure 9. Modifications of the Hexahedron fractal obtained by changing coefficients:

a. $A_4(1,2)$ decreased for 0.32;
b. $A_2(1,1)$ decreased for 0.2, $A_2(3,3)$ increased for 0.12 and $A_4(2,2)$ decreased for 0.19


Animation

Rotate ☑

Speed 0.007

Translation

Scaling

Figure 10. The Animation sub-menu

3.2 Code Analysis

When generated by RIA, the image of the fractal attractor is entirely composed of points. In the presented application, we used the aforementioned algorithm to calculate the coordinates of the fractal points (Algorithm 1) ([2]). Once the array of points is populated, we leave out its first 20 elements since some of them might not belong to the fractal attractor and look scattered on the scene. Please note that with this algorithm, the probabilities to select a certain mapping ω_i are equal.

Algorithm 1 Calculating the coordinates of the fractal points

```
procedure CalculateFractalPoints()
    rnd ← a random number between 0 and 1
    fractalPoints[0] ← [rnd, rnd, rnd]
    for i ← 0 to (fractalPoints length - 1) do
        fractalPoint ← fractalPoints[i]
        random ← a random number between 0 and 1
        for j ← 0 to matrixA length do
            if (random >= j / (matrixA length) && random < (j + 1) / (matrixA length))
                product ← multiply fractalPoint with matrixA[j]
                sum ← calculate sum of product and matrixB[j]
                fractalPoints [i + 1] ← [sum[0], sum[1], sum[2]]
            end if
        end for
    end for
end procedure
```

One of the options for fractal manipulation is a transformation with a standard-like minimal simplex. Its four vertices have the following positions ([1]):

A (*xMin*, *yMin*, *zMin*);
B (*maxSum* - *yMin* - *zMin*, *yMin*, *zMin*);
C (*xMin*, *yMin*, *maxSum* - *xMin* - *yMin*);
D (*xMin*, *maxSum* - *xMin* - *zMin*, *zMin*);

where *xMin*, *yMin*, *zMin* are the minimal values of all *x*-, *y*- and *z*-coordinates of the fractal points, respectively, and *maxSum* is the maximum of the sums of the three coordinates of the fractal points.

Algorithm 2 is based on the relation (3) for converting the rectangular into barycentric coordinates.

Algorithm 2 Calculating the barycentric coordinates from rectangular coordinates

```
procedure CalculateBarycentricCoordinates
    vertexMatrix ← addValues(vertexA.x, vertexB.x, vertexC.x, vertexD.x, vertexA.y,
    vertexB.y, vertexC.y, vertexD.y, vertexA.z, vertexB.z, vertexC.z, vertexD.z)
    matrixInverse ← calculate inverse matrix of vertexMatrix
    for i ← 0 to (fractalPoints length) do
        fractalPoint ← fractalPoints[i]
        pointVector ← addValues(fractalPoint.x, fractalPoint.y, fractalPoint.z)
        barCoordinates ← multiplyMatrices( matrixInverse, pointVector )
         barycentricPositions[i] ← addValues([barCoordinates[0], barCoordinates[1],
         barCoordinates[2], barCoordinates[3]]);
    end for
end procedure
```

To determine the positions of the fractal points after a simplex vertex is moved, i.e. to convert barycentric to rectangular coordinates resembling the position of points of the transformed fractal, Fractal Transformer implements the Algorithm 3, based on the relation (2).

Algorithm 3 Calculating the Cartesian coordinates of the fractal points from barycentric coordinates

```
procedure RecalculateCartesianCoordinates
    for $i \leftarrow 0$ to ( fractalPoints length ) do
        $x \leftarrow 0, y \leftarrow 0, z \leftarrow 0$
        for $j \leftarrow 0$ to 4 do
            $x$ += simplexVertices[$j$].$x$ * barycentricPositions[$i$][$j$]
            $y$ += simplexVertices[$j$].$y$ * barycentricPositions[$i$][$j$]
            $z$ += simplexVertices[$j$].$z$ * barycentricPositions[$i$][$j$]
        end for
        fractalPoints[$i$] $\leftarrow (x, y, z)$
    end for
end procedure
```

Since the steps of the algorithms 1 and 2 are executed only once, it is evident that the third algorithm which is performed at every transformation of the simplex, bears the responsibility for the speed of the transformation.

3.3. Time Complexity Analysis

The algorithms given in this subsection are obviously dependent on the number of fractal points, so it only remains to determine the complexity of that dependence.

The first two instructions of Algorithm 1 will run one or constant time regardless the number of fractal points. The sequence of statements within the first *for* loop executes N times, where N is the number of fractal points. Every time the outer loop performs, the inner *for* loop executes between 1 and n times depending on the value of the *if* condition, where n is the number of matrices **A**, or IFS mappings. Since the value of IFS mappings is relatively small and usually smaller than 10, we will consider the running time of the nested *for* loop as a constant. Therefore, the total time complexity of Algorithm 1 is $(1 + N \cdot 1)$, or $O(N)$. This means that the execution time is linear and directly proportional to the input size, or the running time grows linearly as the input size increases.

The time complexity of the first two statements of Algorithm 2 is constant, since they are not dependable on the input size. The following loop and its sequence of statements is executed N times. The complexity of each statement is 1, so the total time for the *for* loop is $N \cdot 1$ which is $O(N)$ overall.

The third algorithm calculates Cartesian coordinates for each point of the fractal knowing the barycentric coordinates in a nested *for* loops. The complexity is less than or equal to $(N \cdot 4)$ because the first *for* loop iterates N times while the second one, 4 times. The run time of a single statement within these loops is 1. Since the 'Big O' metric ignores the constant factors, which is the complexity of the nested *for* loop in this case, the overall time complexity of Algorithm 3 is linear $O(N)$.

The running time of the presented application has shown to be linearly bound with the number of points that constitute the fractal image. This number is defined and can be modified directly in the code. The linear time is the best achievable result of time complexity for a case like this, where the algorithm has to sequentially read the entire input. The efficiency of such algorithms can be improved by the use of hardware technologies that support parallel computing.

4. Conclusion and Future Work

Fractals can undoubtedly change the way we see the world and ourselves. It is fascinating how the fractal geometry can produce spectacular and often complex shapes through a simple and straightforward process of iteration. The resulting images are striking yet intriguing, attracting many scientists and artist for 30 years already.

This paper presents a novel 3D application for interactive and real-time transformation of fractals generated with the IFS approach. The rendered image of the fractal can be interactively manipulated through control points and by setting values. The control points are either manually added or they are vertices of a standard-like minimal simplex (in the second case the convex hull property when modeling is preserved). The vertices in both cases can be dragged and dropped one at a time, defining a unique affine transformations. Every change in the position of the vertices is simultaneously and continuously followed by the same affine transformation of the fractal which is visible in real-time. Besides this, the fractal can be interactively transformed by changing the values of settings that represents the basic affine transformation such as rotation, scaling, translation, shearing and reflection. In addition, Fractal Transformer offers options for rotation animation and for saving the fractal as an image in a specific format. The editing feature can produce interesting variations or completely new fractals by changing their coefficients.

Although Fractal Transformer offers various approaches for fractal analysis and exploration, many possibilities for extending the functionality of the application still remain. One of the improvements would be transformation of a selected part of the fractal attractor. The inclusion of this feature would enrich the options for fractal manipulation and may lead to interesting modifications of known fractals. So far, the probabilities for choosing an affine transformation from the set of mappings are equal. Adding possibility for assigning different probabilities to the affine transformations will be surely useful. To provide even more interactivity to the application, the changes of the edit modal window might be visible in real-time. Furthermore, a possibility for modeling the fractal with a minimal polyhedron or some other solid that will contain the fractal and will have more than 4 vertices, might be implemented. Also, the functionality of the Fractal Transformer would be enhanced by adding other types of transformation, animation or morphing. These are only few of the possible refinements of the application, but depending on the area where it will be used, the added features would be more specific.

References

[1]. Babače, E., Kocić, L. M., 2009. Minimal Simplex for IFS Fractal Sets. NAA 2008, Lecture Notes in Computer Science 5434, pp.168-175, Eds.: S. Margenov, L. G. Vulkov, J. Wasniewski, Springer-Verlag Berlin Heidelberg

[2]. Barnsley, M. F., 1993. Fractals Everywhere. Second Edition, Academic Press, San Diego.

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

[4]. Bourke, P., 2017. Visualising Volumetric Fractals. GSTF Journal on Computing (JoC) Vol.5 No.5.

[5]. Brannan, D. A., Esplen, M. F., Gray, J. J, 2012. Geometry. Second edition, Cambridge: Cambridge University Press.

[6]. Byer, O., Lazebnik, F., Smeltzer, D. L., 2010. Methods for Euclidean Geometry. Mathematical Association of America, Chapter 12.

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

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

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

[10]. Farin, G., Hansford, D., 2005. Practical Linear Algebra. A Geometry Toolbox. A K Peters, Wellesley, Massachusetts.

[11]. Hadzieva, E., Grupcev, V., Petkoski, J., 2017. Extension and Software Analysis of an Interactive Application for Transforming Fractals. Submitted for publication in European Journal For Information Science and Technology.

[12]. Hadzieva, E., Petkoski, J., 2016. A novel software application for interactive affine transformations of fractals. International Journal of Mathematical Models and Methods in Applied Science, Volume 10.

[13]. Hadzieva, E., Shuminoska, M., 2016. Real-time tool for affine transformations of two dimensional IFS fractal. Journal of Electrical Engineering Volume 4, Number 3, pp. 150-155, David Publishing Company, doi: 10.17265/2328-2223/2016.03.001

[14]. Hutchinson, J., 1981. Fractals and Self-Similarity. Indiana Univ. Math. J. 30 No. 5, 713–747.

[15]. McGraw, T., 2015. Interactive Procedural Building Generation Using Kaleidoscopic Iterated Function Systems. In: Bebis G. et al. (eds) Advances in Visual Computing. Lecture Notes in Computer Science, vol. 9474. Springer, Cham.

[16]. Pang, W., Hui, K.C., 2010. Interactive Evolutionary 3D Fractal Modeling. Visual Computer 26(12), 1467-1483.

[17]. Peitgen, H., Jurgens, H., Saupe, D., 2004. Chaos and Fractals: New Frontiers of Science. Second Edition, Springer, Berlin Germany.

[18]. Soo, S.C., Yu, K.M., Chiu, W.K., 2006. Modeling and fabrication of artistic products based on IFS fractal representation. Computer-Aided Design 38(7), 755-769.

[19]. Van Verth, J.M, Bishop, L. M., 2008. Essential Mathematics for Games and Interactive Applications: A Programmer's Guide. Second Edition, Burlington, Massachusetts: Morgan Kaufmann Publishers.

[20]. http://www.mandelbulber.com/index.php

[21]. http://fractals.nsu.ru/builder3d_en.htm

[22]. http://hirnsohle.de/test/fractalLab/

[23]. http://www.chaospro.de/index.php

[24]. http://www.xenodream.com/

[25]. https://srumi9.github.io/FractalTransformer/