Computational Aesthetics in Graphics, Visualization, and Imaging (2008)
P. Brown, D. W. Cunningham, V. Interrante, and J. McCormack (Editors)

Fast Visualisation and Interactive Design of Deterministic Fractals

Sven Banisch[1] & Mateu Sbert[2]

[1]Faculty of Media, Bauhaus–University Weimar, D-99421 Weimar (GERMANY)
[2]Department of Informàtica i Matemàtica Aplicada, University of Girona, 17071 Girona (SPAIN)

Abstract

This paper describes an interactive software tool for the visualisation and the design of artistic fractal images. The software (called AttractOrAnalyst) implements a fast algorithm for the visualisation of basins of attraction of iterated function systems, many of which show fractal properties. It also presents an intuitive technique for fractal shape exploration. Interactive visualisation of fractals allows that parameter changes can be applied at run time. This enables real-time fractal animation. Moreover, an extended analysis of the discrete dynamical systems used to generate the fractal is possible. For a fast exploration of different fractal shapes, a procedure for the automatic generation of bifurcation sets, the generalizations of the Mandelbrot set, is implemented. This technique helps greatly in the design of fractal images. A number of application examples proves the usefulness of the approach, and the paper shows that, put into an interactive context, new applications of these fascinating objects become possible. The images presented show that the developed tool can be very useful for artistic work.

1. Introduction

The developments in the research of dynamical systems and its attractors, chaos and fractals has already led some people to declare that god was a mathematician, because mathematics could be found behind the stunning beauty of nature. And indeed have researchers found with it a receipt to gain an insight into a series of real-world processes and problems, many of which have been inaccessible by other mathematical disciplines. The fact that many natural and social phenomena, from the growing of plants [PL90] and population evolution of entire species [Zha03] over the ups and downs of the stock market [JCVT93] to weather changes and particular weather effects [Lor63], can be understood better in the light of theories that emerged from the study of iterated systems is, of course, a major motivation to deal with this topic.

It is now more than 30 years ago that the first images of Julia sets and the Mandelbrot set have been presented. Since then, more and more pictures of fractal objects have been generated, and there is still an increasing interest in using these structures in the graphical design process.

Fractal image generation is fairly time–consuming, and an interactive application was not really feasible till the exploitation of the capabilities of new, fast graphics hardware.

The development of an interactive fractal visualisation method, is the main objective of this paper. This can give to the artist new possibilities in the design process, and help the researcher in the analysis of discrete systems used to produce these images.

This paper is organized as follows. In the next section we review fractal visualization tools. In section 3 we present the mathematical approach of our fractal generation technique, section 4 deals with the visualization of the generated fractals, and in section 5 we present some of visualizations together with possible applications of our technique. Finally the last section is devoted to conclusions and future work.

2. Related Work

The fundamentals of the modern research in discrete dynamical systems have been formulated in 1918 by G. M. Julia, analysing the behaviour of iterated rational functions in the complex domain [Jul18]. He described the most important properties of those sets, which are called Julia sets nowadays. Since then dynamical systems, chaos and fractal theory have been applied in many different scientific areas (e.g., [Lor63, Bar88, JX93, JCVT93, Man97]). In fact, there exist entire volumes dedicated only to the application of chaos and fractals in scientific research [AJCJ93]. For this reason, and also because we are mainly interested in the visualisation of fractals, this section concentrates on existing tools for fractal image generation.

The number of software tools available for fractal image generation is tremendous and they may differ greatly in the mathematical knowledge required to work with them. Some of them form an attempt to improve the user's understanding of the mathematics behind fractal objects, others are mainly oriented to the artist providing powerful routines for colour modifications and fractal animation.

Since it is impossible to consider all applications in detail, we will look at a selection of tools, which, to the authors' point of view, are the most advanced and related to the program presented in this paper. A comprehensive overview however can be found on the web page of Paul N. Lee [Lee], where a wide range of software examples is listed and briefly evaluated.

2.1. A Selection of Fractal Generation Tools

In the following, we will discuss three fractal generation and visualisation tools. We will focus on the following characteristics (other characteristics such as output resolution are not discussed here for lack of space):

- the program's relevance for education and dynamical systems study,
- the interactivity of the application,
- the application's capability to produce fractal animations,
- the colour setting routines and the visual quality of the final images.

The first tool we consider is Fractint [Gif], which is a free–ware fractal generator available for MAC, UNIX, Linux, Windows, etc. There is a large number of different fractal types which can be visualized by this application, including several strange attractors of physical systems solved by numerical integration of the underlying differential equations. Moreover, a series of higher-dimensional dynamical systems is implemented, and images of the attracted regions of these systems can be created by projecting them on the 2D plane.

Regarding the relevance for education purposes, above all, the "Parameter Explorer/Evolver" should be mentioned. It links the classic Mandelbrot set and its Julia sets in a very interesting way. The visualisation of the changing Julia set is quite fast, however, not real-time at all. The update rates do not allow a continuous real–time animation. Nevertheless, it succeeds to point out the correlation between Julia sets and the Mandelbrot set. The interactive analysis mode presented in this paper makes use of this correlation and generalizes this concept.

On top of all, Fractint is a scientific tool. Routines for colour settings are not so well developed, although probably outstanding images can be produced by the experienced user. In Fractint, it is unfortunately not possible to change parameters of the visualised set at run time. This means that it is necessary to redefine the iterated system and to revisualise this new set afterwards.

With Ultra Fractal [Ult07], the next tool to be considered here, in contrary to Fractint, even the inexperienced user can produce wonderful fractal images. The main reason for this is the artistic purpose it follows. Ultra Fractal is a commercial software and the carefully designed user interface follows the spirit of Photoshop. It implements layering and filtering techniques, which are important features for graphical tools.

Additionally, a compiler for new, user–specific formulas exists. This can make the tool interesting for researchers and students, even though the real purpose of this software is not in education, and its relevance for learning about fractals and attracted sets is reduced. For particular fractals, there exists the possibility to navigate through the Mandelbrot set, in order to get the desired Julia shape, similar to the analysis mode presented here. Regarding interactivity and animation, Ultra Fractal is clearly superior to Fractint.

However, the only real interactive component is the zooming in and out routine. Changing parameters and updating the image at runtime is not possible and the tool is not suited for real–time fractal animations.

But there exists the possibility of producing key frame animations. This is a powerful feature. In each key frame, the user can set particular parameters values. The animation is then computed by linear interpolation between these values.

The last program taken into account here is Fractal Explorer [Fra06], which has been released only two years ago. Not least the combination of 2D fractal generation, 3D attractor visualisation, the visualisation of geometric iterated function systems, and the implementation of maps acting in the quaternions makes this free software an interesting tool for image generation and scientific work.

Basically, the program implements all the standard complex fractal types, the parameters of which can be chosen before starting the visualisation. Additionally, there exists a formula parser, and it is possible to work with the specific system the user is interested in. The only interactive feature, implemented in this application, is the analysis mode. This is made possible by a very reduced image size for the real–time update. In the normal fractal visualisation mode, the user cannot change parameters and, as in the other tools, a re–specification of the system is necessary.

In the Fractal Explorer, there is a separate dialog for animation production, in which video of zooming animation can be created. Here, as in the case of Ultra Fractal, it is also possible to produce animations of Julia sets under the change of particular parameters. However, real–time animation is not possible.

This overview shows that existing fractal visualisation tools provide sophisticated means for the creation of static fractal images. With Ultra Fractal and the Fractal Explorer, moreover, the offline production of fractal animations is possible. However, none of the existing fractal tools allows fractal parameters to be changed at runtime.

3. Mathematical Representation

In the literature, as well as in most applications for the visualisation of fractals in the plane, a complex system definition is used. This means that the system is defined by a function $F : \mathbb{C} \to \mathbb{C}$. Iterating such systems over and over, and visualising those point regions that are attracted under the repeated iteration of F, very interesting fractal images can be created. However, in general, the same fractals can be obtained when $F : \mathbb{R}^2 \to \mathbb{R}^2$. And in fact, the possible number of different shapes is increased by such a definition.

Primarily for this reason, a definition in $\mathbb{R}^2$ is used. This facilitates an extended control over the fractal parameters while incorporating some of the most common fractals considered by previous authors. The system map hence takes the form

$$F(x,y) = \begin{pmatrix} g(x,y) \\ h(x,y) \end{pmatrix}, \quad (1)$$

where $F(x,y) : \mathbb{R}^2 \to \mathbb{R}^2$ is a map acting on the x–y–plane, since $g : \mathbb{R}^2 \to \mathbb{R}$ and $h : \mathbb{R}^2 \to \mathbb{R}$. Therefore, the initial values of the system $p_0 = (x_0, y_0)^T$ are points in the real plane.

The orbit of an initial point p_0 is defined by

$$s(p_0) = (p_0, F(p_0), F^2(p_0), \ldots, F^\infty(p_0)), \quad (2)$$

where

$$F^n(p_0) = F \circ F^{n-1}(p_0) = \underbrace{F \circ \ldots \circ F}_{n\ times}(p_0) \quad (3)$$

must not be confused with the nth power of F. It is, instead, the repeated application of F to an initial value p_0. Since $F : \mathbb{R}^2 \to \mathbb{R}^2$, the elements of the orbit are also points in the real plane. For the computation of these point sequences, the iterative scheme

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = F(x_i, y_i) = \begin{pmatrix} g(x_i, y_i) \\ h(x_i, y_i) \end{pmatrix} \quad (4)$$

is used. Hence, the $(i+1)$th element $p_{i+1} = (x_{i+1}, y_{i+1})^T$ is computed by applying F to the preceding element p_i. This iteration process, beginning with $p_1 = F(p_0)$, yields the orbit of p_0.

Depending on p_0, the orbit $s(p_0)$ either converges to some state (e.g., to a fixed point) or it tends to infinity. The region of those initial values the orbit of which converges is called attracted region, basin of attraction or filled Julia set. And in many cases these regions are fractal–shaped.

The characteristics of the dynamical system and therefore also the appearance of the attracted region exclusively depend on the functions $g(x,y)$ and $h(x,y)$. In this paper, we concentrate on polynomials up to order 4, namely:

$$g(x,y) = \textstyle\sum_{k=1}^4 a_k^g x^k + \sum_{k=1}^4 b_k^g y^k + c_1^g xy + c_2^g x^2 y + c_3^g xy^2 + c_4^g (xy)^2 + d^g, \quad (5)$$

$$h(x,y) = \textstyle\sum_{k=1}^4 a_k^h x^k + \sum_{k=1}^4 b_k^h y^k + c_1^h xy + c_2^h x^2 y + c_3^h xy^2 + c_4^h (xy)^2 + d^h. \quad (6)$$

The coefficients of the polynomials, a_k^g, b_k^g, c_k^g, d^g in Eq. (5) and a_k^h, b_k^h, c_k^h, d^h in Eq. (6), are the free parameters of the system. All 26 (13 in g and 13 in h) of them can be changed at runtime to generate differently–shaped basins of attraction. This is also suited to analyse the influence of particular parameters on the behaviour of the dynamical system.

There is a major reason to look at this polynomial class of dynamical systems. It was chosen because a typical class of fractals, commonly defined in the complex plane, can be represented by this means, among them the map $F_c(z) = z^2 + c$ with $z, c \in \mathbb{C}$ studied by B. B. Mandelbrot. Splitting F_c into its real and its imaginary component, $F_\Re$ and $F_\Im$, this map corresponds to

$$\begin{aligned} F_\Re &= g(x,y) = x^2 - y^2 + c_\Re, \\ F_\Im &= h(x,y) = 2xy + c_\Im \end{aligned} \quad (7)$$

with the used system definition (Eq. (5) and (6)). In fact, all polynomial complex maps up to order 3 can be represented in that way.[1]

Of course, this representation does not incorporate all system types in which researchers and artists might be interested. On the other hand, however, it allows *all* possible variations within this polynomial class, and there does not yet exist any software implementing this. And not least due to the internal vector representation used in the fractal computation, the extension of the program to other systems (including, for instance, trigonometric or exponential functions) is possible without much effort.

Such a vector formulation is obtained when passing to the GPU the 26 coefficients of the system in vector form as

$$\vec{a}^{\,g} = \begin{pmatrix} a_1^g \\ a_2^g \\ a_3^g \\ a_4^g \end{pmatrix}, \vec{b}^{\,g} = \begin{pmatrix} b_1^g \\ b_2^g \\ b_3^g \\ b_4^g \end{pmatrix}, \vec{c}^{\,g} = \begin{pmatrix} c_1^g \\ c_2^g \\ c_3^g \\ c_4^g \end{pmatrix}, d^g \quad (8)$$

for $g(x,y)$, and respectively for $h(x,y)$, defining $\vec{a}^{\,h}, \vec{b}^{\,h}, \vec{c}^{\,h}, d^h$ in the same way. For the computation

[1] By allowing components $(x^3 y)$ and (xy^3), all complex polynomial maps of order 4 could be incorporated as well.

of $(i+1)$th element of the iterated sequence, p_{i+1}, we construct the *base vectors*

$$\vec{x}_i = \begin{pmatrix} x_i \\ x_i^2 \\ x_i^3 \\ x_i^4 \end{pmatrix}, \vec{y}_i = \begin{pmatrix} y_i \\ y_i^2 \\ y_i^3 \\ y_i^4 \end{pmatrix}, \vec{xy}_i = \begin{pmatrix} x_i y_i \\ x_i^2 y_i \\ x_i y_i^2 \\ (x_i y_i)^2 \end{pmatrix} \tag{9}$$

and calculate $p_{i+1} = (x_{i+1}, y_{i+1})^T$ by

$$\begin{aligned} x_{i+1} &= (\vec{a}^{\,g} \cdot \vec{x}_i) + (\vec{b}^{\,g} \cdot \vec{y}_i) + (\vec{c}^{\,g} \cdot \vec{xy}_i) + d^g \\ y_{i+1} &= (\vec{a}^{\,h} \cdot \vec{x}_i) + (\vec{b}^{\,h} \cdot \vec{y}_i) + (\vec{c}^{\,h} \cdot \vec{xy}_i) + d^h, \end{aligned} \tag{10}$$

using hardware supported dot products. This not only facilitates fractal computation at real–time rates, but also allows the extension to very different fractal types by using other base functions in Eq. (9).

4. Fast Fractal Visualisation

In the AttractOrAnalyst software tool, fractals are drawn on a square surface. This surface is placed perpendicular to the viewing direction of the user, and the rendering is done on the GPU.

The texture coordinates $(u,v) \in [0,1]^2$ of the square are used to define the coordinates of the initial points ($p_0 = (x_0, y_0)$), the orbit of which is to be computed for the fractal generation. In order to be able to arbitrarily change the domain for fractal visualisation, the coordinate transformation

$$\begin{aligned} x_0 &= x_{min} + u(x_{max} - x_{min}) \\ y_0 &= y_{min} + v(y_{max} - y_{min}) \end{aligned} \tag{11}$$

is applied, where x_{min}, x_{max} and respectively y_{min}, y_{max} represent the borders of the visualisation range. The user is allowed to change these parameters at runtime (explicitly or by using mouse gestures), which makes possible interactive zooming and translation.

The basin of attraction of a system F is obtained by computing the orbits of all the initial points, p_0, and by testing whether these orbits tend to infinity or not, i.e., whether they "escape in time" or not. Therefore, this algorithm is called the escape–time algorithm.

The orbit of an initial point p_0 is given by Eq. (2). In practice, the orbit computation is stopped after a certain number of iterations N (usually, $25 \leq N \leq 100$). Therefore, we decide if p_0 belongs to the attracted region by looking at $F^N(p_0) = p_N$. If the squared distance of the point p_N from the origin is smaller than a threshold value β, p_0 is said to belong to the basin of attraction. The definition of a basin of attraction thus reads

$$A_F^N = \left\{ p_0 : \left\| F^N(p_0) \right\| < \beta \right\}, \tag{12}$$

where the norm $\left\| F^N(p_0) \right\| = (x_N^2 + y_N^2)$ is the squared euclidean distance between p_N and the origin.

4.1. Computing Filled Julia Sets

The escape–time algorithm is a perfect candidate to be implemented on the GPU, for the same operations have to be performed for all the initial points p_0. The coordinates of p_0 obtained by the transformation of the incoming texture coordinates (see Eq. (11)) are used as an input to Eq. (10) to compute $F^N(p_0) = p_N$. For each point independently, Eq. (12) is then used to decide if the point belongs to the attracted region A_F^N (i.e., the filled Julia set).

4.2. Automatic Computation of Bifurcation Sets

The idea of integrating an automatic procedure for the computation of bifurcation sets is largely inspired by the work of R. Devaney [Dev89, Dev90]. He also points out the importance of these sets in the analysis of dynamic systems, showing that bifurcation sets can be seen as a catalogue of Julia sets. This is because each point in the bifurcation set represents a specific system map F with its specific dynamical properties. At that time, the required computations could still take hours (or even days), but due to the capabilities of modern graphics hardware it is possible to integrate these methods into an interactive application nowadays.

For the computation of bifurcation sets, we do not iterate the system (10) for different initial points $p_0 = (x_0, y_0)$, but for all different parameter values (points in parameter space) of two of the 26 free parameters in the system. This means that the transformed texture coordinates are assigned to two user–specified parameters (say, for example, d^g and d^h), and for all possible values (d^g, d^h) the same point $p^* = (x^*, y^*)$ is used as initial input to (10). We discuss the meaning of the *critical point* p^* later on. The bifurcation set with respect to d^g and d^h is then defined by

$$B_F^N = \left\{ (d^g, d^h) : \left\| F^N_{d^g, d^h}(p^*) \right\| < \beta \right\}. \tag{13}$$

This means that we compute the orbit of the critical point for different parameter values d^g, d^h and look whether these orbits tend to infinity or not. All those points (d^g, d^h) which remain below β belong to B_F^N. B_F^N can therefore be looked at as the basin of attraction in parameter space.

Computing the critical point p^* is actually the most difficult part in an automatic computation of bifurcation sets. The critical point of a map $F(p)$ is a point at which the first derivative of F vanishes and whose second derivative is not zero, thus $F'(p^*) = 0$ and $F''(p^*) \neq 0$. The reason for which an automatic computation is, in fact, not always possible for a system map defined in the real and not in the complex plane (i.e., $F : \mathbb{R}^2 \to \mathbb{R}^2$) is that, in general, $F'(p) \notin \mathbb{R}^2$.

In these cases, usually, the Jacobian J of the system is taken as differentiation of F, since it is the matrix of the partial derivations of $g(x,y)$ and $h(x,y)$, denoted by g_x, g_y and h_x, h_y. According to the theory, systems of the form (1) are differentiable (in a way that $F'(p) \in \mathbb{R}^2$) only if the partial derivations are continuous, and the Cauchy–Riemann equations are satisfied by them.[2] With the used mathematical representation, this is not the case, and we experimented if the Jacobian matrix J of the system can be used for the computation of p^*.

Consequently, we looked at the equation system $J(x,y) = 0$, solving $g_x = h_y = g_y = h_x = 0$. Obviously, this system is over–determined, for having four equations and only two unknowns, x^* and y^*. Nevertheless, in particular cases this reasoning yields sensible results. For instance, the point $x^* = 0, y^* = 0$ solves $J(x,y) = 0$, if the linear terms in the system are zero ($a_1^g = b_1^g = a_1^h = b_1^h = 0$). Since this already covers a considerable range of possible parameter constellations, it was implemented in the software, and the point $(0,0)$ is usually used as the default critical point. In the AttractOrAnalyst, moreover, the user is given the possibility to set x^* and y^* to experiment different critical points.

And since we are using a system definition with 26 free parameters, the user can also choose with respect to which two parameters the bifurcation set is generated. The incoming transformed texture coordinates are then assigned to the chosen parameters and B_F^N is computed. This makes possible the automated generation of bifurcation sets for all possible pairs of parameters. However, we advise the user to very carefully interpret the resulting bifurcation sets in the case the critical point is unknown.

4.3. Applying Colours

Once having computed A_F^N (respectively B_F^N), the simplest way of putting this information into colour is to assign (say) black to all the points for which $F^N(p_0) \leq \beta$ ($F^N_{d_g,d_h}(p^*) \leq \beta$) and white if not. However, the dynamic properties of the point sequences that yield to F^N can be used to achieve more appealing fractal images.

Basically, a more adequate colouring is achieved by allowing the user to specify two colours, C_s and C_f, for the attracted and the non–attracted region independently.[3] For both regions, different dynamic properties are used to create colour gradients starting from C_s to C_f.

4.3.1. The Non–Attracted Region

In the case of the non–attracted region (with $F^N > \beta$) the velocity of divergence is used, in a way that C_s will be assigned to regions of very fast divergence and C_f to regions that need a larger number of iterations to exceed the bound threshold β. Colours C in between both extremes are defined by

$$C = \left(C_s - \frac{n}{N}(C_s - C_f)\right)^e, \tag{14}$$

where n is the iteration number at which the orbit exceeds β for the first time. In order to give the user more control over the colour gradient, an exponent e, which can be set by the user, is used in the interpolation.

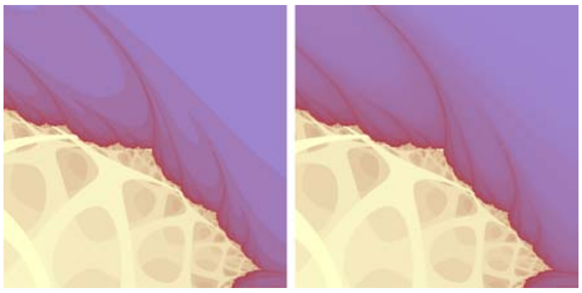

Figure 1: *The result of gradient smoothing.*

The left–hand image in Fig. 1 shows the result of Eq. (14). Since there are large regions of pixels which are coloured in the same way (for the velocity of divergence, computed by $\frac{n}{N}$, is itself not smooth), a gradient smoothing method was applied. Smooth gradients, as shown in the right–hand image, are achieved by taking into account the amount by which β is exceeded. The colour of a pixel, the orbit of which passes β at the nth iteration, is then determined by

$$C = \left(C_s - \left(\frac{n + \frac{\beta}{\|p_n\|}}{N}\right)(C_s - C_f)\right)^e, \tag{15}$$

where p_n is the nth element of the orbit of the considered point p.

4.3.2. The Attracted Region

Basically, the same colouring method is used for the attracted region, but no gradient smoothing is applied. The user defines C_s and C_f, and pixel colours are interpolated in between these two values. The difference comes with the dynamic property that is used in the interpolation. In this case, the difference between the norms of two consecutive elements of the orbit is mapped into colour. The colour of a pixel in the attracted region is defined by Eq. (14), but in this case, n corresponds to the first iteration for which $(\|p_n\| - \|p_{n-1}\|)^2 < 0.0005$. This dynamic property may not be that interesting from the scientific point of view, but fascinating colour patterns can be created in this way.

[2] If this is the case, F is called a complex analytic function, the derivation of which is also complex, since the partial derivations are pairwise equal, $g_x = h_y$ and $g_y = h_x$. For detailed considerations, the reader is referred to general literature on complex analysis, for instance [Sil72, BSMM97].

[3] Here, the RGBA–representation is used, i.e., $C = (R,B,G,A)^T$ with $R,B,G,A \in [0,1]$.

5. Selected Applications

Due to its high interactivity, the AttractOrAnalyst provides useful means in the image design process, for the new image is available without any time delay (as opposed to existing fractal visualisation tools). Visual results are shown in Fig. 6–9. The possibility to change the fractal parameters at runtime, furthermore allows for real–time fractal animations. In this section, we discuss a selection of applications of the presented method in the artistic context, even though we believe that the tool is also suited for scientific research of discrete dynamical systems.

5.1. Image Design using Bifurcation Sets

As explained in [Dev89] and [Dev90], a bifurcation set can be seen as a catalogue set for all different Julia sets. For instance, if the parameter point (d^g, d^h) does not belong to B_F^N, it follows that the attracted region A_F^N (the filled Julia set) computed with this parameter configuration is empty.

Visualising, at the same time, Julia sets and their corresponding bifurcation set, and allowing the user to navigate through the bifurcation set while updating the Julia images, facilitates greatly the image design process. In fact, this procedure is by far more intuitive to the non–expert, than changing specific formula parameters.

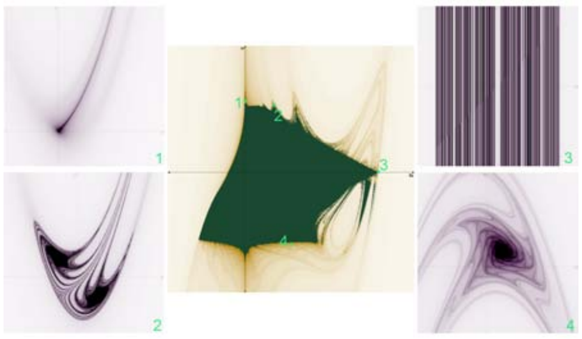

Figure 2: *The (d^g, b_1^g)–bifurcation set and four filled Julia sets are shown for the Hénon map.*

Fig. 2 illustrates this way of image design, called analysis mode in the application. With the mouse the user can move the (green) parameter points in the bifurcation set, and the Julia images corresponding to the points are updated in real–time. Since the user can choose freely the bifurcation parameters, the analysis mode provides a fast and intuitive way to explore all the shapes possible with the used polynomial representation.

5.2. Real–Time Video Production

Changes applied to the system map are updated without a noticeable delay, and also zooming and translation of the fractal works interactively. Provided the user changes the parameters in a smooth way, therefore, a consistent fractal animation is created on the run. Currently, these animations are transformed to common video formats by screen capturing software.

Saving the parameter sequence, however, which is possible at runtime, allows the created animation to be reproduced in the application. On basis of this data, the image sequence could be converted to standard video output formats in an offline process. Integrating this will make the presented tool a powerful instrument for artistic video production.

Moreover, the authors believe that, with a bit of training, the AttractOrAnalyst can be used directly in an interactive artistic context, such as the life–production of images synchronized to music. This makes the tool interesting to VJ (video jockey) artists.

Creating a procedure which maps audio data directly into parameter changes of the system will further support these types of application, and it is one of the issues to be considered in the future.

5.3. Multiple Layers

Another interesting application is the combination of real video or images with fractal animations. Interesting video blending effects and image modification techniques can be achieved by allowing a real image (or video) layer in addition to the fractal layer, and by merging the two layers into a single output. Fig. 3 shows an example of using multiple layers, for the creation of leaf textures that decompose with time.

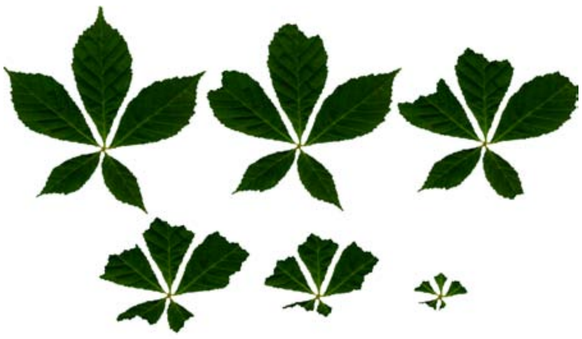

Figure 3: *A leaf gets "eaten up" in a fractal way.*

In this example, the image of a leaf is placed behind the fractal layer with a very small offset between the two layers. The attracted region of the system is completely transparent, and the non–attracted region is white just as the background of the leaf image. Therefore, only that part of the leaf image that lies within the attracted region is visible. Since the attracted region is fractal–shaped, a realistic impression of the natural decomposition process is achieved.

6. Conclusions and Future Work

This paper presented a software tool for the real–time visualisation and the interactive design of artistic fractal images, and it shows that, put into an interactive context, new applications of this fascinating objects become possible.

Fractal images are produced by mapping into colour certain dynamical properties of 2D discrete dynamical systems, most importantly, by computing the basins of attraction (filled Julia sets) of those systems. The escape–time algorithm, used for this computation, is perfectly suited to be implemented on the GPU, since, for each initial point independently, the same iteration sequence is to be evaluated. In order to further speed up the algorithm, the underlying dynamical system, represented by two real–valued functions $g(x,y)$ and $h(x,y)$, was transformed into a vector form which allows the use of hardware supported dot products in the computation.

In the program, which we called the AttractOrAnalyst, the user has access to a large number of parameters, including 26 free parameters that define the dynamic system, routines for interactive zooming and translation, as well as for the colour definition. Having at hand a visualisation method that works interactively, means that the user is immediately confronted with the visual result of the changes applied to the program. This enables real–time fractal animation which is not possible with existing fractal tools.

Another novelty presented in this paper is a method for the automatic computation of bifurcation sets for all possible pairs of system parameters. (The most popular example of a bifurcation set is the Mandelbrot set.) The program implements an analysis mode in which a particular bifurcation set and different filled Julia sets are visualised at the same time (see Fig. 5). To scientists and students, this gives the possibility of understanding by visual means the dynamics of the underlying mathematical formulas.

And it helps the artist in the design of beautiful dynamic images. Since bifurcation sets can be looked at as catalogues of different Julia sets, they provide very intuitive means for exploring new fractal shapes. In the future, the program interface will be based more strongly on this metaphor, which, we believe, will further facilitate the image design process.

Also the production of real–time fractal animations will profit from a more intuitive interface design. Real–time fractal animation is already possible, since all parts of the AttractOrAnalyst perform at interactive rates. In the current state of the application, video capturing tools can be used to convert the created live animations into video data. The future implementation of methods to import from and export to standard video file formats, however, will not only improve the production of fractal videos, but it will also make possible to mix fractal animations and real video in an artistic way. In this way, the presented tool can become even more useful in the creation of aesthetic visual content.

Acknowledgements

This work has been supported in part by TIN2007-68066-C04-01 project from the Spanish Government.

References

[AJCJ93] A. J. Crilly R. A. E., Jones H. (Eds.): *Applications of Fractals and Chaos – The Shape of Things*, first ed. Springer Verlag, 1993.

[Bar88] Barnsley M.: *Fractals Everywhere*. Academic Press Professional, Inc., San Diego, CA, USA, 1988.

[BSMM97] Bronstein I., Semendjajew K., Musiol G., Mühlig H.: *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Frankfurt am Main, Thun, 1997. 3. Ausgabe.

[Dev89] Devaney R. L.: *An Introduction to Chaotic Dynamical Systems*, second ed. Addison–Wesley Publishing Company, 1989.

[Dev90] Devaney R. L.: *Chaos, Fractals, and Dynamics – Computer Experiments in Mathematics*, first ed. Addison–Wesley Publishing Company, 1990.

[Fra06] Fractal explorer, 2006. http://www.eclectasy.com/Fractal-Explorer/index.html, (accessed November 17, 2007).

[Gif] Giffin N.: Fractint. Fractint Homepage, http://spanky.triumf.ca/www/fractint/fractint.html, (accessed November 17, 2007).

[JCVT93] J. C. Vassilicos A. D., Tata F.: *No Evidence of Chaos But Some Evidence of Multifractals in the Foreign Exchange and the Stock Markets*, first ed. Springer Verlag, 1993, pp. 249–265.

[Jul18] Julia G. M.: Mémoire sur l'itération des fonctions rationnelles, 1918. Memoir on iterations of rational functions, Translated in English by A. Rosa, 2001, http://www.emis.de/journals/AMEN/.

[JX93] Jarret D., Xiaoyan Z.: *The Dynamic Behaviour of Road Traffic Flow: Stability or Chaos?*, first ed. Springer Verlag, 1993, pp. 237–248.

[Lee] Lee P. N.: Fractal links. Fractal Software Programs & Links on Paul N. Lee's website, http://home.att.net/P̃aul.N.Lee/Fractal_Software.html, (accessed November 17, 2007).

[Lor63] Lorenz E. N.: Deterministic nonperiodic flow. *J. Atmos. Sci. 20* (1963), 130–141.

[Man97] Mandelbrot B. B.: *Fractals and Scaling in Finance*. Springer, New York, Berlin, Heidelberg, 1997.

[PL90] Prusinkiewicz P., Lindenmayer A.: *The Algorithmic Beauty of Plants*. Springer-Verlag New York, Inc., Berlin, 1990.

[Sil72] Silverman R. A.: *Introductary Complex Analysis*, first ed. Dover Publications, Inc./ New York, 1972.

[Ult07] Ultra fractal: Advanced fractal animation software, 2007. http://www.ultrafractal.com, (accessed November 17, 2007).

[Zha03] Zhao X.-Q.: *Dynamical Systems in Population Biology*, first ed. Springer Verlag, 2003.

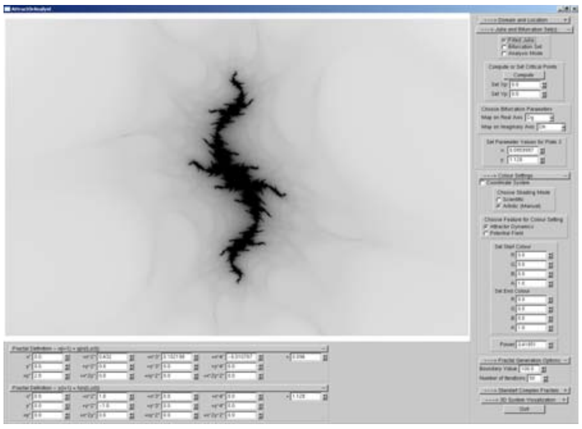

Figure 4: *The AttractOrAnalyst showing a filled Julia set.*

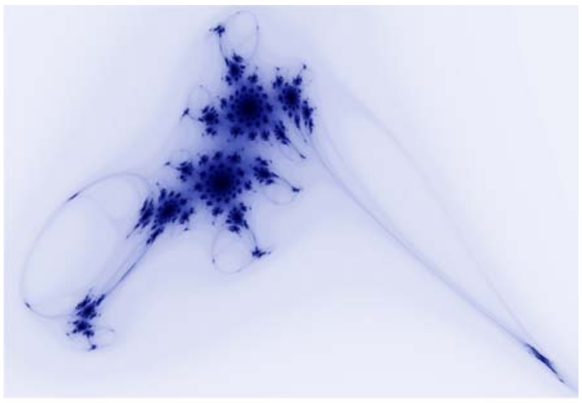

Figure 7: *A typical fractal image.*

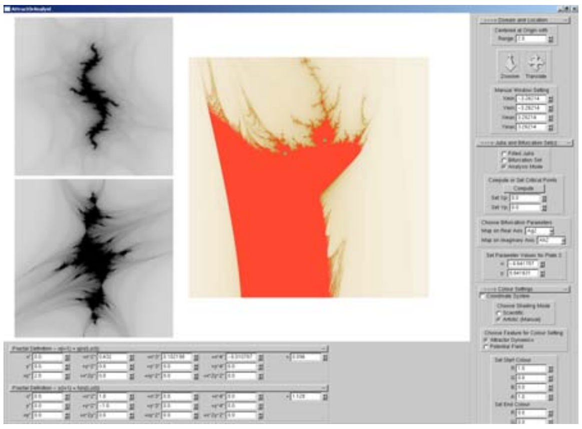

Figure 5: *The AttractOrAnalyst in the analysis mode.*

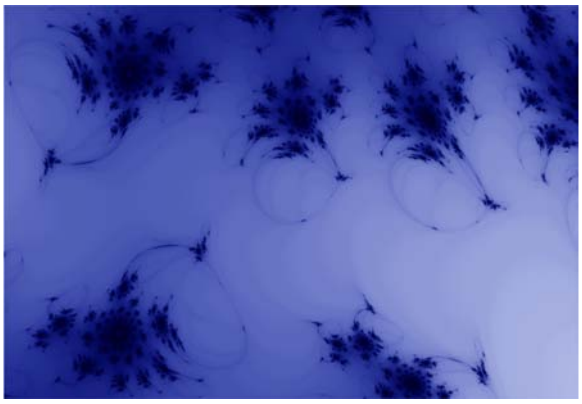

Figure 8: *The program allows interactive zooming into the fractal.*

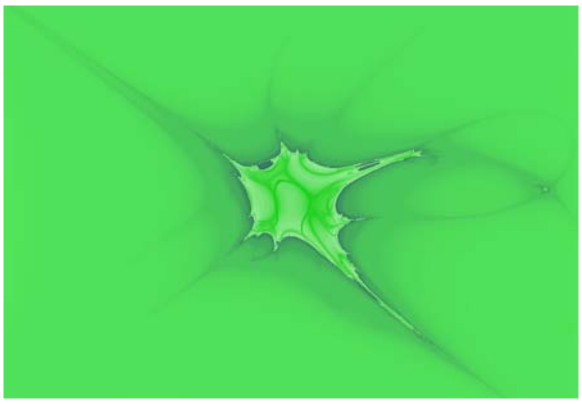

Figure 6: *Painting with mathematical formulas.*

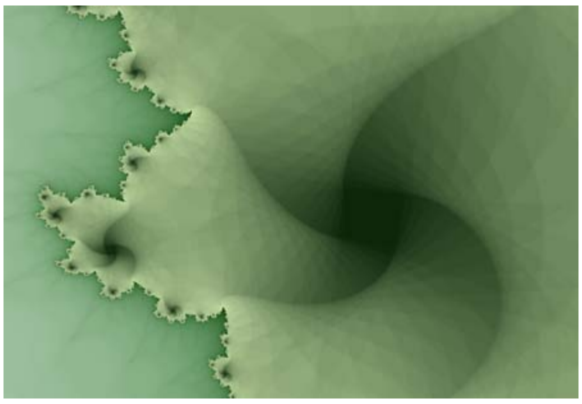

Figure 9: *Another fractal example.*