Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

Evolving the Mandelbrot Set to Imitate Figurative Art

Jeffrey Ventrella
jeffrey@ventrella.com

Abstract. This chapter describes a technique for generating semi-abstract figurative imagery using variations on the Mandelbrot Set, evolved using a genetic algorithm. The Mandelbrot Set offers an infinite supply of complex fractal imagery, but its expressive ability is limited, as far as being a "material" for visual manipulation by artists. The technique described here achieves a unique and varied brand of imagery by manipulating the mathematical function that generates the Set in a way that might seem unsavory from the standpoint of complex analysis – but it is very rich in terms of visual possibilities. Inspired by the author's earlier interest in creating animalistic and pseudo-figurative forms by tweaking the function, a technique was developed to evolve figurative forms using a digital image as the objective fitness function for a genetic algorithm. Experiments reveal that the function has limits in terms of its ability to generate forms that imitate specific detailed images. But this is actually a desired quality, as it lends an enigmatic quality to the resulting artworks. These limitations also elicit questions about the ability for parametrically-based imaging systems (like fractals) to produce representational art. The analogy to genetics and animal morphology is presented, and this is used as a framework to describe the behavior of the Set as it is tweaked, and pulled "out of the complex plane" (the canvas upon which the Set is normally painted). It provides a genetic vocabulary for describing, and thinking about, figurative art-making.

1 Introduction

The art of portraiture includes many art mediums and many styles. In the case of self-portraiture, an artist's choice of medium is sometimes the most important aspect of the work. A love for math and art, and an irreverence concerning the massacring of math equations for visual effect, inspired the medium described in this chapter. It evolves manipulation of the Mandelbrot Set to imitate shapes in digital photographs of human and animal figures. To exploit the Mandelbrot Set's potential for this, the technique emphasizes shading the "flesh" (the interior of the Set), rather than the outside, as is normally done. No image has been generated that looks exactly like a specific figure in detail - nor is this the goal. But the images do have the essence that they are "trying" to imitate something. As an example, Figure shows evolved images that are all based on a single digital image of the author's head. The technique for generating these images will be explained near the end of the chapter.

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

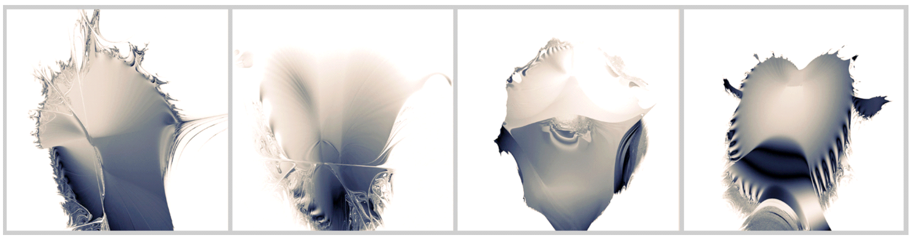

Fig. 1. Four examples based on an image of the author's head

The technique is partly inspired by two movements in Modernist painting: abstract expressionism and surrealism. But in this case, the effect is not achieved by the physics of paint on canvas and an artist's dialog with an emerging image. Instead, the effect is achieved by dynamics in the complex plane (the canvas upon which the Mandelbrot Set is painted) and an artist's algorithmic searching methods.

After developing many artworks by painstakingly adjusting numerical parameters, an interactive interface was added to help automate the selective process and to eliminate the need to control the numbers directly. Then a question came up: is it possible to evolve these kinds of images automatically using a genetic algorithm and an image as the objective fitness function? Also, given a specific image as a fitness function, what are the limits of the Mandelbrot Set - and the array of parameters added to increase its dimensionality and plasticity - to imitate images? How evolvable are these mathematically-defined forms? And for the sake of image-making, are not other parametric schemes more evolvable - such as Koch fractal construction, IFS, chaotic plots, L-systems, cellular automata, etc? General questions like this will be threaded throughout this chapter, touching upon the nature of the Mandelbrot Set, and the notion of genetic art as imitation of natural form.

The Mandelbrot Set has a special place in the universe of visual materials available to artists for manipulation - it has a peculiar complexity all its own. And the fascination with its arbitrary symmetry and beauty can reach near-religious levels. Roger Penrose believes, as do many mathematicians and physicists, that there is a Platonic truth and universality to mathematics, and that the Mandelbrot Set demonstrates this. It was not, and could never have been, invented by a single human mind [Penrose, 2004]. It could only have been discovered. On the other hand, [Lakoff and Nunez, 2000] present a convincing thesis that mathematics springs out of the *embodied* human mind - math is an invention of the human mind which expresses our physical relationship with the world, and the metaphors that have evolved in our brains. Math is not a universal truth waiting to be "discovered", but a language of precision, contingent upon the nature of the human brain and the ecology from which it evolved.

We will not try to address this debate in this chapter. In fact, the technique described in this chapter circumvents the debate by way of a decidedly un-platonic manipulation of the math. This kind of manipulation may be unsavory from a mathematical standpoint, but from an artistic standpoint, it breaks open the canvas of the complex plane to a larger visual vocabulary.

Even when taking this irreverent stance, and choosing to yank the function out of

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

the realm of complex analysis, a tweaked Mandelbrot Set still possesses remarkable properties, and it appears not to be as plastic and malleable as a lump of sculptor's clay. It is more like an organism, whose entire morphology, at every scale, is determined by a specific genetic code and the constraints of its iterative expression. Rotational behavior is characteristic of the complex plane, and is caused simply by the multiplication of two complex numbers. Curvilinear, rotational, and spiral-like features are common, and they are still present when the Mandelbrot function has been manipulated, as seen in the detail at right of Figure 2.

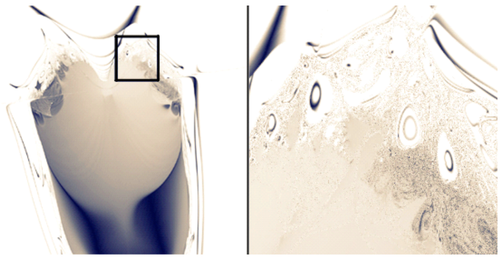

Fig. 2. Curvilinear forms remain upon manipulation of complex plane equation

And so, while this approach may be irreverent, what can be learned about the nature of this particular artistic canvas elevates admiration for the magic of the Mandelbrot Set, and its extended family of related fractals.

Like the playful and evocative variations on complex plane fractals created by [Pickover 1990] and others, the technique described here is heavy on the tweaking. It is not focused on finding interesting regions in the pure un-altered Set (as in "Mandelzoom"). It is more like sculpting than nature photography. Or perhaps it is more like genetic engineering than painting. For this reason, in the remainder of this chapter the technique will be called "Mandeltweak".

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

1.1 Genetic Space

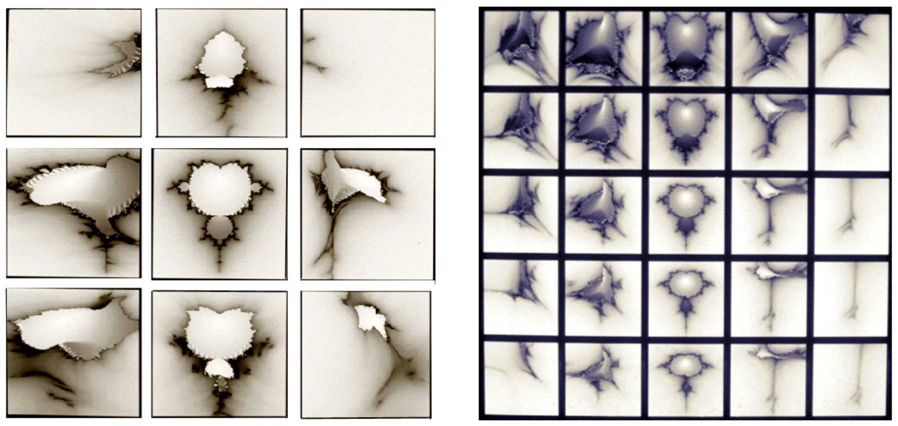

Fig. 4. Mandelbrot Set in the middle of two 2D genetic spaces

Inspired by a metaphor which [Dawkins, 1986] used to describe species genotypes as points existing within a vast multi-dimensional "genetic space", Mandeltweaks are likewise considered as existing within a genetic space. Figure 4. shows two genetic spaces, 3X3 on the left side and 5X5 on the right. In each image, two genes vary in even increments, one in the horizontal dimension, and the other in the vertical dimension. The values are default in the middle of each space, and this is where the Mandelbrot Set lies. The image on the left is of a large nine-panel photo series shown in various gallery settings.

1.2 Genetic Parameters

The number of ways one can alter a digital image is practically infinite. Just consider the number of plug-in filters available for software tools like Photoshop. But while an arbitrary number of filters, distortions, layerings, etc. could have been applied with parameters for manipulating Mandelbrot images in the pixel domain, the choice was to keep all variability to within the confines of the mathematical function. The game is to try to optimize the parameters of the manipulated Mandelbrot equation so that the resulting images resembles an ideal form (or in some cases an explicit target image). This is a challenge because the mapping of genetic parameters to image attributes (genotype to phenotype) is non-trivial, and unpredictable.

Mandeltweak is not just an art-generating technique - it is a sandbox for exploring the notion of genetic expressivity, imitation, and image perception. Throw in a little Dawkins, a bit of Picasso, a touch of Rorschach, and a good dose of Mandelbrot math, and you have the basic motivational profile behind this exploration.

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

2 Background

The emergence of abstractionism in Modern painting is often cited in reference to the invention of photography at the turn of the century. When photography came onto the scene, painters felt less need to replicate reality, and they became more interested in forging a true painterly art - true to the medium. This is part of the program of Modernism in general.

Perhaps because of an innate desire in humans to replicate reality, and the strong science influence in early computer graphics development, an implicit goal was optical realism. The pursuit of realism continues with refinements to ray-tracing, radiosity, and other 3D rendering techniques. Recently we hear of new techniques that emulate painterly styles, described as "non-photorealistic" - a curious term - as if there were something fundamental about photorealism to base descriptions of other visual styles.

But in fact, the computer has nothing in its physical nature that would lend itself to any particular imaging style - constrained either by canvas and brush, or by film and lens. There is however one thing about the computer that would qualify as part of its true nature, and therefore, a determinant in an artistic style - that is *information processing* (of great quantity and speed). The sciences of chaos and complexity, and techniques of fractal-making have brought about a revelation that the information-processing power of computers could generate beauty, complexity, and organic form reminiscent of familiar natural processes. New visualization realms have since blossomed as forms of visual language that are truly unique to the medium.

While one may not be able to claim that these image-making techniques produce "realistic" art (in the same sense that a painting by Piero della Francesca is realistic) they do mimic some very compelling and familiar complex aspects of nature. The genetic algorithm is an example of a recursive tool which, when combined with some parametric-based image-making system, such as fractals or cellular automata, can create naturalistic processes, and sometimes realistic forms and motions. In the European renaissance, *perspective* was a tool that significantly advanced both art and scientific thinking. In the late 20th century, the computer (and recursive processes by which simple rules are applied repeatedly) is a tool that has propelled another renaissance. Chaos, complexity, and fractals reveal another aspect of reality–which is churning away at every second - it reveals the deep structure and iterative evolution of form in the world.

The Mandelbrot Set

The Mandelbrot Set has been called "the most complex object in mathematics" [Dewdney, 1985]. It is like the mascot of this new renaissance - replicated in popular science books like a celebrity. When looking at it in its whole, it looks like a squashed bug - not pretty. But it's deep remote recesses reveal amazing patterns that provoke an aesthetic response. Is the Mandelbrot Set a form of abstract art? No. Not according to the definition of abstractionism as human-made art that is "abstracted" from nature, with human interpretation. According to the platonic view, the Mandelbrot Set "just is". It has been hiding in the folds of complex mathematics until humans and computers revealed it. But consider the canvas upon which the Mandelbrot Set is painted. We can alter our mathematical paint brush from $z = z^2+c$ to something else which is not so clearly defined, and make a departure from its platonic purity. We can render images with the kind of interpretation, *imprecision*, and poetry that distinguishes art from pure mathematics. Iteration in the complex plane, then, is considered a new art form, which emerges in the *Recursion* Renaissance.

It is possible that the Mandelbrot Set was discovered a handful of times by separate explorers, apparently first rendered in crude form by Brooks and Matelski [Penrose, 2004, p 17]. Benoit Mandelbrot discovered it in the process of developing his theory of fractals, based on earlier work on complex dynamics by Fatou and Julia [Mandelbrot, 1977]. His work, and subsequent findings by [Douady and Hubbard, 1985], helped popularize the Set, which now bears Mandelbrot's name.

Briefly described, the Set is a portrait of the complex function, $z = z^2+c$, when iterated in the two-dimensional space known as the complex plane, the space of all complex numbers. When the function is applied repeatedly, using its own output as input for each iteration, the value of z changes in interesting ways, characteristically different depending on c (i.e., where it is being applied in the plane). The dynamics of the function as it is mapped determines the colors that are plotted as pixels, to make images of the Set. Specifically, if the magnitude of z grows large enough (>2) and "escapes" to infinity, it is considered outside of the Set. The inside is shown as the black shape on the left of Figure 5.

The Set has become a favorite subject for computer art. On its boundary is an infinite amount of provocative imagery. The most common visual explorations involve zooming into remote regions and applying color gradations on the outside of the Set near the boundary. This is somewhat like simple point-and-shoot photography, with a bit of darkroom craft added. However a variety of techniques have been developed along these lines to search-out interesting remote regions, including an evolutionary algorithm developed by [Ashlock, 2006]. A particle swarm for converging on the boundary, under different magnifications, was developed by [Ventrella, 2005].

Deeper exploration into the nature of the Set is achieved by manipulating the mathematics, to reveal hidden structures. [Pickover, 1990] has fished out a great wealth of imagery by using varieties of similar math functions. His books on fractal/chaotic image-making techniques have contributed to the popularization of computer-generated fractal art.

[Peitgen, et. al, 1988] describe a variety of complex plane fractals, with ample mathematical explanations. [Dickerson 2006], as well as many others, have explored higher-order variations of the function to generate other "Mandelbrot Sets" as they are sometimes called. According to Dickerson, the equation can be generalized to: $z = a * \mathrm{f}(z) + c$, where a is a scale constant and $\mathrm{f}(z)$ is one of a broad range of functions, such as higher powers of z, polynomials, exponentials and trigonometric functions. As an example of a higher-order function: $z_{t+1} = z_t^3 + c$ creates the shape shown at the right in figure 5. This "Mandelbrot cubed" function is used in some experiments described below.

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

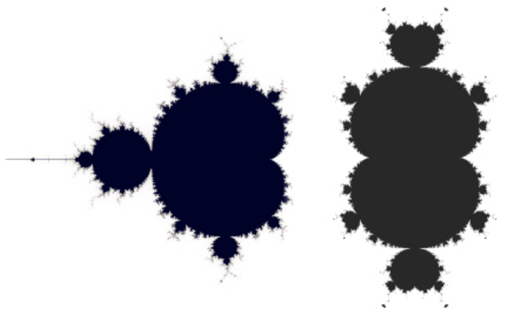

Fig. 5. Mandelbrot Set (left), and Mandelbrot "cubed" (right)

The algorithms used in Mandeltweak can also be considered as a generalization from $z=z^2+c$ to $z=$ f(z)+c, only in this case, the complex nature of the number z is violated: the real and imaginary components of the number, as exposed in the software implementation, are manipulated. In the *Phoenix* fractal, discovered by Shigehiro Ushiki [5], the real and imaginary parts of the equation are likewise tweaked separately. More examples of separate treatment of the real and imaginary parts of the equation are emerging. These are described in various web sites on the internet, including a technique developed by [*eNZed Blue*, 2005].

2.3 Evolutionary Art

One way an artist can approach mathematically-based image-making is to identify a number of variables that determine visual variations and to tweak them to suit his/her own aesthetic style. The more variables available for tweaking, the more the artist can potentially tweak to reach some level of personal expression. The problem is that in most cases the variables are interdependent, and it is hard to predict the effects of variables in combination. Besides, most artists would rather not use numbers to manipulate visual language. Evolutionary computation to the rescue.

Evolutionary Art (EA) is a relatively new addition to a long history of art-making tools. In EA a computer software program becomes a creative collaborator to the artist. The most common process is *interactive evolution* (also called "aesthetic evolution" or "aesthetic selection"). In contrast to the standard genetic algorithm (GA) [Goldberg, 1989], the selection agent is not determined by an objective fitness function, but rather by a human observer/participant (the artist), whose aesthetic choices guide the direction of evolution in a population of variations of an artwork. [McCormack, 2005] outlines a number of problems that remain open as we articulate and refine the tools for EA. Among these are the problem of finding interesting and meaningful phenotypes, which are capable of enough variation to allow for artistic freedom.

Among the earliest examples of evolutionary art are the work of Latham [Todd and Latham, 1992]. [Sims 91], has applied genetic programming [Koza, 1992] to various visual realms. [Rooke, 2001], [Ventrella, 1994], and others have developed evolutionary techniques for generating visual art. Most of this work is abstract (or "abstractsculptural" - rendered with 3D shading), and use interactive evolution as the technique for breeding images.

3 Technique

The standard black-and-white figure of the Mandelbrot Set is created as follows: on a rectangular pixel array, determine whether each pixel lies inside or outside of the Set. If inside, color the pixel black, otherwise, color it white. This 2D array of pixels maps to a mathematical space (x, y) lying somewhere within the range of -3 to1 in x, and -2 to 2 in y. The values x and y represent the real and imaginary components of a window on the complex plane. The function $z_{t+1} = z_t^2 + c$ is iterated repeatedly. The value c is a complex number (the 2D location in the complex plane corresponding to the pixel), and z is a complex number which starts at (0,0i) and is repeatedly fed back into the function. This is repeated until either the number of iterations reaches a maximum limit, or the magnitude of z exceeds 2. If it exceeds 2, it is destined to fly off to infinity, and this signifies that it is outside of the Set. Expressed in pseudocode:

```
For each pixel do:
{
    map screen pixel values (i,j) to real number values (x,y)
    zm = 0
    zx = 0
    zy = 0
    timer = 0

    while ( zm < outsideTest AND timer < maxIterations )
    {
        z1 = y + zy * zy + zx * -zx
        z2 = x + 2.0 * zx * zy
        zm  = z1 * z1 + z2 * z2
        zx = z2
        zy = z1
        timer = timer + 1
    }

    if ( zm < outsideTest )
        set pixel color black
    else
        set pixel color white

    plot pixel(i,j)
}
```

This is not implemented as optimally as it could be, but it exposes more variables for manipulation - which is part of the Mandeltweak technique.

MaxIterations could be any positive number, but higher numbers are needed to resolve a detailed definition of the boundary, which is especially important for high magnifications. Mandeltweak does not require high magnifications, and so this value ranges from 30 to 50. The mapping of (i,j) to (x,y) determines the location and the magnification in the complex plane. Complex number z is represented by zx and zy. The variable zm is the squared magnitude of z. The variable 'outsideTest' is set to 4 for normal Mandelbrot plotting, but it could be set to other values, as explained below.

Note that swapping x and y rotates the Set 90 degrees. The Mandeltweak approach is to make the real number axis vertical, orienting the Set as if it were a fellow entity with similar bilateral symmetry, as shown in Figure 6.

3.1 Coloration

The typical coloration scheme for Mandelbrot Set images applies a color gradient on the outside of the Set which maps to the value of timer. A smoother version of the outside gradient, described by [Peitgen, 1988] can be achieved by replacing the value timer with: 0.5 * log(z) / 2^{timer}. Setting outsideTest to higher values, such as 1000, makes it smoother. Mandeltweak uses this technique, and in most cases, the background is rendered with a light color, which shifts to a dark color very close to the boundary. The effect is a mostly-light colored background, with some emphasis on the complex boundary, as seen in figure 6d.

The inside of the set is colorized with a gradient that maps to *zm*+*za*, where *za* is a special value determined by analyzing the orbit of *z* during iteration. As the value if *z* jumps around the complex plane, the angle between each jump is recorded, and when iteration is complete, the average angle is calculated. This is normalized, and then used as a value that roughly corresponds to the characteristic of *z*'s orbit. In addition, before being applied to the color gradient, both *zm* and *za* are modulated by sine waves whose frequencies and phases are evolvable. The result is that a large variety of possible coloration scenarios are possible, and it accentuates hidden features in the *mathematical flesh*. Both *zm* and *za* are normalized (0-1) before used for coloration.

Figure 6 shows the default Set (a), followed by rotation (b), colorization (c), and finally tweaking (d), which is explained below.

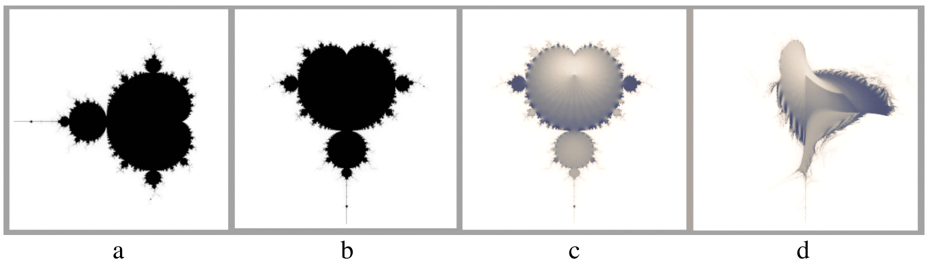

Fig. 6. (a) The default Set, (b) rotated, (c) colorized, (d) tweaked

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

3.2 Morphological Tweakers

The kernel of the Mandelbrot equation is in the two lines above that express $z = z^2 + c$:

```
z1 = y + zy * zy + zx * -zx
z2 = x + 2.0 * zx * zy
```

Arbitrary morphological tweakers are added, which are "pro-visual", and "a-mathematical" ("anti-mathematical" would be too-negative a term). Mother Nature, in her evolutionary creativity, works with phenotypes and their interactions in ecological reality - that is her artistic canvas. Likewise, it is the phenotype that provides the design-space for Mandeltweak. A typical set of tweakers are as follows:

```
z1 = y + (zy+p1) * (zy*p2) + (zx*p3) * ((zx*p4) + p5) * p6
z2 = x + p7 * (zx+p8) * (zy+p9)
```

p1 through p9 are real number variables, most of which are set to 0 as their default values. p2, p3, and p4 are set to 1; p6 is set to -1; and p7 is set to 2 as their default values. Each tweaker can deviate from its default value within a range (extending in both the negative and positive directions). Each tweaker has its own range, and it controls a unique visual manipulation. For instance, p2 is responsible for the distortion shown in Figure 6d. In addition to these morphological tweakers, the following lines:

```
zm  = z1 * z1 + z2 * z2
zx = z2
zy = z1
```

are expanded as follows:

```
zm = z1 * z1 + z2 * z2
zm = zm * (1-p10) + z2 * p11
zx = z2 * (1-p12) + (z1*p13)
zy = z1 * (1-p14) + (z2*p15)
```

Also, before the iterative loop, the lines:

```
zx = 0
zy = 0
```

are expanded to:

```
zx = p16
zy = p17
```

where p16 and p17 are set to 0 as default. When set to non-zero values, these tweakers give an artificial jump-start to the value of z before entering the iteration loop, affecting the resulting morphology.

Since the kernel of the Mandelbrot equation could also expressed as

```
z1 = y + zy*zx + zx*zy
z2 = x + zx*zx - zy*zy
```

a different set of tweakers could be applied as follows:

```
z1 = y + ((zy*p1 )+p9 )*((zx*p2 )+p10) + ((zx*p3 )+p11)*((zy*p4 )+p12)
z2 = x + ((zx*p5 )+p13)*((zx*p6 )+p14) - ((zy*p7 )+p15)*((zy*p8 )+p16)
```

This is a more orderly expansion, and it provides a larger genetic space than the p1-p9 shown in the original expression above. New and intriguing forms exist in this space as well.

The examples of tweakers that we have just seen are not the whole story. However, to describe all the variations that have been tried could take potentially many more pages. These examples should give a general sense of how the technique is applied. The reader, if interested in trying out variations, is encouraged to choose a set of tweakers that satisfy individual aesthetics. Mandeltweaking is more an Art than a Science.

If you were to scale the variable x (the real number part of *c*) by something like 1.4, the Set would get squashed along the x axis - this is not considered a genetic tweaker, as it amounts to a trivial pixel-wise distortion, or a uniform warping of the complex plane. The tweakers described above instead change the dynamics of the math as it iterates, focusing on the expansion of the complex number *z*, which causes unpredictable changes in morphology.

These tweakers are stored in an array (a phenotype), which is generated from an array of genes (a genotype). The genotype holds the normalized representations of all the tweakers, in the range 0-1. Before an image is generated, the genotype is mapped to the phenotype, taking into consideration the tweakers' default values and ranges. This normalized genotype representation will come in handy as explained later when a genetic algorithm is applied to the technique.

Besides color and morphology tweaking, the entire image can be magnified, rotated, and translated by values which transform the window onto the complex plane before the calculations are applied. There are four tweakers for this, and they determine angle of rotation, magnification, translation in x, and translation in y. These can be thought of as transformations of the digital microscope that views the complex plane.

When these tweakers are set to their default values, the function produces the Mandelbrot Set. The values p1 through p17 determine the morphologies within a *superset* of the Mandelbrot Set, called *Mandeltweak*. Any mutation offset from a default value will push the mathematics into a dimension that does not obey the normal rules of complex numbers. All of the tweakers have clearly-defined defaults, and so all experiments can be considered as deviations from the true Set - deviations from its home in the complex plane. This is the control point - the point of registration from which to build a visual vocabulary describing this multidimensional genetic space.

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

3.3 Interactive Evolution for Artistic Breeding

Originally, the Mandeltweak software was given an interface to generate images which could be repeatedly reviewed and altered, until interesting and provocative forms were resolved. These sessions sometimes involved hundreds of adjustments, in terms of both morphology and color. Considering the accumulated memory in the artist's mind of the variations being explored, you could say that a sort of wetware genetic algorithm was being run, resulting in a convergence towards a desired image. These sessions were sometimes very long. Hundreds of images were produced, and stored mostly as photographs, and exhibited in art shows and on the web [Ventrella, 2004]. Figure 7 shows six examples of images created with this process.

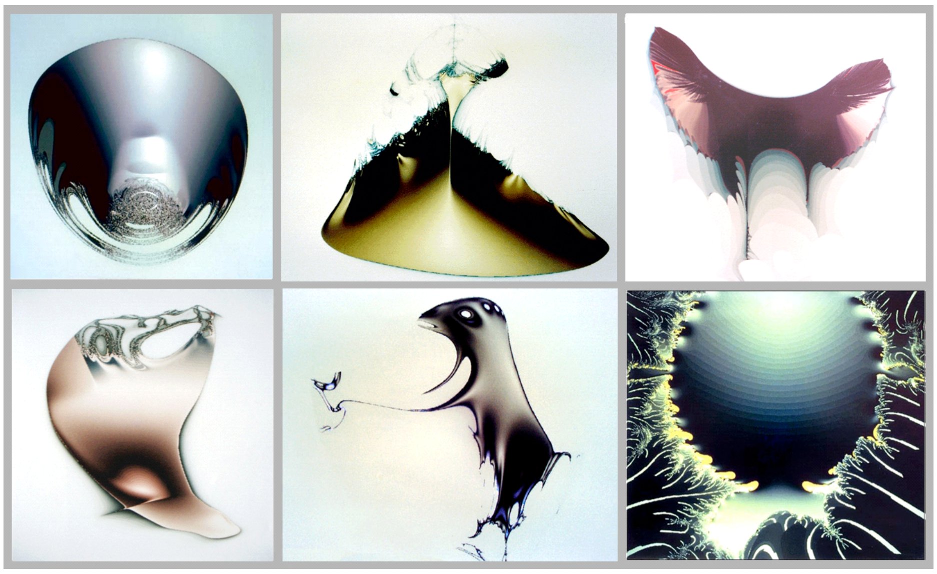

Fig. 7. Some early examples of Mandeltweaks

This experience set the stage for developing a modified genetic algorithm with an interactive evolution interface, whereby the artistic choices could be stored in a population of tweak settings, and re-circulated within the population to offer up combinations of favorite images. This was a great improvement - a natural application of evolutionary programming to the problem domain. This interactive evolution scheme is described below.

1. Initialization

A population of genotypes is generated, with their genes initialized randomly, distributed evenly in the range (0-1). Population size is usually set to around 100. The genotypes are also initialized with random fitness values ranging from 0 to 1 (which is meaningless at first, but as we shall see, these fitness values will gradually take on meaning).

2. Iteration

The iterative loop has three basic components: (a) mating, (b) evaluation, and (c) death. This is explained below.

(a) Mating via *Tournament Selection*

Two random, relatively fit genotypes are chosen as parents, each by way of a competition for relative fitness, as follows: (1) two competitor random genotypes are chosen, and their fitness values are compared. The one with the highest fitness is kept as "parent 1". (2) Another competition is run, and the winner is chosen as "parent 2". These two relatively-fit parent genotypes then mate to produce one offspring genotype using crossover, with some chance of mutation. During mating, standard crossover and mutation techniques are used, with crossover rate $C = 0.2$, and mutation rate m ranging from 0.01 to 0.1 in most experiments. While parent genotypes are being read to generate the offspring genotype, gene-by-gene, there is C chance of the parent genotype which is being copied to the offspring to swap to the other parent. And there is m change that the gene being read will mutate. If mutated, a random number is added to the gene value ranging between -1 and 1, weighted towards zero, with Gaussian distribution, to favor smaller jumps over larger jumps, but not ruling out bigger jumps. If after mutation, the gene value falls out of the normal interval 0 to 1, it wraps around to keep the value normalized.

(b) Evaluation

The resulting offspring genotype is then mapped to a phenotype to determine the tweakers for generating a new image. The user evaluates this image by giving it a value in the range of (0-1). Different versions have been explored as far as inputting this value, including binary (0=bad vs. 1=good); a three-choice scheme (bad-medium-good); and continuous (clicking the mouse on the screen, with the location from left to right or bottom to top determining the value from 0 to 1). There are pros and cons to each of these input techniques, which will not be covered here - what's important is that a value ranging from 0-1 is provided by the user.

(c) Reproduction and Death

Once a fitness value has been provided for this image, the associated offspring genotype replaces the least-fit genotype in the population.

This process is iterated indefinitely. At first, the user experiences no progress, especially in the case of large populations (like more than 100), but in time, the choices that the user has been making start to effect the population, and images begin to come up that are preferable, having visual qualities that the user had been responding positively to.

In some experiments, the selection of parent genotypes is set to not take relative fitness into consideration, and so any two genotypes can become parents. In this case, the only driving force for evolution is the fact that the least-fit individual is killed off to make room for the offspring. This causes slower convergence, but better exploration of the genetic space, but it appears not to have a major impact on the outcome.

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

3.4 One Image at a Time, One Mating at a Time

A common design in interactive evolution schemes is to present the user with a collection of images with variation, and for the user to compare these and make some selection based on that comparison. A major difference in this technique is that the user is presented with only one image at a time, and uses *memory* to compare with other images seen. The reason for this interaction design is that it enhances the experience of perceiving an individual image as a work of art, and allows the aesthetic sense to operate more like viewing art, rather than like shopping for a product. This interface is meant to allow the process of interactive evolution to be pure and direct - an evolving dialog between the artist's visual memory and a series of images, with an arc of aesthetic convergence that threads through the experience.

Many genetic algorithm schemes use *generational selection*: all the genotypes in the population are sized-up for fitness in one step, and then the entire population is updated to create a new generation of genotypes, which selectively inherit the genetic building blocks from the previous generation. For software implementation, a backup population is required in order create each new generation. In contrast, this scheme uses *steady-state selection*: it keeps only one population in memory, and genetic evolution is performed on that population one individual at a time. This is admittedly slower than the classic scheme, but it does have the effect of preserving the most fit genotypes at all times. And the grim reaper only visits the least fit.

3.5 Fitness Decay

At the start of the process, fitness values are randomly distributed from 0 to 1. Since relatively-fit individuals are chosen to mate and offer up Mandeltweak images for evaluation, the first images reviewed are essentially random in their relative quality. But this soon begins to change as the user provides meaningful fitness values. In addition to this, a global decay scalar *d* is applied to all fitness values at each iteration. (d = just under 1, typically 0.99). The initial effect of *d* at the beginning of the process is to allow genotypes that were randomly initialized with high fitness values to "back away" as new genotypes rise to the top of the fitness range as a result of positive user selection. The distribution of fitness values begins to reflect user choice.

Once the initial random distribution of fitness values has given way to meaningful values from user interaction, the decay operator then begins to serve a different purpose: that of allowing for meandering aesthetic goals. Individuals that were once considered fit are allowed to fade into the past, while their offspring take the spotlight. The decay effect roughly corresponds to memory. It avoids having the highest fit individuals dominate the population and prohibit new discoveries to take the lead. If there were no fitness decay, the population would lose any flexibility to respond to changes in the aesthetic directory. The fitness decay operator is like the evaporation of ant pheromones - chemicals released by ants for communicating which build up in the environment and permit ants to establish trails for foraging. If pheromone scent never decayed, ant colonies would not be able to adapt to changing distributions of

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

food, and their collective behavior would become rigid. Same with this fitness decay: it gives the user a chance to push the population in new directions when an aesthetic dead-end is reached.

The value *d* is important. If it is set too low (like, 0.9 - decaying too quickly), then the results of the user's choices will not stay around long enough to have an effect on the general direction of evolution. If it is too weak (like 0.999 - decreasing too slowly) then inertia sets in: user selections that were either "mistakes" (or choices that are no longer relevant) will stick around too long and make evolution inflexible. This value is sensitive to population size, user psychology, the nature of the evolvable imagery, mutation rate, and other factors.

The interactive evolution technique just described has a few notable properties:

(1) it always preserves the most fit individuals (imagine looking at a picture, liking it, and choosing to keep it in a box for future use - you can rely on it being there for a long time).

(2) it always overwrites the least-fit individual, which has the effect of increasing average fitness over time. (note that the lowest fit individual fell to its place either because the user put it there directly by selection, or else it slowly "faded into the past" as a result of fitness decay - in both cases, it is appropriate to replace it).

(3) it allows user aesthetics to change direction over time, and to re-direct the population.

3.6 Using a Digital Image as a Fitness Function

The most recent stage in this progression towards automating the process is to use an image as an objective fitness function. Instead of a user providing the fitness of a Mandeltweak based on aesthetics, the Mandeltweak is compared to an ideal image to determine similarity. The genetic algorithm for this scheme is the same as the interactive evolutionary scheme described above, except for three important differences:

(1) The human user is replaced by an image-comparison algorithm, which uses a single ideal image.

(2) There is no fitness decay operator. Fitness decay is considered a "psychological" mechanism, and is not needed in this case. Also, since the ideal image is static (as opposed aesthetic whim), there is no need for the flexibility that decay affords.

(3) Instead of setting all fitness values randomly at initialization, the initial genotypes are used to generate an initial population of Mandeltweaks to establish meaningful fitness values. This becomes the starting point for the iterative loop.

The ideal image is either painted in Photoshop, pulled off of a website, or snapped with a digital camera and then post-processed with Photoshop. Only gray-scale images are used, for three reasons: (1) it reduces the complexity of the experiment to fewer variables, (2) the addition of color was not found to contribute significantly to the perception of figurative form in the images, and (3) it was an artistic choice: to encourage the resulting images to resemble black-and-white portrait photography. Also, the outside of the Set is colorized only with white (with black gradation very close to the boundary), and all ideal images have a white background, to simplify the technique and to disambiguate figure vs. ground. To give the images a sepia-tone quality with a subtle blue shadow effect, the gray scale in the final image is altered slightly. While keeping pure black and white at the extremes, a slight blue shift in the lower range, and a slight orange shift in the upper range, are applied.

True Color Scheme

A true color scheme was explored at one time, in which extra genes were added to accommodate the red, green, and blue components of the image. This scheme was not fully explored, and it was decided that a hue-saturation-luminance scheme, as opposed to red-green-blue, would be better – this is high on the list of things to come back to in a subsequent version of the software.

3.7 Image Resolution

It was found that comparing a Mandeltweak image with the ideal image could be done adequately using a resolution r of only 50 pixels. So, the ideal image consists of r^2 (50*50 = 2500) pixels, where each pixel color is a shade of gray ranging from black to white in g = 256 possible values (0<=g<=255). When a Mandeltweak is generated so as to compare with the ideal image to calculate a fitness value, it is rendered at the same resolution r. The images are compared, pixel-by-pixel, and so there are r^2 pixel value differences used to determine the difference between the images, and thus the fitness.

Note that even though the Mandeltweak is rendered at a specific resolution for comparison, this does not mean that it could not be re-rendererd at a higher resolution. In fact, since the *genes* are what constitutes the representation of the image (NOT the pixel values), it could be re-rendered at any arbitrary resolution. What r represents, then, is the amount of image detail that could potentially be compared. And since the mimicking ability of Mandeltweak is limited only to general forms and approximate gray-scale values, it is not necessary to present it with high-resolution ideal images - the extra detail would be wasted.

3.8 Image Comparison

To illustrate how the comparison scheme works, consider the ideal image in Figure 8 of a black disk against a white background.

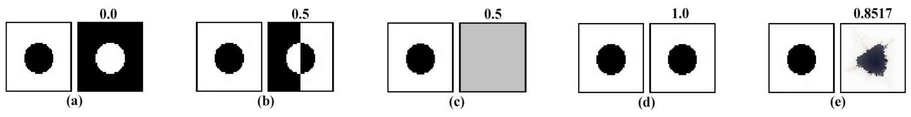

Fig. 8. Five examples showing images compared to an ideal image (black disk).

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

Compare each image to the right of the black disks in examples (a) through (e). In example (a), every pixel value is as different as possible (covering the complete range of pixel difference: 255), and so the resulting fitness is 0. In example (b), half of the pixel values are identical, and the other half are as different as possible, and so fitness is 0.5. In example (c), the image is filled entirely with gray value 128 (mid-range between black and white). In this case, fitness is 0.5, since the difference between a gray pixel and either black or white is 128. In example (d), all pixels are identical, and so fitness is 1.0. Example (e) shows a Mandeltweak that resulted from evolution in a population using the black disk as the fitness function. It was able to approach the ideal, reaching a fitness value of 0.8517.

Let us define p ($0 <= p <= 255$) as the difference in value of a pixel in the Mandeltweak image and its corresponding pixel in the ideal image. The fitness f of a Mandeltweak image is

$$f = 1 - P/r^2 \quad (0 <= f <= 1)$$

where P is the sum of all normalized pixel differences: $|p|/g$.

This technique uses a simple pixel-wise comparison. A few variations have been explored in order to encourage sensitivity to certain features. But nothing conclusive has come of this. There is certainly a lot of research, and many techniques, for feature-based image comparison, and it would make for an interesting enhancement to this technique. But for the preliminary purposes of these experiments, this simple scheme is sufficient.

4 Experiments

To help visualize the evolution of a population of Mandeltweaks, each individual's genotype is plotted as a row of rectangles. The gray value of a rectangle corresponds to the value of its associated gene, with the range 0-1 mapped to a gray scale from black to white. An example is illustrated in Figure 9.

Fig. 9. visualization of a genotype with gene values mapped to grayscale.

This genotype visualization in used in figure 10, which shows a population of 1000 Mandeltweak genotypes evolving to imitate an ideal image of the author's face (upper left). Four stages of the evolution are plotted, at times 0, 1000, 10,000, and 50,000. Fitness is visualized as the vertical height of the genotype, and convergence is revealed as similarity in genotype coloration. The Mandeltweak with the highest fitness in the population is shown at the top of each plot, and its fitness value is shown at the left of the plot. The final Mandeltweak is shown at right.

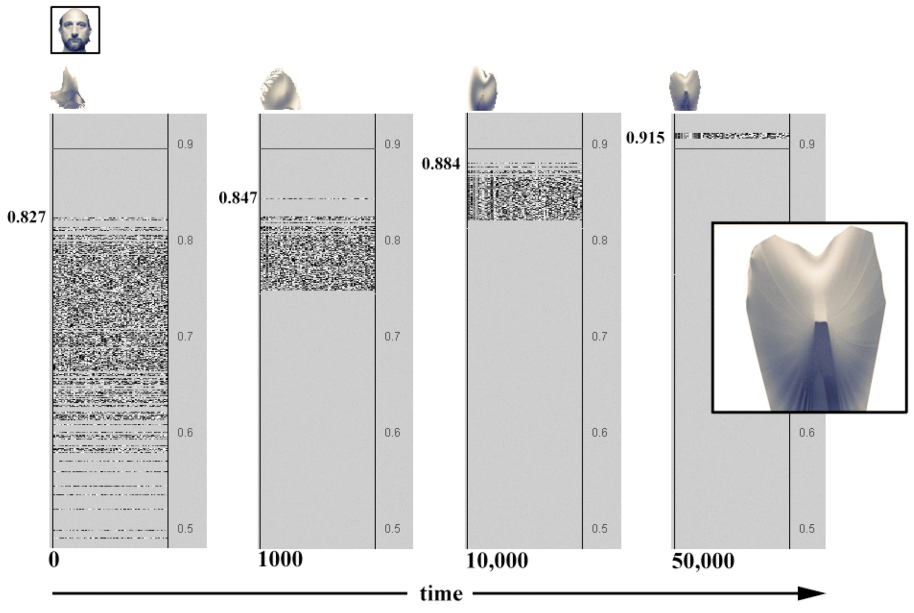

Fig. 10. plotting fitness and genetic convergence in a population of 1000 genotypes

At initialization genotypes are randomized and their associated images are compared to the ideal image to determine fitness. In this particular experiment, the fitness values range from just under 0.5 to 0.827 at initialization. This distribution of fitness values is due to the three following factors: (1) the characteristics of the genotype-to-phenotype mapping, and thus the resulting Mandeltweak images, (2) the ideal image, and (3) the nature of the fitness comparison scheme.

The graph shows that after 1000 iterations the lower end of the fitness range has raised. This is because the least-fit genotype is always replaced with the newly-created genotype for each step, and since each new genotype is the product of two relatively-fit genotypes, it usually has higher fitness. This is especially the case in the beginning.

The highest fitness is shown at each stage, along with a small image of the Mandeltweak with the highest fitness. By time 50,000 we see that the highest fitness is 0.915. The highest fit Mandeltweak is shown at lower-right. Notice also that the genotypes cover a much smaller range of fitness and that they have converged considerably (revealing visible bands of similar colors across the population).

Figure 11 shows the results of six experiments, each using a different ideal images as the fitness function.

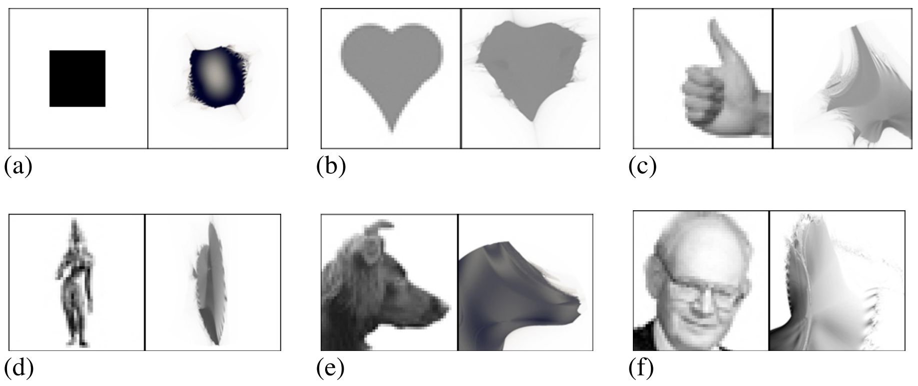

Fig. 11. Six examples of ideal images and evolved Mandelweaks

In these examples, the ideal images are shown to the left of their associated Mandelt-weaks . The ideal images may appear jagged or pixelated because of their low resolution. The Mandeltweaks, in contrast, are shown at a higher resolution - recall that pixel resolution is arbitrary in Mandeltweaks, and that the encoding of the image is purely genetic/mathematical.

In all of these experiments, population size was set to 1000, and mutation rate was set to 0.05 except for examples (a) and (c), in which population size was set to 100 and mutation rate was set to 0.1. In all cases, the highest fitness achieved was in the approximate range of 0.95. The number of iterations in each case ranged, averaging around 2000.

Range of Genetic Variation

Each tweaker used in the math function has a range within which it can deviate from its default value. To manipulate this range, a global range scale *s* was created so that the whole array of range values could be scaled at once. In most experiments, *s* is set to 1 (resulting in the normal tweak ranges as originally designed). But *s* can be varied to explore Mandeltweak's imitative performance over different genetic ranges. Figure 12 shows the results of 11 experiments with the ideal image set to a portrait of the 20th century painter Francis Bacon.

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

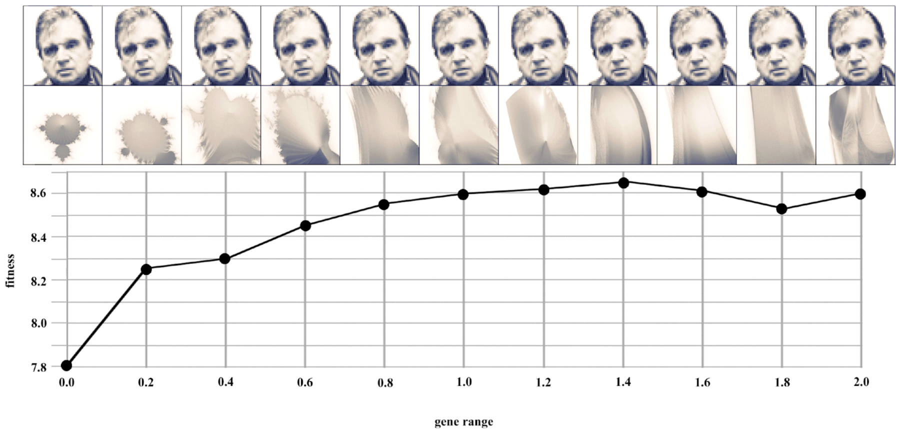

Fig. 12. Varying tweak ranges in a series of experiments

In each experiment, population size was set to 1000, and mutation rate was set to 0.05. When the *s* is set to 0.0, the result is that when genotypes are mapped to phenotypes, the values of the tweakers are clamped to their defaults. And so the image at the left-most side of Figure 12 is the Mandelbrot Set, at default angle and default coloration. As *r* is increased by increments of 0.2, we see that fitness increases on average, because there is an increasingly larger genetic space available for searching. The reason the default Mandelbrot image has a fitness of ~0.78 has to do with the nature of the ideal image, the nature of the default Mandeltweak settings, and the comparison technique. What is of interest, though, is not this value, but the rate at which fitness increases, and then reaches what appears to be a limit at around *s*=1.4 (this is not absolutely confirmed, but all experiments indicate the same approximate limitation, including when they are run for many more iterations and with much larger populations).

Artistically-speaking, one might find the visual "sweet-spot" to occur before fitness has reached its maximum. In fact, the images in Figure 1 were intentionally evolved using a slightly smaller range. They also used smaller populations and were run for fewer iterations - this allowed the peculiar vestiges of Mandelbrot features to remain, making the head-like shapes more intriguing and ambiguous.

Imitating...The Mandelbrot Set?

Since the vast genetic space of all Mandeltweaks contains the Mandelbrot Set at the point in the space where all values are at their default settings, it is possible that an initial population of random Mandeltweaks can converge on the Mandelbrot Set. But in a number of experiments with different populations and mutation rates, this was not achieved. Instead, the population converged on another region of the space which approached the shape of the Set. Figure 13 shows the most fit Mandeltweak in a population in multiple stages of evolution, starting at time 0, then 2000, and then doubling the time intervals, up to 128000. It reached a maximum fitness of 0.895 The image at right is the Mandelbrot Set (a high-res version of the ideal image used) to show what it was trying to imitate.

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

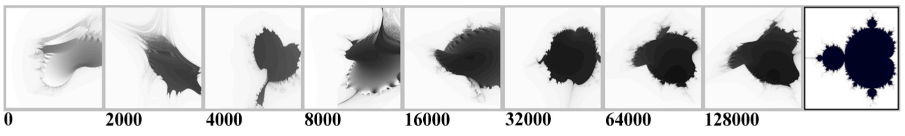

Fig. 13. Mandeltweak imitates the Mandelbrot Set, only *backwards*

In this and other similar experiments, the Mandeltweak got stuck on a local hill in the fitness landscape, which corresponds roughly to the shape, but it is rotated almost 180 degrees! The fitness landscape is very large and rugged, and the shapes in the initial random population are too varied from the original shape - and so a common protrusion resulting from tweaking (such as the one shown in Figure 6), ends up being a proxy for the main bulb. As a test, a critical gene, the "angle" gene (responsible for varying the rotation of the shape in the complex plane), was clamped to 0, and not allowed to vary. The population was then able to more easily converge on the Mandelbrot Set. This is an indication that the angle gene enlarges the fitness landscape considerably.

Using Higher-Order Mandelbrot Functions

The shape created by the higher-order function, $z = z^3 + c$, is shown in Figure 4. This uses more variables in the software implementation, to which tweakers can be attached, and so it was considered as another base function to explore. It can be expressed as follows: replace the kernel of the expanded Mandelbrot function shown above:

```
z1 = y + zy*zx + zx*zy
z2 = x + zx*zx - zy*zy
```

with:

```
a = zx
b = zy
z2 = a*zx - b*zy
z1 = b*zx + a*zy
zx = z2
zy = z1
z2 = a*zx - b*zy
z1 = b*zx + a*zy
z1 = z1 + x
z2 = z2 + y
zx = z2
zy = z1
```

and tweak like this:

```
a = zx
b = zy
z2 = ((a*p1 )+p2 )*((zx*p3 )+p4 )-((b*p5 )+p6 )*((zy*p7 )+p8 )
z1 = ((b*p9 )+p10)*((zx*p11)+p12)+((a*p13)+p14)*((zy*p15)+p16)
zx = z2
zy = z1
```

```
z2 = ((a*p17)+p18)*((zx*p19)+p20)-((b*p21)+p22)*((zy*p23)+p24)
z1 = ((b*p25)+p26)*((zx*p27)+p28)+((a*p29)+p30)*((zy*p31)+p32)
z1 = z1 + x
z2 = z2 + y
zx = z2
zy = z1
```

Figure 14 shows the results of five experiments in which the normal Mandeltweak algorithm is compared to the one which uses the cubed algorithm just described. Population was set to 1000, and mutation rate was set to 0.05. Each experiment was run until the population converged significantly, and the number of iterations, while varying among pairs of tests, was kept constant for each algorithm in the pair.

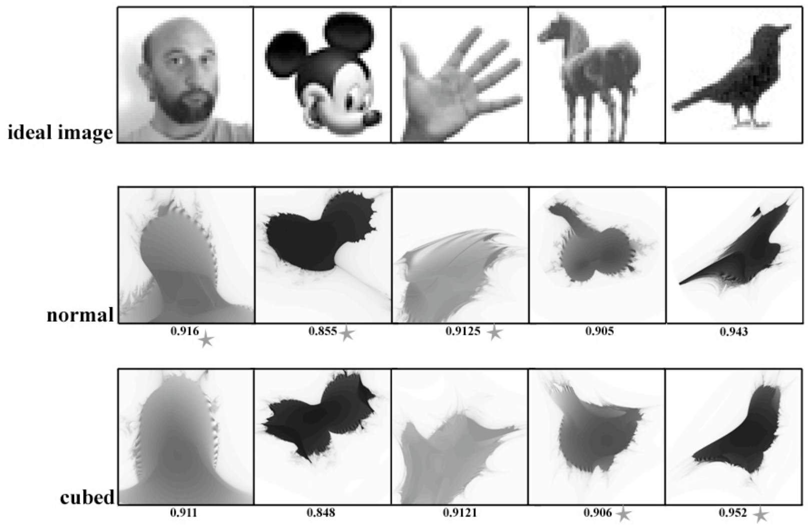

Fig. 14. Comparing normal vs. cubed algorithm

The cubed algorithm doesn't appear to be much better at imitating the ideal image, and it might even be inferior, if the stars placed next to the fitness values in the figure are any indication. In the case of both algorithms, there is an inability to imitate local features - notice that the fingers of the hand and the legs of the horse are not picked up very well using either algorithm. The reasons for this are not clear. A few possibilities are: (1) the image comparison scheme is not feature-based, (2) the population is too small, (3) the genetic algorithm is not designed appropriately, or (4) these mathematical equations are simply not able to conjure up these particular shapes, even though the genetic space is very large.

There is more exploration to be done with higher-order Mandelbrot functions, as well as the other varieties of fractal equations, so as to increase the image-making vocabulary of genetic art based on iteration in the complex plane.

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

5 Conclusion

The Mandeltweak images shown that are based on images of the author's head may be considered self-portraits. But in truth, the entire technique is a self-portrait. A strong background in art as a child, plus failing grades in high school math, set the stage for a naive and reckless approach to mathematics (when it was necessary to encounter math - as was the case when first programming the Mandelbrot Set). But while math understanding played a small role in these early tweaking experiments, it gradually became familiar, and now, complex mathematics is a favorite topic. Nonetheless, the Mandeltweak technique maintains its irreverent and subversive manipulation of complex plane fractals, because the phenotype space is considerably expressive, and the visual vocabulary is large, and is likely to grow larger as more variations are explored.

Math-based computer art created using evolutionary computation is often non-objective or abstract - not meant to represent anything in particular. The Mandelbrot Set and its kin are neither art, nor are they abstract art, in the sense of being hand-crafted by a human. But the curious animal-like nature of the Mandelbrot Set, as far as how it behaves upon being tweaked, invites one to read it as an organic entity – and thus it enters into an interpretive space. This was part of the initial motivation behind the technique described. Its ability to imitate explicit images is limited. But this tension - the tension between being the platonic Mandelbrot Set and being coerced into a representational form - is part of the game. It is a kind of conceptual art. The question of how evolvable an image-making scheme can be is a common problem in evolutionary art: is the phenotype space large enough? - and can a subset of it map to the artist's aesthetic space? The Mandeltweak technique was created to take on this question. In the process of asking these questions, and to better understand its limits, the artist's aesthetic – and mathematical – vocabularies have grown larger.

References

1. Ashlock, D. *Evolutionary Exploration of the Mandelbrot Set* in the Proceedings of the 2006 Congress On Evolutionary Computation, pages 7432-7439, 2006.

2. Dawkins, The Blind Watchmaker - Why the Evidence of Evolution Reveals a Universe Without Design. WW. Norton and Company. 1986

3. Dewdney, A. K. "Computer recreations: A computer microscope zooms in for a look at the most complex object in mathematics", Scientific American, August 1985, pp 16-25.

4. Dickerson, R. Higher-order Mandelbrot Fractals: Experiments in Nanogeometry. published online at http://mathforum.org/library/view/65021.html. 2006

5. Douady, A., and Hubbard, J. "Etude dynamique des polynomes complexes, I, II" Publ. Math. Orsay 1984, 1985

6. *eNZed Blue*. The Koruandelbrot. published online at:
http://www.enzedblue.com/Fractals/Fractals.html. 2005

Published in: Design by Evolution, Natural Computing Series. ISBN 978-3-540-74109-1. Springer Berlin Heidelberg, 2008, p. 145

6. Goldberg, D. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989

7. Koza. J. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press 1992.

8. Lakoff, G., and Núnez, R. Where Mathematics Comes From - How the Embodied Mind Brings Mathematics Into Being. Basic Books, 2000.

9. Mandelbrot, B. The Fractal Geometry of Nature. W. H. Freeman and Company. 1977

10. McCormak, J. *Open Problems in Evolutionary Music and Art*, in F. Rothlauf et al. (eds), Lecture Notes in Computer Science, Vol 3449. 2005. Springer-Verlag. pp 428-436

11. Peitgen, H.O, and Saupe, D., editors, The Science of Fractal Images. Springer-Verlag, 1988

12. Penrose, R. The Road to Reality, A Complete Guide to the Laws of the Universe. Knopf, 2004

13. Pickover, C. Computers, Pattern, Chaos, and Beauty - Graphics From and Unseen World. St. Martins Press. 1990

14. Rooke, S. Eons of genetically evolved algorithmic images. Creative Evolutionary Systems, Morgan Kaufman Publishers, Inc. San Francisco CA, 2001

15. Sims, K. "Artificial Evolution for Computer Graphics". Computer Graphics. ACM SIGGRAPH '91 conference proceedings, 1991, pp.319-328.

16. Todd, S. and Latham, W. Evolutionary Art and Computers. Academic Press 1992.

17. Ushiki, S. IEEE Transactions on Circuits and Systems vol 35 No. 7 July, 1988 pp 788

18. Ventrella, J. 1994. Explorations in the Emergence of Morphology and Locomotion Behaviour in Animated Characters. Brooks, R., and Maes, P. (eds)

19. Ventrella, J. Mandeltweaks: published on the internet at:
http://www.ventrella.com/Tweaks/MandelTweaks/tweaks.html. 2004

20. Ventrella, J. Mandelswarm - (particle swarm seeks the boundary of the Mandelbrot Set). Published online with Java Applet at:
http://www.ventrella.com/Tweaks/MandelTweaks/MandelSwarm/MandelSwarm.html. 2005