Riga Technical University
Riga, Latvia

Bric-a-brac of Creative Computing: Studying Fractal Shapes with form•Z

by Modris Dobelis
Project by Dmitriy Averyanov and Vladimir Katz

And how the One of Time,
Of Space the Three,
Might in the Chain of
Symbols girdled be.

William Rowan Hamilton

Generally, a fractal most often referred to as "a rough shape that can be subdivided in parts, each of which is (at least approximately) a reduced-size copy of the whole." The father of fractal geometry Benoît Mandelbrot formulated this term in 1975. A fractal has a complicated shape. Certain features characterize a graphical representation of the fractal by mathematical equation: it is self-similar, has fine structure, and is very difficult to describe it using Euclidian geometric language. It might seem like a nice mathematic abstraction, but fractal geometry, as we believe – is the foundation of everything that exists.

The theory of evolution, which emphasizes on chaos, natural selection, and gene, has come to dominate the world scene. The universe described as an intricately complex system emerged from some preset condition a long time ago and differed greatly over the course of its history. The diverse life forms are all a part of the evolved complexity, one that can be described using mathematics.

Quoting Ian Malcolm from Michael Crichton's "Jurassic Park", "Fractals are everything!" – meaning that all of the creation, from macrocosmic scale down to quarks follows the same rule of organization. A rock resembles larger pieces of rock in its structure and form, and ultimately it looks pretty much like the mountain that it is forming.

The reason for starting this project was our interest in fractal geometry as well as a desire to produce a program that would generate 3D fractal shapes for creation of convertible models. As to our knowledge there is limited number of software that draws fractals in 3D space, where you can "touch and feel" or explore them from different viewpoints.

Mathematicians have been generating 3D fractals for a long time. Just a few programs to mention are Quat 1.2[1] and TetraBot Explorer[2] – first thing, you are likely to find upon typing "3D fractals" in any internet search engine. These applications serve as a good medium for generating 2D images of 3D fractals. The problem is that you cannot get a complete 3D fractal, rotate it, zoom into it or use it as a separate model for your design project. There are plugins and applications for CAD software that assist in drawing fractal trees and mountains. However, there is practically no software that generates real-life fractals, which are described by algebraic methods rather than geometric.

There is a fine difference between 2D and a 3D fractal. Ripples on the water, or patterns formed by the wind on

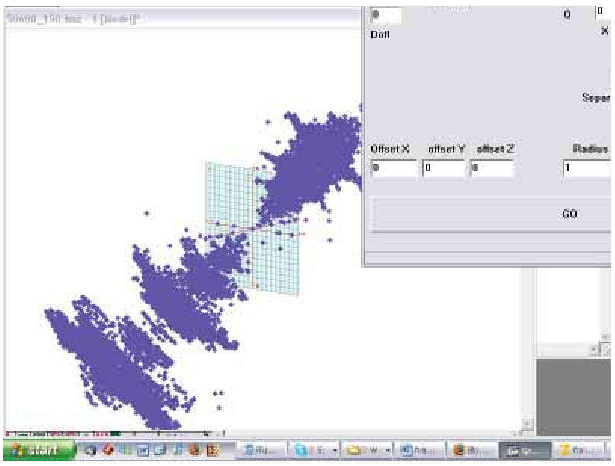

Figure 1: FractalZ application running simultaneously with **form•Z**

the sand, or the fabric of a leaf are all examples of 2D fractals when viewed from the appropriate viewpoint. Some plants, trees and the above-mentioned rocks – all are examples of 3D fractals. Their complex structure is not visible at first glance; it exists nevertheless. Living organisms belong to this second group. Although some of the main features, like repetition of separate parts in smaller scale are not present, human beings display symmetry and fine structure of their organs.

At the beginning, an application was written that performs the required calculations (Figure 1). For this specific purpose, quaternion numbers were used. The fractals defined in programs like Quat or FractIt were generated likewise. It is not the only way to generate fractals. Geometric methods can be used as well, but – as it was already mentioned before – this is not the case considered in this study (Figure 2).

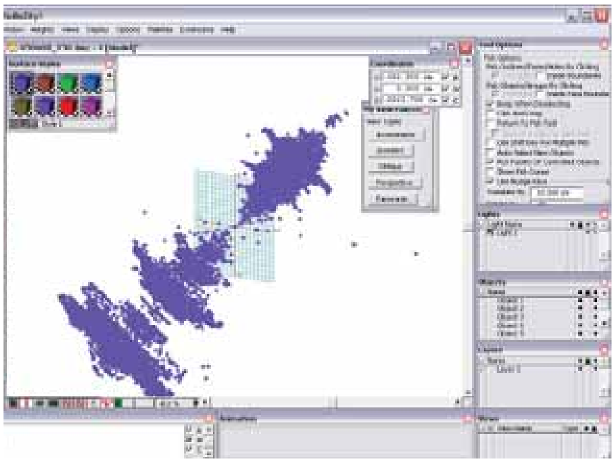

Figure 2: Fractal shape rendered by **form•Z**

The tale of quaternion numbers is an amazing story in itself – one that deserves a separate topic. In mathematics, quaternions are a non-cumulative extension of complex numbers. They were first discovered and described in the middle of the 19th century by Sir William Rowan Hamilton[3], a mathematician and physicist from Ireland. Hamilton was searching for a way to expand complex numbers, which represent points on a plane, into a higher dimension. He failed for three dimensions, but succeeded with four – thus finding the quaternion. Quaternion was met coldly in the mathematicians world and were quickly proven to be pathologic because they disobeyed the commutative law ab=ba, and therefore were completely replaced by vectors. Quaternion resurfaced in areas of 3D object orientation and geometric analysis during the computer age, finding their application in the computation of the objects of higher order.

Quaternion number consists of four parts – “1”, “i”, “j”, and “k”[4]. The resulting figure of the computational analysis is a 4-dimentional shape that can later be projected. Quat and similar applications do just that – they draw a projection of the resulting shape on a 2D plane. We went one-step ahead or in other words, an attempt to project or model a 4D quaternion equation in 3D space was performed in this study.

The calculation of the shape begins with an equation: $x_{n+1} = x_n^2 - c$. In this equation x_0 is the starting value, which is the point that has to be calculated; n assigns index to x $(n=0, 1, 2, 3, ...)$. The c value is the predefined quaternion that defines the shape of the resulting fractal. It is evident that the sequence of all x_n defined by the iteration formula, sometimes referred to as “orbit”, can have three resulting values:

1) the sequence converges toward a fixed value (e.g. zero);
2) oscillates periodically between some value;
3) approaches infinity.

The resulting object is the amount of all points (numbers) x_0, for which the sequence defined by the formula does not approach infinity or zero – in other words – which does not follow the scheme 1) or 3). This formulation is not complete because the computer is not capable of carrying out calculations up to infinity to see if the sequence is convergent. A maxiter value bypasses the limitation of our material world. After a certain amount of iterations, the calculation was interrupted. To verify, whether the sequence approaches infinity or not, the value bailout was introduced. This term comes from FractIt and Quat applications. If the value is exceeded, the computer assumes that the sequence goes toward infinity. Therefore, the calculation is the amount of all point-numbers x_0, for which the sequence defined by the iteration formula of the x_n did not exceed the value bailout after at most maxiter iterations.

The output was a little bit trickier. While the basics of fractal calculation are well known for a long time, handling the resulting data is far more difficult. The programming was simplified by the fact that it was not necessary to write the code responsible for the graphical output. The data of the generated points were fed directly into **form•Z** application. This involved writing a keyboard control module using a Delphi compiler. No changes were made in **form•Z** software. FractalZ is the name of the developed application that ran into separate window, and feeding point coordinates into active **form•Z** application. Thus, two programs – **form•Z** and FractalZ – ran simultaneously in some sort resembling a chalkboard that records a single point in 3D space for every coordinate FractalZ calculated.

At this point, we encountered serious problems. One thing to note is the speed with which the calculations or the input process was carried out. Using the method described above, it takes 200 milliseconds for **form•Z** to draw a single point and some 15 to 20 minutes to generate a 3,000-point fractal model. Another thing to mention is the amount of data that **form•Z** was able to handle. With shapes consisting from 20,000 and more points, application and computer crashes became more frequent. In fact, it takes several million points to generate a smooth fractal shape like seen in Quat. Our models, consisting from several thousand points, do not look much like fractals. Microscopic details very often are hidden at the lower levels.

It should be explained in what way 2D fractals are different from 3D and why handling them is more difficult. When the point is projected on a 2D plane, it is possible to control the size of the resulting image by lowering the resolution and adjusting its dimensions. That way, while some fine fractal details will disappear, the overall picture will remain. In case we want to go deeper, it is possible to zoom into a specific part and calculate it. The same cannot be done easily with the resulting points of 3D fractal shapes. They consist of a collection of individual points, a point-cloud that need to be visualized somehow. One way is to replace each individual point with a 3D shape (an analogy is voxel rendering), or drape the shape, creating a NURBS surface. To simplify the work with the resulting model, data filtering was performed. Excluded from the process of calculation were all those points that were found “inside” the model and could not be seen from outside, thus leaving only the outer layer. This approach considerably reduced the resulting model size and facilitated its easier manipulation.

Rendering of the object was finalized using point-cloud processing software Points2Polys from Paraforms (Figure 3) and Point Cloud (v.1.0) from Floating Point Solution. These applications connected the adjoining points, forming poly-meshes from the input data material, while Point Cloud is also capable of draping a point-cloud with a 2D plane. To achieve this it was necessary to export **form•Z** model into *.obj file format. Further visualization was complicated. It was not possible to reach the level of detail found in similar applications, for the simplest reason that our hardware and software was not adequate to handle the complex models consisting of several thousand points. To truly appreciate the 3D fractals in all their glory it would require many more iterations to be carried out on a finer scale with the resulting models much more complex, and much “heavier” (Figures 4-6). Similarly to a man in a dark cavern poking at walls with a stick. we were able to reproduce a part of that hidden, complex topology, while most of the nooks and crannies remain hidden.

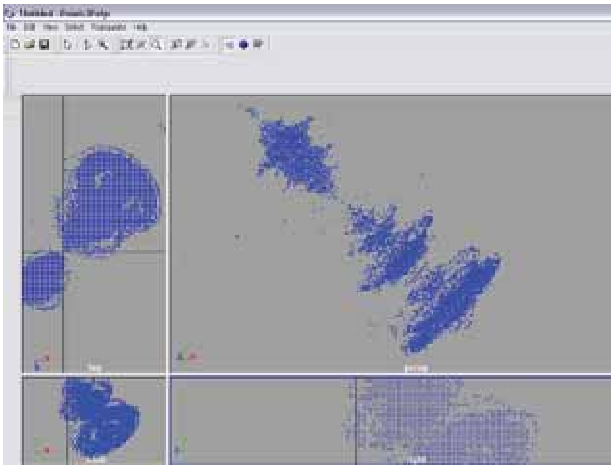
Figure 3: Fractal point-cloud in Points2Polys application.

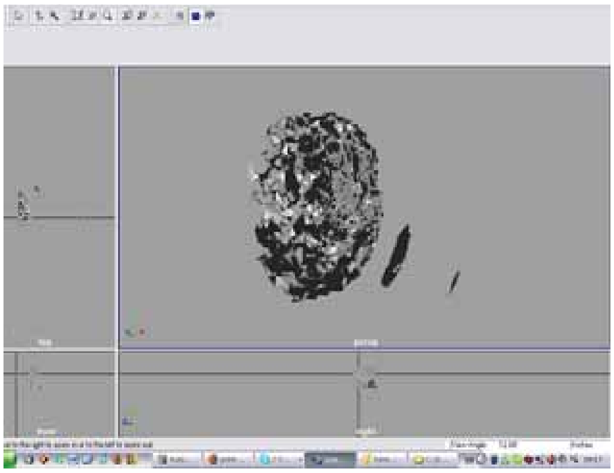
Figure 4: Mesh-cloud converted from point-cloud.

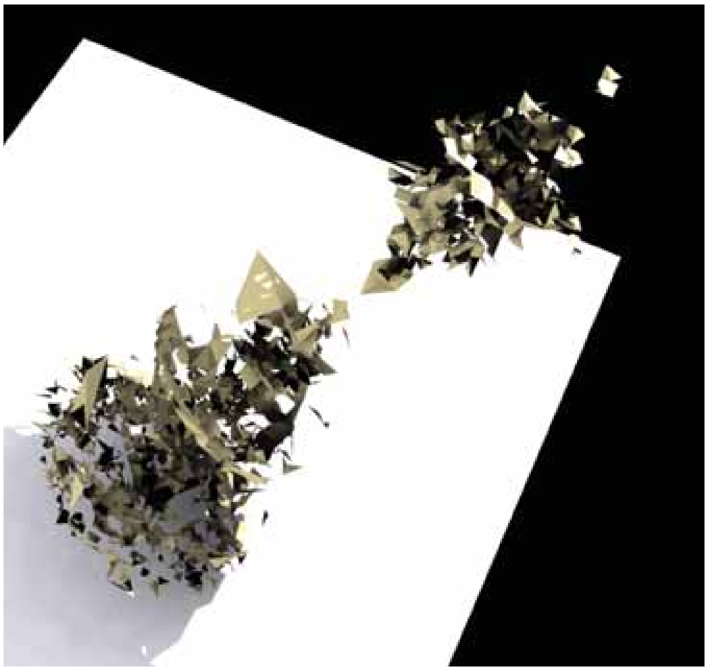
Figure 5: Artistic rendering of the mesh model.

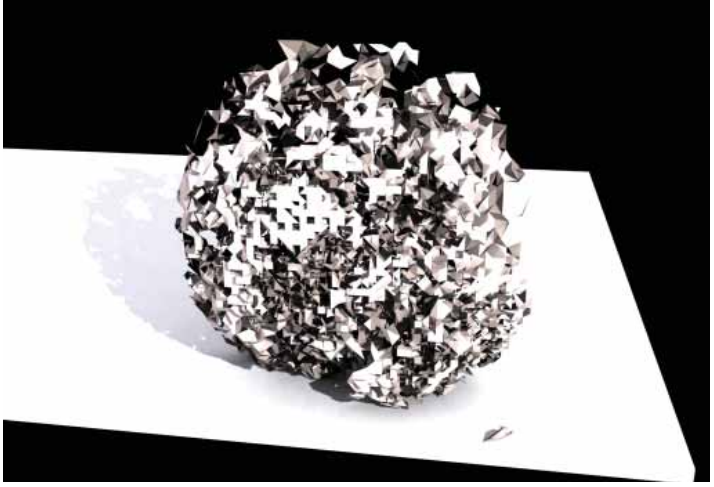

Figure 6: Artistic rendering of the mesh model.

The practical value of this work, as to our knowledge, is in its similarity with the study of the modern theoretical physics[5]. On the forefront of contemporary cosmology is the idea that our universe is just one of the many universes that forms a far bigger cluster called “the multiverse”, or “megaverse” as some scientists prefer to call it. String theory describes particles as vibrating strings of energy – because strings vibrate at different levels of intensity – there exist varieties of particles in our universe. The expansion of the universe is fueled by the so-called “dark energy”, an energy that represents the cosmologic constant (Λ). Now, the value of cosmologic constant is very important for our existence. If its value had been different during and right after the Big Bang, our universe would have ended up in a completely different state. If the cosmologic constant were lower than its present value, all the matter following the gravitational pull would collapse onto itself. On the other hand, if its value would have been greater, all of the matter would be blown apart, the expansion of space would occur so fast that galaxies and stars would have no chance of forming, and likewise, there would be no time for higher intelligence to evolve.

While calculating the 3D shape generated by the equation, FractalZ application was looking for specific values that would produce meaningful results. In case the results of the equation converged to zero or infinity, there would be nothing. We did a listing of all the valid points on the surface of that 4D shape.

The same can be applied to cosmology. Cosmologic constant, which seems to be so fine-tuned to satisfy our needs, could be completely different in some other universe of the multiverse thus making that other places uninhabitable. Scientists are busy constructing models of multiverse, called “landscape”. Now “landscape” is a difficult term. It is not something that exists in reality; rather, it is a “map” or the model of possibilities for the “shape” of the multiverse. Our universe, with our particular value of the cosmologic constant is just one of the many possibilities or points on the landscape.

What we did in our study was the mapping of the “habitable” places on a 4D shape, in the same way as physicists today are trying to map the landscape of the multiverse. To follow the analogy – cosmologic constant is our x, the result of the equation, where c – the quaternion – represents the four fundamental forces in our universe (gravity, electromagnet force, strong and weak nuclear forces). Now, this was not an arbitrary choice: since each force has messenger particles (gravitons, photons, gluons, W and Z bosons) that can be described by string theory as being the same vibrating string, it is possible to unify all of these forces into one quaternion number, assuming that they are permanent in their affect on the universe. That way, the topology of the landscape, where the universal constant changes and the four fundamental forces are locked into a hyper-complex number, the only thing missing is the equation itself, the formula that governs the shape of the landscape, and consequently – the multiverse. The four forces do not even have to have the same value; they can change as well, affecting the value of the cosmologic constant. It might seem a bit far-fetched, but linking these two principles might produce a coherent picture.

A trivia, a curio, bric-a-brac of creative computing are the words we would use to describe this study. The main value comes in the form of expansion of conscience. We achieved our goal – some very attractive visual masterpieces are presented for your evaluation (Figures 7-9). No conclusions have been formulated so far and the work on the study is continuing. Too many things are still not clarified yet. For some reason, certain quaternion values bring no result at all, while others blossom in various forms and shapes emerging from minor changes. New quaternion values are tested and new methods of graphical representation are under development.

A universe is an open-ended system. It exists in itself and has no perceivable end – neither in time nor in space. Mathematics, being an abstract plane of symbols and relations, shares that same quality – there is no end to it. Likewise, we would like to keep our project going on rather than placing a full stop at the end of this sentence…

References

[1] Three-dimensional fractals (quaternionic fractals). http://www.physcip.uni-stuttgart.de/phy11733/index_e.html

[2] Tetrabot: The generalized Mandelbrot set. http://www.3dfractals.com/

[3] William Rowan Hamilton (1805-1865). http://en.wikipedia.org/wiki/William_Rowan_Hamilton.

[4] Macfarlane, Alexander (1906), “Vector analysis and quaternions”, 4th ed. 1st thousand. New York, J. Wiley & Sons; LCCN es 16000048.

[5] Susskind, Leonard – “The Cosmic Landscape: String Theory and the Illusion of Intelligent Design”; Little, Brown and Company (December 12, 2005) ISBN-10: 0316155799.

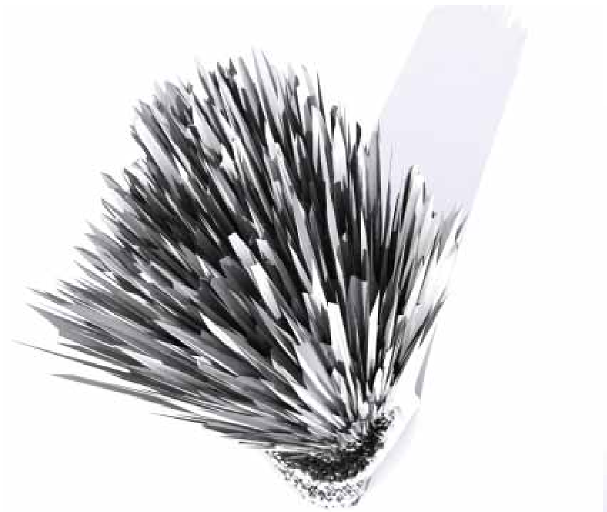
Figure 7: Rendering of point-clouds after processing by Point Cloud 1.0.

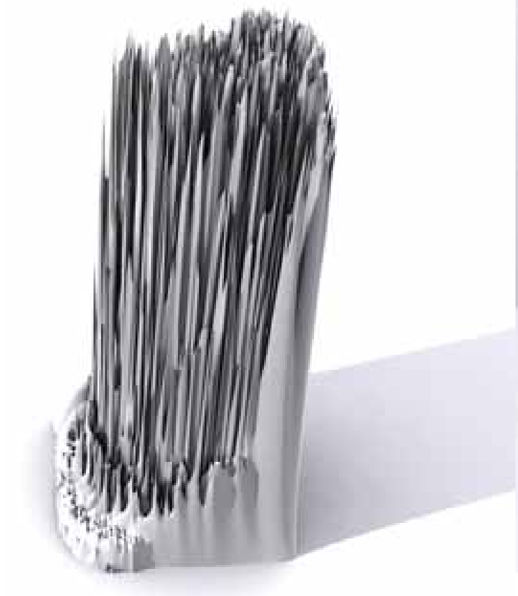
Figure 8: Rendering of point-clouds after processing by Point Cloud 1.0.

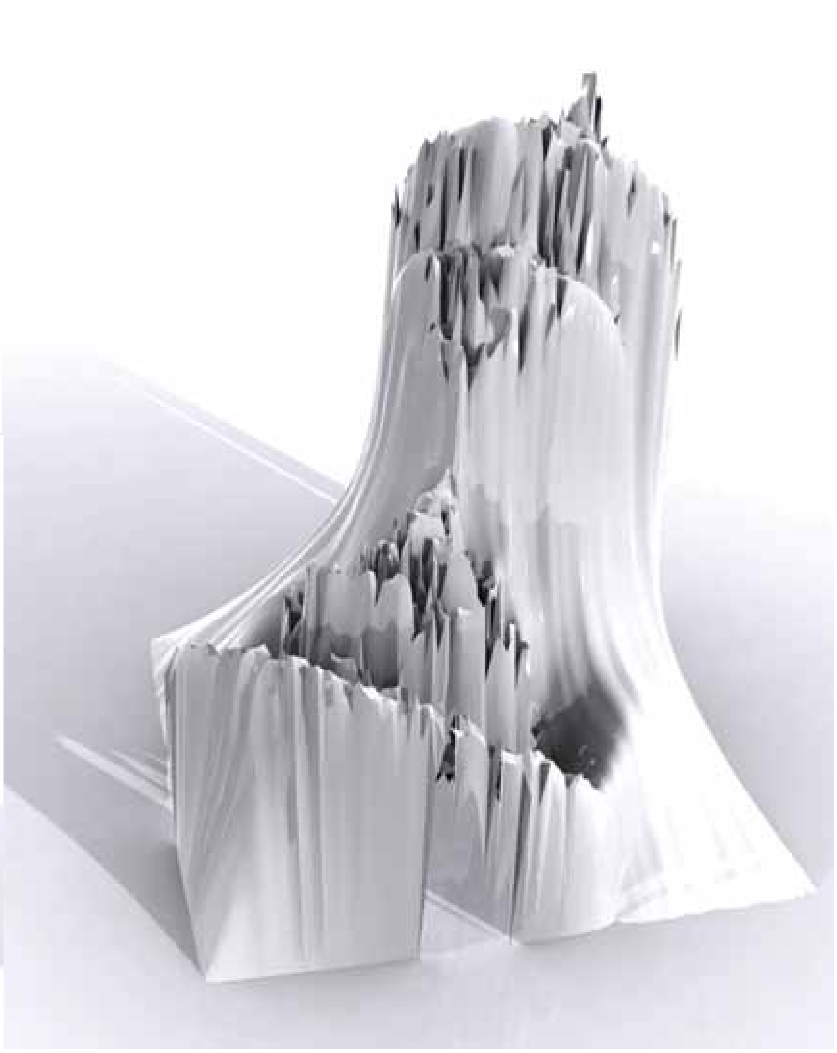
Figure 9: Rendering of point-clouds after processing by Point Cloud 1.0.

Modris Dobelis is Professor and Head of Department of Computer Aided Engineering Graphics at the Riga Technical University in Riga, Latvia. He teaches graduate and undergraduate students computer aided architectural design. He has received a Certificate of Appreciation and AP600 Program Award from Westinghouse Electric Corp. in recognition of superior performance as a member of the AP600 Project working on 3D modeling of piping and equipment (1994-1996), and Honorary Award from the Ministry of Science and Education of Latvia about promotion of CAD ideas in education (2007). He is a president of International Baltic Association for Geometry and Engineering Graphics BALTGRAF. He strives to promote the Building Information Modeling idea into architectural design process. Email: Modris.Dobelis@rtu.lv.

Dmitriy Averyanov is a bachelor student of architecture at the Riga Technical University, Faculty of Architecture and Urban Planning. He is a freelance artist doing part-time jobs in several architecture bureaus in Latvia, as well as in Russia. He is mostly involved in urban planning and renovation.

Vladimir Katz is a 4th-year student of mathematics at the University of Latvia, Faculty of Physics and Mathematics. He is an assistant programmer at a telecommunication company, Belam Inc., and a full-time fractal enthusiast.