# Visualising Volumetric Fractals

Paul Bourke
The University of Western Australia
Perth, Western Australia
paul.bourke@uwa.edu.au

**Abstract -- Fractal images have for many years been a rich source of exploration by those in computer science who also have an interest in graphics. They often served as a way of testing the performance of new computing hardware and to explore the capabilities of emerging display technologies. While there have been forays by some into 3D geometric fractals, the 3D equivalents of the Mandelbrot set have been largely ignored. This is largely due to the lack of suitable tools for rendering these sets except perhaps as isosurfaces, a rather unsatisfactory and limited representation. The following will illustrate the application of GPU based raycasting, a now relatively standard approach to volume rendering, to the representation of volumetric fractals. Leveraging existing software that has been designed for general volume visualisation allows the interested 3D fractal explorer to focus on the mathematical generation of the volume data rather than reinventing the entire volume rendering pipeline.**

***Keywords -- Volumetric fractals, Chaotic attractors, Iterated Function Systems, L-Systems, Voxel, Visualisation, Raycasting, Multidimensions, Hypercomplex.***

## I. INTRODUCTION

Computer generation and representation of fractals have been created from the very early days of computer graphics. Indeed one could say they predate computer graphics as we know it today because they were originally formed only as black and white prints on paper by line printers. This representation employed a technique the relies on the relative density of characters in the alphabet and is now referred to as ASCII art [1]. Almost every student of computer science in the late 80's and early 90's at some stage programmed the famous Mandelbrot [2] set. Many proceeded to the multitude of variants [3] with what will be familiar names to many: Lindenmayer-Systems (L-Systems) [4], chaotic attractors and iterated function systems (IFS) [5]. These were fertile grounds in computer graphics because they not only created compelling and beautiful images but also often forced one to consider the efficiency of algorithms, the data storage requirements and the numerical issues as one zoomed into ever increasing depths to explore and confirm the self-similarity, the key characteristic of a fractal.

As computer capabilities grew it was natural for those with an interest in computer graphics to think about extending the ideas into 3D. For geometric fractals based upon recursive replacement, this was relatively straightforward. The rendering tools developed for engineering, architecture and the movie industry were suited to the geometric primitives involved. The challenge was often just to create efficient versions so as not to overload the software, software that was designed to handle the volume of data likely to be created manually rather than by automated algorithms. An example of such a geometric construction might be the Menger sponge [6] or the 3-dimensional version of the Sierpinski carpet. The sponge is made up of cube elements, on each iteration the cube is split into a 3x3x3 grid of cubes each one third the size of the cube in the previous iteration. On each iteration the central cube as well as the cube in the center of each face are removed. This simplistic formation sees the number of cube primitives increase by a factor 20 on each iteration, a factor that quickly overwhelms any engineering CAD application. In order to create higher iterations one needed to consider how to avoid duplicate faces and how to merge connected coplanar faces together.

The extension to 3D of the 2D fractals that are continuous functions on a plane has received much less attention. These 2D fractals in the plane typically have a range of values that are normally mapped to some colour ramp resulting in some of the beautiful images we are accustomed to. The Mandelbrot style fractals being one example, each point on the complex plane (the image) has an associated number that, normally, relates to how quickly the underlying series escapes to infinity, or doesn't. The extension into 3D involves points within a volume of space instead of a region of a plane. Each point has a value that may be related to some metric, again, such as an escape speed. These fractals can be based upon quaternion or hypercomplex algebra for transforming points in 3 or 4 dimensions respectively. The question then is how to visualise these volumes, the computer graphics software found in almost all electronic devices today can readily create 2D fractal images but the higher dimensional geometry is more challenging.

The geometric rendering tools in common usage in architecture, engineering or the movie/entertainment industry are largely inadequate. One could choose a threshold and draw a point or small 3D brick for cells that are on one side of that threshold, the results are unsatisfactory for a few reasons. One being that the brick resolution in most geometric packages would not be small enough for a pleasing representation, a modest resolution may be considered 500x500x500 partitioning of the region in question with a possible 125 million bricks. Another reason why such a technique does have visual appeal is the shading can only use the 3 planes of the brick, as such the surfaces do not naturally contain surface normals and thus do not look smooth. Meaningful surface normals are not straightforward to derive. The usual approach is to take samples of the surface a small delta away in three directions to form an estimate of the normal. This doesn't work for fractal objects with infinite detail because a different estimate of the normal will arise for each value of delta, unlike

DOI: 10.5176/2251-3043_5.2.370

© The Author(s) 2017. This article is published with open access by the GSTF

a continuous surface where the estimate of the normal normally improves as delta is reduced. Similar arguments apply to the use of points or spheres [7]. Figure 1 illustrates the unsatisfactory appearance of using bricks and spheres to represent the Bedouin fractal, see example 1 later.

An improvement that solves these problems might be to create an isosurface using techniques such as the Marching Cubes [8] algorithm. This results in a triangular mesh well suited to viewing in many packages and continuous surface normals allow for lighting and shading models. However, an isosurface only conveys a fraction of the geometric information contained within the volume, there is an infinity of possible isosurfaces each one conveying different aspects of the same fractal. Figure 2 illustrates two such isosurfaces, note that the object appears different in each and due to the fractal nature the non-smooth appearance of the surface is misleading. The last two techniques can be improved slightly by adding a degree of transparency based upon the isosurface. But both the visual quality and representation of the interior structure is limited.

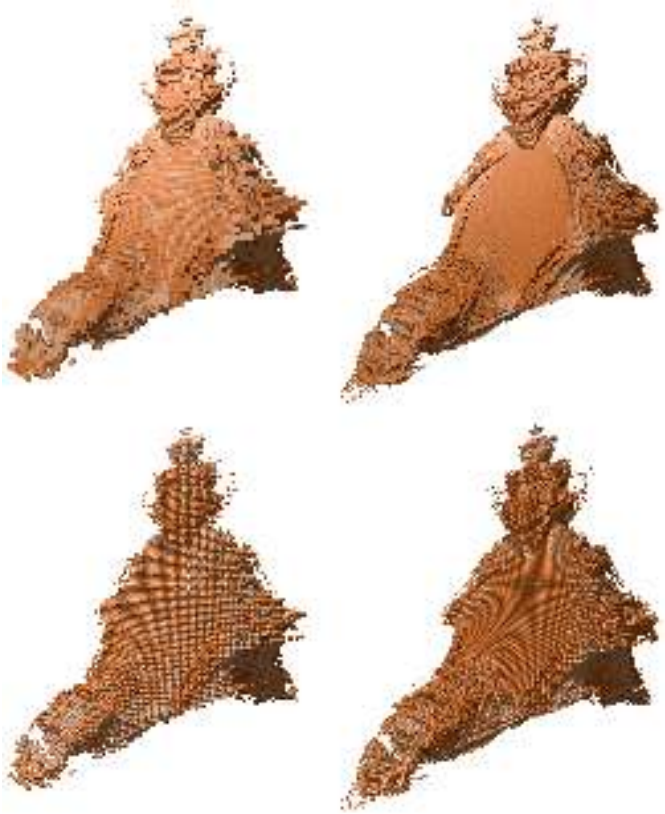

Figure 1. Top, a sampling of the Bedouin fractal by $100^3$ bricks (left) and $500^3$ bricks (right). Bottom, sampling of the Bedouin fractal by $100^3$ spheres (left) and $500^3$ spheres (right).

## II. Volumetric Data

The sampling within some rectangular bounded region of space is what is known in many disciplines as a volumetric data set and the process of representing such data is known as volume rendering [9] or volume visualisation. Each unit within the volume (brick) is known as a 3-dimensional pixel, known as a VOXEL (VOlume piXEL). Volumetric data arise in many areas of engineering and science, for example the result of a 3D MRI scan or the representation for simulation in fluid flow. For a CT scan the value at each 3D pixel is density at that point, other sources of 3D scans or simulations would have other, usually but not always, scalar metrics. The process of volume visualisation involves mapping that density range to colour and opacity, through what are known as transfer functions, and subsequently rendering the result with a model of how light propagates through the volumetric space.

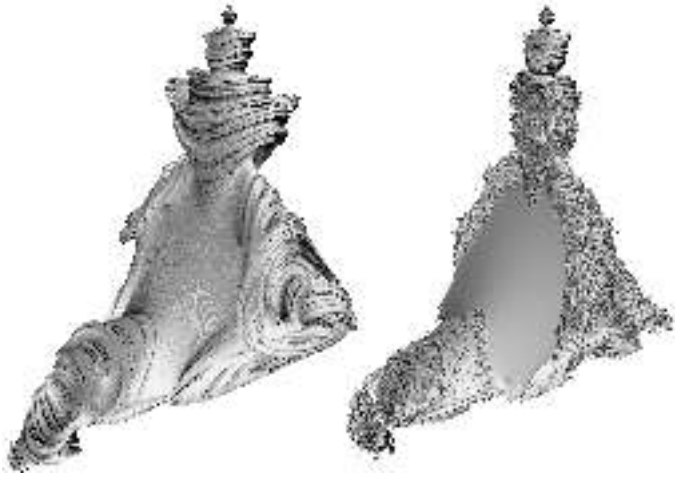

Figure 2. Two isosurfaces Bedouin fractal from an infinite possible number.

There are a plethora of software packages, frameworks and libraries developed over the years to support volume visualisation. Many such as OsiriX, 3DSicer and VisageRT are targeted towards medical volumes, still others are tightly coupled to the medical scanners themselves. Most general visualisation packages, for example, ParaView and Amira provide volume rendering capabilities, a reflection that volumetric data is considered a key data type in the science and engineering fields.

Volume rendering has always been challenging to perform in real time even with the recent use of graphics hardware [10]. While the general techniques have existed for some time, the performance is directly related to the size of the volume. As computing and graphics capabilities have advanced over the years, so has the size of the volumes. Volumes derived from simulation, for example in engineering and astrophysics, have grown as computing power has allowed more fine grain simulations to be performed. Volumes have similarly increased in size as 3D scanners have improved, especially in medical and geoscience. At the same time quality of the renderings has improved due largely to the current ray casting approach which is well suited to GPU implementations. Additional visual appeal has been made possible by considering not just the scalar value at each voxel but the gradient in the local neighbourhood. Examples of so-called 2-dimensional transfer functions (voxel value plus local gradient) are Simian [11] and more recently Drishti [12].

## III. Data Generation

Volume rendering is still a challenging process and efficiencies need to be considered. For a given graphics

© The Author(s) 2017. This article is published with open access by the GSTF

hardware capability the number of voxels on each axis is limited, one cannot create an arbitrarily large volume in order to support deep zooms, such solutions generally involve bespoke rendering algorithms [13]. So given a manageable volume resolution, say $1024^3$ voxels, one needs to choose where that available resolution is positioned and scaled in space in order to represent the region of the fractal in question. This is no different to a 2D fractal representation where one maps the image bounds to a region of the plane in which the fractal resides.

The dynamic range of many volume rendering packages is also limited, largely due to memory limitations. A larger volume can reside in memory if it only has 1 byte per voxel rather than a floating point number requiring 4 bytes. The approach taken by the author is to use a single byte per voxel. The volume may be created with a higher dynamic range, typically floats (4 bytes), but exported to a reduced dynamic range given a knowledge of the voxel value distribution. This may be as simple as a linear mapping between the minimum and maximum values, or it may be nonlinear in the case of extremely high dynamic ranges.

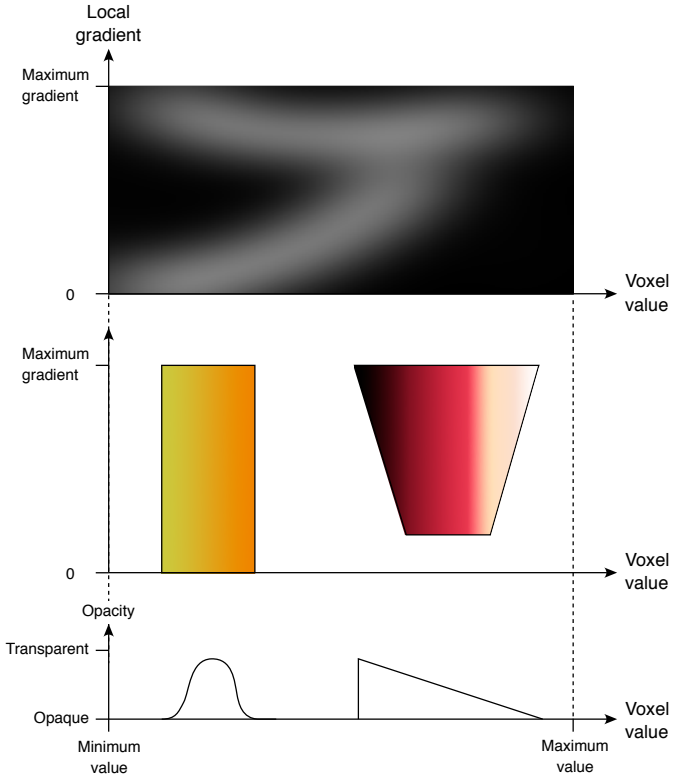


Figure 3. An example of a transfer function specification.

There are a number of data formats that may be used to store the volume, unfortunately there are no standards in widespread usage. Various disciplines have their own widely supported formats, for example DICOM in the medical space and FITS in astronomy. The lowest common denominator is just a raw, unstructured file containing the voxel values. Most volume renderers will provide a means of reading such data, once the dimensions of the volume are supplied by the user or read from the header of the file. There are other ad-hoc headers that precede the raw data that provide the dimensions in voxels, voxel depth, and for instrument scanners the physical dimensions and other meta data. While not ideal, the unstructured raw data format is proposed due to generality and ease of use, both data reading and writing.

## IV. Volume Rendering

It is not the intent of this document to describe a particular rendering approach but the algorithm used here is based upon the volume being loaded into texture memory on the graphics card and view aligned triangles are textured and composited together [14]. The basic operation of all volume rendering is for the operator to choose a mapping between voxel values (assumed to be scalars), and colour and opacity, known as a transfer function. The colour mapping can be thought of as similar to the application of a colour map to a continuous 2D fractal. Opacity is now required because in 3D one may wish to see through outer layers into the inner structure. An additional capability of many volume rendering tools is for the operator to also vary the transfer function based upon local gradient. This provides addition control over the appearance of regions that are changing quickly compared to regions of similar voxel values that are changing more slowly. A notional interface for a 2D transfer function is shown in figure 3. The colour mapping is normally performed in the same 2D space as the voxel value and gradient. The transparency, alpha channel, is a 1D function of the voxel value.

Once the transfer function is defined, the rendering process (in this case) consists of classical ray casting. Rays are cast from the virtual camera, through each pixel on the screen and through the volume. A model is applied that specifies how the ray is affected by the voxels, that is, by their colour and opacity.

In what follows are three examples illustrating the application of volume visualisation algorithms as applied in many areas of science, to 3 or higher dimensional fractals. Three fractals have been chosen that are not widely known and of three distinct classes, a standard escape style Mandelbrot, a 4-dimensional volume sampled with a cutting plane to yield a 3D volume, and a chaotic attractor.

### *A. Example 1*

The first example is from equations proposed by Russell Walsmith and affectionately named the Bedouin fractal although known previously in other circles as the RockBrot. The volume data is created by considering a point $(x_0,y_0,z_0)$ within the region of space of interest and evaluating the following series.

$$x_{n+1}=a+\sin(x_0)$$
$$y_{n+1}=a+\sin(y_0)$$
$$z_{n+1}=a+\sin(z_0) \quad (1)$$

where

$$a=x_n^2-y_n^2-z_n^2$$
$$b=x_n z_n$$
$$c=2x_n y_n \quad (2)$$

© The Author(s) 2017. This article is published with open access by the GSTF

This is similar to the usual quaternion Mandelbrot but with two of the axes swapped.

The voxel is assigned a value depending on how quickly it escapes to infinity, the range is mapped linearly to a byte value from 0 to 255. The volume resolution is $1024^3$ (data size of 1GB) and is suited to interactive performance on a laptop style graphics card, for example an AMD Radeon R9 M370X with 2048MB of memory. Figure 4 shows a rendering of the Bedouin fractal, this is the same example used to illustrate the poor representation by bricks, spheres and isosurfaces in figure 1 and 2.

Note that most of the observed structure actually has the same voxel value, the differences in colour arise from colours mapped to the gradient.

Figure 4. Bedouin fractal on the domain $-1.7 \le x \le 1.0$, $-1.4 \le y \le 1.4$, $-1.4 \le z \le 1.4$.

### B. Example 2

This example is based on equations by Marius-F Danca [15,16] aimed at illustrating that for the alternated Julia sets, the Mandelbrot set consists of the set of all parameter values for which each alternated Julia set is not only connected, but also disconnected and totally disconnected.

These volume fractals also illustrate how quickly a series escapes but in this case it is a 4-dimensional volume, we can render out slices in one dimension resulting in a 3D volumetric dataset. Since each slice yields one 3 dimensional object changing the slice position and angle results in an animation sequence. The series is defined as

$$z_{n+1}=(z_n^2+c_1)+c_2 , \quad z_i \text{ and } c_i \in C, n \in N \qquad (3)$$

where the complex numbers $c_1$ and $c_2$ define the 4-dimensional volume. Figure 5 illustrates one such slice achieved by setting the real value of $c_2$ to zero.

In this example the ability to employ transparency is critical to representing the very fine structures that have only very slightly different voxel values to the main body.

### C. Example 3

This final example is based upon equations originally provided by Roger Bagula, the form is currently unnamed. This is a 3 dimensional iterated function system, a chaotic attractor. On each iteration of the function, a new point in 3D is created, these points lie on the attractor surface. The straightforward method of representing such attractors in 3D might be to place a small sphere at each point in the series. While such representations can be rendered by traditional surface rendering software, the results are generally unappealing for some of the same reasons discussed earlier. Specifically the limits on the number of points and the lack of surface shading.

By computing this series to billions of terms and accumulating the number of times a point on the attractor lands in each voxel region a continuous "smooth" attractor surface is formed across the volume of space of interest.

$$p_{n+1}(x,y,z) = (2p_n(x),2p_n(y),1-p_n^2(x)-p_n^2(y)) / A_{xy}+(-1,-1,0)$$
$$p_{n+1}(x,y,z = (1-p_n^2(z)-p_n^2(y),2p_n(y),2p_n(z)) / A_{yz}+(0,-1,-1)$$
$$p_{n+1}(x,y,z) = (2p_n.x,1-p_n^2(z) - p_n^2(x),2p_n(z)) / A_{xz}+(-1,0,-1)$$
$$p_{n+1}(x,y,z) = p_n(x,y,z)/ 2$$
$$p_{n+1}(x,y,z) = p_n(x,y,z)/ 2 + (1,1,1) \qquad (4)$$

Where

$$A_{xy} = 1 + p_n^2(x) + p_n^2(y)$$
$$A_{yz} = 1 + p_n^2(y) + p_n^2(z)$$
$$A_{zx} = 1 + p_n^2(z) + p_n^2(x) \qquad (5)$$

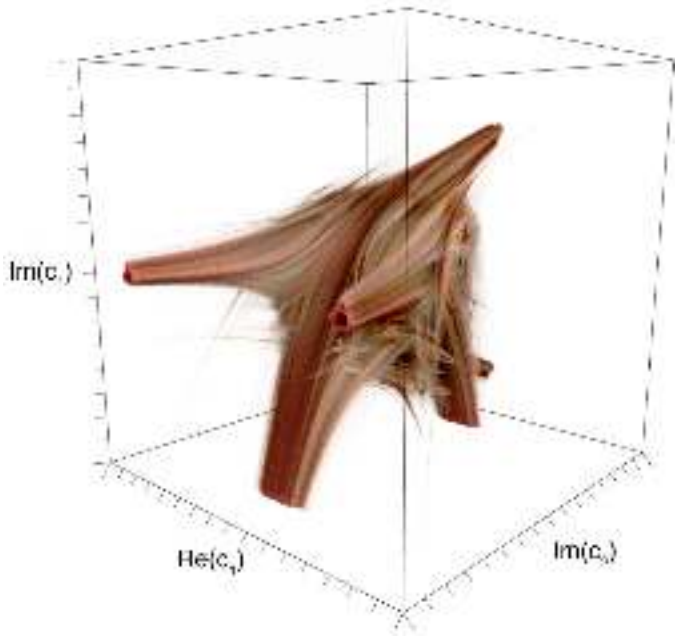


Figure 5. One of an infinite number of slices of the 4-dimensional fractal. This is a slice by the plane $Re(c_2)=0$. The domain is [-2,2] on all axes.

© The Author(s) 2017. This article is published with open access by the GSTF

Figure 6 is a volume render of a $512^3$ volume created from the first 500 million points of the attractor series. Colour mapping designed to provide a biological look, the redder portions are those with a higher density of attractor points. Traditional means of drawing solid points at each position on the attractor doesn't reveal the structure seen here. This is essentially a volume rendering of a 3D histogram.

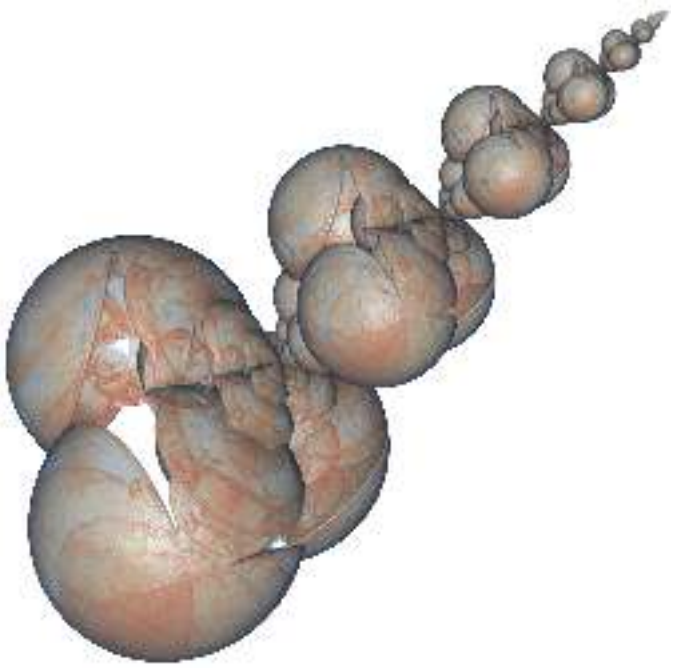

Figure 6. Unnamed attractor by Roger Bagula. Domain is [-2,2] on all axes.

## CONCLUSION

Presented is a means of interactively exploring volumetric fractals that is within the reach of anyone with a computer with a reasonably capable graphics card. It removes the need to develop one's own volume rendering solution but rather leverages the extensive research already conducted in that area. The approach is general in that the volume creation, mathematics, is decoupled from the rendering system. It is additionally based upon existing volume rendering software, the example employed here is free. The downside is that while the visualisation is performed in real time, zooming requires a new volume to be created.

30 years on from the early days of fractal generation it is now possible to escape from the 2D plane opening up a whole new area of investigation and creative opportunities.

## ACKNOWLEDGEMENT

The volume rendering is performed using Drishti, an interactive volume exploration and presentation tool developed by Ajay Limaye at the Australia National University.

## REFERENCES

[1] F.P. Miller, A.F. Vandome, J. McBrewster. ASCII art: Emoticon, Webcomic, List of text editors, ASCII art converters, ANSI art, Shift JIS art, Unicode, ASCII stereogram,FILE ID. DIZ, . nfo, ASCII porn. Alpha Press ©2009. ISBN:6130070446 9786130070441

[2] B. B. Mandelbrot (1977). Fractals form, chance, and dimension. Published 1977 by W. H. Freeman in San Francisco.

[3] H.-O. Peitgen, D. Saupe. (1988) The science of fractal images, Springer-Verlag, New York.

[4] P. Prusinkiewicz, A. Lindenmayer. (1990). The Algorithmic Beauty of Plants. Springer-Verlag, New York.

[5] M. Barnsley. A. Vince (2010). The Chaos Game on a General Iterated Function System. Ergodic Theory Dynam. Systems 31 (2011), no. 4, 1073–1079.

[6] K. Menger (1926), "Allgemeine Räume und Cartesische Räume. I.", Communications to the Amsterdam Academy of Sciences. English translation reprinted in Edgar, Gerald A., ed. (2004), Classics on fractals, Studies in Nonlinearity, Westview Press. Advanced Book Program, Boulder, CO, ISBN 978-0-8133-4153-8.

[7] S. Nikiel, A. Goinski. Generation of volumetric quadratic map basins. Computers & Graphics. volume 27 issue 6. pp 977-982.

[8] W. E. Lorensen, H. E. Cline. (1987) Marching Cubes: A high resolution 3D surface construction algorithm. In: Computer Graphics, Vol. 21, Nr. 4, July 1987

[9] R.A. Drebin, L. Carpenter, P. Hanrahan. (1988) Volume Rendering. SIGGRAPH '88 Proceedings of the 15th annual conference on Computer graphics and interactive techniques, ACM NY 22, 65–74

[10] J. Kniss, G. Kindlmann, C. Hansen. (2001) Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets. Visualization 2001.

[11] P. Bhaniramka, Y. Demange, "OpenGL volumizer: a toolkit for high quality volume rendering of large data sets", Proceedings of the 2002 IEEE symposium on Volume visualization and graphics: 45-54, 2002.

[12] A. Limaye. (2012) Drishti: a volume exploration and presentation tool. Proc. SPIE 8506, Developments in X-Ray Tomography VIII, 85060X (October 17, 2012); doi:10.1117/12.935640. From Developments in X-Ray Tomography VIII. San Diego, California, USA | August 12, 2012.

[13] B. Rama., J Mishra. (2012) Game-enabling the 3D-Mandelbulb Fractal by adding Velocity-induced Support Vectors. International Journal of Computer Applications 48(1) 1-3 · June 2012

[14] B. Cabral, N. Cam, J. Foran, Accelerated volume rendering and tomographic reconstruction using texture mapping hardware, 1994 Symposium on Volume Visualization: 91-98, 1994.

[15] M.-F. Danca, M. Romera, G. Pastor. (2009) Alternate Julia sets and connectivity properties. International Journal of Bifurcation. Chaos 19, 2123–2129.

[16] M.-F. Danca, P. D. Bourke, M. Romera. (2013) Graphical exploration of the connectivity sets of alternated Julia sets; M, the set of disconnected alternated Julia sets. Nonlinear Dynamics, Springer, March. DOI: 10.1007/s11071-013-0859-y.

## Author's Profile

**Paul Bourke** has worked as a visualisation researcher for most of his diverse career. In his various University roles, including positions as a Centre Director and Professor, he has applied visualisation to architecture, brain science, astrophysics and heritage. A visualisation research interest includes the application of novel and emerging data capture and display technologies. This involves displays that leverage the capabilities of the human visual system and how these may be used to facilitate insight in scientific research and increase engagement for public outreach and education. Paul has additionally been an early adopter of the application of fractal geometry to the representation of form, particular applications being to landscape architecture and patterns found in the human body and nature in general.

© The Author(s) 2017. This article is published with open access by the GSTF