International Journal of Computer and Communication Technology

Volume 2 | Issue 1 Article 4

January 2011

Generation of 3D Fractal Images for Mandelbrot and Julia Sets

Bulusu Rama
Department of Computer Science and Engineering, MLR Institute of Technology Hyderabad, India, rama_bulusu@yahoo.com

Jibitesh Mishra
College of Engineering and Technology BPUT, Bhubaneswar, India, mishrajibitesh@gmail.com

Follow this and additional works at: https://www.interscience.in/ijcct

Recommended Citation
Rama, Bulusu and Mishra, Jibitesh (2011) "Generation of 3D Fractal Images for Mandelbrot and Julia Sets," *International Journal of Computer and Communication Technology*: Vol. 2 : Iss. 1 , Article 4.
Available at: https://www.interscience.in/ijcct/vol2/iss1/4

This Article is brought to you for free and open access by Interscience Research Network. It has been accepted for inclusion in International Journal of Computer and Communication Technology by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

Generation of 3D Fractal Images for Mandelbrot and Julia Sets

Bulusu Rama[#], Jibitesh Mishra[*]
[#]*Department of Computer Science and Engineering, MLR Institute of Technology*
Hyderabad, India
[1]rama_bulusu@yahoo.com
[*]*College of Engineering and Technology*
BPUT, Bhubaneswar, India
mishrajibitesh@gmail.com

***Abstract*— Fractals provide an innovative method for generating 3D images of real-world objects by using computational modelling algorithms based on the imperatives of self-similarity, scale invariance, and dimensionality. Images such as coastlines, terrains, cloud mountains, and most interestingly, random shapes composed of curves, sets of curves, etc. present a multi-varied spectrum of fractals usage in domains ranging from multi-coloured, multi-patterned fractal landscapes of natural geographic entities, image compression to even modelling of molecular ecosystems. Fractal geometry provides a basis for modelling the infinite detail found in nature. Fractals contain their scale down, rotate and skew replicas embedded in them. Many different types of fractals have come into limelight since their origin. This paper explains the generation of two famous types of fractals, namely the Mandelbrot Set and Julia Set, the3D rendering of which gives a real-world look and feel in the world of fractal images.**

***Keywords*— Fractals, 3-Dimensional Images, Mandelbrot Set, Julia Set, Recursion, Affine Transformations, 3D Rendering, IFS**

I. INTRODUCTION

A fractal is a rough or fragmented geometric shape that can be subdivided into parts, each of which is (at least approximately) a reduced size copy of the whole or in other words, is self-similar when compared with respect to the original shape. The term was coined by Benoit Mandelbrot in 1975 and was derived from the Latin word "fractus" meaning "broken" or "fractional". The primary characteristic properties of fractals are self-similarity, scale invariance and general irregularity in shape due to which they tend to have a significant detail even after magnification-the more the magnification the more the detail. In most cases, a fractal can be generated by a repeating pattern constructed by a recursive or iterative process. Natural fractals possess statistical self-similarity whereas regular fractals such as Sierpinski Gasket, Cantor set or Koch curve contain exact self-similarity. This paper presents the generation of two of the best-known fractals – the Mandelbrot Set and Julia Set using the deterministic method of IFS (Iterated Function Systems) and affine transformations. The rendering was done in 3D. The program was implemented in C++ based on the algorithms presented in the following sections. The displayed output of based on multiple test cases varied by number of iterations and a given parameter that corresponds to a coefficient value, are presented. The paper ends by giving concluding remarks.

II. ABOUT MANDELBROT AND JULIA SETS

A brief description of Mandelbrot and Julia Sets and their respective properties based on iterative function systems is presented in the sub-sections that follow. A detailed account of the Mandelbrot and Julia Sets can be found in the books as in [2], [7] and in the research project compilation as [4].

A. The Mandelbrot Set

The Mandelbrot Set was invented by the French mathematician Benoit Mandelbrot in 1979, when he was working on the simple equation z = z2 + c. In this equation, z and c denote complex numbers. In other words, the Mandelbrot set is the set of all such complex numbers c, that iterating z = z2+c does not diverge. Hence, it is a connected set of points, which is bounded.

An Iterated Function System (IFS) based on the maximum number of iterations and an initially defined region denoting the lower and upper limits of the bounded space, is iterated as many times as the maximum number of iterations. The resulting set of points can be span an indeterminable amount of space that is a function of the number of iterations involved. Then the set of all points for which the line spacing from the origin to that point in each of the co-ordinate directions is zero, constitutes the Mandelbrot Set. Applying the set of affine transformations iteratively on each point in the starting region set, the resulting fractal is a self-similar shaped image that is resembles an approximation to the original image. This effect is best visualized when rendered in 3D.

To generate the Mandelbrot set graphically, the computer screen becomes the complex plane. Each point on the plane is tested into the equation $z = z^2+c$. If the iterated z stayed within a given boundary forever, i.e., is convergent, the point is inside the set and the point is plotted black. If the iteration went of control, i.e., is divergent, the point was plotted in a colour with respect to how quickly it escaped. When testing a point in a plane to see if it is part of the set, the initial value of z is always zero. This is so **because zero is the critical point of the equation used to generate the set**.

B. The Julia Set

The Julia Set was invented by the French mathematician Gaston Julia in 1918 while studying the iteration of polynomials and relational functions. It is very closely related to the Mandelbrot Set. It is also obtained by iterating the equation $z = z^2 + c$. **The primary difference between the Julia set and the Mandelbrot set is the manner in which the function is iterated. The Mandelbrot set iterates as per $z = z^2 + c$ with z always starting at 0 and varying the c value. The Julia set iterates as per $z = z^2 + c$, where c is constant and z is variable. In other words, the Mandelbrot set is in the parameter space or the c-plane, while the Julia set is in the dynamical space, i.e., the z-plane.**

An Iterated Function System (IFS) based on a co-efficient c and the maximum number of iterations, is iterated as many times as the maximum number of iterations. The resulting set of points can span an indeterminable amount of space that is a function of the number of iterations involved. Now, for any randomly chosen point z from this set, it can either be located inside or outside of the generated area, depending on the value of c and the co-ordinate-axes range. The Julia Set comprises all such points z, each of which lies outside of the bounded space before the IFS was applied. Applying the set of affine transformations on the starting set of points, in each iteration step, the resulting fractal is a self-similar shaped image that is resembles an approximation to the original image. This effect is best visualized when rendered in 3D.

1) Relationship between Mandelbrot and Julia Sets:

Mandelbrot set is a one-page dictionary of Julia sets. It is so called because if we enlarge the Mandelbrot set. Sufficiently at any given point we obtain something that looks very much like the Julia set at that point. Let us see what that means mathematically. Suppose we have a Julia set for which we set the value of c to w. Consider the point of the Julia set at which z0 is also w. We then have

$$z_0 = w$$
$$z_1 = w^2 + w$$

Now let us look at the Mandelbrot set at the point where c is w. We have

$$Z_0 = w$$
$$Z_1 = w^2 + w$$

Thus, the result of iterating is the same for both the Julia and Mandelbrot sets at this particular point. Now let us move from this point by a distance of d, where d is also a complex number. For the Julia set, we then have

$$Z_0 = w + d$$
$$Z_1 = (w+d)^2 + w$$

And for the Mandelbrot set, we then have

$$Z_0 = w + d$$
$$Z_1 = (w+d)^2 + w+d$$

We see that if d gets smaller and smaller, the result of iterating again and again is the same for the two sets, but as d increases, the results diverge more and more. Furthermore, d can be varied so that the result (w+d) represents, in turn, every point on the screen. We then observe that at every point on the screen, the iteration begins in the same way for both the sets, but diverges more and more as the distance d increases. Hence, for very large magnifications, the pictures of the Mandelbrot and Julia should be the same.

III. Method of generation of Mandelbrot and Julia Sets

A step-by-step description of the methods used to generate the Mandelbrot and Julia Sets are outlined in the sub-sections that follow.

A. Method of generation of Mandelbrot Set

The steps used in the generation of the Mandelbrot Set are as follows:

1. For a given maximum number of iterations denoted by the constant maxit, plot the Set defined by the (min, max) range of values of each point the axes (Xmin, Xmax), (Ymin, Ymax).
2. This is done by first generating a grid based on the line spacing values for each Xz and Yz. The line spacing is calculated using the distance of each such point based on (Xmin, Xmax) & (Ymin, Ymax), set to lower and upper limits from the origin of z, whose initial value is always 0. For each co-ordinate axis involved, we get a line spacing value, e.g., Xlinespacing, Ylinespacing etc.
3. Then the actual Mandelbrot Set is plotted by selecting all points in the grid obtained from the step above, and checking if it qualifies as a candidate for being part of a Mandelbrot Set. If the difference between these two line spacing values is 0, then the corresponding point belongs to the Mandelbrot Set being generated. And the union of all such points gives the domain of the resulting Mandelbrot Set.
4. The next step is to plot the actual Mandelbrot Set based on the set of points obtained from step 3 above. This is done by iterating the above steps based on the max number of iterations (this is now an input variable) and the (Xmin,Xmax), (Ymin, Ymax) ranges, till all the iterations are complete. However, a super limit for the maximum number of iterations being input is fixed at value greater than maxit, to ensure that the generated Mandelbrot Set doesn't blow up into an infinite space on the screen.
5. The program runs as a Windows console application with the 3D image of the Mandelbrot Set being

rendered as a colour-mapped image projected onto the 3D plane, using MATHLAB 3D Image Rendering software.

The use of OpenGL algorithms and the standard Graphics library makes the implementation of the routines for construction of the initial grid and the subsequent Mandelbrot Set as also the colour-coding and 3D rendering very flexible and efficient. Additional details and other techniques to generate the 3D Mandelbrot Set can be found in the book as in [1] and the publications [3]-[6].

B. Method of generation of Julia Set

The steps used in the generation of the Julia Set are as follows:

1. For a given maximum number of iterations denoted by the constant maxit, and a co-efficient parameter denoted by c, plot the Set defined by the (min, max) range of values of each point in the region defined by (Xmin, Xmax), (Ymin, Ymax).
2. This is done by first generating a grid based on the line spacing values for each Xz and Yz. The line spacing is calculated using the distance of each such point based on (Xmin, Xmax) & (Ymin, Ymax) set lower and upper limits from the point z, whose value depends as it moves in the dynamical space (z-place). For each co-ordinate axis involved, we get a line spacing value, e.g., Xlinespacing, Ylinespacing etc.
3. Then the actual Julia Set is plotted by selecting all points in the grid obtained from the step above, and checking if it qualifies as a candidate for being part of a Julia Set. If the difference between these two line spacing values is 0, then the corresponding point belongs to Julia Set being generated. And the union of all such points gives the domain of the resulting Julia Set.
4. The next step is to plot the actual Julia Set based on the set of points obtained from step 3 above. This involves evaluation for each of these points and checking to see if it qualifies as a member of the Julia Set being generated. This is done by iterating the above steps based on the max number of iterations (this is now an input variable) and the (Xmin,Xmax), (Ymin, Ymax) ranges, and the co-efficient c, till all the iterations are complete. However, a super limit for the maximum number of iterations being input is fixed at value greater than maxit, to ensure that the generated Julia Set doesn't blow up into an infinite space on the screen.
5. The program runs as a Windows console application with the 3D image of the Julia Set being rendered as a colour-mapped image projected onto the 3D plane, using MATHLAB 3D Image Rendering software.

The use of OpenGL algorithms and the standard Graphics library makes the implementation of the routines for construction of the initial grid and the subsequent Julia Set as also the colour-coding and 3D rendering very flexible and efficient. Additional details and other techniques to generate the 3D Julia Set can be found in the book as in [1] and the publications as in [3]-[6].

IV. EXPERIMENTAL RESULTS

This section depicts the results obtained by applying above described methods to generate the Mandelbrot and Julia Sets based on the defined parameters, and the subsequent rendering of each image output in 3D using MATHLAB Software. Full colour images of each of the outputs obtained are also presented as figures. Additional details and other methods to generate the 3D Mandelbrot and Julia Sets can be found in the book as in [1] and the publications [3]-[6].

C. Generation of the 3D Mandelbrot Set

The corresponding program is written in C++ and built as a Visual C++ Project of type Windows Console Application. The resulting Mandelbrot Set is rendered as a 3D image using MATLAB 3D Image Rendering software). Multiple 3D Mandelbrot Sets are generated by varying the number of iterations and the boundary of the initially defined region.

Fig. 1 shows the initial 2D image of the Mandelbrot Set. Figs. 2-4 show the outputs of the same based on 25 and 100 iterations and rendered in 3D.

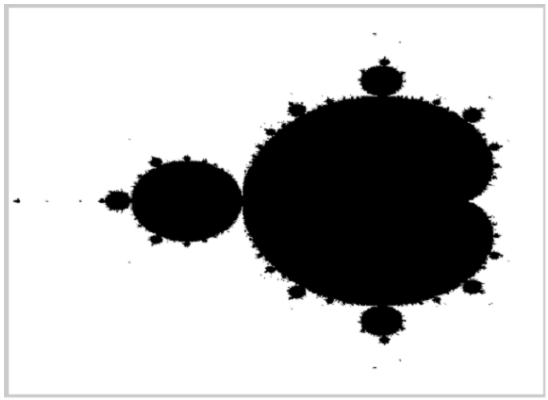

Fig. 1 2D Mandelbrot Set

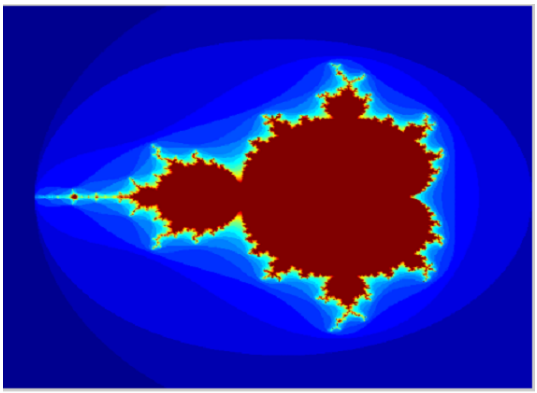

Fig. 2 3D Mandelbrot Set generated using 25 iterations, and (X, Y) ranges as [-2.2, 1] and [-1.6, 1.6]

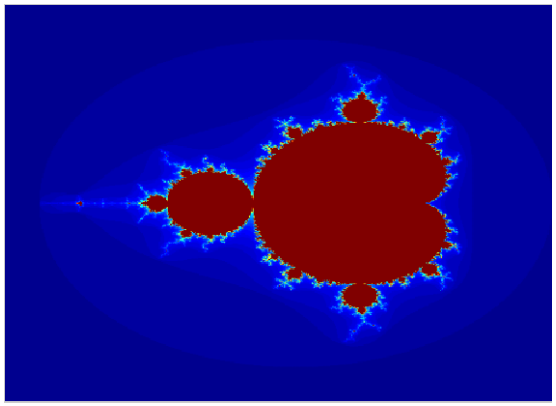

Fig. 3 3D Mandelbrot Set generated using 100 iterations, and (X, Y) ranges as [-2.2, 1] and [-1.6, 1.6]

By keeping the number of iterations same as 100, if the (X, Y) region for the grid is changed, a different 3D image of the Mandelbrot Set is obtained, as shown in Fig. 4.

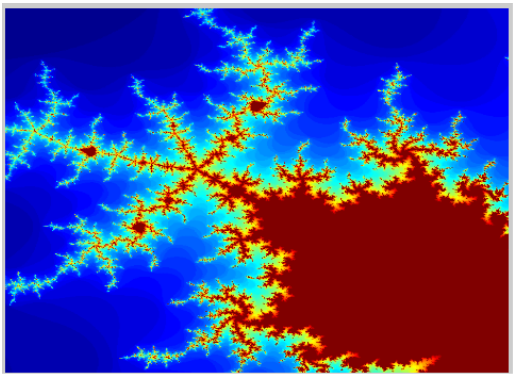

Fig. 4 3D Mandelbrot Set generated using 100 iterations, and (X, Y) ranges as [-1.25, -1.20] and [0.14, 0.19]

D. Generation of the 3D Julia Set

The corresponding program is written in C++ and built as a Visual C++ Project of type Windows Console Application. The resulting Julia Set is rendered as a 3D image using MATLAB 3D Image Rendering software. Multiple 3D Julia Sets are generated by varying the number of iterations and the random co-efficient c. Fig. 5 shows the initial 2D image of the Julia Set. Figures 6-8 show the outputs of the same based on 25 and 100 iterations.

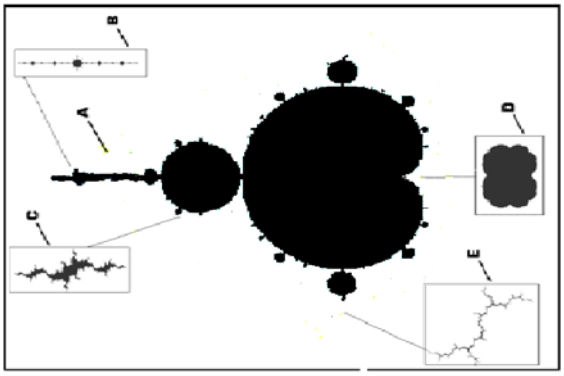

Fig. 5 2D Julia Set – Initial Image – The sub-images A, B, C, D, E are the derived 2D Julia Sets obtained by an iterative zoom-in process of the 2D version of Mandelbrot Set.

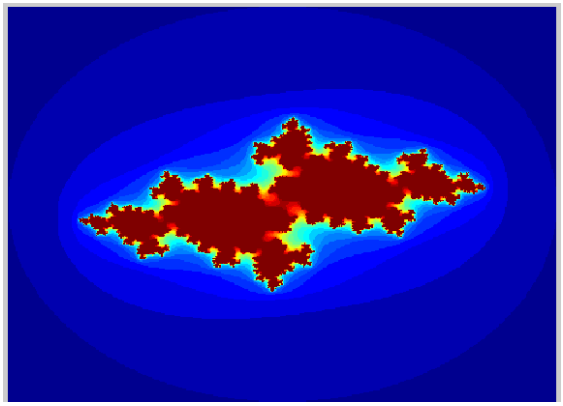

Fig. 6 3D Julia Set generated using 25 iterations, and (X, Y) ranges as [-2, 2] and [-2, 2] and c = -0.7-0.3*iter (iter = 1 to 25)

\

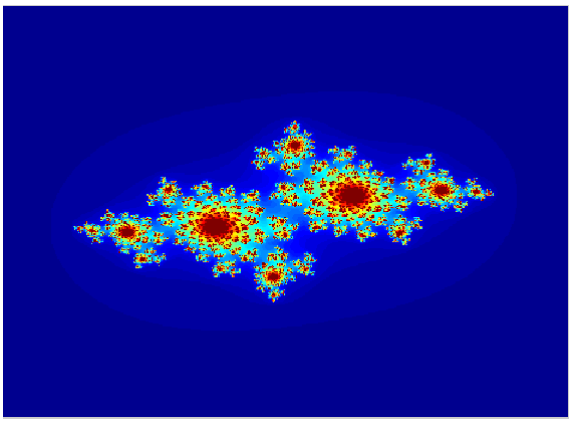

Fig. 7 3D Julia Set generated using 100 iterations, and (X, Y) ranges as [-2, 2] and [-2, 2] and c = -0.7-0.3*iter (iter = 1 to 100)

If the co-efficient is changed, keeping the number of iterations as 100 and the (X, Y) region the same as in Fig. 7, a 3D image of the Julia Set is obtained, as shown in Fig.8. Here we change the value of c to be calculated as
c = -0.85 – 0.22*iter (iter = 1 to 100).

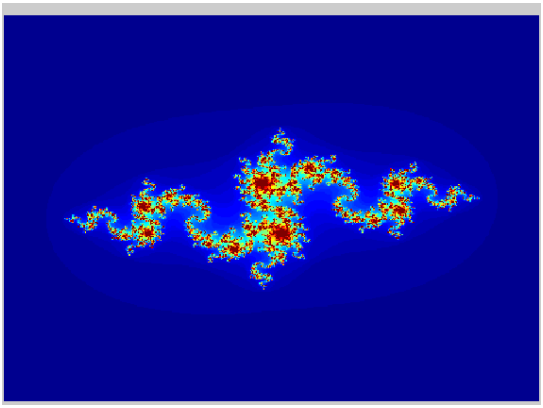

Fig. 8 3D Julia Set generated using 100 iterations, and (X, Y) ranges as [-2, 2] and [-2, 2] and **c = -0.85-0.22*iter (iter = 1 to 100)**

V. CONCLUSIONS

The above experiments demonstrated the process of generating the 3D fractal images of the Mandelbrot and Julia Sets. It appears that there is no significant difference between the basic look of the 2D and 3D versions of the Mandelbrot and Julia Sets, the 2D versions of the same look as shown in Figs. 1 and 5 respectively. On running these 2D images on

MATLAB 3D image rendering software, the 3D versions appear as shown in Figs. 2-4 for the Mandelbrot Set, and as in Figs. 6-8 for the Julia Set. The true 3D Mandelbrot is not created so far. One can have different images rendered by zooming-in iteratively, that come within close proximity to the original 2D versions. The results presented here can be improvised based on the variables involved such as the number of iterations, the 3D co-ordinates range, and the choice of calculating the co-efficient c. This enables generation of more beautiful images and in higher dimensions too.

ACKNOWLEDGMENT

The first author thanks Prof. Jibitesh Mishra, his Ph.D guide (and also the second author), for his invaluable guidance and suggestions that helped him complete the experimental results presented in this paper.

REFERENCES

[1] M. Barnsley, *Fractals Everywhere,* Academic Press, Boston, USA, 1988.
[2] Roger T. Stevens, *Advanced Fractal Programming in C*, M&T Books, Redwood City, CA, USA.
[3] Krzysztof Gdawiec, "Fractals", Tech. Rep,. May 2008.
[4] Dominic Rochon, "3D Fractals", Université du Québec à Trois-Rivières, Quebec, CANADA, CRSNG Research Project, May 2006.
[5] Tan Lei, "Similarity between the Mandelbrot Set and Julia Sets", *Communications in Math. Phys*., vol. 134, pp. 587-617, 1990.
[6] Chris King, "Exploding the Dark Heart of Chaos", University of Auckland, Research Project, Mar-Dec 2009.
[7] Peitgen H.O. et.al., *The Science of Fractal Images*, Springer-Verlag, New York , Berlin, 1988.