Game-enabling the 3D-Mandelbulb Fractal by adding Velocity-induced Support Vectors

Bulusu Rama
Assistant Professor of CSE
SLC Institute of Engg. And Technology
Piglipur(V),Hayatnagar(M),
Hyderabad – 501512(A.P.) India

Jibitesh Mishra
Head of Dept. of Information Technology
College of Engg. & Tech.
BPUT,Ghatikia.
Bhubaneswar- 751030(Odisha) India

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. Of the many different types of fractals that have come into limelight since their origin, the family of Mandelbrot Set fractals has eluded both mathematicians and computer scientists alike. And the 'true' 3D realization of the Mandelbrot set has been a challenging centre piece of research with its limits extending only to that of the sky. An earlier paper co-authored by us in 2011 explained a method of realizing a 'true' 3D simulation of the Mandelbrot set and the rendering of the same onto 3-dimensional space. This paper takes a step further in using this variant of the Mandelbulb as input and outlines a method of the game-enabling of the same Mandelbulb by using direction-oriented vectors that are analogous in function to that of Support Vectors in the Support Vector Graphics (SVG) domain. A real-world application of the same can translate to examples of understanding an entire coast-line set to motion in space by adding 3D-animation enabled elevation to the corresponding fractal image.

General Terms
Algorithms, Fractal Geometry

Keywords
Fractals, Mandelbrot Set, Mandelbulb, Three Dimensional Velocity, Rendering, Support Vectors

1. 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. A 3D rendering of the Mandelbrot set is popularly termed as the Mandelbulb. This paper takes a step further in game-enabling the Mandelbulb by adding variant input based velocity induction that sets the Mandelbulb in a dynamically controllable motion. It does this by using a dynamically set speed 2-tuple that acts as a Support Vector lever to kick-off the game-enablement. The source image of the 3D Mandelbulb is generated based on rotation of the Mandelbrot set away from the azimuthal or the z-axis, followed by an IFS-based repeated execution and the rendering of the same in 3D. The game-enabling program was implemented using the Python Visual Development Kit and the Python Game API and rendered using the VIDLE GUI based on the algorithms presented in the following sections. The displayed output of the same based on a typical set of inputs is presented. The experimental results are shown in section 4 and concluding remarks in section 5.

2. ABOUT MANDELBROT SET AND 3D MANDELBULB
A brief description of Mandelbrot Set and the 3D Mandelbulb along with their properties followed by a description of game-enabling the Mandelbulb based on adding variant support vectors game-enabling on iterative function systems is presented in the paragraphs that follow. The Mandelbrot Set was invented by the French mathematician Benoit Mandelbrot in 1979, when he was working on the simple equation $z = z^2 + c$. 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 $z = z^2+c$ 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 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 = z2+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. The true canonical form

of 3D Mandelbrot set is deemed to be non-existent as the 3D equivalent of the 2D complex plane is non-existent.

However, using an nth power of a 3D hyper-complex number, a true 3D version of the same can be obtained and is often referred to as the Mandelbulb. This is obtained by using a rotation transformation away from the z-axis.

The corresponding rotational matrix of the Mandelbulb is

$$\{x, y, z\}^{\wedge}n = R_z(n\,\theta)*Ry(-n*\varphi)*\{r^{\wedge}n, 0, 0\}$$

where
$r = SQRT(x^2+y^2+z^2)$,
θ (theta) = ATAN2(y, x),
φ (phi) = ASIN(z/r)

This does an azimuthal angle (phi) rotation away from the azimuthal axis (z-axis).
The 3D version is obtained using the iteration $z = z^n + c$ where z and c are 3-dimensional hyper-complex numbers with the power mapping $z \rightarrow z^n$ defined as stated above. For $n > 3$, the result is a 3-dimensional bulb-like structure with fractal surface detail and a number of "lobes" controlled by the parameter n. Multiple graphic renderings can be generated using n = 8.
The rotational matrices of the 3D Mandelbulb can be defined as
$R_x(\theta)$ = matrix[(1 0 0), (0 cos(θ) - sin(θ)), (0 sin(θ) cos(θ))],
$R_y(\theta)$ = matrix[(cos(θ) 0 sin(θ)), (0 1 0), (-sin(θ) 0 cos(θ))],
$R_z(\theta)$ = matrix[(cos(θ) -sin(θ) 0), (sin(θ) cos(θ) 0), (0 0 1)]

The expanded trigonometric form of the same is given below.

$$\{x, y, z\}^{\wedge}n = r^n\{a_1\cos(n\theta) – a_2\sin(n\theta), a_1\sin(n\theta) + a_2\cos(n\theta), x\sin(n\varphi)/r_{xy}\}$$

The corresponding equation for n=2 is as follows:

$$\{x, y, z\}^2 = \{x^2 – y^2 – (x^4 – 6x^2y^2 + y^4)z2 / r^4_{xy}, 2xy(1 – 2(x^2 – y^2)z^2/r^4_{xy}), 2xz\}.$$

3. METHOD OF GAME-ENABLING THE MANDELBULB

A step-by-step description of the method used to game-enable the Mandelbulb is outlined in the following steps:

1. Adding a support vector property that is input-variant to a chosen 3D Mandelbulb fractal image.
2. The support vector consists of a 2-tuple speed vector [sx, sy] corresponding to the [horizontal, vertical] velocities that sets the Mandelbulb in motion (simultaneously) in either direction.
3. Setting the reference frame of motion wherein the Mandelbulb games through driven by the added support vector.
4. Using brit-and-flip graphics functionality to dynamically contain the Mandelbulb within the boundaries of the frame-of-motion.
5. The dynamic variables involved in this method are:
    a. The speed vector 2-tuple.
    b. The referential frame-of-motion itself.
    c. The back ground colour.
    d. The maximum time period that allows the Mandelbulb to game in motion.

The program runs as a Windows console application using the VIDLE GUI, with the 3D Mandelbulb being input as a colour-mapped JPEG image obtained by projecting the Mandelbrot set onto the 3D plane, using MATHLAB 3D Image Rendering software. The game-enabling was achieved using the Python Visual Development Kit and the Python Game API also called pygame. The flex-based gaming-in-motion is as depicted in Figure 2.

The pygame() is a Python-enabled Visual Graphics API that is Open-source and compatible with OpenGL algorithms and other standard Graphics libraries. This eased the implementation of the routines for constructing the initial reference frame-of-motion and the subsequent velocity-induced support vectors that set the Mandelbulb in motion-driven gaming in a very flexible and efficient fashion. Additional details and other related techniques for further research can be found in the books as in [3], [4], and the publications [5]-[8].

4. EXPERIMENTAL RESULTS

This section depicts the results obtained by applying above described method(s) to implement the Mandelbulb game-enabling based on the defined variant parameters, and the subsequent gaming-in-action is captured as a Video Clip in AVI format. The program was written in the Python programming language, the Python-based 3D “visual” Graphics library *VPython*, and the Python Game API called *pygame*. It runs as a Windows console application using the VIDLE GUI, with the 3D Mandelbulb being input as a colour-mapped JPEG image obtained by projecting the Mandelbrot set onto the 3D plane, using MATHLAB 3D Image Rendering software.

The flex-based gaming-in-motion is as depicted in Figure 2. That contains an embedded AVI Object of the corresponding Video Clip.

Figure 1 shows the ‘true’ 3D simulated Mandelbulb fractal image that is used as primary input source for the game-enabling process. Figure 2 shows the flex-version of the motion-based output of the game in action.

Fig. 1: The 'true' 3D simulated Mandelbulb fractal image that is used as primary input source for the game-enabling process.

gameEnablingMandelbulb.avi

Fig. 2: The embedded link to the flex-version Video Clip of the motion-based output of the Mandelbulb in action. Double-clicking the image activates the video clip.

5. CONCLUSION

The 3D version of the Mandelbrot set termed as Mandelblub gives an idea about the nature and mathematical aspect of the Mandelbrot formula. Though the authors mentioned in their earlier communication that a true 3D version of Mandelbrot set is not created so far [2], the JPEG image of the 3D Mandelbulb used as the input source is one obtained by an pragmatic attempt made to create a 3D version of Mandelbrot set based on rotation away from the azimuthal or the z-axis. It is to be noted that there are many such Mandelblub made by researchers from time to time. However none of them truly represents the 3D version of Mandelbrot set. This image is an interim result of an ongoing work. This paper presents an extension of the Mandelbulb by enabling it to be used in 3D Games in a flexible manner opens the doors for an entirely different domain of fitting the infinite subversions in the finite world - where the so-termed infinitely dimensional Mandelbulb (a zoomed-in version of the Mandelbrot set that tends to extend to infinity), namely the Mandelbrot Set and the Mandelbulb, is algorithmically interpolated by way of theory and experimentation, to be dynamically set to motion by adding the support vector as the extra lever. A real-world Imagineering of the same can translate to examples like an entire coast-line set to dance in space by adding 3D-animation enabled elevation to the corresponding fractal image.

6. REFERENCES

[1] B. Rama and J. Mishra, "Generation of 3D fractal images for Mandelbrot Set," In Proceedings of the Intl. Conf. Communication, Computing & Security, pp. 235-238, 2011, doi:10.1145/1947940.1947990 (ACM-International Conference Proceedings Series).

[2] B. Rama and J. Mishra, "Generation of 3D Fractal Images for Mandelbrot and Julia Sets," Intl. J. of Computer and Communication Tech. vol. 1, nos. 2-4, pp. 178-182, 2010 (Special Issue).

[3] Akenine-Moller, T., Haines, E., and Hoffman, N. 2009 Real-Time Rendering, Natick, MA: A.K. Peters, 3rd Edition.

[4] Roelofs, G., 1999. PNG – The Definitive Guide. Sebastopol, CA: O'Reilly & Associates.

[5] Scherer, D., Andersen, D., Brandmeyer, J., Chabey, R., Heitner, A., Peters, I., and Sherwood, B. 2000 VPython – 3D Programming for Ordinary Mortals, available at http://www.vpython.org/index3.html

[6] OOoAuthors Group. 2010. "Changing Object Attributes," OpenOffice.org 3 Draw Guide, Chapter 04, available at :http://wiki.services.openoffice.org/wiki/Documentation/OOo3_User_Guides/Draw_Guide/Changing_Object_Attributes

[7] Lei, T. 1990. "Similarity between the Mandelbrot Set and Julia Sets", Communications in Math. Phys., vol. 134, pp. 587-617.

[8] Peitgen, H.O., Saupe, D., Fisher, Y., McGuire, M., Voss, R.F., Barnsley, M.F., Devaney, R.L., and Mandelbrot, B.B. 1988. The Science of Fractal Images, New York, NY: Springer-Verlag.

[9] N. Yokoya, K. Yamamoto, and N. Funakubo, "Fractal-based analysis and interpolation of 3D natural surface shapes and their application to terrain modeling," Computer Vision, Graphics, and Image Processing, vol. 46, pp. 289–302, 1989.

[10] G.B. Wyvill and C. McPheeters, "Solid texturing of soft objects," IEEE Computer Graphics and Applications, vol. 7, no. 12, pp. 20–26, 1987.

[11] Norton, "Generation and display of geometric fractals in 3D," Computer Graphics, vol. 16, no. 1, pp. 61–67, 1982.[12] E. Groeller and H. Wegenkiltl, "Interactive design of non-linear functions for iterated function systems," Proc. 4th Intl. Conf. Computer Graphics and Visualization, (WSCG'96), pp. 93–102, Plzen, 1996.

[12] M.F. Barnsley, J.H. Elton, and D.P. Hardin, "Recurrent iterated function systems," Constructive Approximation, vol. 5, pp. 3–31, 1989.[14] M.F. Barnsley and J. Hutchinson, "New methods in fractal imaging," Proc. CGIV, (CGIV'06), Sydney 2006.

[13] S. Nikiel, "An efficient fractal modeller (modification of the IFS)," Proc. 4th Intl. Conf. Computer Graphics and Visualization, (WSCG'96), pp. 294–300, Plzen, 1996.

[14] J.J. VanWijk and D. Saupe, "Image-based rendering of iterated function systems," Computers & Graphics, vol. 28, no. 6, pp. 937–943, 2004.

[15] D.M. Monro and F. Dudbridge, "Rendering algorithms for deterministic fractals," IEEE Computer Graphics and Applications, vol. 272, no. 17, pp. 32–41, 1995.

[16] Y.Q. Chen and G. Bi, "3D IFS fractals as real-time graphics model," Computers & Graphics, vol. 21, no. 3, pp. 367–370, 1997.

[17] Mandelbrot, M. 1982. The Fractal Geometry of Nature, New York, NY: W.H. Freeman & Co.