SIERPINSKI JELLY
Iterated Function Systems as Elastic Bodies

Pawel Filipczuk[1], Slawomir Nikiel[1] and Korneliusz Warszawski[2]
[1]*Institute of Control & Computation Engineering, University of Zielona Gora, Podgorna 50, Zielona Gora, Poland*
[2]*Faculty of Electrical Engineering, Computer Science and Telecommunication, University of Zielona Gora Podgorna 50, Zielona Gora, Poland*

Keywords: Fractals, Iterated Function System, Real-time Simulation, Animation.

Abstract: Relatively simple ideas of fractal geometry result in an infinite number of complex images and objects. Fractals are used in computer graphics to increase visual fidelity of the vector models. Although derived from dynamical systems, fractals are usually presented as static objects. The paper presents a new concept of embedding physical description in the model of IFS (Iterated Function System). Dynamically changing fractal structures offer better sense of 'material' than static images or key-framed animations. The model can augment IFS attractors with the illusion of softness, weight and other material-related features. The proposed model is flexible, deterministic and offers high rendering performance.

1 INTRODUCTION

We can observe tremendous upgrade in computer graphics. Computing power of currently available PCs and game consoles support real-time rendering of high-resolution images. Image quality is still a result of trade–off between geometry and performance. Vector graphics is a core of 3D computer simulation models (Hearn and Baker, 1997); (Heckbert, 1994). Shading algorithms add visual details to the rendered scene. Fractals has always been a source for procedural shaders, greatly improving quality of images. There are fractal models that perfectly describe some organic structures (Prusinkiewicz and Lindemeyer, 1990). Genetic Programming and fractals are also used in shape grammar-based rendering (Glassner, 1989; 1992); (Mignonneau and Somerrer, 2000); (Sims, 1991). Most fractal models, however, are usually presented as static structures. This is probably due to their geometric complexity, easily consuming available computation power. IFS proposed by Barnsley were originally focused on application of affine transformations to image analysis and synthesis (Barnsley, 1993). They proved to be enough flexible to be rendered even on limited-resource mobile devices (Nikiel, 2007). IFS are part of dynamic systems (Clempner and Poznyak, 2011); (Di Trapani and Inanc, 2010), but their graphic representations are rather static. Recent developments of Super-IFS and IFS homeomorphism open path to dynamical morphing of flat images and textures (Barnsley, 2006). Extension of classical IFS with a vector model along with the purely deterministic rendering algorithm enabled real-time IFS shape modelling (Nikiel, 2005). Adding non-scaling parameters such as a type of vector object to the affine maps enhances the process of fractal shape construction. Key-framed and parameter-driven animations of IFS attractors have also been discussed in literature (Barnsley, 2006). Adding physical description to IFS model offers quite new interaction and simulation properties. Fractals behave 'naturally' reacting to gravity and deformations in realistic manner. IFS attractors can be elastic bodies. User can interact with them the same way as with their real-life counterparts.

The paper is organized as follows: First Section describes theoretical background of the model including necessary precautions to determine the set of IFS functions. Then the IFS with embedded physics model is described. It is followed by presentation and discussion on real-time rendering and simulation. Concluding remarks sum up advantages of the proposed method. Directions for further developments are indicated at the end of the paper.

Filipczuk P., Nikiel S. and Warszawski K..
SIERPINSKI JELLY - Iterated Function Systems as Elastic Bodies.
DOI: 10.5220/0003829803610364
In *Proceedings of the International Conference on Computer Graphics Theory and Applications* (GRAPP-2012), pages 361-364
ISBN: 978-989-8565-02-0
Copyright © 2012 SCITEPRESS (Science and Technology Publications, Lda.)

2 ITERATED FUNCTION SYSTEMS

2.1 Background

The Iterated Function Systems theory defines mathematically some concepts of chaos and irregularity in geometry. Research done mainly by Barnsley led to significant new methods for image understanding (Barnsley, 1993; 2006). A basic set of tools for image construction is created through a set of simple geometric transformations. IFS are based on mathematical foundations laid by Hutchinson. An IFS fractal is constructed from a collage of transformed copies of itself. The transformation is performed by a set of affine maps. An affine mapping on a plane is usually a combination of rotation, scaling and translation in R^2. There are no particular conditions imposed on the maps except their contraction (Falconer, 1990).

A set of affine transformations is accompanied by respective contraction factors. They are relatively easy to estimate for classical linear two-dimensional IFS. When all contraction factors are less than 1 the IFS are called to be hyperbolic IFS. If each mapping has a specific measure assigned a probability IFS are called IFS with probabilities. Probabilities can be proportional according to Jacobians of transformation matrices. Proper adjustment of probability according to a given contraction factor enhances the rendering process. The average contraction suggests that even non-hyperbolic and non-linear IFS can posses their constant points, called the attractors. Estimation of the contraction factor for non-linear IFS is not trivial (Skarbek, 2006). It is often a matter of trial and error process to check the existence of the IFS attractor. If we consider an IFS to be a hyperbolic and linear, then the situation is quite safe. We will always obtain the image of the IFS attractor. With the help of the Collage Theorem it is possible to design interactively nature-like images and shapes.

2.2 IFS with Physics

First implementations of the IFS used binary images and were rather inefficient. There have been many improvements made over the IFS models and alternative fractal rendering methods. IFS (Partitioned IFS) and VRIFS (Vector Recurrent IFS) are incorporated in fractal image compression and decompression schemes (Barnsley, 1993). Polar IFS along with Genetic Programming is a very interesting alternative targeted at inverse problem: how to find IFS set for a given image (Collet et al., 1999). Adding color-space to the IFS codes enhanced the process of colorization of fractal models (Nikiel, 1998). In the field of rendering methods, Dubuc and Elquortobi recognized that only new points of the IFS attractor are necessary to visualize the attractor (Dubuc, 1990). Monroe and Dudbridge developed an optimized version for on-screen display of IFS images, called the Minimal Plotting Algorithm (Monroe and Dudbridge, 1995). Bell proposed a recursive rendering scheme called Tesseral Synecdoche Algorithm (Bell, 1995); (Bell and Holroyd, 1991). Vector Recursive Rendering is another algorithm that can be used to transform an arbitrary set of points in n-dimensional spaces (Nikiel, 2005). Considering above-mentioned developments and purely deterministic character of some IFS models it is possible to render their attractors in real time. This enables interaction and animation of IFS fractals. There have been many attempts to use key-framed or parameter-driven animations (Barnsley, 2006). Adding physical description to IFS model opens quite new ways of interaction and simulation. Fractals behavior might be sensitive to deformations, gravity or other kind of forces.

Let (F, d) be a complete metric space. Let $F \rightarrow F$ be a collection of mappings (ω_i; i=1,2,…,L) operating on points (p) in F, then the

$$\Omega=(F, (\omega_i); i=1,2,\dots,N) \tag{1}$$

is called the Iterated Function System. An affine transformation ω_i scaling and translating points (p) in R^3 has the form, and can be treated as a transformation matrix M_T:

$$P' = M_T \cdot P = \begin{bmatrix} s_x & 0 & 0 & t_x \\ 0 & s_y & 0 & t_y \\ 0 & 0 & s_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2}$$

where (t_x, t_y, t_z) determine translations and (s_x, s_y, s_z) determine scaling. A 2D example of such IFS model might be Sierpinski triangle built with three transformations. The IFS described above can be treated then as an elastic body by using classical physical formulas to define its behavior. Fractal objects can be influenced by external forces of gravity and wind. They can also collide with each other as well as with other objects or with the ground.

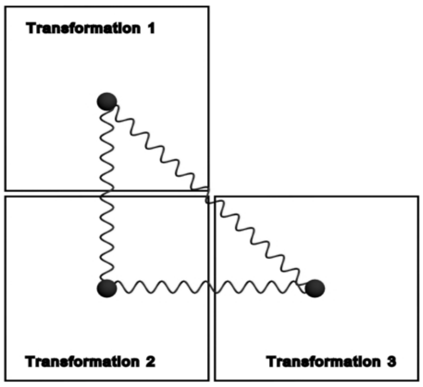

Figure 1: The model of IFS representing rigid bodies connected with springs.

Fig. 1 depicts an example, in which transformations are treated as rigid bodies connected by springs (for the model defined by Eq. 2). Spring forces are calculated using following formula:

$$F = F_{spring} + F_{damping} = -k \cdot stretch - bv \tag{3}$$

where *k* is the spring stiffness coefficient, *stretch* is the difference between neutral length of the spring and its current length, *b* is the dumping coefficient and *v* is a relative velocity between both ends of the spring. The resultant force $F_{resultant}$ acting on objects is a sum of forces of connected springs and other forces like gravity, wind, etc. In the example presented in this paper, simple Euler integration method have been used.

Let $P=[t_x, t_y, t_z]$ represent a position of the fractal object. Then, following formulas describe new value of *P* after time:

$$\begin{gathered} a = \frac{F_{\text{resultant}}}{m} \\ v' = v + a\Delta t = v + \frac{F_{\text{resultant}}\Delta t}{m} \\ P' = P + s = P + v'\Delta t = P + \left(v + \frac{F_{resultant}\Delta t}{m}\right) \cdot \Delta t \end{gathered} \tag{4}$$

where *v* is velocity and *m* is mass of the object. Final transformation used in the IFS to generate the object in current frame of simulation depends on forces acting on it, time between the current and the previous frame, on its mass and on previous position and velocity. Final transformation matrix based on Eq. 2 can be described by:

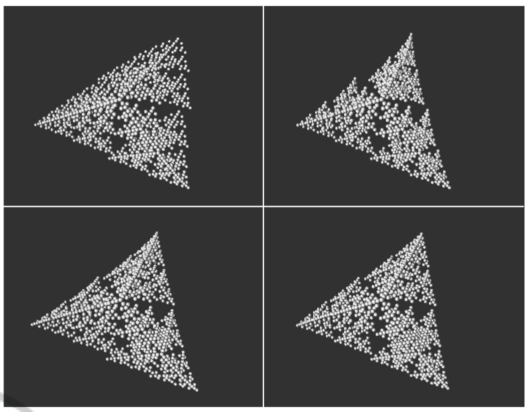

Figure 2: A few screens from the simulation.

$$M_T = \begin{bmatrix} 1 & 0 & 0 & P_x + \left(v_x + \frac{F_x\Delta t}{m}\right) \cdot \Delta t \\ 0 & 1 & 0 & P_y + \left(v_y + \frac{F_y\Delta t}{m}\right) \cdot \Delta t \\ 0 & 0 & 1 & P_z + \left(v_z + \frac{F_z\Delta t}{m}\right) \cdot \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

It is possible to apply this method to other transformations, like scaling, rotation or skew but their physical interpretation is less obvious and could not be explained directly as for translation. Final transformation set used in our prototype application consists of all these transformations and is calculated as follows:

$$M = M_S \cdot M_{RC}^{-1} \cdot M_R \cdot M_{RC} \cdot M_T \tag{6}$$

where M_S is scaling, M_R is rotation, M_T is translation and M_{RC} is the center of rotation matrix.

2.3 Rendering Algorithm

The physical IFS objects are rendered with vector recursive rendering (VRR) (Nikiel, 2005). It provides fast and deterministic way to calculate structure of the object. According to the VRR algorithm set transformations are used *N* times on initial point to estimate fractal attractor. The final set of points might be drawn directly on the screen or rendered as coordinates of billboards or CSG sets. The amount of physical calculations depends on the number of transformation, not on *N* or the final complexity of the object.

A deterministic rendering scheme is necessary to construct physics-based three-dimensional fractals. Either Tesseral Synecdoche Algorithm or Vector Recursive Rendering Algorithm can be adapted to handle the model described in the previous Section.

3 IMPLEMENTATION

The simulation and rendering procedure was implemented using C++ and DirectX API. The application delivers full structure in real-time. The overall performance is very fast and some of the operations may be performed at the GPU. The scene was composed of 256 geometric objects, which contained 10,752 vertices and 20,480 faces all together. Average efficiency was about 165 FPS. Without rendering it was about 780 FPS. The prototype application was running on AMD Athlon 64 3000+ (1.8GHz) with ATI Radeon X700Pro graphics card. Fig. 2. presents sample screens from the simulation.

4 CONCLUSIONS

The developments described in the paper broaden the application area of fractal modeling in three dimensional vector graphics with the idea of physical behavior of fractals. It opens up even more possibilities to create artificial objects. When simulated or interactively manipulated, physics-based IFS attractors acts 'naturally' showing their mass (inertia) or elasticity (they behave like jelly, strings, plants or fur). The model presented in the paper is relatively simple and provides real-time interaction. It changes appearance of fractals from complex static images to dynamically changing structures. It is possible to describe IFS physics in more advanced way, including collision detection. Transformations might influence each other with gravity or electromagnetic field instead of springs. That would simulate objects like galaxies or atoms. Considering plants, the hierarchical structure might be used.

REFERENCES

Barnsley, M. F., 2006. Superfractals, Cambridge University Press, N.Y.

Barnsley, M. F., 1993. Fractals Everywhere, 2nd Edition, San Diego, CA, *Academic Press*.

Barnsley, M. F., 1993. Fractal Image Compression, Wellesley, MA, *Academic Press*.

Bell, S. B., 1995. Fractals: A Fast, Accurate and Illuminating Algorithm, Image and Vision Computing, 13(4), 253-277.

Bell, S. B., and Holroyd, F. C., 1991. Tesseral Amalgamators and Hierarchical Tilings, Image and Vision Computing, 9(5), 313-328.

Clempner, J. B., Poznyak A. S, 2011. Convergence method, properties and computational complexity for Lyapunov games, *AMCS*, Vol. 21

Collet, P., and Lutton, E. et al. 1999. Polar IFS and Individual Genetic Programming, Technical Report, *INRIA Research Reports*.

Di Trapani, L. J., Inanc T., 2010. NTGsim: A graphical user interface and a 3D simulator for nonlinear trajectory generation methodology, *AMCS*, Vol. 20

Dubuc, S., 1990. Approximations of Fractal Sets, *J. Computational and Applied Math.*, 29, 78-89.

Falconer, K., 1990. Fractal Geometry, Mathematical Foundations and Applications, New York, *John Wiley&Sons*.

Glassner, A. S., 1989. An Introduction to Ray Tracing, San Diego, CA, *Academic Press*.

Glassner, A. S. 1992. Geometric Substitution: A Tutorial, *IEEE Computer Graphics and Applications*, 12(1), 22-36.

Hearn, D. and Baker, P., 1997. Computer Graphics C-Version, *Upper Saddle River*, New Jersey, Prentice Hall.

Heckbert, P., 1994. Graphics Gems IV, London, *Academic Press*.

Mignonneau, L., and Sommerer, Ch., 2000. Modeling Emergence of Complexity: The Application of Complex System and Origin of Life Theory to Interactive Art on the Internet, Artificial Life, *Proc. of the 7th Int. Conf. on Artificial Life*, Cambridge, MA, MIT Press, 547-554.

Monro, D. M., and Dudbridge, F., 1995. Rendering Algorithms for Deterministic Fractals, *IEEE Computer Graphics and Applications*, January 1995, 272(17), 32-41.

Nikiel, S., A Proposition of Mobile Fractal Image Decompression, *AMCS*, Vol. 17, No. 1, 129-136.

Nikiel, S., 2005. Integration of Iterated Function Systems and Vector Graphics for Aesthetics, to be published in *Computers&Graphics*, 30(2).

Nikiel, S., 1998. True-color Images and Iterated Function Systems, *Computers&Graphics*, 22(5), 635-640.

Prusinkiewicz, P. and Lindemeyer, A., 1990. The Algorithmic Beauty of Plants, New York, *Springer Verlag*.

Sims, K., 1991. Artificial Evolution for Computer Graphics, Computer Graphics, 25(4), 319-328.

Skarbek, W.: On Convergance of Affine Fractal Operators, *Image Processing and Communications*, Vol. 1, No. 2, 33-41