Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Graphics 30 (2006) 277–283
COMPUTERS & GRAPHICS
www.elsevier.com/locate/cag

Integration of iterated function systems and vector graphics for aesthetics

Slawomir Nikiel

Department of Control and Computation Engineering, University of Zielona Góra, ul. Podgórna 50, 65-246 Zielona Góra, Poland

Abstract

Fractals are an integral part of many computer graphics applications. Relatively simple implementations of fractal modeling result in an infinite number of complex and attractive images. The subject of fractal image generation is well covered in the literature. Many fractal models are based on a raster representation, which is rather inconvenient for interactive modeling and implementation in 3D graphics. On the other hand, vector-geometry models are very popular in commercial graphics design packages. This creates a demand for vector representation of fractal models. The paper introduces the iterated function systems (IFS) vector-based model along with the supporting rendering algorithm. The solution offers high performance and modeling flexibility. IFS, when combined with vector graphics, offer new forms of artistic expression.

Keywords: Fractal modeling; Image synthesis; Rendering algorithms; Interactive design; Vector graphics

1. Introduction

The works of Barnsely have often focused on applications of affine transformations in fractal modeling and fractal image compression. These iterated function systems (IFS) have been thoroughly studied, especially for two-dimensional (2D) binary and gray-scale image representations [1–4]. Programs generating raster images of various 2D IFS fractals are widely available [5]. The IFS description provides a useful method for researching image shape and texture. It forms, through a set of simple geometric transformations, a basic set of tools for interactive image construction. The 2D binary IFS is the most classic raster implementation of the IFS. Vector-based fractal models are better known from work on PL systems [6], which are used to generate models of plants and simple creatures such as snails and worms. This paper describes a useful vector representation applied to the IFS. This has the advantage that the vector objects generated can be described by normals; thus, it is possible to apply lighting and shadows. It is also possible to include fractal objects in 3D modeling packages.

2. Background

IFS are based on mathematical foundations laid by Hutchinson [1]. IFS fractals have an elegant recursive definition—a fractal is constructed from a collage of transformed copies of itself; it is inherently self-similar and infinitely scaleable. The transformation is performed by a set of affine maps. An affine mapping of the plane is a combination of rotation, scaling, sheer, and translation in R^2. Any affine transformation

E-mail address: S.Nikiel@issi.uz.zgora.pl.

0097-8493/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2006.01.002

$\omega : R^2 \to R^2$ of the plane has the form

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \omega \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \qquad (1)$$

where $(x, y)(x', y') \in R^2$ are any points on a plane.

We can say that an IFS on the plane is given by a finite collection of affine transformations acting on a metric space. There are no particular conditions imposed upon the maps except their contraction. A set of affine transformations (Eq. (1)) is accompanied by respective contraction factors s. It is relatively easy to estimate such factors for 2D linear IFS. However, the situation is not trivial when considering other cases, such as nonlinear IFS. An essential feature of an IFS is its unique attractor set A, the fractal, which is closed and bounded [1].

Definition 1. If an IFS $\Omega = \{F,(\omega_i); i = 1, 2,\ldots, N\}$ is contractive, there exists a unique set

$$A = \omega(A) = \bigcup_{i=1}^{N} \omega_i(A) \qquad (2)$$

called the fixed point (the attractor) of Ω.

IFS can be treated as a subset of general dynamical systems that can have fixed points often associated with strange attractors. If we keep the contraction factors close to 1.0, we can observe that the IFS behave like a classical nonlinear system. For clearly defined attractors, we need to keep all contraction rations below 1.0, where $\max\{s_i,\ i = 1,\ldots,\ N\} < 1.0$. Such an IFS is called a hyperbolic IFS.

3. Vector IFS model

The IFS rendering algorithms usually operate on points or sets of points that may represent flat images or volumetric spaces. The authors of the Tesseral Synecdoche algorithm suggested that it is possible to insert non-scaling parameters to IFS codes, thus making possible further developments [7]. The vector recursive rendering (VRR) algorithm (described in the next section) exploits recursive function calls applied to transform a spatial vector [8]. The vector can be described either by a co-ordinate and angles, or by a pair of point co-ordinates in space.

The main idea is to transform only the two points that define the vector. The recursive process changes both the location and orientation of pairs of points. Because hyperbolic IFS contract all points, the pair defining a vector needs normalization after each mapping. The vectors are easy to calculate and, once obtained, they can be attached to any vector computer-graphics model, including those that make use of constructive solid geometry (CSG) primitives.

4. VRR algorithm

The VRR algorithm on a plane can be described by a pseudo-code as follows:

```
Choose two initial pixel values q0(x0,y0),q1(x1,y1)
Choose the max_no_of_recursions L
Procedure VRR(q0(x0,y0),q1(x1,y1),L)
Begin
   For i = 1 to N do
   Begin
      Apply q2 = wi(q0)
      Apply q3 = wi(q1)
      If L = = 0 then Apply Plot_line(Normali-
      ze(q2,q3))
      Else call VRR(q2,q3,L–1)
   End for i
End
```

The VRR is a classic recursive algorithm. It terminates when the current level of recursion L is equal to zero. The VRR algorithm is initially called with parameters VRR[(0,0),(1,1),10]—in other words, a recursive call to the function VRR with a sample initial parameters $(x_1,y_1),(x_2,y_2)$ and maximum level of recursion. It is also possible to extend this representation to any vector in a 3D space. The VRR algorithm approaches the IFS attractor rapidly and in a finite time.

We can estimate the *max_no_of_recursions L* for a given IFS codes, but its sense is a bit different for vector objects. The main difference from raster image rendering algorithms is that we do not operate on points in space but on a vector to be attached to any computer-graphics object (a circle, a triangle, a cube or a CSG solid). All computer-graphics objects have sizes larger than one pixel; therefore, it is enough to apply only a dozen steps of the recursive algorithm in order to obtain interesting results (Fig. 1) [9].

There is a problem with the simplest implementation of the VRR algorithm (without the normalization process). We know that a hyperbolic IFS is contractive, thus all points become closer with each recursive mapping. We have to implement a normalization process (denoted in pseudocode by the function *Normalize*) in order to achieve the uniform size for all vectors at each step of recursive iteration (Fig. 2). We can introduce a scaling factor *sf* that negates the contraction of two points in the space (q_0,q_1). *sf* is used to normalize the vectors at each level of recursive rendering.

For q_0 represented by a pair (x_0,y_0), and q_1 represented by a pair (x_1,y_1), on a plane (co-ordinates