

Interactive Design of Nonlinear Functions for Iterated Function Systems

Eduard Gröller and Rainer Wegenkittl

Technical University Vienna, Institute for Computer Graphics

Karlsplatz 13/186/2, A-1040 Vienna, Austria

e-mail: groeller@eigvs4.una.ac.at and wegenkittl@cg.tuwien.ac.at

Abstract: The basic requirement for the functions of an Iterated Function System (IFS for short) is contractivity. Nevertheless the majority of recent scientific investigations is concentrating on IFSs defined through a set of contractive linear functions. Simpler handling of this kind of functions and a more predictable result is the main reason for this approach. In this work we use distorted grids (representing nonlinear functions) to specify an IFS with a higher degree of flexibility and a higher modeling capability. A program for modeling these grids with so-called high-level operations is presented. Attention is directed to the interactivity of designing the IFS and rendering its limit set. Therefore a z-Buffer for displaying the result of a stochastic algorithm is included. Example images designed with the implemented software system are presented.

Keywords: iterated function systems, nonlinear, interactivity

1 Introduction

1.1 Iterated Function Systems

Main principles of fractal geometry are self-similarity and invariability of scale (which means, that parts of a fractal object resemble the whole object). The mechanism of IFSs can be derived from these principles. A fractal object is generated through a set of functions, which transform an initial object into parts of itself, on which the same functions are applied repeatedly. Self-similarity and invariability of scale are automatically generated.

In the following a short mathematical introduction to IFSs is given [1]. An IFS is defined through a finite set of contractive functions $f_i : X \mapsto X$, or short: IFS $\{X; f_1, \dots, f_N\}$, where (X, d) is a complete metric space with X being a set and $d : X \times X \mapsto \mathbb{R}$ a distance function. If and only if there exists a so-called contractivity-factor $s_i \in [0, 1)$ with

$$d(f_i(x), f_i(y)) \leq s_i \cdot d(x, y) \quad \forall x, y \in X$$

the functions f_i are called contractive. Then a complete metric space $(\mathcal{H}(X), h)$ based on (X, d) is defined, where $\mathcal{H}(X)$ contains all nonempty compact subsets of X and $h(X)$ denotes the *Hausdorff metric* (which describes the 'similarity' of two sets), given by

$$h(A, B) = \max(\max_{x \in A} \min_{y \in B} d(x, y), \max_{y \in B} \min_{x \in A} d(x, y)).$$

In our work set X (and for that reason also $\mathcal{H}(X)$) contains 3-dimensional objects. The functions $f_i : X \mapsto X$ can easily be extended to functions $f_i : \mathcal{H}(X) \mapsto \mathcal{H}(X)$ by

$$f_i(A) = \{f_i(x) \mid x \in A\}, \quad A \in \mathcal{H}(X)$$

These (of course also contractive) functions can be used to define a function $F : \mathcal{H}(X) \mapsto \mathcal{H}(X)$ with

$$F(A) = \bigcup_{i=1}^N f_i(A), \quad A \in \mathcal{H}(X),$$

being the so-called *Hutchinson Operator*.

The contractivity-factor s of the Hutchinson Operator is given by $s = \max\{s_1, \dots, s_N\}$. Due to the contractivity there exists a unique set \tilde{A} with $\tilde{A} = F(\tilde{A})$. \tilde{A} is called the attractor of the IFS and it satisfies $\tilde{A} = \lim_{n \rightarrow \infty} F^n(A)$ for all $A \in \mathcal{H}(X)$.

An IFS can thus be interpreted as a compact definition for a very complex (often fractal) object [1]. The compact description of complex objects with IFSs is exploited in image compression techniques which aim at solving the so-called *inverse problem*. Given a 2D raster image functions f_i are determined so that the limit set \tilde{A} of the IFS $\{\mathbb{R}^2; f_1, \dots, f_n\}$ is a close approximation of the original raster image. Figure 1 shows an example of a (2-dimensional) IFS, i. e., the well known Sierpinski triangle.

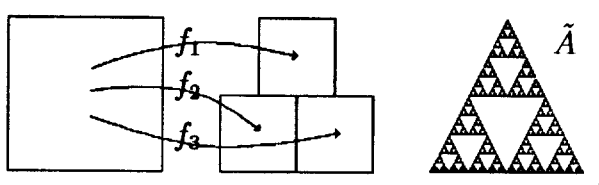


figure 1: Sierpinski triangle as limit set \tilde{A} of the IFS $\{\mathbb{R}^2; f_1, f_2, f_3\}$

1.2 Nonlinear Functions in IFS

As shown above, IFSs are based on *contractive* functions f_i . Although linear functions are usually taken to specify IFSs, nonlinear functions might be used as well, as long as the contractivity condition holds. Now let's make the step from *linear* to *nonlinear* functions. Such IFSs have already been examined. A well known example are Julia sets in the complex plane, which rely upon two nonlinear functions $f_1 = +\sqrt{z-c}$ and $f_2 = -\sqrt{z-c}$. Julia sets are also a good example for the difficulty in predicting the result, when nonlinear functions are involved.

Linear (or affine) functions behave in a more predictable way, because they can be interpreted as combinations of translation, scaling, rotation and reflection. The resulting attractor of the IFS can thus be foreseen by intuition. What we are searching for is a better representation for nonlinear functions so that the attractor can be predicted more easily [10].

1.3 Modeling Functions Using Grid Distortions

One possible way to represent a wide class of nonlinear functions is *grid-distortion*, where a nonlinear function $f(x, y, z)$ is defined by specifying how f transforms a 3-dimensional rectilinear grid. The advantage of this approach is easy design and modeling of these functions.

Every mesh-point $A_{i,j,k}$ is assigned a transformed point $\hat{A}_{i,j,k}$ where $1 \leq i \leq N_i$, $1 \leq j \leq N_j$ and $1 \leq k \leq N_k$, with $N_i, N_j, N_k \in \mathbb{N} < \infty$. Interior points B are transformed by a trilinear interpolation specified by its adjacent gridpoints.

Trilinear interpolation of interior points B is simple and fast. One can think of more elaborate and more costly interpolation schemes like spline interpolation. We, however, believe that in the context of the work presented here more complex interpolation techniques are not worth the extra cost they produce. Extrapolation of exterior points is not needed, because the deformed grid is assumed to lie within the initial grid to guarantee contractivity.

Figure 3 shows an example of a nonlinear IFS. One of the three defining functions of the Sierpinski triangle has been replaced by a nonlinear function which was defined by grid distortion.

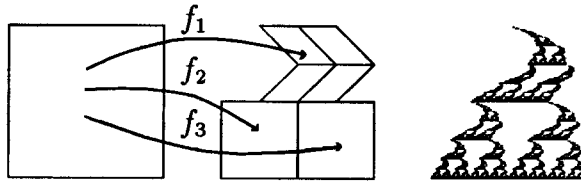


figure 3: IFS $\{\mathbb{R}^2; f_1, f_2, f_3\}$, f_1 nonlinear

Despite the use of nonlinear functions, the result stays predictable — so we achieved the higher capability of nonlinear functions without losing the advantages of their linear counterparts.

Deforming a grid by moving single grid points $A_{i,j,k}$ is quite a tedious work, so we investigate *high-level* operands for distorting rectangular grids.

2 Interactive Modeling of Nonlinear Functions

2.1 Affine Transformation

Affine transformations (*translation, rotation, scaling*) can be specified through grid distortions as well, although this may not be the most efficient way of implementing linear transformations. Translation, rotation and uniform scaling preserve the shape of an object. The following functions are characterized by shape deforming properties.

2.2 Tapering

Let x, y and z be the coordinates of a 3-dimensional gridpoint and \hat{x}, \hat{y} and \hat{z} denote the distorted ones. A transformation can be written as

$$\hat{x} = T_x(x), \quad \hat{y} = T_y(y), \quad \hat{z} = T_z(z)$$

Tapering [2] describes non-uniform scaling where the scaling factor varies along a user-defined axis. The following formulas describe tapering along the z -axis:

$$\hat{x} = f(z) \cdot x, \quad \hat{y} = f(z) \cdot y, \quad \hat{z} = z$$

where $f(z)$ is a linear or nonlinear function. Figure 5 gives an example of tapering.

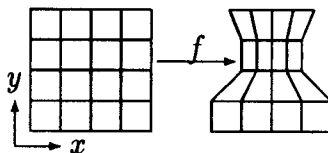


figure 5: tapering (2D example)

2.3 Twisting

Tapering can be considered as scaling along an axis, whereas *twisting* [2] is specified as a nonuniform rotation, where the rotation angle varies along a user defined axis. The following formulas describe twisting along the z -axis:

$$\hat{x} = x \cdot \cos \varphi(z) - y \cdot \sin \varphi(z), \quad \hat{y} = x \cdot \sin \varphi(z) + y \cdot \cos \varphi(z), \quad \hat{z} = z$$

$\varphi(z)$ may be any function, but most applications define the twisting-function by $\varphi(z) = k \cdot z$ (k represents the 'strength' of the twist). In our program system we also implemented the type $\varphi(z) = k \cdot \sin(z)$.

The example shows a twist of the type $\varphi(z) = k \cdot z$ applied to a $6 \times 2 \times 2$ grid:

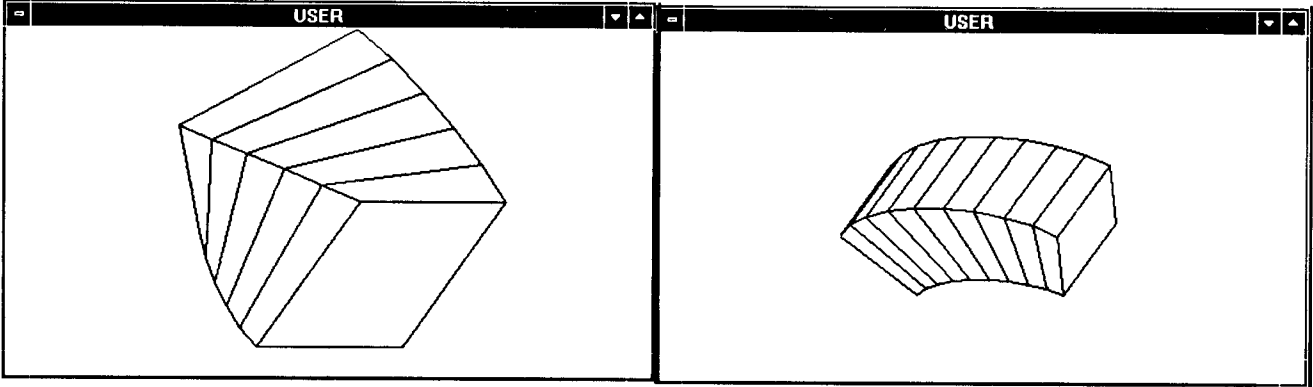


figure 6: grid distortion with a twist operation figure 7: grid distortion with a bend operation

2.4 Bending

The third operation described in [2] is a *bend*-operation along an axis as a composite transformation consisting of a bent region and a region outside the bent region where the transformation is given by a rotation and a translation. In our system the whole grid is distorted by a bend operation. This transformation is given by

$$\hat{x} = x, \quad \hat{y} = -\sin \varphi\left(z - \frac{1}{k}\right) + y_0, \quad \hat{z} = \cos \varphi\left(z - \frac{1}{k}\right) + \frac{1}{k}$$

with $\frac{1}{k}$ specifying the radius of curvature of the bend and $y = y_0$ denoting the centre of transformation. An example is given in figure 7.

2.5 Attracting Point (Magnet)

Another possibility for a high level deformation is given by a kind of mathematical magnet. Let $M(x_m, y_m, z_m)$ be the coordinates of the magnet (which can also lie outside the grid). In proportion to the distance, point M has an attracting (or repulsive) influence on each gridpoint. The maximum range of influence is given by d_{\max} , the strength is determined by s and weakening (of influence) by distance is represented by p . The influence on a point $P(x, y, z)$ is given by

$$f = s \cdot \left(\frac{d_{\max} - d}{d_{\max}} \right)^p$$

where d denotes the distance from P to M .

The distortion is calculated by

$$\hat{x} = x + f \cdot (x_m - x), \quad \hat{y} = y + f \cdot (y_m - y), \quad \hat{z} = z + f \cdot (z_m - z)$$

Figure 8 shows a 2-dimensional example:

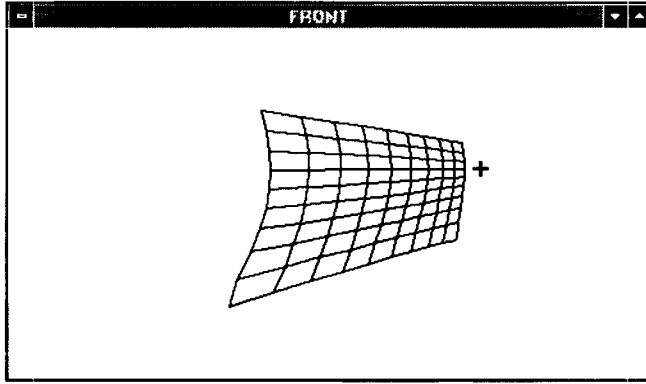


figure 8: magnet (2D example)

2.6 Point Morphing

Morphing, which is a rather popular image processing technique, specifies a geometry based transformation between two images. There are three common methods for defining a morphing function: *line-*, *net-* and *pointmorphing*.

Line-morphing [3] is defined by n corresponding lines in the initial and final picture. For a point X in the final picture its relative position to each line (given by the normal-distance and the proportion of the line in the perpendicular) is calculated. \bar{X} , denoting the corresponding point of X in the initial picture, should have the same relative position to the corresponding line in the initial picture. Weighting over n lines defines the morphing-function.

Net-morphing [11] is based on two similar grids, one laid over the initial image, the other one over the final image. For each intermediate image an interpolated grid is calculated and interior points are bilinearly interpolated.

We integrated point morphing [8] [6] into our system, which requires n corresponding points in the initial and final picture. The corresponding point-pairs can be interpreted as vectors.

Let P_i with $1 \leq i \leq n$ denote the end points of n vectors and $R_i(d_i)$ describe weighting-functions (depending on the distance to the endpoint of the vector i) [7]. P is a point that has to be transformed. Every Point P_i has influence on P , so the mapping function f can be described by a summation over weighted terms, namely

$$f(P) = G(P) + \sum_{i=1}^n \Phi_i(d_i) \cdot P_i$$

After splitting $\Phi_i(d_i)$ into two terms α_i and $R_i(d_i)$ (a so-called radial basis-function) and setting $G(P) = P$ and $R_i(d_i) = 1/(d_i^2 + r_i^2)$ a set of n equations can be solved for each coordinate x , y and z which leads to solutions for α_i^x , α_i^y and α_i^z . d_i denotes the distance from P to P_i and r_i denotes the distance from P_i to the nearest point P_j .

With these results a point $P(x, y, z)$ can be transformed by

$$\hat{x} = x + \sum_{i=1}^n \frac{\alpha_{ix}}{d_i^2 + r_i^2}, \quad \hat{y} = y + \sum_{i=1}^n \frac{\alpha_{iy}}{d_i^2 + r_i^2}, \quad \hat{z} = z + \sum_{i=1}^n \frac{\alpha_{iz}}{d_i^2 + r_i^2}$$

Figure 9 shows a 2-dimensional example before morphing where three pairs of corresponding points are specified and figure 10 illustrates the result of this morphing function.

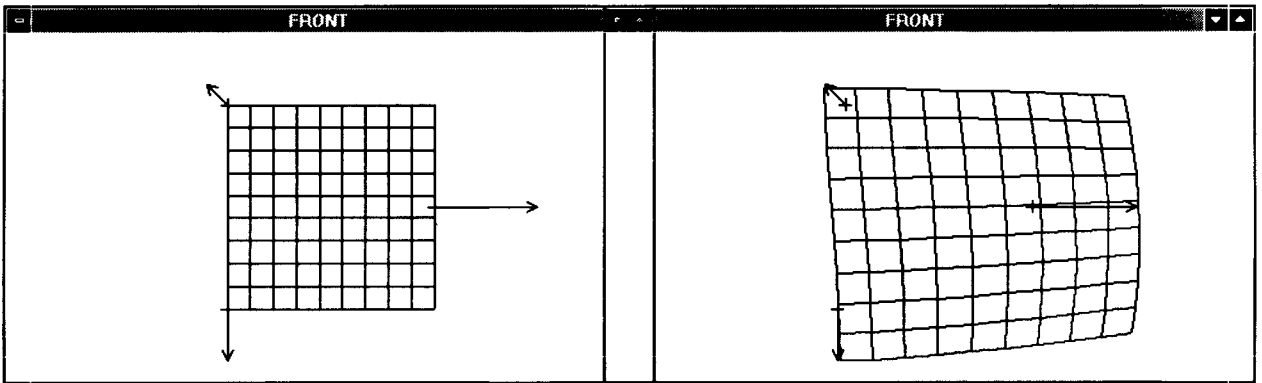


figure 9: grid before morphing

figure 10: same grid after morphing

3 Rendering of nonlinear IFS

The attractor of a linear IFS can be visualized by two types of algorithms. Both types, the *stochastic algorithm* and the *deterministic algorithm*, can be adapted to non linear IFSs as well.

3.1 Stochastic algorithm

The stochastic algorithm is a very quick and easy to implement way to render the attractor of linear or non linear IFSs. Starting with an arbitrary point P one of the functions $f_1 \dots f_n$ is selected at random and applied to point P . Repeatingly selecting a function and transforming the previously calculated point $P_j = f_i(P_{j-1})$ yields a sequence of points that, due to the chaotic dynamics of the process, closely approximates the attractor of the IFS (with a very high probability). Depending on the choice of starting point P the first few calculated points may not yet be close to the attractor – they should therefore be eliminated.

In pseudo-code the stochastic algorithm can be written as shown in figure 11.

```

P0 = P
FOR i=1 TO NMAX DO
BEGIN
  Pi = { f1(Pi-1) with prob. p1
        :
        fn(Pi-1) with prob. pn
  DRAW Pn
END

```

figure 11: stochastic algorithm for IFS $\{X; f_1, \dots, f_n\}$

When assigning probabilities p_i to the functions f_i according to their contractivities of f_i (higher contractivity means lower probability), the point set generated by the stochastic algorithm covers the attractor in a more uniform way. As nonlinear functions may have locally varying contractivity, determining these probabilities is not as easy as for linear functions. In our software system all probabilities p_i are set to equal values $1/n$.

The resulting points of the stochastic algorithm are projected onto a 2D viewing plane preserving depth information. This information is used for a z-buffer algorithm, which determines visibility of a point, and is used to set the color of a point as well. Thereby the color values of points farther away from the viewer are attenuated according to distance. The resulting images of the stochastic algorithm give a very fast, but rather low quality impression of the attractor of an IFS.

3.2 Deterministic algorithm

The second algorithm for approximating an IFS is an immediate result of the fact that every object A will approximate the attractor \tilde{A} of an IFS under iteration of function F [5]. Therefor the deterministic algorithm approximates the attractor as follows:

$$\tilde{A} \approx F^{nmax}(A)$$

where $nmax$ determines the level of accuracy of the approximation.

A software for ray tracing a nonlinear fractal object was implemented in [9] and described in [5], so our software contains an export filter to that program, which provides lightning and shadow casting. Rendering an image by ray tracing gives a high quality result, but calculation times are rather long.

4 The Program InterIFS – Software Description

4.1 General

The program InterIFS was implemented in TURBO C++ and is intended for the WINDOWS 3.1 environment. It has been written on a PC with 486/50 main CPU and 8 Mbyte of RAM. Implementing the software on other PC has shown that 4 Mbyte RAM is sufficient. Program speed depends not only on the CPU than rather on a good graphics card!

4.2 User Interface

The screen is made up by a so-called MDI (*Multi Document Interface*) window. The child windows can be divided into three groups: one group consists of the viewing windows of the grids, one group consists of the views of the IFS and the third group consists only of one window, namely the rendered result. All these child windows can be positioned and scaled individually and they can be shrunk to a symbol. The grids and the system are shown from a top view, a side view, a front view and a user defined view. Each window can be zoomed. The two user defined views can also be rotated. Rotating the IFS user view affects also the result window. All functions can be activated by a menu, the most important of them also by symbols in a tool-bar. At the bottom of the window there is a status line, which contains context dependent information.

4.3 Definition of nonlinear functions in InterIFS

All functions mentioned in chapter 2 have been implemented in our software system. These transformations can be applied to the whole grid or to parts of it (so-called selections). Included affine transformations are scaling, translation and rotation. Applying an affine transformation only to a part of the grid usually leads to a nonaffine transformation.

Tapering, twisting and bending is implemented only with respect to the x , y and z -axis. A user-defined axis can be used by first rotating the grid until the desired axis

coincides with the x , y or z axis. These transformations can be applied by a linear or by a sinusoidal function. Furthermore a ‘ping-pong’ effect can be generated (see also chapter 2.3 and figure 13).

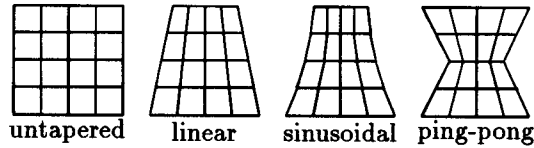


figure 13: possible variations of tapering operations

When the menu item ‘magnet’ is selected a control panel with three scroll bars (distance, strength and decrease) appears which can be used to interactively specify the parameters of the magnet (see section 2.5). The coordinates of the magnet are given by the center of transformation, which can be selected in different ways (center of grid, center of selection or interactive specification).

Point-morphing is specified by up to eight vectors which are given by entering either start- and end-coordinates or start-coordinates and direction. Strength of morphing can be defined by another control-panel.

One other very helpful tool for specifying grid transformations is what we call ‘a universal calculator’. It can be used to enter a mapping function for a grid-point with the coordinates (x, y, z) . Mathematical expressions like \sin , \cos and \tan can be used as well as their negations and hyperbolic counterparts. Also logarithm- and random-functions and constants like π are provided. Furthermore there exist some variables, like x , y and z for the point-coordinates or cx , cy and cz for the center of transformation. Three variables v_1 , v_2 and v_3 and the corresponding ranges can be defined and used to interactively change the equations of the calculator by moving three scroll-bars in a control-panel.

The example in figure 14 was transformed by

$$\hat{x} = x \cdot \cos(v_1 \cdot d) - y \cdot \sin(v_1 \cdot d), \quad \hat{y} = x \cdot \sin(v_1 \cdot d) + y \cdot \cos(v_1 \cdot d), \quad \hat{z} = z$$

where $d = \sqrt{x^2 + y^2}$ and v_1 was set to 0.2.

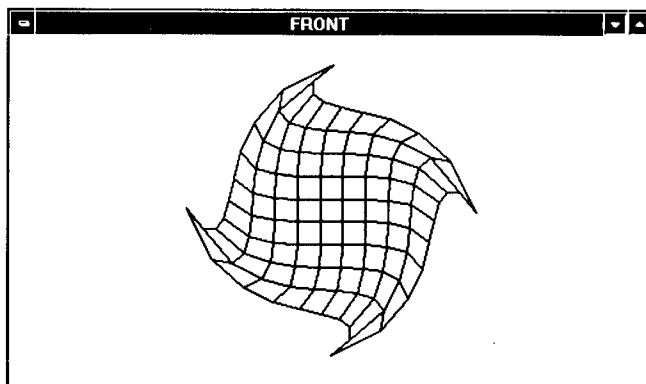


figure 14: grid distorted by universal calculator control

4.4 Selection of Gridpoints

A transformation may be applied to the whole grid of a function or to a subset of the defining gridpoints. Simple subsets are specified by A_{i_1, j_1, k_1} and A_{i_2, j_2, k_2} . All following transformations are then only applied to grid-points $A_{i, j, k}$ with indices $i_1 < i < i_2$,

$j_1 < j < j_2$ and $k_1 < k < k_2$. Simple subsets as mentioned above can be combined by boolean operations (like *and*, *or*, *xor*).

As long as a subset is active, all of the following operations are applied only to the selected gridpoints.

4.5 Combination of grid distortions to specify an IFS

InterIFS handles grids which can have different numbers of gridpoints (N_i, N_j, N_k). An IFS may consist of functions whose underlying grids do contain largely varying numbers of gridpoints. A grid with a high number of gridpoints allows more accurate modeling but increases the cost of function application.

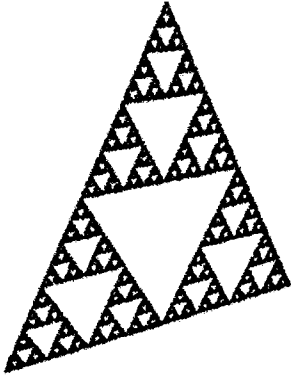
Several nonlinear functions are combined to define a nonlinear IFS. The combination is done by positioning each grid within a common initial grid. The process of positioning requires affine transformations like scaling, rotation, translation. These transformations are again defined interactively.

5 Results

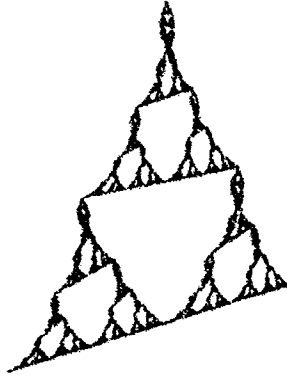
Several plates demonstrate the varied shapes which can be modeled with a nonlinear IFS. Each of these IFSs consists of up to 5 functions which are defined by grid distortions.

References

- [1] Barnsley, M., *Fractals Everywhere*, Academic Press, 1988.
- [2] Barr, A. H., *Global and local deformations of solid primitives*, Computer Graphics, 18 (3), 1984, pp. 21–30.
- [3] Beier, T. and Neely, S., *Feature-based image metamorphosis*, ACM Computer Graphics, 26 (2), 1992, pp. 35–42.
- [4] Glassner, A., *An Introduction to Ray Tracing*, Academic Press Ltd., 1989.
- [5] Gröller, E., *Modeling and Rendering of Nonlinear Iterated Function Systems*, Computers&Graphics, 18 (5), 1994.
- [6] Gröller, E., *Interactive Transformation of 2D Vector Data*, Proceedings of Spring School on Computer Graphics, Comenius University Bratislava, June 6–9, 1994.
- [7] Kunze, M., *Täuschend echt*, c't magazin für computer technik, 11, 1993, pp. 232–238.
- [8] Ruprecht, D. and Müller, H., *Image Warping with Scattered Data Interpolation Methods*, Forschungsbericht 443, Fachbereich Informatik der Universität Dortmund, November 1992.
- [9] Schipfer, J., *Iterierte Funktionensysteme nichtaffiner Funktionen*, diploma thesis, Technical University of Vienna, 1994.
- [10] Wegenkittl, R., *Interactive Design of Nonlinear Functions for Iterated Function Systems*, diploma thesis, Technical University of Vienna, 1994.



linear IFS



twisted Functions



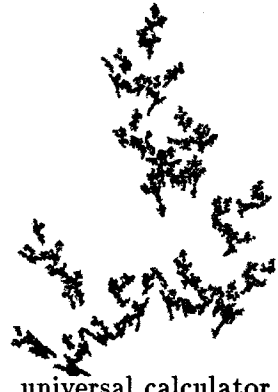
tapered Functions



attracting point



morphed Functions



universal calculator



bended Functions



Functions (combined distortions)