KETpic Matlab Binding for Efficient Handling of Fractal Images

Akemi Gálvez[1], Andres Iglesias[1], Setsuo Takato[2]

[1]*Department of Applied Mathematics and Computational Sciences, University of Cantabria, Avda. de los Castros, s/n, E-39005, Santander, SPAIN*
galveza@unican.es, iglesias@unican.es

[2]*Department of Mathematics, Faculty of Pharmaceutical Sciences, Toho University, Miyama 2-2-1, Funabashi 274-8510, JAPAN*
takato@phar.toho-u.ac.jp

Abstract

Fractals are among the most exciting and intriguing mathematical objects ever discovered. Because of the spectacular development of technology - and particularly computers and computer science - during the last decades, they have become very popular not only in scientific and technological domains but also in mass media related fields such as advertising, computer art, computer design, computer animation and many others. Nowadays there is a wealth of programs and tools to generate fractals by computer. However, none of them is able to yield high-quality graphical output to be readily embedded into standard LaTeX source code in a text-like form. This paper is aimed at filling this gap by introducing a new and freely available KETPic add-on for generating and rendering IFS fractal objects. This new software, developed in the popular scientific program Matlab, generates LaTeX-readable code so that image files are no longer invoked nor required. The resulting files are astonishingly small when compared with their image file counterparts, thus leading to higher compression ratios than other conventional formats such as JPEG, GIF, PNG, EPS and the like. In addition, since IFS fractals are auto-similar they still offer better quality. In fact, a great advantage of this method is that any final image encoded as a collection of IFS becomes resolution independent and hence it can be enlarged at will in a lossless fashion. This paper describes our program and gives some examples to illustrate the excellent performance of our approach.

Keywords: *Fractal geometry, Iterated Function Systems, chaos game, KETpic, Matlab, LaTeX, computer algebra systems, mathematical visualization, scientific artwork.*

1. Introduction

Computer tools for mathematical editing and publishing (such as scientific editors, word processors, graphical editors, mathematical utilities for the web) are ubiquitous in today's technological era. Among them, *LaTeX* has become the standard “de facto” for high-quality typesetting of scientific documents, as it offers remarkable publishing features and extensive facilities for automating most aspects of typesetting and publishing, including numbering, cross-referencing, tables, figures, page layout, table of contents and handling of bibliographic entries. However, dealing with graphics is a different story. Although graphical output is supported, what *LaTeX* actually provides is a basic and very limited set of graphical capabilities to yield drawings. Pictures are generated by combining different simple objects

within the *LaTeX* code, very often a cumbersome, tedious and very time-consuming process. The only available alternative is to use a graphical editor to generate our artwork and then invoke the resulting image file from *LaTeX*. Typically, this option produces a large collection of heavy images files, thus requiring a huge size to store them and preventing users from their transferring on the web. Further, incompatibilities may arise as image files should comply with a limited set of prescribed formats. Although diverse solutions have been proposed so far, there is still a need for computer tools able to yield high-quality graphical output that can readily be embedded into standard *LaTeX* source code in a text-like form.

In 2006, a research group from the "Kisarazu National College of Technology" at Kisarazu (Japan) released a new software for high-quality mathematical drawing in *LaTeX* [27]. The program, called *KETpic* (*K*isarazu *E*ducational *Tpic*), is a library of functions developed on various computer algebra systems (CAS) to generate *LaTeX* source codes for high-quality scientific artwork. Depending on the CAS they rely on, these plug-ins might typically run internally in a quite different way, but this process is usually transparent to end-users. Once *KETpic* is loaded, users are simply requested to execute commands in the CAS of their choice in order to plot graphs and other mathematical data. CAS-embedded *KETpic* commands generate additional *LaTeX* source code and files, which are subsequently compiled in *LaTeX* in the usual manner. As a result, accurate graphical figures can be obtained either on a PC display or on printed matter. So far, versions for *Maple*, *Mathematica*, *Scilab* and *Matlab* have been developed. The corresponding libraries and some interesting examples and documentation are freely downloadable at: http://www.kisarazu.ac.jp/~masa/math/. The interested reader is referred to [10,23,28,29,30,31] for a nice collection of illustrative examples and additional information about this software.

Although smooth curves and surfaces are the most usual graphical objects displayed in scientific documents, it is also interesting to represent graphically irregular mathematical objects, such as fractals [24]. Among them, the *Iterated Function Systems* (IFS) models, popularized by Barnsley in the 1980s, are particularly interesting due to their appealing combination of conceptual simplicity, computational efficiency and great ability to reproduce natural formations and complex phenomena [1]. For instance, the attractors of nonlinear chaotic systems exhibit a fractal structure [7-9,12,13,17,19-21,26]. In the two-dimensional space IFS fractals are made up of the union of several copies of themselves, where each copy is transformed by a function (function system). Such a function is mathematically a 2D affine transformation (see Section 2 for details), so the IFS is defined by a finite number of affine transformations (rotations, translations, and scalings), and therefore represented by a relatively small set of input data [14-16]. This fact has been advantageously used in the field of fractal image compression, an efficient image compression method that uses IFS fractals to store the compressed image as a collection of IFS codes [2,5,22]. The method, based on the idea that in images, certain parts of the image resemble other parts of the same image, has the great advantage that the final image becomes resolution independent. Moreover, this compression method is able to achieve higher compression ratios than conventional methods and still offer better quality.

In this paper we introduce a new, freely available *KETpic* add-on for generating and rendering IFS fractal objects. Our program, developed in the popular scientific program *Matlab*, generates *LaTeX*-readable code so that image files are no longer invoked nor required. The resulting text files are astonishingly small when compared with their image file counterparts, thus leading to higher compression ratios than other conventional formats such as JPEG, GIF, PNG, EPS and the like. These claims will be properly discussed throughout the paper.

The structure of this paper is as follows: in Section 2 some basic definitions and concepts about IFS are given. The chaos game algorithm and the optimal choice of the probabilities required by the algorithm are also briefly discussed in that section. Section 3 describes the software introduced in this paper, including the description of system components, some implementation issues and a typical session workflow. Some illustrative examples showing the good performance of our program are reported in Section 4. The paper closes with the main conclusions and our future work.

2. Iterated Function Systems

In this section we provide our readers with a gentle introduction to some basic definitions and concepts about Iterated function Systems (IFS). The interested reader is referred to [1,4] for a more comprehensive introduction to the field. See also [2,5,11,15,16] for details on fractal image compression with IFS.

3.1 Basic definitions

From the mathematical standpoint, an IFS is a finite set of contractive maps $w_i : X \to X, i = 1,\ldots,n$ defined on a complete metric space (X,d). We refer to the IFS as $W = \{X; w_1,\ldots,w_n\}$. In the two-dimensional case, the metric space (X,d) is typically R^2 with the Euclidean distance d_2, which is a complete metric space, so the affine transformations w_i are of the form:

$$\begin{bmatrix} x^* \\ y^* \end{bmatrix} = w_i \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix} \qquad (1)$$

or equivalently:

$$w_i(x) = A_i . x + B_i$$

where B_i is a translation vector and A_i is a 2×2 matrix with eigenvalues λ_1, λ_2 such that $|\lambda_i| < 1$. In fact, $|det(A_i)| < 1$ meaning that w_i shrinks distances between points. Let us now define a transformation, T, in the compact subsets of X, $H(X)$, by

$$T(A) = \bigcup_{i=1}^{n} w_i(A). \qquad (2)$$

If all the w_i are contractions, T is also a contraction in $H(X)$ with the induced Hausdorff metric [1, 18]. Then, T has a unique fixed point, $|W|$, called the *attractor of the IFS*.

2.2 Chaos game

Let us now consider a set of probabilities $P = \{p_1,\ldots,p_n\}$, with $\sum_{i=1}^{n} p_i = 1$. We refer to $\{W,P\} = \{X; w_1,\ldots,w_N; p_1,\ldots,p_n\}$ as an *IFS with Probabilities* (IFSP). Given P, there exists a unique Borel regular measure $\nu \in M(X)$, called the *invariant measure of the IFSP*, such that

$$\nu(S) = \sum_{i=1}^{n} p_i \nu(w_i^{-1}(S)), \quad S \in B(X),$$

where $B(X)$ denotes the Borel subsets of X. Using the Hutchinson metric on $M(X)$, it is possible to show that M is a contraction with a unique fixed point, $\nu \in M(X)$. Furthermore, $support(\nu) = |W|$. Thus, given an arbitrary initial measure $\nu_0 \in M(X)$ the sequence $\{\nu_k\}_{k=0,1,2,\ldots}$ constructed as $\nu_{k+1} = M(\nu_k)$ converges to the invariant measure of the IFSP. Also, a similar iterative deterministic scheme can be derived from Eq.(2) to obtain $|W|$.

However, there exists a more efficient method, known as *probabilistic algorithm*, for the generation of the attractor of an IFS. This algorithm follows from the result $\overline{\{x_k\}}_{k>0} = |W|$ provided that $x_0 \in |W|$, where (see, for instance, [3]):

$$x_k = w_i(x_{k-1}) \quad with \quad probability \quad p_i > 0. \qquad (3)$$

Picking an initial point, one of the mappings in the set $\{w_1, \ldots, w_n\}$ is chosen at random using the weigths $\{p_1, \ldots, p_n\}$ according to Eq. (3). The selected map is then applied to generate a new point, and the same process is repeated again with the new point obtaining, as a result of this iterative process, a sequence of points. The sequence obtained using this stochastic process converge to the fractal as the number of points increases. This algorithm is known as *probabilistic algorithm* or *chaos game* [1] and generates a sequence of points that are randomly distributed over the fractal, according to the chosen set of probabilities. Thus, the larger the number of iterations (a parameter we can freely set up), the better the resolution of the resulting fractal image. As it will be shown later on, input data of main commands in our program include the number of iterations used to display the final image along with the method applied to generate the sequence of data points.

2.3 Optimal choice of probabilities

The fractal image is determined only by the set of contractive mappings; the set of probabilities gives the efficiency of the rendering process. Thus, a good choice for the probabilities is relevant for the efficiency of the rendering process, since the random sequence of points is generated according to these probabilities. One of the main problems of the chaos game algorithm is that of finding the optimal set of probabilities to render the fractal attractor associated with an IFS. Several different heuristic methods for choosing efficient sets of probabilities have been proposed in the literature [4,6,11]. The most standard of these methods was suggested by Barnsley [1] and has been widely used in the literature. For each of the mappings, this method (called *Barnsley's algorithm*) selectes a probability value that is proportional to the area of the figure associated with the mapping. Since the area filled by a linear mapping w_i is proportional to its contractive factor, s_i, this algorithm proposes to take:

$$p_i = s_i \sum_{j=1}^{n} s_j \quad ; \quad i = 1, \ldots, n. \qquad (4)$$

Another algorithm, proposed in 1996 and known as *multifractal algorithm* [15,16], provides a method for obtaining the most efficient choice for the probabilities as:

$log(p_i) = Dlog(w_i) \Leftrightarrow p_i = w_i^D$, (where D is a real constant) along with $\sum_{i=1}^{n} w_i^D = 1$. Then, the most efficient choice corresponds to

$$p_i = s_i^D; i = 1,\ldots,N \quad (5)$$

where D denotes the *similarity dimension*. This method will be called *optimal algorithm* onwards.

3. The Program

In this section we describe the software presented in this paper. Firstly, the main components of the system are outlined; then, some implementation issues are given; lastly, a typical session workflow is briefly described.

3.1 System components

Before getting started two basic components need to be installed:

1. the scientific program *Matlab* (version 5 or later), which supports many different platforms, such as PCs (with Windows 9x, 2000, NT, Me, XP and Vista) and UNIX workstations. A version for Apple Macintosh with Mac OS X system is also available under X11 (the implementation of the X Window System that makes it possible to run X11-based applications in Mac OS X).
2. a *TeX* editor and compiler. You can use the text editor of your choice to generate your *LaTeX* documents. However, some specialized text editors are usually preferred because they provide some interesting features such as syntax highlighting, shortcuts for some usual commands and procedures, menus with frequently used *LaTeX* macros and styles, etc. Regarding the compiler, there are several LaTeX compilers (most of them freeware) available for Windows, Mac, UNIX and Linux.

Once these programs are properly installed and configured, the only task you have to do is to copy our *KETpic* files into your workspace folder. Note that some additional programs might be required for specific tasks. For instance, you may need a Postscript viewer in case you include EPS graphics in your documents. However, these optional features are related to *LaTeX* rather than to *KETpic* itself and hence out of the scope of this paper so they will be omitted here.

3.2 Implementation Issues

Regarding the implementation, our software has been developed in *Matlab* [25] v2008b on Windows XP operating system by using a PC with Intel Core 2 Duo processor at 2.4 GHz. and 2 GB of RAM. However, the program supports many different platforms, such as PCs (with Windows 9x, 2000, NT, Me, XP and Vista), Linux, Mac OS X and UNIX workstations. The numerical functions have been implemented in the native *Matlab* programming language, while the graphical output is on top of the built-in low-level *Open GL* functions.

3.3 Session Workflow

This section describes a typical *KETpic* session workflow by means of a simple yet illustrative example. *KETpic* processing pipeline starts up by opening *Matlab* for a new session. Then, we load the add-on:

Figure 1. Different schemes for the determination of probabilities for Barnsley's fern: (left) random algorithm; (middle) Barnsley algorithm; (right) optimal algorithm

Table 1: IFS code of Barnsley's fern

	a	b	c	d	e	f
w_1	0.81	0.07	-0.4	0.84	0.12	0.195
w_2	0.18	-0.25	0.27	0.23	0.12	0.02
w_3	0.19	0.275	0.238	-0.14	0.16	0.12
w_4	0.0235	0.087	0.045	0.1666	0.11	0

```
>> Ketinit
```

so that all new *KETpic* commands are automatically available from the very beginning. The main command, **`Plotifs`**, returns the graphical data associated with the IFS fractal. Its syntax is as follows:

`Plotifs`(*sys, init, np, tr,opt1,opt2,..., optn*)

sys: the system of iterated functions representing the fractal, expressed as its IFS code, formatted according to Eq. (1),

init: the initial point, x_0, for the iterated sequence (see Eq. (3)),
np: total number of iterations,
tr: the trasient (the number of initial iterations that are not displayed in order to skip points that do not really belong to the attractor and might otherwise be displayed before convergence)
opts: optional parameters; they account for options that either are not strictly required or have a default value (see below for details).

Next input generates the classical *Barnsley's fern*, represented here by variable **`fern`** and whose IFS code is given in Table 1:

```
>> G1=Plotifs(fern,[1,1],7000,10,'Method','Random');
```

Table 2: (left) Generation of a *Tpic* file; (right) *LaTeX* code of Figure 1

`1 Openfile('Fern.tex');`
`2 Beginpicture('3cm');`
`3 Generateifs(G1);`
`4 Endpicture(1);`
`5 Closefile();`
	`\documentclass[11pt]{article}`
	`\newlength{\Width}`
	`\newlength{\Height}`
	`\newlength{\Depth}`
	`\begin{document}`
	`\input{Fern}`
	`\end{document}`

This fractal is displayed in a new *Matlab* graphical window. The final semicolon precludes graphical data to be printed in the command window. It is however stored in variable **G1** for further use. *KETpic* command **Setwindow** sets up canvas dimension for *LaTeX* drawing: the fractal will be plotted on the prescribed area *[0,1]x[0,1]*:

```
>> Setwindow([0,1],[0,1]);
```

In order to make this output data available in *LaTeX*, it must be converted into a *LaTeX*-readable format (*Tpic* in our case) and then stored into a file. Table 2(left) summarizes this process. Line 1 opens a *TeX* file called **Fern.tex** in the folder indicated in the namepath. Second line defines the units of length for the final picture (with default value 1 centimeter when empty). Command **Beginpicture** is also used to create the **\begin{picture} ...\end{picture}** environment in *LaTeX*. Command **Generateifs** in line 3 converts 2D data points into a sequence of *Tpic* commands to be inserted into the picture environment created in line 2 for standard compilation. Command **Endpicture()** performs two different actions: on one hand, it closes the **picture** environment in the *TeX* file. On the other hand, it allows us to set up the display of cartesian axes, according to its value: 1 (empty value is also feasible) if axes are to be displayed and 0 otherwise. Finally, Line 5 closes the file.

The final output of this process is a file called **Fern.tex** in our workspace folder containing a description of the graphical objects created in *Matlab* in terms of *LaTeX* and *Tpic* commands. The file can be embedded into a standard *LaTeX* file for compilation. Code in Table 2(right) will yield a printout of Barnsley's fern fractal. It is the typical *LaTeX* code with a **documentclass** declaration and the **document** environment. The only difference is given by the three lines in the preamble (lines in-between the start of the file and the **\begin{document}** command) that specify new directives for the length units, and the **\input** command in the main body of source code that causes the indicated file to be read and processed, exactly as if its contents had been inserted in the current file at that point. Compilation of the code above generates Figure 1(left) as a DVI file.

4. Illustrative Examples

As discussed in Section 2.3, the efficiency of fractal rendering is related to the choice of probabilities. **Plotifs** command allows us to specify the method used to determine such probabilities. Figure 1 shows, from left to right, three standard choices for Barnsley's fern (all with 7000 points) selected by setting the option **'Method'** to **'Random'**, **'Barnsley'**

(given by Eq. (4)) and **'Optimal'** (i.e. Eq. (5)) respectively. From a simple visual inspection of this figure, it becomes clear that efficiency improves dramatically in later cases.

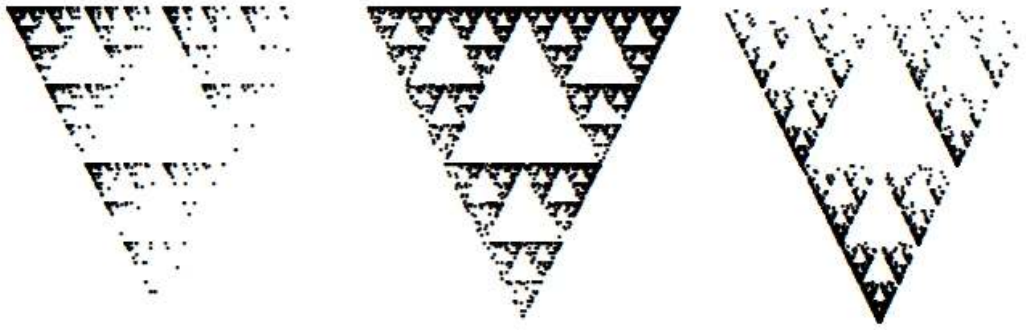

Figure 2. Applying the chaos game algorithm to Sierpinsky's gasket with different probability sets for the three iterated mappings: (left) (0.8,0.1,0.1); (middle) (0.3,0.5,0.2); (right) (0.2,0.1,0.7)

Table 3: IFS code of Sierpinsky's gasket

	a	b	c	d	e	f
w_1	0.5	0	0	0.5	-1	1
w_2	0.5	0	0	0.5	1	1
w_3	0.5	0	0	0.5	0	-1

In fact, the multifractal algorithm provides the best efficiency rate and rendering quality of the fractal attractor (see [15] for further details).

In addition, **Plotifs** command provides users with a method to enter their own choice of probabilities, by setting the option **'Method'** to **'User'**. In this case, a new option, **'Probabilities'**, is activated and the user can input the probabilities of his/her choice. Figure 2 shows three different choices of probabilities for Sierpinsky's gasket, whose IFS code is given in Table 3. The corresponding input is given by:

```
>> lprob=[0.8,0.1,0.1; 0.3,0.5,0.2;0.2,0.1,0.7];
>> for i=1:size(lprob,1)
 S(:,:,i)=Plotifs(sierpinsky,[1,1],5000,10,'Method','User',...
            'Probabilities',lprob(i,:));
 end
```

The corresponding output of previous code, depicted in Fig. 2, shows the enormous visual difference among the fractal images associated with three different sets of probabilities for the same number of iterations in all cases (5000 iterations in this example). Differences among the rendering processes are due to the different distributions of points generated by the

different sets of probabilities over the fractal attractor. For example, with the first choice of probabilities $p_1 = 0.8$, $p_2 = 0.1, p_3 = 0.1$ that corresponds to the left image in Figure 2, we are overestimating the rate of points associated with the left-most mapping, since in this case

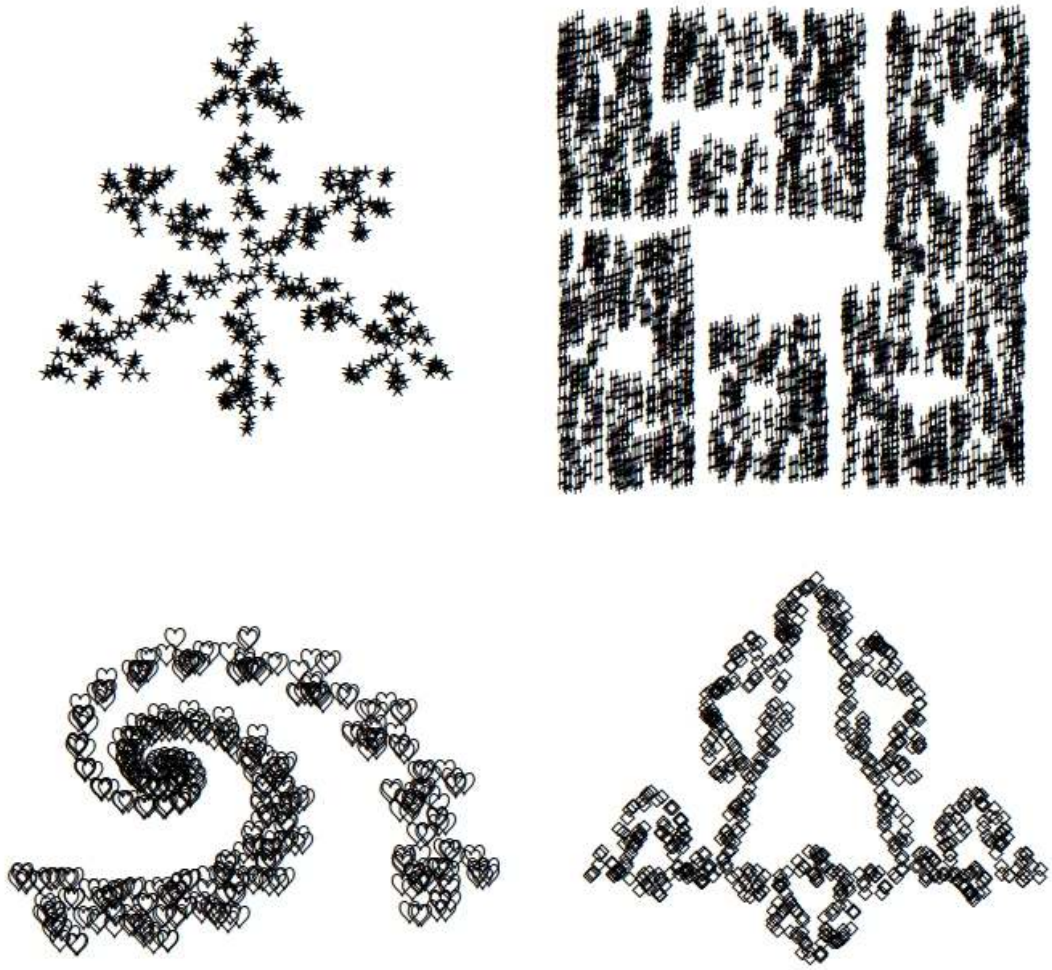

Figure 3. IFS fractal images obtained by iterating different LaTeX symbols and operators (left-right, top-bottom): crystal fractal (\star symbol), blocks fractal (\sharp symbol), spiral fractal (\heartsuit symbol), Koch curve (\diamon symbol)

all the mappings fill equivalent regions of the fractal (so the probabilities should also be the same for all mappings). As a consequence, the points are not uniformly distributed in the resulting fractal image. Similar effects can be seen in the other two examples of this figure, where the right-most and bottom-most mappings are overestimated, respectively.

Generateifs command provides us with some other interesting options, such as **'Symbol'**, which admits all feasible *LaTeX* commands for symbols and operators. Furthermore, such symbols are written in standard *LaTeX* syntax, thus minimizing the time required to get accustomed to such input. Figure 3 shows four examples of IFS fractal images obtained by iterating different *LaTeX* symbols and operators (from left to right, top to bottom): the **\star** operator for the crystal fractal, **\sharp** symbol for the blocks fractal, **\heartsuit** symbol for the spiral fractal and **\diamon** operator for the curve of Koch.

As the reader can see, beautiful pictures can readily be obtained by applying the fractal generation scheme onto carefully chosen *LaTeX* symbols and operators.

Default value for `'Symbol'` is `'Point'`, which admits at its turn some additional options. For instance, we can set up the size of the point displayed on the screen (option `'Pointsize'`, with default value `0.001`), the filling of the point (option `'Filled'`, with default value `'On'`) and many other options not described here to keep the paper at reasonable length.

5. Conclusions and Future Work

In this paper a new *Matlab*-based *KETpic* add-on for generating and rendering IFS fractal objects has been introduced. Our software generates *LaTeX*-readable code so that image files are no longer invoked nor required. The program allows us to display any two-dimensional IFS fractal in *LaTeX* by using a number of options that are explained throughout the paper. The excellent performance of our software is illustrated by several examples of beautiful two-dimensional IFS fractal objects.

The reported add-on is very easy to work with; no proficiency on the software is actually required as all *KETpic* commands are virtually transparent to end-user. As a result, anyone with a minimal knowledge about *Matlab* can generate sophisticated *LaTeX*-readable IFS graphical output in a very short span with reasonable effort. Further, the resulting files are amazingly small when compared with those in formats such as EPS, JPG, GIF, TIFF, PNG and the like. Moreover, since no compression algorithms are applied, those files preserve the highest level of quality, thus providing the best size-quality ratio. For instance, total size for all source files of this paper is less than 1 MB. Besides, this software is freely available and easy to install, with extremely low demands in terms of computer memory and data storage capacity. All these pleasant features make it a highly advisable choice for graphical printout of IFS fractals at professional level.

Regarding our future work, we plan to extend our software in several different ways: on one hand, we want to include some other interesting features such as the computation and rendering of fixed points and bounding boxes of contractive iterated mappings. On the other hand, we want to explore other ways to generate 2D fractals, such as escape algorithms, L-systems, etc. The implementation of other types of fractals like Julia's and Mandelbrot's sets are also goals for further work. Finally, we plan to extend our program in order to display the three-dimensional case. Fractals in the 3D space are both challenging and more difficult to generate. They are typically constructed by superimposing polyhedra upon themselves recursively, a computational expensive technique in terms of CPU and memory requirements. Stereographic images, that will provide a higher degree of realism and a truly three-dimensional feeling to our fractal artwork, are also part of our future work. This feature is currently available in *KETpic* version for *Maple*, which is based on the cross-eye view technique. We plan to apply the same ideas to our software in near future.

This research has been supported by the Computer Science National Program of the Spanish Ministry of Education and Science, Project Ref. #TIN2006-13615, the University of Cantabria, the Japanese Society for Promotion of Science, Project Ref. KAKENHI #20500818 and Toho University. The paper has been written during a three-months stay of first two authors at Toho University (Funabashi, Japan). They are deeply thankful to last author for his kind hospitality, sincere friendship and great collaboration.

6. References

[1] Barnsley, M.F.: "Fractals Everywhere", Second Edition. *Academic Press* (1993)

[2] Barnsley, M.F. , Hurd, L.P.: "Fractal Image Compression". *AK Peters*, Wellesley, MA. (1993)

[3] Elton, J.H.: "An Ergodic Theorem for Iterated Maps", *Ergodic Theory Dynam. Syst.,* 7 (1987) 481-488

[4] Falconer, K.: "Fractal Geometry: Mathematical Foundations and Applications". *Wiley* (1990)

[5] Fisher, Y.: "Fractal Image Compression: Theory and Applications". *Springer-Verlag* (1995)

[6] Forte, B. Vrscay, E. R.: "Solving the inverse problem for measures using iterated function systems : a new approach". *Adv. Appl. Prob.*, 27, (1995) 800-820

[7] Gálvez, A.: "Numerical-symbolic Matlab program for the analysis of three-dimensional chaotic systems". *Lectures Notes in Computer Science*, 4488 (2007) 211-218

[8] Gálvez, A., Iglesias, A.: "Symbolic/numeric analysis of chaotic synchronization with a CAS". *Future Generation Computer Systems*, 25(5) (2007) 727-733

[9] Gálvez, A.: "Matlab Toolbox and GUI for Analyzing One-dimensional Chaotic Maps", *Computational Science and its Applications-ICCSA'2008.* IEEE Computer Society Press, Los Alamitos, California, USA (2008) 321-330

[10] Gálvez, A., Iglesias, A., Takato: "New Matlab-Based KETpic Plug-In for High-Quality Drawing of Curves", *Computational Science and its Applications-ICCSA'2009*, IEEE Computer Society Press, Los Alamitos, California, USA (2009) 123-131

[11] Graf, S.: "Barnsley's scheme for the fractal encoding of images". *Journal of Complexity*, 8 (1992) 72-78

[12] Gutiérrez, J.M., Iglesias, A., Rodríguez, M.A.: "Logistic Map Driven by Correlated Noise". Second Granada Lectures in Computational Physics. Garrido, P.L., Marro, J. (Eds.) *World Scientific*, Singapore (1993) 358-364

[13] Gutiérrez, J.M., Iglesias, A., Rodríguez, M.A.: "Logistic Map Driven by Dichotomous Noise". *Physical Review E,* 48(4) (1993) 2507-2513

[14] Gutiérrez, J.M., Iglesias, A., Rodrguez, M.A., Rodrguez, V.J.: ``Fractal Image Generation with Iterated Function Systems". In "Mathematics With Vision". In: *Proceedings of the First International Symposium of Mathematica*, V. Keranen and P. Mitic (Eds.), Computational Mechanics Publications (1995) 175-182

[15] Gutiérrez, J.M., Iglesias, A., Rodrguez, M.A.: "A Multifractal Analysis of IFSP Invariant Measures with Application to Fractal Image Generation". *Fractals*, 4(1) (1996) 17-27

[16] Gutiérrez, J.M., Iglesias, A., Rodríguez, M.A., Rodríguez, V.J.: "Efficient Rendering in Fractal Images". *The Mathematica Journal*, 7(1) (1997) 7-14

[17] Gutiérrez, J.M., Iglesias, A.: "A Mathematica package for the analysis and control of chaos in nonlinear systems". *Computers in Physics*, 12(6) (1998) 608-619

[18] Hutchinson, J.: "Fractals and Self-Similarity". *Indiana Univ. Math. Jour.*, 30 (1981) 713-747

[19] Iglesias, A., Gálvez, A.: ``Analyzing the synchronization of chaotic dynamical systems with Mathematica: Part I". *Lectures Notes in Computer Science*, 3482 (2005) 472-481

[20] Iglesias, A., Gálvez, A.: "Analyzing the synchronization of chaotic dynamical systems with Mathematica: Part II". *Lectures Notes in Computer Science*, 3482 (2005) 482-491

[21] Iglesias, A., Gálvez, A.: ``Revisiting some control schemes for chaotic synchronization with Mathematica". *Lectures Notes in Computer Science*, 3516 (2005) 651-658

[22] Jacquin, A.E.: "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations". *IEEE Transactions on Image Processing*, 1(1) (1992) 18-30

[23] Kaneko, M., Izumi, H., Kitahara, K., Abe, T., Fukazawa, K., Sekiguchi, M., Tadokoro, Y., Yamashita, S., Takato, S.: "A Simple Method of the TeX Surface Drawing Suitable for Teaching Materials with the aid of CAS". *Lecture Notes in Computer Science*, 5102 (2008) 35-45

[24] Mandelbrot, B. B.: "The Fractal Geometry of Nature". *W. H. Freeman and Co*. (1982)

[25] The Mathworks Inc: "Using Matlab". Natick, MA (1999)

[26] Peitgen, H. O., Jurgens, H. and Saupe, D.: "Chaos and Fractals. New Frontiers of Science". *Springer-Verlag* (1993)

[27] Sekiguchi, M., Yamashita, S., Takato, S.: "Development of a Maple Macro Package Suitable for Drawing Fine TeX-Pictures". *Lecture Notes in Computer Science*, 4151 (2006) 24-34

[28] Sekiguchi, M., Kaneko, M., Tadokoro, Y., Yamashita, S., Takato, S.:"A New Application of CAS to LaTeX Plottings". *Lecture Notes in Computer Science*, 4488 (2007) 178-185

[29] Tadokoro, Y., Abe, T., Kaneko, M., Sekiguchi, M., Fukazawa, K., Yamashita, S., Takato, S.: "A LaTeX plotting software K-.20em.5exE-.125emTpic and its development". In: *Proceedings of the 12th Asian Technology Conference in Mathematics, ATCM'2007*. National Taiwan Science Education Center, Taipei, Taiwan (2007)

[30] Takato, S., Iglesias, A., Gálvez, A.: "Use of ImplicitPlot in Drawing Surfaces Embedded into Latex Documents". *Computational Science and its Applications-ICCSA'2009*, EEE Computer Society Press, Los Alamitos, California, USA, (2009) 115-122

[31] Yamashita, S., Abe, T., Izumi, H., Kaneko, M., Kitahara, K., Sekiguchi, M., Fukazawa, K., Takato, S.; "Monochrome Line Drawings of 3D Objects due to the Programmability of KETpic". *Computational Science and its Applications-ICCSA'2008*. IEEE Computer Society Press, Los Alamitos, California (2008) 277-283

Authors

AKEMI GALVEZ is a lecturer at the Department of Applied Mathematics and Computational Sciences of the University of Cantabria (Spain). She holds a B.Sc. degree in Chemical Engineering at the National University of Trujillo (Peru), and a M.Sc. and a Ph.D. degrees in Computational Sciences at the University of Cantabria (Spain). She has published several papers in reputed international scientific journals and conferences on geometric processing, surface reconstruction, symbolic computation and artificial intelligence with emphasis in its applications to geometric modeling and computer graphics. She has participated in several national and international projects on geometric modeling and processing and its applications, in particular to the automotive industry, where she has developed various software packages and toolboxes with an extensive portfolio of industrial and academic users. She has also participated in several academic projects aimed at promoting the use of new technologies in order to improve the quality of university system with regards the European Area of Higher Education. She is currently the main developer of the Matlab binding of KETpic. Her fields of interest also include software engineering, information technologies, numerical/symbolic computation and industrial applications.

ANDRES IGLESIAS is currently the head of the Department of Applied Mathematics and Computational Sciences of the University of Cantabria (Spain). Since Nov. 2005, he has also been the Post-graduate studies coordinator at his department, awarded with the Quality Certificate by the Spanish Ministry of Education and Science. He has been a Visiting Researcher at (among others) the Department of Computer Science of the University of Tsukuba (Japan), Wessex Institute of Technology (UK), International Center of Theoretical Physics-ICTP (Italy) and Toho University (Japan). He holds a B.Sc. degree in Mathematics (1992) and a Ph.D. in Applied Mathematics (1995). He has been the chairman and organizer of 30 international conferences in the fields of computer graphics, geometric modeling and symbolic computation, such as the CGGM (2002-09), TSCG (2003-08) and CASA (2003-10) annual conference series and co-chair of ICMS'2006, VRSAL'2008, ICCIT'2008 and CGVR (2009-10). In addition, he has served as a program committee and/or steering committee member of over 100 international conferences such as 3CM, 3IA, ACN, CGA, CAGDAG, CGI, CGIV, CIT, CyberWorlds, FGCN, GMAG, GMAI, GMVAG, Graphicon, GRAPP, ICCS, ICCSA, ICICS, ICCIT, ICM, ICMS, IMS, IRMA, ISVD, MMM, NDCAP, SEPA, SMM, VIP, WSCG and WTCS. He has been reviewer of 99 international conferences, 24 international journals (including 13 ISI-indexed journals) and outstanding research institutions and agencies such as NSF (USA) and the European Commission. He is currently the Editor in Chief of the

"International Journal on Computer Graphics", Associate Editor of the journals "International Journal of Computer Graphics and CAD/CAM", "Transactions on Computational Science", "Advances in Computational Science and Technology", "International Journal of Computational Science", "International Journal of Biometrics", "Journal of Convergence Information Technology", "Int. Journal of Future Generation Communication and Networking" and "International Journal of Digital Content Technology and its Applications" and member of the Editorial Reviewing Board of the "International Journal of Information Technology and Web Engineering". He has also been guest editor of some special issues of international journals about computer graphics and symbolic computation. He is the author of over 110 international papers on different topics and 7 books. For more information, take a look at his personal web site available at: http://personales.unican.es/iglesias

SETSUO TAKATO is currently a Full Professor at the Department of Mathematics of the Faculty of Pharmaceutical Sciences, at Toho University, Funabashi campus (Chiba, Japan). He holds a B.Sc. from the University of Tokyo in 1973, and a M.Sc. from the University of Tsukuba in 1975, two of the top-ranked universities in Japan. For many years, he has been strongly involved in the field of Mathematics Education, where he published several books and lecture notes. Later, he turned his attention to the use of Computer Algebra Systems for Mathematical Education. Based on his motivation to produce high-quality student' notes, he conceived during his time at the Kisarazu National College of Technology (Kisarazu, Japan) a new software system called KETpic for the generation of high-quality scientific artwork to be embedded into LaTeX documents. He has been the main designer and architect of the system since the original release of KETpic binding for Maple in 2005. Subsequently, KETpic has been extended to other symbolic computation systems such as Mathematica, Matlab, Scilab and Axiom. As a consequence, KETpic users base has enlarged significantly and the software is receiving increasing attention from the mathematical community. His fields of interest also include Applied Mathematics, Fundamental Mathematics and applications of computers to Science and Engineering.