Intuitive modeling of vapourish objects

Dmitry Sokolov, Christian Gentil

Intuitive modeling of vapourish objects

Dmitry Sokolov

LORIA - Alice, Campus Scientifique, BP 239
54506 Vandoeuvre-les-Nancy Cedex, France
dmitry.sokolov@loria.fr

Christian Gentil

Laboratoire LE2I — UMR CNRS 5158
Aile de l'Ingénieur, Faculté des Sciences Mirande, Université de Bourgogne
21000 Dijon, France
christian.gentil@u-bourgogne.fr

Abstract

Attempts to model gases in computer graphics started in the late 1970s. Since that time, there have been many approaches developed. In this paper we present a non-physical method allowing to create vapourish objects like clouds or smoky characters. The idea is to create few sketches describing the rough shape of the final vapourish object. These sketches will be used as condensation sets of Iterated Function Systems, providing intuitive control over the object. The advantages of the new method are: simplicity, good control of resulting shapes and ease of eventual object animation.

Keywords: Iterated Function System, Condensation Sets, Morphing, Cloud Modelling

1. Introduction

Modeling natural phenomena such as clouds, steam, fire, smoke is a challenging task since the dawn of computer aided design systems (CAD). Standard CAD modeling approaches were developed for smooth regular shapes and are not well suited to effectively capture the intricate features of vapourish phenomena. Many scientists met the challenge and now a number of powerful methods

exist. We can classify it into two categories: techniques for modeling the geometry of gases and techniques for rendering scenes with gases and atmospheric effects.

The first family uses physics-based approaches, simulating the natural process that gives birth to desired shape. These methods are generally represented by dynamic systems, i.e. systems of equations that describe the evolution of a shape. As an example of such methods one can point at works of Harris et al. [1] and Miyazaki et al. [2]. An animation can also be created by adjusting the parameters of the simulation system. Whereas simulations give excellent, very realistic results, they are computationally expensive and difficult to use for many animators and modelers. Setting correct physics parameters is a time-consuming task.

The second family of methods is based on (non-physical) 3D procedural textures. A good overview can be found in a book by David Ebert [3]. In these approaches the artist is not limited by the laws of physics, however, it can be harder to create convincing images. Textures are generated by pseudo-random noises, defining a "perturbed" 3D shape. Generally a two-level model is used: macrostructure plus microstructure. The idea is to construct a "support shape" (macrostructure) which in large represents the desired form. In the majority of cases "support shapes" are composed of spheres. Procedural modification then alters the density distribution to create the detailed wisps. That is, procedural noise functions (e.g. Perlin's noise, fractal noise, turbulence) create the microstructure to make initial "support shape" irregular. Depending on the noise function used, a number of parameters allow to adjust perturbation characteristics. Furthermore, if necessary, a specific treatment can be applied in order to simulate the nature of desired material (cloud, smoke, fire etc).

There are two ways to create animations with this family of methods. Either the artist deforms the support shape, or he modifies the microstructure. In this case the artist has to modify parameters of a pseudo-random process. The control is very delicate, notably, in order to keep the smoothness of the animation, the transformations are to be continuous. A possible solution is to generate a

noise with an additional dimension corresponding to the time. For example, we can use 4D noise functions to animate a 3D cloud.

Numerous problems arise with this approach. The method perfectly fits when it is necessary to create a (static) cloud, however, it is very difficult to create a vapourish character and, especially, to animate it. The primitives become visible during the deforming the "support shape". Then, it is almost impossible to control subtle wisps on the object. For example, the method does not allow to change directly *local* features of the microstructure, i.e. if an artist dislikes a wisp, he can not erase it without affecting other parts of the object. Pseudo-random nature of the microstructure makes the control very delicate and non intuitive. Artists must be familiar with such mathematical terms as "frequency", "wavelet", "Gaussian kernel" etc. Even mathematicians can fail to construct a 4D microstructure with desired characteristics.

In order to enhance the control, some authors combine the approaches [4]. Here the user disposes an interface allowing to define a "support shape", which is deformed by a simulation process.

We have developed an alternative way to create vapourish objects, giving a preference to shape control over realistic rendering. We aim to propose a very intuitive tool, a tool which can be used by any artist without special training. The goal is to have a tool allowing to create recognizable 3D textures of gaseous objects, resembling a given draft. Figure 1 shows a smoky ship created with the method we propose. Note that the galleon is fully three-dimensional, so it can be rendered from any point of view.

To fit the constraints, we chose a model based on the fractal geometry, namely, the Iterated Function System model (IFS). The model is introduced by Hutchinson and popularized by Barnsley. In this article we generalize the notion of classical IFS, enriching it with condensation functions. This generalization inherits all the properties, established by the IFS model such as continuous dependence on parameters. As we show in the following sections, it guarantees perfect control on shapes and animations, keeping simplicity and intuition over the modeling process.

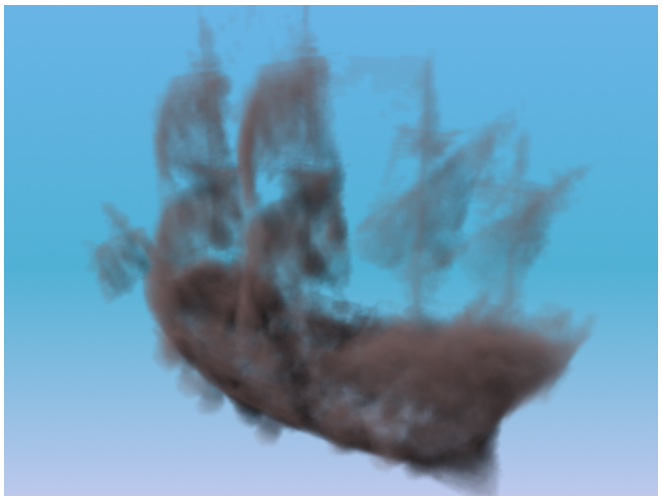

Figure 1: A smoky galleon can be easily created with the new modeling tool.

The paper is organized as follows: section 2 provides the necessary background on iterated function systems. Then, sections 3 and 3.4 describe our generalization of the standard approach. Examples may be found in section 4. Finally, section 5 concludes the paper and provides suggestions for further research.

2. Background

2.1. General definitions

An *iterated function system* or *IFS* consists of a finite set of transformations $f_i : \mathbb{X} \to \mathbb{X}$ for $i = 1, 2, \ldots, N$, where $N \geq 1$ is an integer and $(\mathbb{X}, d)$ is a complete metric space. Let us denote it by $\mathcal{F} = \{\mathbb{X}; f_1, f_2, \ldots f_N\}$.

A transformation $f_i : \mathbb{X} \to \mathbb{X}$ is *strictly contractive* if and only if there exists a number $0 \leq s_i < 1$ such that $d(f_i(x), f_i(y)) \leq s_i d(x, y)$ for all $x, y \in X$. The number s_i is called a contractivity factor for f_i. If an IFS consists of strictly contractive transformations, it is called a *hyperbolic IFS* and the number $s = \max\{s_1, s_2 \ldots s_N\}$ is called a contractivity factor for the IFS.

Let $(\mathbb{X}, d)$ be a complete metric space. Then $\mathbb{H}(\mathbb{X})$ denotes the space whose points are non-empty compact subsets of $\mathbb{X}$. Given a hyperbolic IFS, let us

define a contractive (with respect to the Hausdorff metric $d_{\mathbb{H}}$) transformation $\mathcal{F} : \mathbb{H}(\mathbb{X}) \to \mathbb{H}(\mathbb{X})$ as follows:

$$\mathcal{F}(B) = f_1(B) \cup f_2(B) \cup \cdots \cup f_N(B) \ \textit{for all} \ B \in \mathbb{H}(\mathbb{X}).$$

Note that we use $\mathcal{F}$ to denote either an IFS or a corresponding transformation. The meaning should be clear from context. In the literature the transformation $\mathcal{F}$ is also called the Hutchinson operator.

It is well-known [5] that there exists a unique fixed point $A_{\mathcal{F}} \in \mathbb{H}(\mathbb{X})$ such that $A_{\mathcal{F}} = \mathcal{F}(A_{\mathcal{F}})$. Moreover, the contraction mapping theorem [6] states that for any non-empty compact subset $B_0 \in \mathbb{H}(\mathbb{X})$ the sequence, recurrently defined as $B_{k+1} = \mathcal{F}(B_k)$, converges to $A_{\mathcal{F}}$ with respect to the Hausdorff metric as $k \to \infty$. The resulting set does not depend on the choice of B_0, and for this reason $A_{\mathcal{F}}$ is called the *set attractor of the (hyperbolic) IFS*.

2.2. Deterministic algorithm

It is possible to approximate attractors in a straightforward manner by means of the contraction mapping theorem. This process is called the deterministic algorithm. Let us consider an example, where the IFS consists of three affine transformations on $\mathbb{C}$ (figure 2 shows an illustration):

$$f_1 = \frac{1}{2}z, \quad f_2 = \frac{1}{2}z + \frac{1}{2}, \quad f_3 = \frac{1}{2}z + \frac{1}{2}i.$$

The first step is to choose an arbitrary non-empty compact subset of $\mathbb{C}$. Then the iteration process consists of applying each transformation f_1, f_2 and f_3 to the current set, and then uniting the resulting sets. Then f_1, f_2 and f_3 are to be applied to the union and so on. The limit of this process gives the attractor of the IFS, as shown above.

2.3. Random iteration algorithm

Another approach to approximate IFS attractors is the random iteration algorithm [7]. An *IFS with probabilities* consists of an IFS together with a

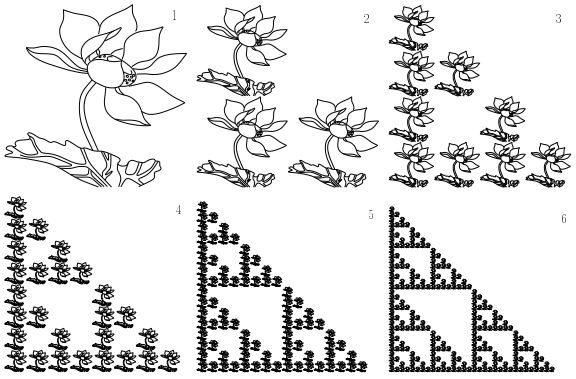

Figure 2: An illustration for the deterministic algorithm. Starting from any non-empty compact subset (the flower, for example), the sequence converges to the Sierpinski's Gasket.

distribution of probabilities $p_1, p_2, \ldots, p_N$, positive real numbers such that $p_1 + p_2 + \cdots + p_N = 1$. An IFS with probabilities may be denoted as

$$\mathcal{F} = \{\mathbb{X}; f_1, f_2, \ldots, f_N; p_1, p_2, \ldots, p_N\}.$$

The probability p_i is associated with the transformation function f_i for each $i \in \{1, 2, \ldots, N\}$. If one uses the plane $\mathbb{C}$ for the iteration space $\mathbb{X}$, then the random iteration algorithm may be expressed in pseudo-code as follows:

1 **Input:** An IFS with probabilities $\mathcal{F}$

2 **Output:** An approximation (a digital image) of the set attractor $A_{\mathcal{F}}$

3 select a random point $z_0 = (x_0, y_0)$ of the plane $\mathbb{C}$

4 iterate {

5 choose a transformation f_i according to probabilities $\{p_1, p_2, \ldots, p_N\}$.

6 $z_{k+1} = f_i(z_k)$

7 if $(k > 100)$ draw pixel with coordinates z_{k+1}

8 }

This schema works because the IFS is contractive: if z_k is close to the attractor $A_{\mathcal{F}}$, then $f_i(z_k)$ is yet closer. Though the algorithm starts out with a random point, the distance between the solution set $A_{\mathcal{F}}$ and the "dancing" point decreases exponentially. In this example we have taken 100 iterations before drawing any point. In general, the distance between $A_{\mathcal{F}}$ and z_{101} is inferior to the size of the pixel. Moreover, the Elton's Ergodic Theorem [6, p. 370] states that, with probability 1, the set z_k is dense in $A_{\mathcal{F}}$. No fixed number of iterations is given for the algorithm. Because of the stochastic nature of the algorithm, more iterations lead to a better approximation of the exact solution. If there are no predefined probabilities, one can choose the transformations with equal frequency in line 5 of the random iteration algorithm.

An example of an IFS with probabilities is $\{\mathbb{C}; f_i; p_i; i = 1, 2, 3, 4\}$, where $p_i = 0.25$ and

$$f_1(z) = \frac{1}{2}z, \qquad f_2(z) = \frac{1}{2}z + \frac{1}{2}$$
$$f_3(z) = \frac{1}{2}z + \frac{1}{2}i, \qquad f_4(z) = \frac{1}{2}z + \frac{1}{2} + \frac{1}{2}i.$$

Then the random iteration algorithm proceeds as follows: an initial point, $z_0 \in \mathbb{C}$, is chosen. One of the transformations is selected "at random" from the set $\{f_1, f_2, f_3, f_4\}$. The selected transformation is applied to z_0 to produce a new point $z_1 \in \mathbb{C}$. Again a transformation is selected, in the same manner, independently of the previous choice, and applied to z_1 to produce a new point z_2. The process is repeated a number of times, resulting in a finite sequence of points $\{z_k : k = 1, 2, \ldots, number\ of\ iterations\}$. Starting from some step, the queue of the sequence lies in the unit square with corners at (0, 0), (1, 0), (1, 1) and (0, 1). If the number of iterations is sufficiently large, a picture of the filled unit square will be the result.

In this example the "dancing" point z_i visits all points of the unit square with equal probabilities. But what will happen if we change the probabilities? Let us consider the above IFS with the different set of probabilities $\{0.30, 0.20, 0.25, 0.25\}$. Let us also change the game rules a bit: instead of just plotting a pixel z_k, we count how many times each pixel was visited by the

"dancing" point z_i. Once the algorithm terminates, we draw a gray-level image with darker pixels in most visited areas.

Amazingly, the resulting image changes completely! Refer to figure 3, where the left image shows the result for the equal probabilities, while the right one shows the result for the new set of probabilities.

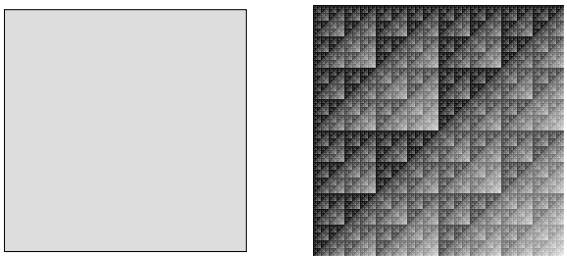

Figure 3: Measure attractors: different sets of probabilities lead to different distributions of "mass" on the set attractor. Left image is computed using equal probabilities $\{0.25, 0.25, 0.25, 0.25\}$, the right one is obtained with the set $\{0.30, 0.20, 0.25, 0.25\}$.

The images suggest a wonderful idea. They suggest that associated with an IFS with probabilities there is a unique "density" on the attractor of the IFS. Barnsley [8, pp. 122–129] has developed the idea of the densities using the measure theory formalism. Measures can be used to describe intricate distribution of "mass" on metric spaces. Starting from an IFS $\{\mathbb{X}; f_i; p_i; i = 1, 2, \ldots, N\}$, Barnsley introduces another IFS $\mathcal{F} = \{\mathbb{P}(\mathbb{X}); f_i; p_i; i = 1, 2, \ldots, N\}$, where $(\mathbb{P}(\mathbb{X}), d_{\mathbb{P}})$ is the space of normalized Borel measures on $\mathbb{X}$ with the Monge-Kantorovich metric on it. He shows how a transformation f being contractive on $\mathbb{X}$ inherits this property on $\mathbb{P}(\mathbb{X})$. Therefore, the induced transformation $\mathcal{F} : \mathbb{P}(\mathbb{X}) \to \mathbb{P}(\mathbb{X})$ has a unique fixed point called a *fractal measure* or a *measure attractor*.

Back to our examples, both IFS have the same *set* attractor (the unit square), but different *measure* attractors.

3. Theoretical framework

In spite of the fact that IFS theory deals with arbitrary complete metric spaces, almost all implementations construct fractals in a 2D plane with affine contractions. Very few approaches use three dimensions or (and) non-linear transformations. For example, one can point at works by Scott Draves [9, 10] in the domain of rendering and special effects.

Here we propose to consider an IFS $\mathcal{F} = \{\mathbb{H}(\mathbb{X}); f_1, f_2 \ldots, f_n\}$, where all operators f_i act on $\mathbb{H}(\mathbb{X})$, the space whose points are non-empty compact subsets of $\mathbb{X}$. It is convenient to visualize the space $\mathbb{X}$ as a white sheet of paper, and the space $\mathbb{H}(\mathbb{X})$ as the set of all possible black ink drawings on the sheet. In order to simplify the presentation, we talk of two-dimensional images, however all the techniques may be applied to arbitrary spaces. Thus, $\mathbb{H}(\mathbb{X})$ is the set of all possible drawings on $\mathbb{X}$.

In our case, induced Hutchinson operator $\mathcal{F}$ maps $\mathbb{H}(\mathbb{H}(\mathbb{X}))$ onto $\mathbb{H}(\mathbb{H}(\mathbb{X}))$. Recall that $\mathbb{H}(\mathbb{X})$ is the space of all possible black ink drawings; then $\mathbb{H}(\mathbb{H}(\mathbb{X}))$ (non-empty compact subsets of $\mathbb{H}(\mathbb{X})$) is the space of all possible collections[1] of drawings. One may visualize the fixed point $A_{\mathcal{F}}$ as a "book" of drawings.

3.1. IFS with condensation

A very particular case of IFS that deals with compact subsets is the IFS with (constant) condensation, introduced by Barnsley in 1988: let $(\mathbb{X}, d)$ be a metric space and let $C \in \mathbb{H}(\mathbb{X})$. Then a constant transformation $f_0 : \mathbb{H}(\mathbb{X}) \to \mathbb{H}(\mathbb{X})$ defined by $f_0(B) = C$ for all $B \in \mathbb{H}(\mathbb{X})$ is called a *condensation transformation* and C is called the *associated condensation set*. A condensation transformation $f_0 : \mathbb{H}(\mathbb{X}) \to \mathbb{H}(\mathbb{X})$ is a contraction mapping on the metric space $(\mathbb{H}(\mathbb{X}), d_{\mathbb{H}})$ with contractivity factor equal to zero. Therefore, it possesses a unique fixed point, namely the condensation set.

Then Barnsley defines a new IFS: let $\{\mathbb{X}; f_1, f_2, \ldots, f_n\}$ be a hyperbolic IFS with contractivity factor $0 \leq s < 1$. Let $f_0 : \mathbb{H}(\mathbb{X}) \to \mathbb{H}(\mathbb{X})$ be a condensation

[1] Uncountable in general.

transformation. Then $\{\mathbb{X}; f_0, f_1, \ldots, f_n\}$ is called a *hyperbolic IFS with condensation*, with contractivity factor s. Note that the IFS rests in the space $\mathbb{X}$. In fact, Barnsley modified the Hutchinson operator only (by adding f_0):

$$\mathcal{F}(B) = C \cup f_1(B) \cup f_2(B) \cup \cdots \cup f_N(B) \ \textit{for all} \ B \in \mathbb{H}(\mathbb{X}).$$

Let us consider an example, where the IFS consists of one two-dimensional contraction $f_1(z) = \frac{1}{2}z$. Clearly, the attractor consists of a single point, namely origin. But if we add a condensation transformation f_0 which transforms any compact subset to a pine tree, then the attractor becomes a series of pine trees, refer to figure 4 for an illustration.

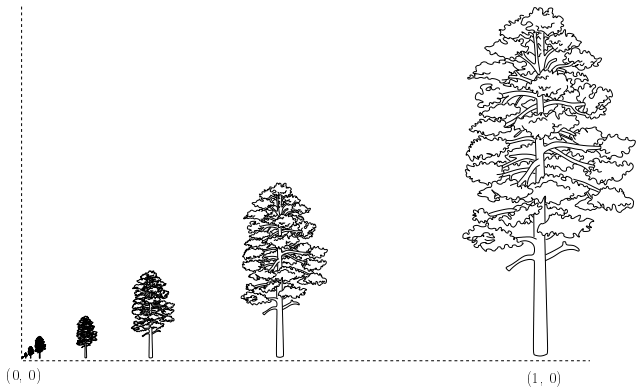

Figure 4: A geometric series of pine trees, the attractor of an IFS with condensation.

3.2. A generalized implementation

We would like to enrich the notion of IFS with condensation. Let us consider IFS with transformations action on the space of compact subsets $\mathbb{H}(\mathbb{X})$. Such IFS consists of contractions on $(\mathbb{H}(\mathbb{X}), d_{\mathbb{H}})$. As before, a contraction f on $(\mathbb{H}(\mathbb{X}), d_{\mathbb{H}})$ is a mapping with unique fixed point. This means that for any non-empty compact subset B the sequence $\{f^k(B)\}$, recurrently defined by $f^0(B) = B$ and $f^{k+1}(B) = f^k(B)$ converges to the fixed point (with respect to the Hausdorff metric). In his approach Barnsley chooses a fixed point C (the pine tree in the case of figure 4). Then, clearly, the corresponding *constant* mapping is a contraction.

The question is: is it possible to find contraction mappings on $\mathbb{H}(\mathbb{X})$ besides obvious constant ones? Refer to figure 5 for an example. Let us suppose that there is a set C given and there is a mapping f such as *any* non-empty compact subset morphes to the given set C under recurrent application of the mapping.

More strictly, the Meyer's converse to Banach's theorem [11] states that if for any $B \in \mathbb{H}(\mathbb{X})$ the sequence $\{f^k(B)\}$, recurrently defined by $f^0(B) = B$ and $f^{k+1}(B) = f^k(B)$, converges to C uniformly on a neighbourhood of C in Hausdorff metric $d_{\mathbb{H}}$, then there is a topologically equivalent metric $d'_{\mathbb{H}}$ on $\mathbb{H}(\mathbb{X})$ that makes f a strict contraction (note that the random iteration algorithm works under a much weaker condition that the whole system is to be contractive on average).

In this section we build an example of such a transformation. The problem of defining contraction mappings on $\mathbb{H}(\mathbb{X})$ is closely related to the image interpolation (binary image morphing) task. Image interpolation is a general term for a set of techniques used in Computer Graphics to generate intermediate pictures between two given images. For example, Iwanowski and Serra [12] propose a morphing method consisting of three steps (figure 6 illustrates the process):

1. First, for two given binary images one detects bounding boxes in order to superpose the images.

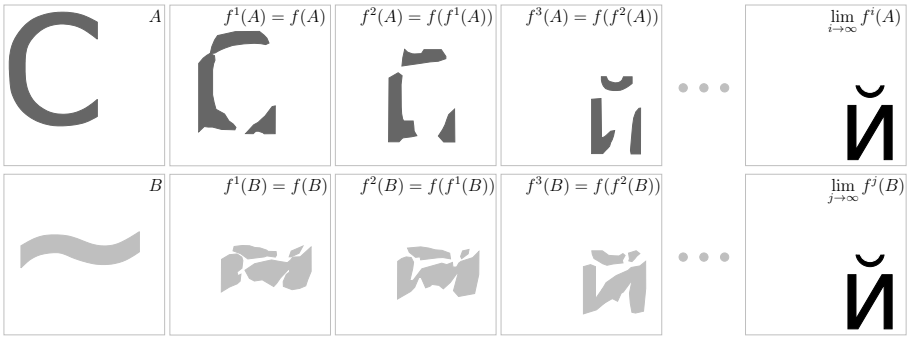

Figure 5: An example of a contraction mapping on $(\mathbb{H}(\mathbb{X}), d_{\mathbb{H}})$. The fixed point (chosen by the user) is shown black in the final image. *Any* non-empty compact subset morphes to the given set by recurrent application of the mapping f.

2. Then the authors calculate morphological median image.

3. Finally, they place the median image to its final position.

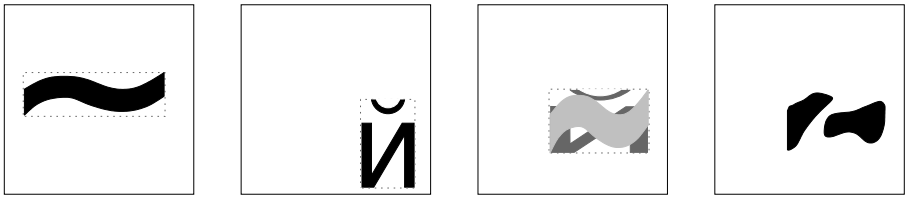

Figure 6: First image represents the source image, the second one is the destination image. The idea is to superpose the input images in the central position (third image) and then to calculate the median image (rightmost one).

The key step is the median set calculation. The authors define the median set $M(A, B)$ between two sets A and B as follows:

$$M(A, B) = \{x \in \mathbb{X} : d(x, A \cap B) < d(x, \overline{A \cup B})\},$$

where d is the Euclidean pseudo-distance of a point to a set. In other words, the median set consists of points which lie more close to the intersection between A and B than to the complement of their union. The interpolation sequence (morphing of the first shape into the second) can be produced by the recurrent generation of new medians.

Unfortunately, this method has a lot of restrictions. First, the union $A \cup B$ must not be equal to $\mathbb{X}$. Next, the intersection $A \cap B$ must not be empty. Moreover, the intersection between all connected components is to be non-empty.

To avoid the limitations, we propose a simple method of binary image morphing. So, we have two images A and B. Let us define a *left* median image $D_A(B)$ as an image drawn by following rules:

1 **Input:** Two binary images A and B

2 **Output:** Left median image $D_A(B)$

3 clear the sheet $D_A(B)$

4 for each black point p_A of A {

5 find corresponding nearest black point p_B in image B

6 draw a point in the image $D_A(B)$ in the center of the segment (p_A, p_B)

7 }

Thus, left median image $D_A(B)$ is the image A deformed by the influence of B. Right median image $D_B(A)$ is defined in the same way. Then final median image is defined as a union of the left and the right median images: $M(A, B) = D_A(B) \cup D_B(A)$. Figure 7 shows an illustration. For convenience reasons all the sets are drawn in the same sheet of paper, but in fact the figure represents four superposed images. The image A is the comma drawn at left of a blank canvas, the image B represents a character at right of another canvas etc.

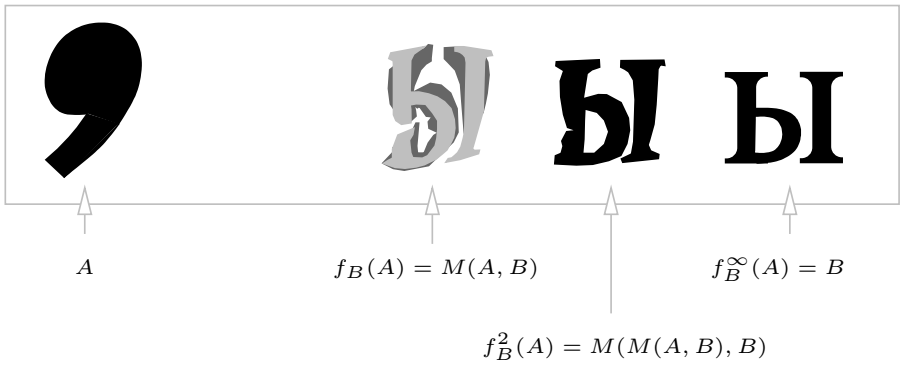

Figure 7: An example of morphing sequence. Left and right medians are shown by dark and light gray colours, respectively.

Having fixed a set $C \in \mathbb{H}(\mathbb{X})$, the mapping $f_C(\cdot) := M(\cdot, C)$ has a fixed point: $f_C(C) = M(C, C) = C$. Moreover, it is contractive with respect to the Hausdorff metric:

$$\lim_{i \to \infty} d_{\mathbb{H}}(f_C^i(X), C) = 0 \quad \forall X \in \mathbb{H}(\mathbb{X}).$$

We refer to the set C as a *condensation set* for a contraction mapping f_C in $\mathbb{H}(\mathbb{X})$.

Let us resume: we use Iterated Function Systems of type $\{\mathbb{H}(\mathbb{X}); f_1, f_2 \ldots, f_n\}$. The Hutchinson operator $\mathcal{F}$ maps $\mathbb{H}(\mathbb{H}(\mathbb{X}))$ onto $\mathbb{H}(\mathbb{H}(\mathbb{X}))$. It is convenient to

visualize the fixed point $A_{\mathcal{F}}$ as a book of black ink drawings (recall it can be uncountable). The set attractor $A_{\mathcal{F}}$ contains all fixed points of f_i. Therefore, if $f_i := M(\cdot, C_i)$, then it is possible to control the attractor's shape by choosing appropriate condensation sets C_i.

Note that all contraction mappings in $\mathbb{X}$ are contraction mappings in $\mathbb{H}(\mathbb{X})$. For example, a contractive two-dimensional affine transformation is a contraction mapping in $\mathbb{H}(\mathbb{X})$, where corresponding condensation set is a compact set, degenerated to a point. Therefore, new implementation contains the old one, inheriting all its properties such as contraction.

3.3. Deterministic algorithm

Figure 8 shows an example of Iterated Function System made of two transformations: $\{\mathbb{H}(\mathbb{X}); f_A, f_B\}$. The condensation sets A and B are shown in the upper corners of the image. As described in the section 2.2, the first step is to choose an arbitrary non-empty compact subset. Here we have chosen a bananas image S. Then the iteration process consists of applying each transformation f_A and f_B to the current set, and then taking the union of the resulting sets. The limit of this process gives the attractor of the IFS, as shown above.

Therefore, we start from a "book" with one page in it, namely bananas S. Since there are two transformations, the number of pages in the book will double with each step. At the second step the book contains two drawings: $f_A(S)$ and $f_B(S)$. In order to clarify the notations we omit the symbol f, concatenating the indices of transformations. For example, images ABS and $f_A(f_B(S))$ are the same (we shall also call ABS an *address* of the image $f_A(f_B(S))$). Thus, at the second step the book contains two drawings: AS and BS. Then recurrent call of the iteration step produces four new images: AAS, BAS, ABS, BBS. The images (sorted alphabetically) are shown in the third row of the figure 8.

Since the limit of the process does not depend on the initial choice of drawing S, the shape of bananas disappears in few iterations only. Note that in the case of two transformations the pages of the book (being sorted by their address) represent a smooth morphing sequence between A and B.

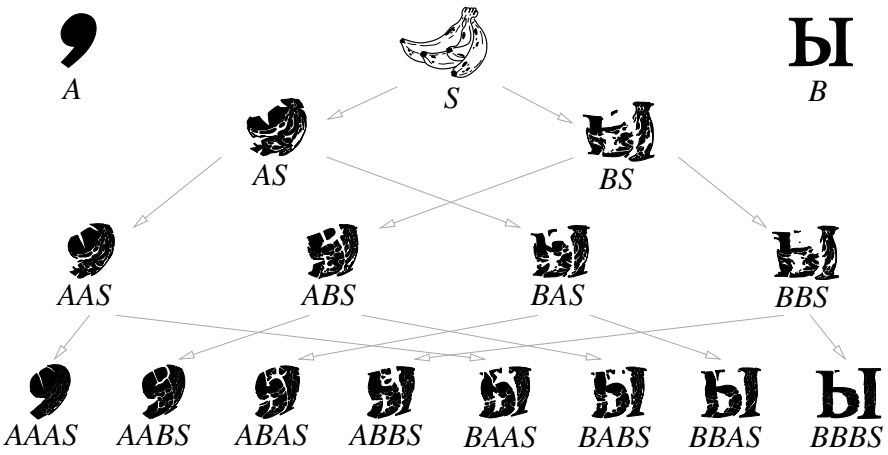

Figure 8: The deterministic algorithm for an IFS made of two transformations with condensation sets A and B. Starting from an arbitrary non-empty compact set S (bananas, for example), the system converges to the fixed point $A_{\mathcal{F}}$. Note that the attractor does not depend on the initial set S. In few iterations only the shape of bananas disappears completely from the "book" of drawings. Note that in the case of two transformations the pages of the drawings book (being sorted by their address) represent a smooth morphing sequence between A and B.

Note that in this example we have used two condensation functions, however, it is possible to use arbitrary number of transformations. Obviously, in such a case the tree will not be binary.

3.4. Random iteration algorithm

Let us take sets A and B as it is shown in figure 7. Now let us build an IFS $\mathcal{F} = \{\mathbb{H}(\mathbb{X}); f_1, f_2\}$ consisting of two contraction mappings with condensation sets A and B, built as it is shown in section 3.2. Thus, $f_1 = M(\cdot, A)$, $f_2 = M(\cdot, B)$ and $\mathbb{X}$ is a rectangular sheet of paper. In in this case the measure attractor $\mu_{\mathcal{F}}$ (rendered in figure 9) shows a uniform transition from A to B. We can approximate measure attractors using the random iteration algorithm:

1 **Input:** An IFS with probabilities $\mathcal{F} = \left\{\mathbb{H}(\text{rectangle});\ f_1, f_2;\ p_1 = \frac{1}{2}, p_2 = \frac{1}{2}\right\}$, output image resolution (w, h)

2 **Output:** An approximation (a digital image) of the measure attractor $\mu_{\mathcal{F}}$

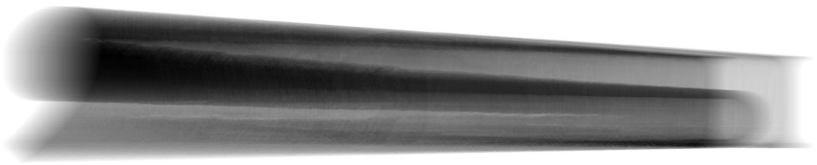

Figure 9: Image of a measure attractor for an IFS consisting of two simple contractions with condensation sets. Refer to figure 7 for the condensation sets.

draw a random non-empty binary image I_0 of resolution $w \times h$

initialize a matrix P of size $w \times h$ with zeroes

iterate {

  choose a transformation f_i according to the probabilities $\{p_1, p_2\}$

  $I_{k+1} = f_i(I_k)$

  if $(k > 100)$ {

    for each black pixel (i, j) of I_{k+1}

      $P(i, j) + +$

  }

}

Draw a gray-scale image $\mu_{\mathcal{F}}$ according to values of P

The main difference between the standard random iteration algorithm game and this algorithm is that at each iteration we have to store not just a point coordinates $x_k \in \mathbb{X}$, but an image of a compact subset $I_k \in \mathbb{H}(\mathbb{X})$.

Compare the result with the bottom row of figure 8: in this case the measure attractor $\mu_{\mathcal{F}}$ is the "X-Ray image" of the book of drawings. We superpose all the drawings and for each point we calculate in how many drawings it appears. Then a gray-level image is rendered using the number of occurrences.

4. Examples

4.1. Two dimensions

Next example illustrates that any contraction in $\mathbb{X}$ is also contraction $\mathbb{H}(\mathbb{X})$, therefore, an IFS in $\mathbb{R}^2$ is a particular case of IFS in $\mathbb{H}(\mathbb{R}^2)$. Let us consider an example of an IFS with two transformations: $\mathcal{F} = \{\mathbb{H}(\text{bi-unit square}); f_1, f_2\}$. Bi-unit square is a square with corners at (-1, -1), (1, -1), (1, 1) and (-1, 1). Transformation f_1 has a condensation set rendered in the left image of figure 10. Second contraction is a non-linear transformation defined as $f_2(r, \theta) = \left(\frac{r}{\sqrt{2}}, 2\theta\right)$. In other words, one takes the bi-unit square and stretches it in such a way, where for a point with polar coordinates (r, θ) the angle θ is doubled. Then the transformed square is shrunk to fit the original bi-unit square. The origin rests in his place. The image in the middle of figure 10 renders the transformation.

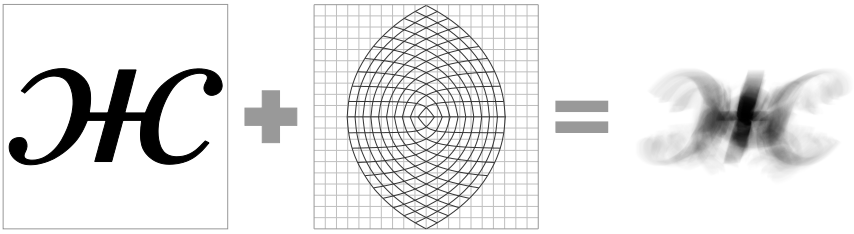

Figure 10: An IFS consists of two transformations: f_1 with a condensation set shown in the left image and $f_2(r, \theta) = \left(\frac{r}{\sqrt{2}}, 2\theta\right)$. The corresponding measure attractor is rendered in the right image.

Up to this moment we used only one type of contractions with condensations in $\mathbb{H}(\mathbb{X})$. However, there are plenty of other contraction mappings which can be constructed keeping the same condensation sets. For example, let us modify a bit the process described in section 3.2: as before, we detect bounding boxes of input sets A and B, superpose the boxes mid-way between A and B. Then we rotate it by angle proportional to the distance between A and B; finally we compute the median set.

Examples of such transformations (for different degrees of rotation) are shown at the top of figures 11 and 12. Note that by definition $\tilde{M}(A, B)$ is a ro-

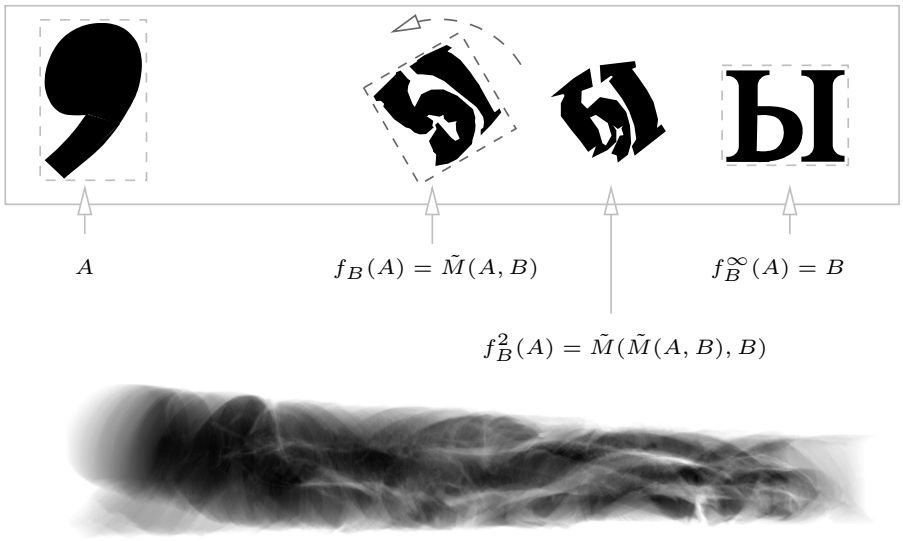

Figure 11: The median set is calculated by exactly the same routine as in the previous example (refer to figure 7), but finally it is rotated by the angle, proportional to the distance between sets A and B. The measure attractor is rendered at the bottom of the figure.

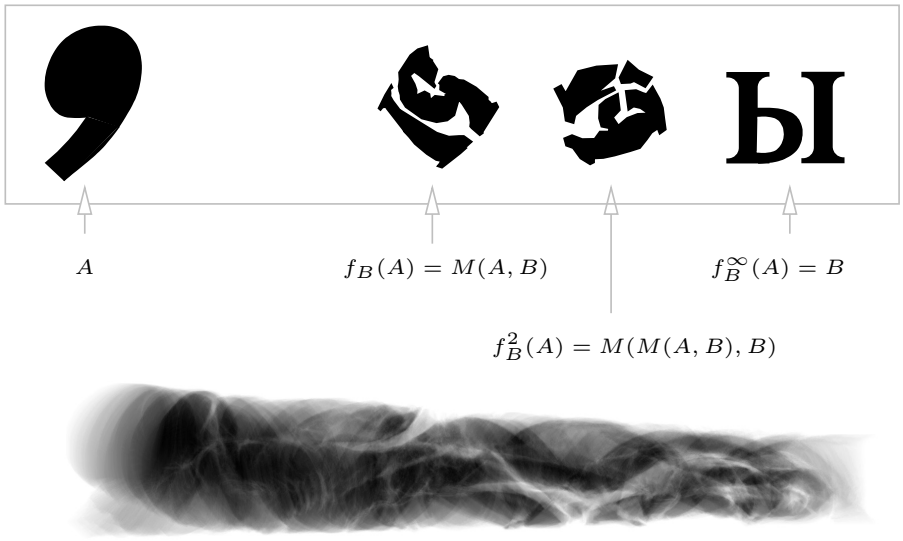

Figure 12: This example is given for another degree of rotation. The measure attractor is rendered at the bottom of the figure.

tated median set $M(A, B)$, however other intermediate images, e.g. $\tilde{M}(\tilde{M}(A, B), B)$, are completely different. Renderings of corresponding measure attractors are given at the bottom of the figures. Resulting images depend continuously on the parameters of transformations, e.g. the condensation sets and rotation degree. Therefore, by changing slightly the parameters we can obtain smooth animations.

4.2. Three dimensions

Up to this moment all examples were constructed in 2D, but nothing forbids to work in 3D, let us consider an example. The task is to create a cloud creature, a bull. Here an artist created a polygonal model of the bull. Second model was created by placing randomly (polygonal) balls inside the original shape. Refer to figure 13 for the renderings.

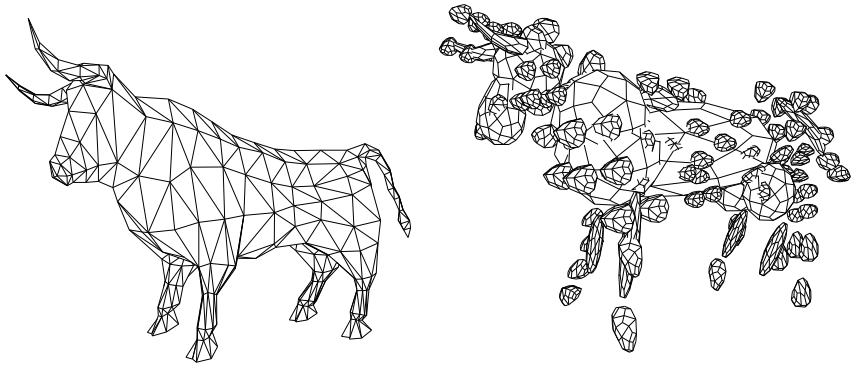

Figure 13: These quick sketches will give birth to a cloud creature.

Then the models were voxelized, in our example we used 256^3 voxels. Now all is ready to calculate a measure attractor of an IFS consisting of two condensation functions. Figure 14 shows a rendering result of the measure attractor. Here we imported the densities for the voxelized space to Maya fluids container to render the scene.

The rendering stage in Maya takes about 1 minute per image (including shading, lighting, self-shadowing) and the random iteration algorithm takes less

than 1 minute (at Intel Xeon CPU 1.6 GHz). Note that no hardware acceleration was used and the implementation can be greatly improved. Still, the running times are hugely inferior to the time of the modeling phase, to create such a bull mesh an experienced artist needs about one day of work.

Figure 14: The measure attractor for the IFS with condensation functions, shown in figure 13. The probabilities are taken equal to $\frac{2}{3}$ and $\frac{1}{3}$, respectively.

Difference between input sketches gives the effect of nebulosity. If a voxel is present in all the sketches, then it will be opaque and dense in the final render. Otherwise it will be more transparent. In this example the probabilities were taken equal to $\frac{2}{3}$ and $\frac{1}{3}$, respectively. This choice forces the final render to be closer to the first sketch. Figure 15 shows a cloudy version of the galleon we have already met in the introduction.

Since resulting images depend *continuously* on the parameters of transformations (input 3D models), it is easy to create animations in the usual for artists manner. Moreover, measure attractors can be imported to Maya fluids, it allows to combine our method of smoky objects creation with physics-based methods.

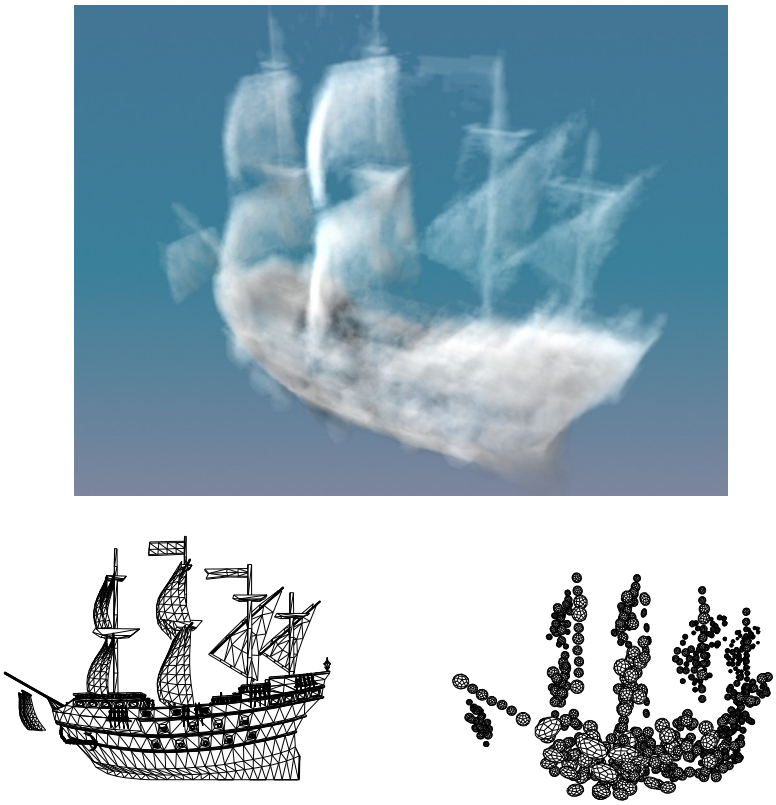

Figure 15: This galleon is created from two input models shown in the bottom row.

5. Conclusions

Up to our knowledge, this paper is the first attempt to use non-constant condensations for Iterated Function Systems. Besides purely aesthetic matters, IFS with condensation functions allow in some cases better control on the attractor. We have also shown an application for such IFS: creation of artistic renderings of vapourish objects. The main advantage of such application is its simplicity, the only source code necessary is a binary image morphing and a simple implementation of the random iteration algorithm, resulting in few hundreds lines of code. Besides clouds, we think that other phenomena might be rendered by means of IFS with condensation functions. For example, fractal

tops and similar techniques could give possibilities to draw textures of tissues.

[1] M. J. Harris, W. Baxter, T. Scheuermann, A. Lastra, Simulation of cloud dynamics on graphics hardware, in: W. Mark, A. Schilling (Eds.), Proceedings of the 2003 Annual ACM SIGGRAPH/Eurographics Conference on Graphics Hardware (EGGH-03), Eurographics Association, Aire-la-ville, Switzerland, 2003, pp. 92–101.

[2] R. Miyazaki, S. Yoshida, T. Nishita, Y. Dobashi, A method for modeling clouds based on atmospheric fluid dynamics, in: PG '01: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications, IEEE Computer Society, Washington, DC, USA, 2001, p. 363.

[3] D. S. Ebert, K. F. Musgrave, D. Peachey, K. Perlin, S. Worley, Texturing & Modeling: A Procedural Approach, Third Edition (The Morgan Kaufmann Series in Computer Graphics), Morgan Kaufmann, 2002.
URL http://www.amazon.ca/exec/obidos/redirect?tag=citeulike04-20{&}path=ASIN/1558608486

[4] J. Schpok, W. Dwyer, D. S. Ebert, Modeling and animating gases with simulation features, in: SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, ACM Press, New York, NY, USA, 2005, pp. 97–105. doi:http://doi.acm.org/10.1145/1073368.1073381.

[5] J. Hutchinson, Fractals and self-similarity, Indiana University Journal of Mathematics 30 (5) (1981) 713–747.

[6] M. Barnsley, Fractals everywhere, Academic Press Professional, Inc., San Diego, CA, USA, 1988.

[7] M. F. Barnsley, S. G. Demko, Iterated function systems and the global construction of fractals, Royal Society of London Proceedings Series A 399 (1985) 243–275.

[8] M. F. Barnsley, Superfractals, Cambridge University Press, 2006.

[9] S. Draves, The electric sheep screen-saver: A case study in aesthetic evolution, in: Applications of Evolutionary Computing, LNCS 3449, Springer Verlag, 2005.

[10] S. Draves, The electric sheep and their dreams in high fidelity, in: The 4th International Symposium on Non-Photorealistic Animation and Rendering, ACM, 2006.

[11] P. Meyers, A converse to Banach's contraction theorem, Journal of Research of the National Bureau of Standards 71B (2) (1967) 73–76.

[12] M. Iwanowski, J. Serra, The morphological-affine object deformation, in: Proceedings of 5th International Symposium on Mathematical Morphology, Kluwer Academic Publishers 2000, Palo Alto (USA), 2000, pp. 81–90.