Modelling Vegetation Through Fractal Geometry

Irina Tolkova

Spring Quarter 2014

Contents
1 Introduction 1
2 Preliminary Material 2
3 Iterated Function Systems 3
  3.1 Contraction Mappings ........................................... 3
  3.2 Iterated Function Systems ....................................... 5
  3.3 The Collage Theorem ............................................. 7
4 Applications of Iterated Function Systems 8
  4.1 The Algorithm ..................................................... 9
  4.2 Barnsley’s Fern ................................................... 10
  4.3 Fractal Tree ....................................................... 10
A Implementing the IFS Algorithm 10

1 Introduction

Most natural structures – plants, forests, mountains, landscapes – provide a stark contrast to shapes in Euclidean geometry. This causes a problem for modeling: the exactness of mathematics seems to be inadequate for capturing the complexity and apparent stochastic quality of real-life surroundings. However, in the 1960s, Benoit B. Mandelbrot proposed the idea of fractal geometry, which is fundamentally fragmented and nowhere smooth. In his book ”The Fractal Geometry of Nature”, he discussed the applications of fractal geometry (in his words, the ”geometry of nature”), especially for their use in studying the natural world. [4]

In his book Fractal Geometry, Kenneth Falconer avoids giving a precise definition of a fractal, writing ”it seems best to regard a fractal as a set that has properties as those listed below, rather than look for a precise definition which will almost certainly exclude interesting some cases” [3, pg xx]. The properties of the set $S$ he referenced are as follows:

1. $S$ has a structure that is detailed on arbitrarily small scales
2. $S$ is self-similar in some way; either directly, statistically, or approximately
3. $S$ cannot be described by traditional geometric shapes due to its irregular structure

While Mandelbrot gave a more formal definition of a fractal through the Hausdorff dimension of a set, the term "fractal" will be used informally throughout the paper to refer to sets with properties like those described by Falconer.

One of the motivations for this paper is to find a method for using fractals to model specific plant structures, such as tree canopies. In "Modelling of Tree Crowns with Realistic Morphological Features: New Reconstruction Methodology Based on Iterated Function System Tool," Collin et al use an Iterated Function Systems to create 3D models of tree canopies from various species, and comment on the effectiveness of the tool in creating realistic models [2]. A discussion of their work will be found in the section Applications of Iterated Function Systems.

This paper will start with a review of basic definitions and theorems relating to sets and metric spaces. Next, we will introduce Iterated Function Systems, starting with definitions and theorems on contraction mappings. The Contraction Mapping Theorem, IFS Theorem, and Collage Theorem will be discussed. Finally, we will mention the uses of an IFS in the natural sciences, and provide an algorithm (with examples) of the generation of visually fractal-like images. The appendix will contain the full code used.

2 Preliminary Material

Let us start by defining some basic properties and terms relating to sets.

The diameter of a nonempty set $A \in \mathbb{R}$ is defined to be $|A| = \sup \{|x - y| : x, y \in A\}$ (the diameter of the null set is 0). [3, pg 5]

A covering of $A$ is a sequence of nonempty sets $\{A_j\}_0^\infty$ such that $A \subset \bigcup_0^\infty A_j$.

A metric space is a space, or set, $X$, along with a function $d : X \times X \to \mathbb{R}$ that measures the distance between two points in $X$. This function must satisfy the following properties:

- $d(x, y) = d(y, x)$ for all $(x, y) \in X$.
- $0 < d(x, y) < \infty$ for all $(x, y) \in X$ such that $x \neq y$.
- $d(x, x) = 0$ for all $x \in X$.
- $d(x, y) \leq d(x, z) + d(z, y)$ for all $x, y, z \in X$.

Then $d$ is called a metric on $X$ and we say that $(X, d)$ is a metric space. [1, pg 18]

A sequence of points $\{x_n\}_{n=0}^\infty$ in a metric space $(X, d)$ is called a Cauchy sequence if for any $\epsilon > 0$ there is an integer $N > 0$ such that if $k, j > N$ then $d(x_k, x_j) < \epsilon$. [1, pg 16]

We say that a metric space $(X, d)$ is complete if every Cauchy sequence in $X$ converges to a limit point $x \in X$. [1, pg 18]
Now let \((X,d)\) be a complete metric space. Define the space \(H(X)\) to be the set of all non-empty compact subsets of \(X\).

Now, suppose we want to make a metric for \(H(X)\) – in other words, to define a distance between the elements of \(H(X)\). Since points of \(H(X)\) are themselves sets, we need have a definition of the distance between two sets.

Let \((X,d)\) be a complete metric space, and let \(x \in X, B \in H(X)\). Then we define the distance from the point \(x\) to the set \(B\) to be:

\[
d(x,B) = \min \{d(x,y) : y \in B\} \tag{1}\]

Now let \(A\) be another point in \(H(X)\) (and therefore another compact subset of \(X\)), and let \(x\) be in \(A\). Define the distance from \(A\) to \(B\) to be:

\[
d(A,B) = \max \{d(x,B) : x \in A\} \tag{2}\]

Finally, let \((X,d)\) be a complete metric space. Then the Hausdorff distance between \(A,B \in H(X)\) is defined to be:

\[
h(A,B) = \min \{d(A,B),d(B,A)\} \tag{3}\]

It can be shown that \(h\) is a metric on the space \(H(X)\) [1, pg 34]. Therefore, we can define the metric space \((H(X),h)\), which is known as the "space of fractals". It can be shown that this \((H(X),h)\) is in fact a complete metric space. Furthermore:

**Theorem 2.1 Theorem on Completeness of the Space of Fractals**

Let \((X,d)\) be a complete metric space. Then \((H(X),h)\) is a complete metric space. Moreover, if \(\{A_n\}_{n=1}^{\infty} \in H(X)\) is a Cauchy sequence, then its limiting set \(A = \lim_{n \to \infty} A_n \in H(X)\) is the set of all points \(x \in X\) such that there exists a Cauchy sequence \(\{x_n \in A_n\}\) that converges to \(x\).

A proof can be found in Michael Barnsley’s *Fractals Everywhere.* [1, pg 38]

### 3 Iterated Function Systems

So far, self-similarity has been treated as a property of fractal sets. But it can also be used to define – and, by extension, construct – fractals. The "iterated function system" or "iterated function scheme," (abbreviated as "IFS") is an approach which relies on self-similarity to approximate fractal sets by repeatedly iterating a starting set through a system of contractions. [3, pg 113]

#### 3.1 Contraction Mappings

The defining quality of a self-similar structure is that a subset of it will be similar to the overall set. Therefore, informally, the structure can informally be thought of as being a union of many pieces, where each piece is the original set compressed in size (and perhaps rotated or translated). For this paper, the shrinking, rotating, and translation will be done by maps called contractions.
So, to create self-similarity, it is useful to define a contraction. Let \((X, d)\) be a metric space. A transformation \(f : X \to X\) is a contraction if there exists a number \(s \in (0, 1)\) such that \(d(S(x) - S(y)) \leq s \cdot d(x - y)\) for any two \(x, y \in D\). [1, pg 75]

Let \((X, d)\) be a metric space and let \(f : X \to X\) be a transformation on \(X\). If a point \(x_f \in X\) satisfies \(f(x_f) = x_f\), \(x_f\) is called a fixed point of \(f\). So, when a transformation takes one set to another set, the fixed points will stay in place. This can help determine the image of the transformation. [1, pg 73]

The constant \(s\) is called the contraction coefficient or contraction factor of \(f\). The \(k\)th iterate of \(S\), \(f^k\), is given by \(f^k(D) = f(f^{k-1}(D)), f^0(D) = D\). [3, pg 113]

Note that a contraction \(f\) must be continuous. This is easy to show: for any \(\epsilon > 0\) we can choose \(\delta = \epsilon\). When \(d(x, y) < \delta\) for some \(x, y \in X\), then \(d(f(x), f(y)) < s\delta = \epsilon < \epsilon\), so \(f\) is continuous. [1, pg 80]

**Theorem 3.1 (Contraction Mapping Theorem)** Let \(f : X \to X\) be a contraction mapping on a complete metric space \((X, d)\). Then \(f\) possesses exactly one fixed point \(x_f \in X\). Moreover, for any point \(x \in X\), the sequence of iterates \(\{f^n(x)\}\) converges to \(x_f\).

**Proof.**
First, for a fixed \(x \in X\), consider the distance \(d(f^m(x), f^m(x))\). Without loss of generality, suppose that \(n > m\). By definition, \(f\) satisfies \(d(f(u), f(v)) \leq c \cdot d(u, v)\) for any two \(u, v \in X\), where \(c \in (0, 1)\) is the contraction factor of \(f\). Therefore,

\[
d(f^m(x), f^n(x)) \leq s^m d(x, f^{n-m})
\]  

(4)

By the triangle inequality of metrics, \(d(u, v) \leq d(u, w) + d(w, v)\) for any \(u, v, w \in X\). Let \(k = n - m\). The right side of the above equation can be rewritten as:

\[
d(x, f^k) \leq d(f, f(x)) + d(f(x), f(f(x))) + d(f(f(x)), f^3(x)) + \ldots + d(f^{(k-1)}(x), f^k)
\]  

(5)

Each term can be expressed in terms of the contraction factor of \(f\):

\[
d(f, f(x)) + d(f(x), f(f(x))) + \ldots + d(f^{(k-1)}(x), f^k) \leq (1 + s + s^2 + \ldots + s^{k-1})d(x, f(x))
\]  

(6)

Since \(|s| < 1\), the polynomial on the right is a partial sum of a geometric series:

\[
(1 + s + s^2 + \ldots + s^{k-1})d(x, f(x)) = \frac{1 - s^k}{1 - s}d(x, f(x)) \leq \frac{d(x, f(x))}{1 - s}
\]  

(7)

Therefore,

\[
d(f^m(x), f^n(x)) \leq \frac{s^m}{1 - s}d(x, f(x))
\]  

(8)

Since \(x\) is fixed, \(d(x, f(x))\) is fixed. Since \(|s| < 1\), \(\frac{s^m}{1 - s}d(x, f(x))\) approaches 0 as \(m \to \infty\). Therefore, for any \(\epsilon > 0\), we can choose an integer \(M\) such that when \(m > M, \frac{s^m}{1 - s}d(x, f(x)) < \epsilon\), and therefore \(d(f^m(x), f^n(x)) < \epsilon\) (note that \(n > m > M\)). So, the sequence \(\{f^m(x)\}_{n=0}^\infty\) is a Cauchy sequence, and therefore converges to a limit \(x_f\).
Recall that the definition of a complete metric space, \( X \), contains the limit of every Cauchy sequence in \( X \). Therefore, \( x_f \in X \).

Now we want to show that \( x_f \) is a fixed point of \( f \). By definition, \( f(x_f) = \lim_{n \to \infty} f^{(n+1)}(x) \).

But by continuity of contractions, this is equal to \( \lim_{n \to \infty} f^{(n+1)}(x) = \lim_{n \to \infty} f^{(n)}(x) = x_f \).

Therefore, \( f(x_f) = x_f \), so \( x_f \) is a fixed point.

We have proved existence, but is the fixed point unique? Suppose \( f \) has two fixed points \( x_f \) and \( y_f \): so, \( f(x_f) = x_f \) and \( f(y_f) = y_f \). Then:

\[
d(x_f, y_f) = d(f(x_f), f(y_f))/\leq s \cdot d(x_f, y_f) \tag{9}
\]

Since \( s < 1 \), \( d(x_f, y_f) \) must equal zero, so \( x_f = y_f \). Therefore, the fixed point is unique, and this completes the proof. \( \square \)

This theorem is the basis for constructing fractals through Iterated Function Systems, which will be described in the next section. One of the key points of the theorem is that the sequence of iterates of \( f \) will converge to the fixed point, regardless of the starting point \( x \). In the case of Iterated Function Systems, the analog of the fixed point of a contraction will be a "fixed set" of a system of contractions, and this set is most often a fractal.

### 3.2 Iterated Function Systems

Now, an iterated function system (sometimes called a hyperbolic iterated function system) is a finite collection of contractions \( \{f_1, f_2, \ldots, f_N\} \) with corresponding contraction factors \( \{s_1, s_2, \ldots, s_N\} \), \( N \geq 1 \), mapping from a set \( X \) to \( X \). An IFS can also be denoted as \( \{X : f_i\} \).

**Lemma 3.1** Suppose \( (X, d) \) is a metric space and \( f : X \to X \) is a continuous mapping. Then \( f \) maps \( H(X) \) onto itself.

The proof is short, and can be found in [1, pg 80].

**Lemma 3.2** Let \( (X, d) \) be a metric space and let \( f : X \to X \) be a contraction mapping with contraction factor \( s \). Then the transformation \( F : H(X) \to H(X) \) defined by

\[
F(B) = \{f(x) : x \in B\} \tag{10}
\]

for \( N \in H(X) \) is a contraction mapping on \( (H(X), h(d)) \) with a contraction factor \( s \).

**Proof.** [1, pg 81]

Since \( f \) is a contraction mapping, it is continuous, so by the above lemma it maps \( H(X) \) to itself. Now, for two sets \( B, C \in H(X) \), the distance between \( F(B) \) and \( F(C) \) is given by the metric:

\[
d(F(B), F(C)) = \max\{\min\{d(f(x), f(y)) : y \in C\} : x \in B\} \tag{11}
\]
Since \( d(f(x), f(y)) \leq s \cdot d(x, y) \),
\[
d(F(B), F(C)) \leq \max\{\min\{s \cdot d(x, y) : y \in C\} : x \in B\} = s \cdot d(B, C) \tag{12}
\]
The same argument holds in the other direction, so:
\[
d(F(C), F(B)) \leq s \cdot d(C, B) \tag{13}
\]
Therefore, the Hausdorff metric of \( F(C), f(B) \) is:
\[
h(F(B), F(C)) = \min\{d(F(B), F(C)), d(F(C), F(B))\} \leq \min(s \cdot d(B, C), s \cdot d(C, B)) \leq s \cdot h(B, C) \tag{14}
\]
Therefore, \( F \) is a contraction mapping with contraction factor \( s \) on \( (H(X), h(d)) \). □

**Lemma 3.3** Let \( B, C, D, E \) be sets in \( H(X) \). Then:
\[
h(B \cup C, D \cup E) \leq \max[h(B, D), h(C, E)] \tag{16}
\]
**Proof.** [1, pg 33]
First, let \( A = D \cup E \). By definition,
\[
h(B \cup C, A) = \max[h(x, A) : x \in B \cup C]
\]
\[
= \max\{h(x, A) : x \in B, (h(x, A) : x \in C]\} \tag{18}
\]
\[
= \max[h(B, A), h(C, A)] \tag{19}
\]
Now substitute \( A = D \cup E \). This becomes
\[
h(B \cup C, D \cup E) = \max[h(B, D \cup E), h(C, D \cup E)] \tag{20}
\]
\[
= \max[h(B, D), h(B, E), h(C, D), h(C, E)] \tag{21}
\]

**Lemma 3.4** Let \( (X, d) \) be a metric space. Let \( \{F_n\}_{n=1}^N \) be contraction mappings on \( (H(X), h(d)) \) with corresponding contraction factors \( \{s_n\}_{n=1}^N \). Then the transformation \( W : H(X) \to H(X) \) defined by
\[
W(B) = F_1(B) \cup F_2(B) \cup \cdots \cup F_N(B) = \bigcup_{n=1}^N F_n(B) \tag{22}
\]
is a contraction on \( (H(X), h(d)) \) with contraction factor \( s = \max\{s_n\} \).

**Proof.** [1, pg 81] Proof by induction: suppose \( N = 2 \). Then \( W(B) = F_1(B) \cup F_2(B) \). For two sets \( B, C \in H(X) \), compute:
\[
h(W(B), W(C)) = h(F_1(B) \cup F_2(B), F_1(C) \cup F_2(C)) \leq \min[h(F_1(B), F_1(C)), h(F_2(B), F_2(C))] \leq \max[s_1, s_2] \cdot h(B, C) \tag{25}
\]
where (24) followed from Lemma 3.3. For the inductive step, simply write
\[ \bigcup_{n=1}^{N} F_n(B) = \left[ \bigcup_{n=1}^{N-1} F_n(B) \right] \cup F_N(B). \] (26)

This completes the proof. □

**Theorem 3.2 (Theorem on Iterated Function Systems)** Let \( \{X : F_n, n = 1, 2, \ldots, N\} \) be an iterated function system with contraction factor \( s \). Then the transformation \( W : H(X) \to H(X) \) given by
\[ W(B) = \bigcup_{n=1}^{N} F_n(B), B \in H(X) \] (27)
is a contraction mapping on the complete metric space \((H(X), h(d))\) with contraction factor \( s \). That is, for all \( B, C \in H(X) \),
\[ h(W(B), W(C)) \leq s \cdot h(B, C) \] (28)
Furthermore, this IFS has a unique fixed point which satisfies:
\[ A = W(A) = \bigcup_{n=1}^{N} F_n(A) \] (29)
This fixed point is given by:
\[ A = \lim_{n \to \infty} W^n(B) \] (30)
for any \( B \in H(X) \).

**Proof.**
The first part of the theorem – that \( W(B) \) is a contraction mapping on \((H(X), h(d))\) – has been shown in Lemma 3.4. Since \( W(B) \) is a contraction mapping, then by the Contraction Mapping Theorem, it has a unique fixed point in \( H(X) \) given by \( A = \lim_{n \to \infty} W^n(B) \). This completes the proof. □

### 3.3 The Collage Theorem

The extension of the Contraction Mapping Theorem to Iterated Function Systems provides a method of generating approximations of fractals through iterating a contraction over a set; a more specific discussion of the algorithm can be found in Section 4. Yet when creating fractal-like images, one would often like to image to look like some specific set, such as a picture of a fern or tree. In essence, this sums up to solving the inverse problem: given an attractor, find an IFS. So, it would be very useful to be able to find the set of contractions corresponding to a given attractor. This problem is the focus of the Collage Theorem.
Theorem 3.3 (The Collage Theorem) [1, pg 97]
Let \((X,d)\) be a complete metric space, and let \(L \in H(X)\) be a compact subset of \(X\). Fix \(\epsilon > 0\), and choose an IFS \(\{X; f_1, f_2, \ldots, f_N\}\) such that:

\[
h(L, \bigcup_{n=1}^{N} f_n(L)) \leq \epsilon
\]

where \(h(d)\) is the Hausdorff metric. Then:

\[
h(L, A) \leq \frac{1}{1 - s} h(L, \bigcup_{n=1}^{N} f_n(L))
\]

where \(A\) is the attractor of the IFS.

In an informal sense, the collage theorem shows that for an IFS to have an attractor "close to" a given set, the set \(\bigcup_{n=1}^{N} f_n(L)\), given by the union of the contractions of the IFS operating on some set \(L\), must be "close to" that set \(L\). This "closeness" is measured by the Hausdorff metric.

The proof of the Collage Theorem is based on proving that for a metric space \((X,d)\) and a contraction mapping \(f\) with contraction factor \(s\) and fixed point \(x_f \in X\),

\[
d(x, x_f) \leq \frac{d(x, f(x))}{1 - s}
\]

for all \(x \in X\). This has already been done as part of the proof for the Contraction Mapping Theorem.

4 Applications of Iterated Function Systems

One common problem occurring in the natural sciences is creating models to represent natural structures. To study environmental interaction between vegetation and the surroundings – such as studies of forest fires or plant photosynthesis and nutrient transfer – it would be useful to have models of vegetation. Most models fall into three categories: global representations, modular representations, and multiscale representations. The first models the plant as one homogenous whole, combining together the trunk, branches, leaves, and ambient air. This representation, while simple and useful for a rough-scale view of a tree, does not incorporate much detail. Modular representations build up the plant out of regular cells called voxels, each of which has attributed physical properties and represents some vegetation component. The model depends on repeating certain components in their corresponding position. Finally, multiscale representations are the most detailed models, accounting for leaves, branches, and other components. This detail is considered on many scales, resulting in the most accurate representation of the tree. [2]

Due to the intricate structure of natural objects at many scales, multiscale representations are difficult to capture through the simple geometric shapes of Euclidean geometry. Yet this apparent scaling self-similarity suggests an analog to the scaling self-similarity of fractals. Since objects such as ferns, mountains, trees, and coastlines can have a visual likeness to attractors of certain Iterated
Function Systems, images can be generated by some number of iterations of a system of contractions over an initial set.

In [2], the authors used the Collage Theorem to develop 3D models of two coniferous tree species. They were able to provide a relationship between experimentally measurable values for a tree’s leaf area index, branching orientation and crown surface to the constants in the contraction mapping of the IFS. Finally, the authors concluded that The IFS method is an efficient modeling tool, and encouraged the development of measurements of the contraction map constants for a greater variety of species.

4.1 The Algorithm

We can now apply the IFS theory to render images which approximate fractals. There are two main algorithms which accomplish this: the Deterministic Algorithm and the Random Iteration Algorithm. While both are based on taking an initial set \( A_0 \) and computing the sequence of sets \( W^n(A) \), the Random Iteration Algorithm incorporates a random component to the sequence by associating a probability to each contraction [1, pg 86]. This paper will focus on the Deterministic Algorithm; more specifically, on the Deterministic Algorithm using affine transformations in two dimensions.

An affine transformations is a transformation of the form \( f(x) = Mx + b \) where \( M \) is a matrix and \( x, b \) are vectors. In other words, an affine transformation is one that rotates, dilates, and translates a set. In the case of our two-dimensional contractions, this will take the form:

\[
f \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}
\]

(34)

The Iterated Function System associated with a certain attractor will be a set of these transformations \( (f_1, f_2, \ldots, f_n) \), each with associated values for \( a, b, c, d, e, f \). To describe such an IFS, it is easiest to put these values in a IFS code table.

If one finds the right values for all of the constants of the transformations, the algorithm becomes straightforward:

1. Start with a set of points. It should be noted that some of the power of IFS algorithm comes from the fact that the iterations will converge to the same attractor with any starting (compact) set. In practice – for simplicity and efficiency – this initial set will probably consist of discrete, disconnected points.

2. Apply the contractions to each point, and take the union of the resulting points to create a new set. This is one iteration of the contractions over the initial set.

3. Repeat the iteration on the resulting set. The larger the number of iterations, the closer the image will approximate the attractor of the IFS.

This method was used for generating the figures in the rest of this paper. Appendix I contains the full code, using Java (for generating the set of points resulting from multiple iterations) and MatLab (for graphing).
4.2 Barnsley’s Fern

One of the most popular examples of the use of Iterated Function Systems is Barnsley’s fern, which is encoded by:

\[
\begin{array}{|c|cccccc|}
\hline
\text{function} & a & b & c & d & e & f \\
\hline
f_1 & 0 & 0 & 0 & 0.16 & 0 & 0 \\
\hline
f_2 & 0.85 & -0.04 & -0.04 & 0.85 & 0 & 1.6 \\
\hline
f_3 & 0.2 & -0.26 & 0.23 & 0.22 & 0 & 1.6 \\
\hline
f_4 & -0.15 & 0.28 & 0.26 & 0.24 & 0 & 0.44 \\
\hline
\end{array}
\]

Take the initial set to be the point (1, 1). The images generated after 5 and 10 iterations are shown in Figure 1 and Figure 2, respectively. Now suppose the initial set consisted of two points: (1, 1) and (2, 2). The images generated after 5 and 10 iterations are shown in Figure 3 and Figure 4, respectively. The likeness in this figures supports the fact that any initial set will approach to the same attractor.

4.3 Fractal Tree

Similarly, the code for a fractal tree is given by:

\[
\begin{array}{|c|cccccc|}
\hline
\text{function} & a & b & c & d & e & f \\
\hline
f_1 & 0 & 0 & 0 & 0.5 & 0 & 0 \\
\hline
f_2 & 0.42 & -0.42 & 0.42 & 0.42 & 0 & 0.2 \\
\hline
f_3 & 0.42 & 0.42 & -0.42 & 0.42 & 0 & 0.2 \\
\hline
f_4 & 0.1 & 0 & 0 & 0.1 & 0 & 0.2 \\
\hline
\end{array}
\]

The image resulting with the initial set consisting of the point (1, 1) and 10 iterations is shown in Figure 5.

A Implementing the IFS Algorithm

For the programming behind the algorithm, two helper classes were made in Java. The first was a Point class, which simply stored a point with (x, y) coordinates, and defined a toString of the form "x y" for easier analysis by MatLab. The second was a Contraction class which stored the constants \(a, b, c, d, e, f\) used in the affine transformation, and which had a method to apply the transformation to a given point by matrix multiplication.

Then, an IFS class was made, which stored a queue of contractions as a field. Other than the constructor, this class had a single recursive method responsible for iterating over a set of points a given number of times. This method returned the set resulting from iteration.

An "IFSmain" class acted as the client, and the specific contractions and initial sets were constructed in this class. It constructed an IFS over a given set of contraction and ran the iteration. Then, it printed an output of points to a text file of given name. When a grid of points was used for an initial set, there was a extensive running time, so only a small number of iterations (below
Figure 1: Barnsley’s fern using the initial set consisting of the point (1, 1) and 5 iterations.
Figure 2: Barnsley’s fern using the initial set consisting of the point \((1, 1)\) and 10 iterations.
Figure 3: Barnsley’s fern using the initial set consisting of the two points (1, 1) and (2, 2) and 5 iterations.
Figure 4: Barnsley’s fern using the initial set consisting of the two points (1, 1) and (2, 2) and 10 iterations.
Figure 5: Barnsley’s fractal tree using the initial set consisting of the point (1, 1) and using 10 iterations.
5) was used for this set. For a higher efficiency with a higher number of iterations, a single point (1, 1) was used. Even 10 iterations took under 20 seconds.

Finally, a script in MatLab was used to read and plot the text file containing the output of IFSmain, and is omitted.

The Point class code:

```java
public class Point {
    public double x;
    public double y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public String toString() {
        return x + " " + y;
    }
}
```

The Contraction class code:

```java
public class Contraction {
    private double a;
    private double b;
    private double c;
    private double d;
    private double e;
    private double f;

    public Contraction(double a, double b, double c, double d, double e, double f) {
        this.a = a;
        this.b = b;
        this.c = c;
        this.d = d;
        this.e = e;
        this.f = f;
    }

    public Point apply(Point p) {
        Point nextP = new Point(a * p.x + b * p.y + e, c * p.x + d * p.y + f);
        return nextP;
    }
}
```
The IFS class code:
public class IFS {
    private Queue<Contraction> contractions;

    public IFS(Set<Contraction> input) {
        contractions = new LinkedList<Contraction>();
        for (Contraction f: input) {
            contractions.add(f);
        }
    }

    public Set<Point> iterate(int iterations, Set<Point> initial) {
        if (iterations == 0) {
            return initial;
        } else {
            Set<Point> result = new HashSet<Point>();
            for (int i = 0; i < contractions.size(); i++) {
                Contraction next = contractions.remove();
                for (Point p: initial) {
                    result.add(next.apply(p));
                }
                contractions.add(next);
            }
            return iterate(iterations - 1, result);
        }
    }
}

The client code: IFSmain
import java.io.*;
import java.util.*;

public class IFSmain {
    public static void main(String[] args) throws FileNotFoundException {
        Contraction c1 = new Contraction(0, 0, 0, 0.16, 0, 0);
        Contraction c2 = new Contraction(0.85, 0.04, -0.04, 0.85, 0, 1.6);
        Contraction c3 = new Contraction(0.2, -0.26, 0.23, 0.22, 0, 1.6);
        Contraction c4 = new Contraction(-0.15, 0.28, 0.26, 0.24, 0, 0.44);

        Set<Contraction> functions = new HashSet<Contraction>();
        functions.add(c1);
        functions.add(c2);
        functions.add(c3);
        functions.add(c4);
Set<Point> set1 = new HashSet<Point>();
Point p1 = new Point(1, 1);
set1.add(p1);

run(10, functions, set1);

public static void run(int iterations, Set<Contraction> functions, Set<Point> initial) throws FileNotFoundException {
    IFS system = new IFS(functions);
    Set<Point> result = system.iterate(iterations, initial);
    PrintStream data = new PrintStream(new File("treeOutput10.txt"));
    for (Point x: result) {
        data.println(x);
    }
}

References