university of groningen

faculty of mathematics and natural sciences

# In by Out again

Faking arbitrarily-deep zooming on Iterated Function Systems

Bachelors's thesis

August 11, 2020

Student: Wiebe-Marten Wijnja

Primary supervisor: dr. J. Kosinka

Secondary supervisor: G. J. Hettinga

# Contents



# 1 Abstract

# 2 Introduction

Iterated Function Systems (IFSs) are a method to generate infinitely detailed fractal images by repeatedly applying simple mathematical functions until a fixed point is reached. [CITE] IFSs see use in rendering/modeling of physical phenomena[CITE], image compression [CITE] and representing gene structures [CITE]. Sometimes they also see use simply for the aesthetic beauty of their graphical representations[CITE].

Various computer algorithms to visualize IFSs exist [CITE], [CITE]. However, these all take either a still image as final result, or, if they want to render an animation, view this as a sequence of separate still images to generate.

This leaves a door open for potential optimization: if there is information that remains the same between animation frames, then we could compute it only once and re-use this information for all frames.

For instance, many kinds of animations consist of transformations of the camera viewport w.r.t the viewed fractal over time like translation, rotation and scaling do not require alterations to the fractal itself. This means that (an approximation of) the fractal might be computed once and then be used for all frames.

Furthermore, because of the self-similar nature of the rendered fractals, it might it be possible to simulate zooming in to an arbitrary depth by 'jumping up' to a more shallow viewport that shares the same self-

similarity as the original one.

Investigating this claim in detail is the essence of this thesis.

### 2.1 Overview

## 3 Background

Informally, an Iterated Function System is a set of transformations that, given any input image, can create a new image by

1. transforming the input image with each of the transformations
2. combining all transformed images together. This is the new image.

This process is then repeated an arbitrary number of times, until changes between the input image and output image are no longer visible to the human eye.

What you end up with is a visual representation of the IFS's attractor.

reference picture

### 3.1 Formal definition of an Iterated Function System

Formally, an Iterated Function System consists of a finite set of contraction mappings that map a complete metric space $(\mathcal{M}, d)$ to itself:

$$= \{f_i : \mathcal{M} \to \mathcal{M} | i = 1, 2, \ldots, N\}, N \in \mathbb{N}$$

That each mapping needs to be contractive means that for each mapping $f_i$, the distance between every two arbitrary points $a$ and $b$ in $(\mathcal{M}, d)$ needs to be larger than the distance of the points after transforming them:

$$d(f_i(a), f_i(b)) < d(a, b)$$

We can then take the union of performing all of these mappings on any compact set of points $\mathcal{S}_0 \subset \mathcal{M}$, and iterate this procedure as often as we'd like:

$$S_{n+1} = \bigcup_{i=1}^{N} f_i(S_n)$$

If we perform this operation an arbitrary number of times, we approach the Fixed-Point, or Attractor, $\mathcal{A}$ of the Iterated Function System:

$$\mathcal{A} = \lim_{n \to \infty} S_n$$

Curiously, which set of points $\mathcal{S}_0$ we started with makes no difference (even if it is a single point) [CITE].

Most research of IFSs restricts itself to using $\mathbb{R}^2$ as metric space[1] which can easily be rendered to screen or paper, and furthermore most commonly-used IFSs are restricted to use *affine transformations* as mappings.

Because of their prevalence, these are also the restrictions that will be used in this thesis.

### 3.2 Rendering an Iterated Function System

A couple of algoritms exist to render (visualize) the attarctor of an Iterated Function System. While it is impossible to render the attractor exactly, as this would require an infinite number of transformation steps, we can approximate it until we are certain that the difference between our approximation and the attractor is smaller than the smallest thing we can visually represent (e.g. smaller than the size of a pixel).

More in-depth information about the rendering of Iterated Function Systems can be found [CITE]. Short summaries of the two most common techniques will now follow.

All of the rendering techniques have in common that the main sequence of

#### 3.2.1 The deterministic method

#### 3.2.2 The chaos game

## 4 Problem Description

## 5 Findings

## 6 Discussion

## 7 Conclusion

## 8 Further Work

[1] More formally, the two-dimensional Euclidean space: $\left(\mathbb{R}^2, d(p,q) = \sqrt{p-q)^2}\right)$.