Plotting and Scheming: an odyssey of fractals and functional programming.

André van Meulebrouck, started 22 Sep 13 Su

# Table of Contents

[1 Table of Contents 1](#_Toc367643794)

[2 Introduction 1](#_Toc367643795)

[3 Plotting 1](#_Toc367643796)

[4 Scheme 1](#_Toc367643797)

[5 Vector Fractals 2](#_Toc367643798)

[6 Complex Plane Fractals 2](#_Toc367643799)

# Introduction

I have always been intrigued by factals. This account of my odyssey into the subject matter will walk the reader through my footsteps in wanting to reproduce fractals on home built hardware using home built software.

# Plotting

I have always been intrigued by any type of plotting because it allows visualizing equations. Humans are notoriously bad at comprehending myriad details; so anything that allows visualization is of interest to me. A picture is worth a thousand words.

Since my early days of plotting mathematical equations to see what they look like, a classical problem in computer science has been of keen interest: how do you allow a user to change the equation to be plotting without having to recompile the program. This question will be explored herein, with an answer proposed from LISP. Create an embedded LISP system and allow the user to express equations in LISP.

# Scheme

Herein, MIT’s dialect of LISP will be used: Scheme. However, there are some aspects of Scheme that have never appealed to me. In particular, you can’t intrinsically define an if function. It has to be built-in. The idea of lazy languages has always had an appeal to me, so the dialect of Scheme used will be lazy. We will then consider whether Scheme should have been defined lazily from its onset.

# Vector Fractals

There is a type of fractals that were experimented with by Bill Gosper in LISP. I call these “vector fractals” because they are based on a vector that continually gets replaced by smaller copies of itself (potentially down to infinite smallness).

While many of these can also be expressed using L-system fractals, the vector expression is interesting because it is very operational in nature, and less declarative than L-system fractals. Herein we will see which fractals can easily be expressed using the different approaches and consider what formal languages are at play. How much power is needed to express a given fractal? Is a vector expression simpler than the equivalent L-system expression? Are there some fractals that can’t be expressed using one approach or the other?

My original vector fractals were written in LISP, and only allowed strict vector replacement. That is: the replacement must begin and end at the endpoints of the original vector. However some vector fractals require a relaxing of this rule so that the replacement doesn’t have to end at the same endpoint. We will consider these, and whether we need to allow moving the endpoint, or keep it in place (even while we might not end on the same endpoint).

# Complex Plane Fractals

Complex plane fractals were made famous by Mandelbrot; but they are much more than simply Mandelbrot fractals. Julia sets are often much more breathtaking. And there are many more equations to be explored rather than the Mandelbrot equation. Once again, the ability to change equations at run time will more readily allow exploring these.