Project Report
CS421 SU21
UIUC

Fractals Using Functional Programming

Submitted by:

Gowtham Kuntumalla
gowtham4
August 1st week, 2021

Overview

In this project, we work on how to create Fractal geometry using the techniques in a Functional Programming Language – Haskell. Fractals are simple mathematical structures which are self-similar at different scales. Snowflake is popular real-life structure which is a fractal. Th intricate geometry repeats itself at successive magnifications. These elegant patterns can be generated using seemingly easy recursion functions.

Our fascination with Fractals started a few years back when we were doing a research internship at Washington University in St. Louis and then again in a Non-Linear Dynamics and Chaos course in the 4th year of our Undergraduate program.

Mandelbrot Sets and Julia Sets are two examples of fractals which we generate using Haskell concepts such as higher order functions, recursion, and laziness.

Implementation

This section lists various components of the code at a higher level. It is broadly derived from the reference paper with some custom modifications and experiments.

The most basic components of a visual object are Points and Images (groups of points). These can be represented on a 2D Euclidian space using a Tuple of Floating-point numbers and 2D list of Points respectively.

Core logic to implement the replicating logic. For example, Mandelbrot Set can be defined by set of points Z for which the iteration of the function (quadratic map) remains *bounded* (Wikipedia).

$$z_{n+1} = z_n^2 + c$$

Here z_n is a point in the Euclidean space with (0,0) as the starting point for Mandelbrot set.

It is initiated by a seed which is a point other than the origin (0,0) for Julia Set. It is slightly displaced from the origin. All the subsequent points are generated from this Seed point. Julia Sets are like a subset of the Mandelbrot sets located at different coordinates in the broader Mandelbrot set.

This iterator is a key generator for mandelbrot set (set of all z that satisfy the following equation). say z = x + i y, p = u + i v (complex numbers). z_(n+1) = z_n^2 + p

2D Array of Points is converted to an image using a character palette.

Tests and Results

In this section, we post some of the results of this experiments. Figures 1-2 and 3-6 are examples of Mandelbrot sets and Julia Sets respectively. Similar points in the pictures are represented by similar character (from the pallete.)

Also, jut to illustrate how simple changes to the function can alter images -> we share the dramatic change in the figures with just the change in the itertor from 'next' (2way symmetry) to 'next2' (3way symmetry – see the equation below).

$$z_{n+1} = z_n^3 + c$$

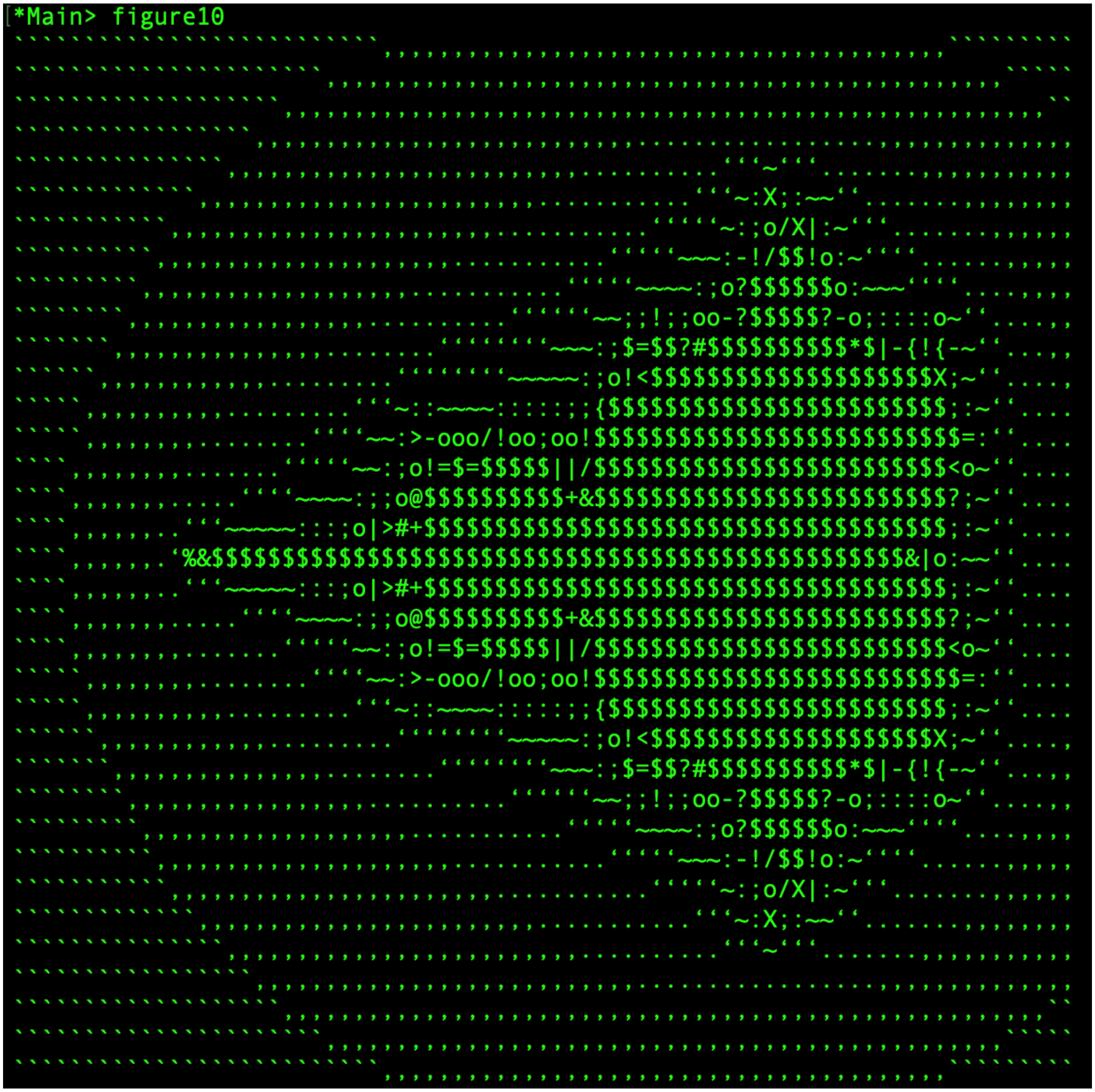

Figure 1 Example of a Mandelbrot Set

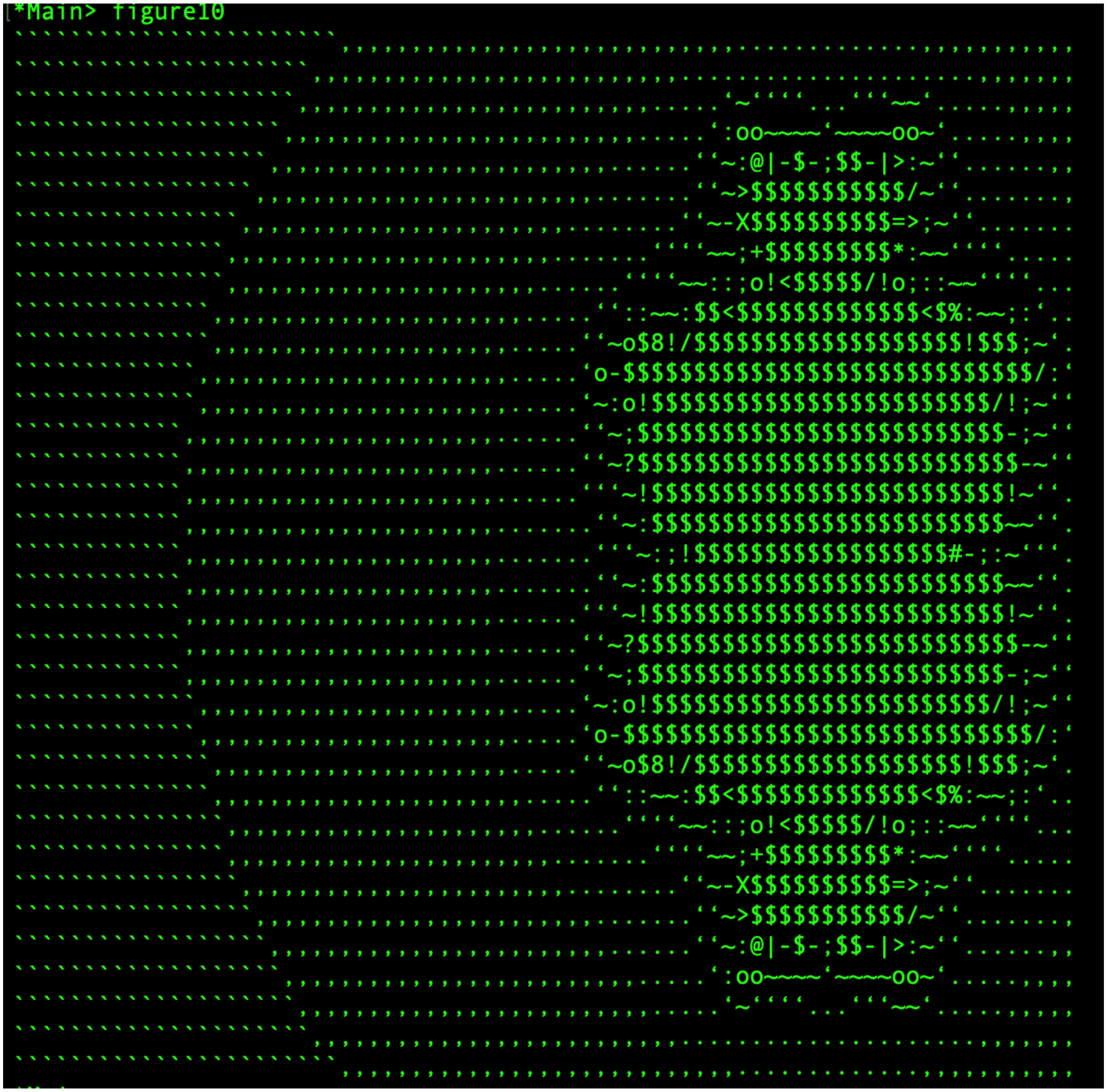

Figure 2 Example of a Mandelbrot Set (3way symmetry - cubed function)

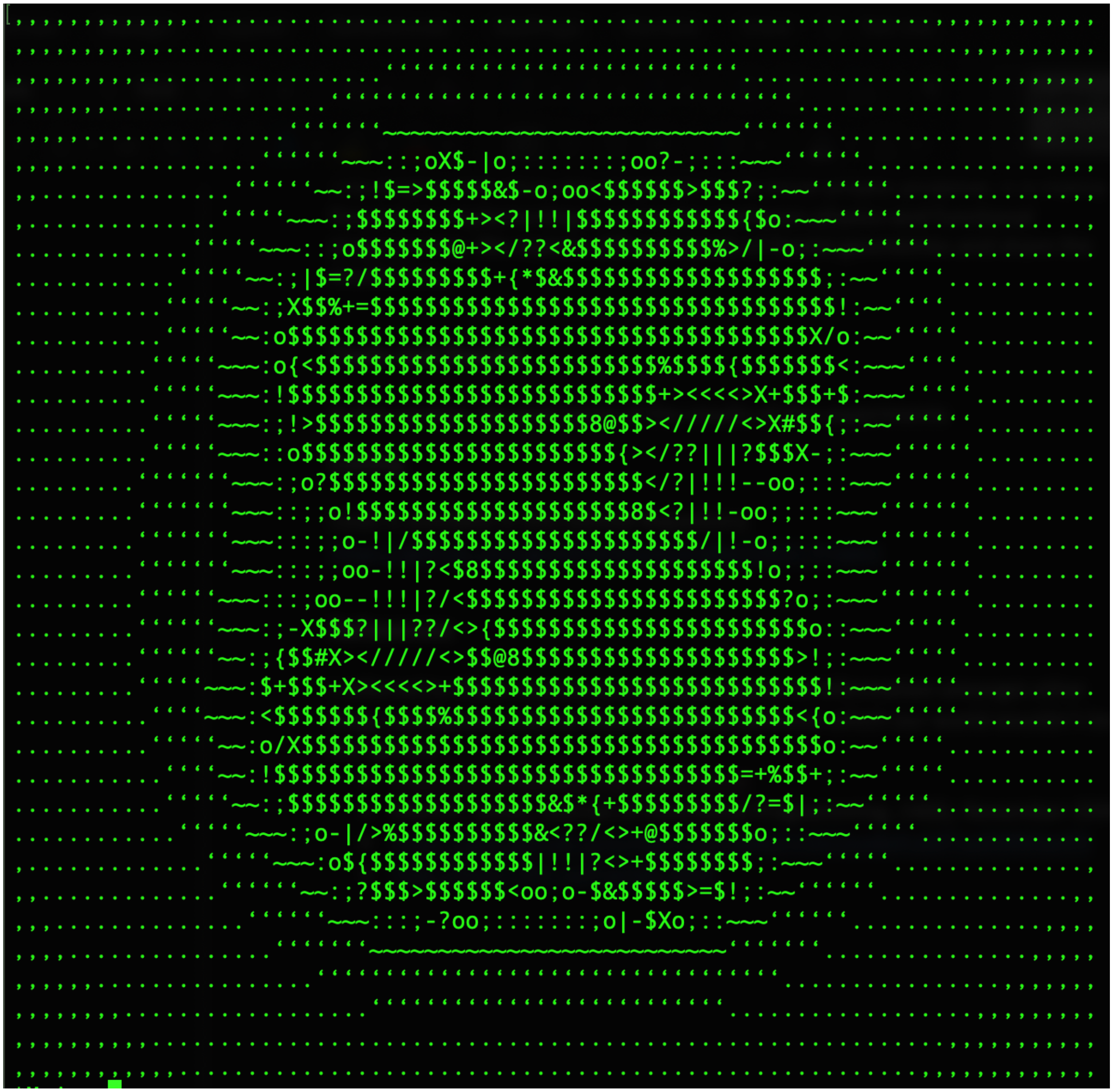

Figure 3 Example of a Julia Set (seed (0.32, 0.043))

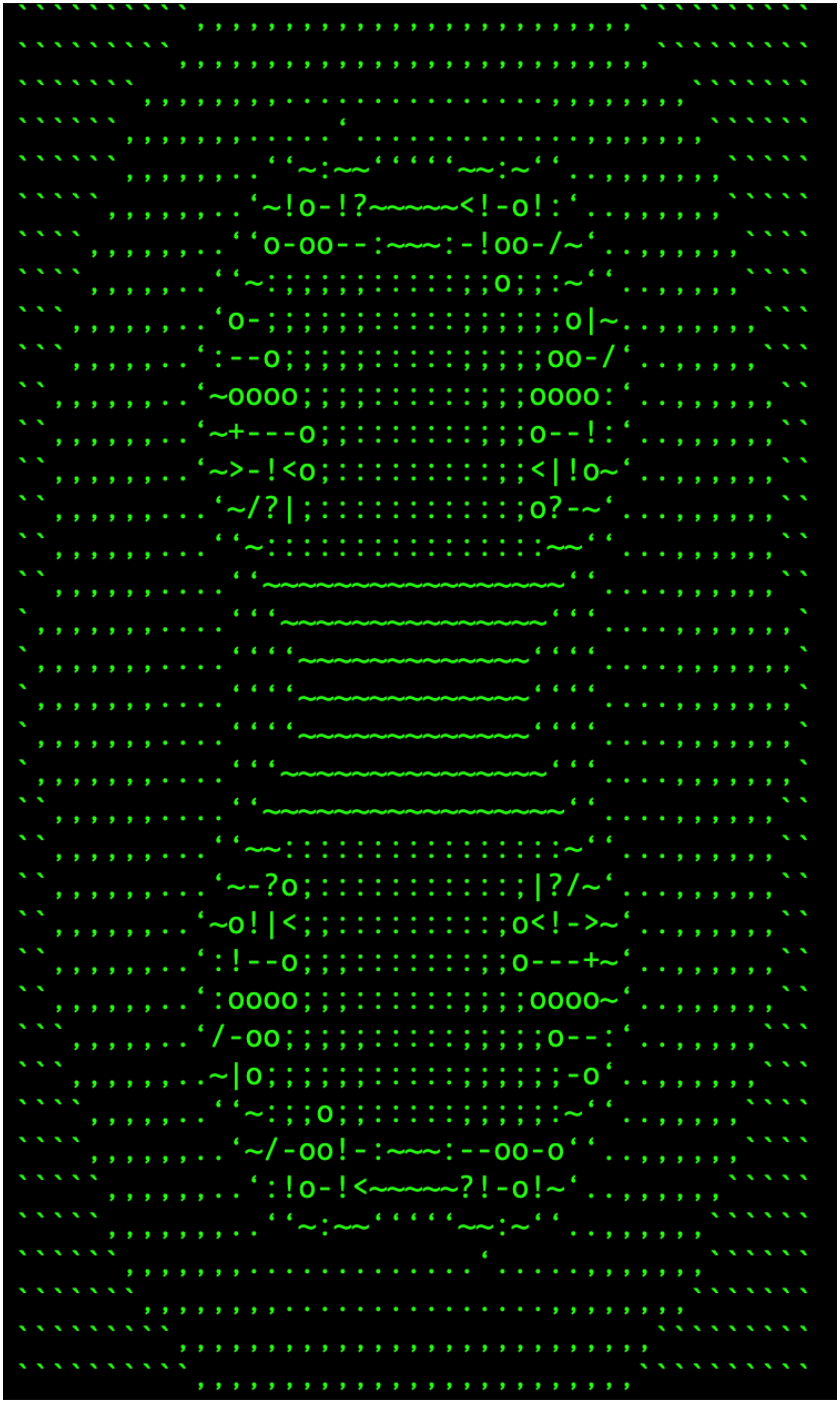

Figure 4 Julia Set (Seed (0.5, 0.01))

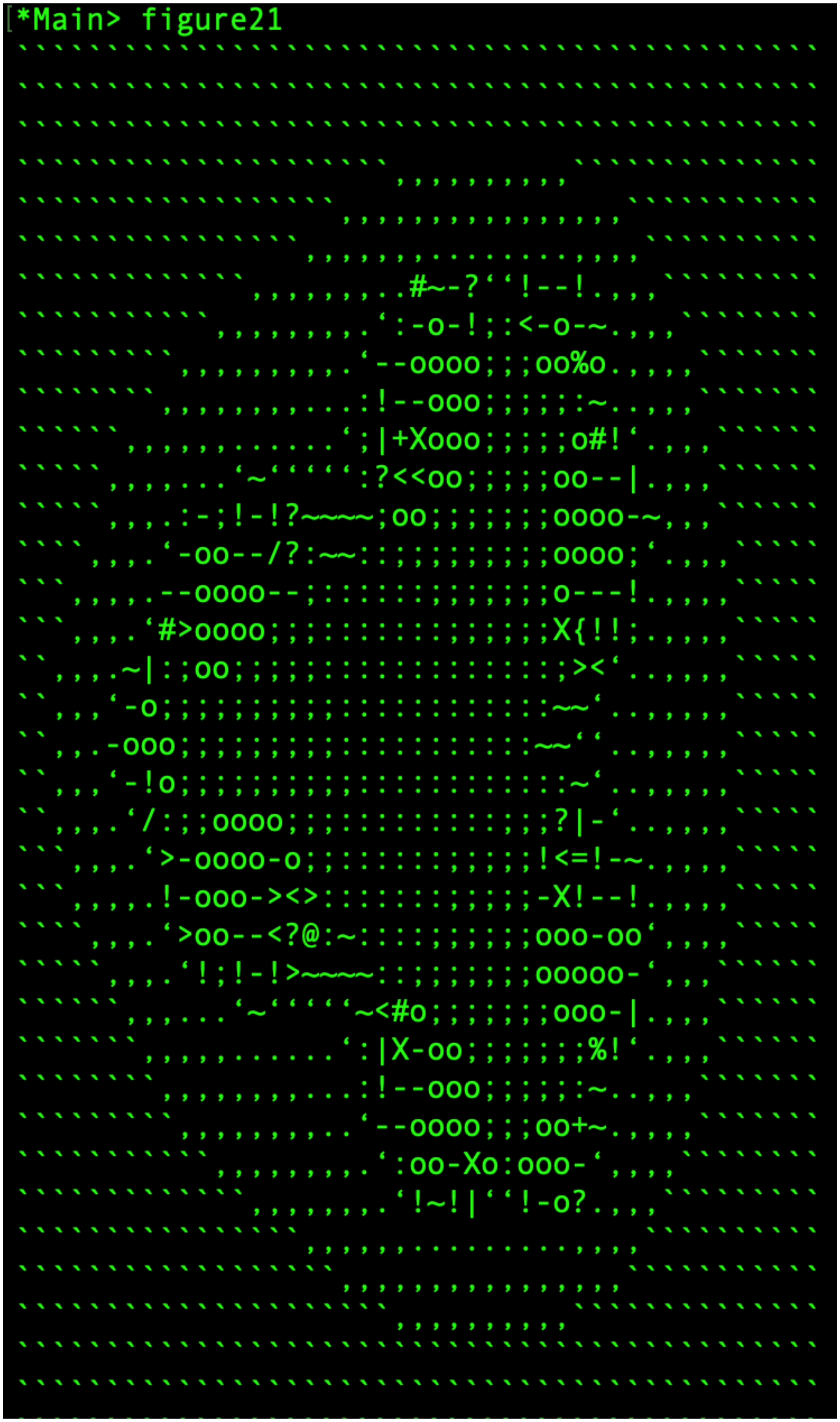

Figure 5 Julia Set (Seed (0.5, 0.01)) with the 3way iterator- cubed function (not was beautiful as before)

This project does what was proposed in the initial project proposal. It is mainly focused on implementing a connection between physics and functional programming. We explore different kinds of fractal geometries and share the results here.

Listing

Code is available in this GitHub Repository.
https://github.com/gowthamkuntumalla/CS421-Project

../ CS421-Project/fractals-code/fractal-main.hs contains relevant code. Folder marked not-working contains some failed attempts at trying to Rasterific package to work for RGB images.

References

I referred to these sources as a general source of information amongst other blogs and forums such as *Stack Overflow* (without which our world wouldn't be the same!)

- http://en.wikipedia.org/wiki/Fractal
- Mark P. Jones, Journal of Functional Programming, 14(6), November 2004.
- https://gist.github.com/mbrc12/c3a40215022ea8efcddf7ad39993e4f3
- http://learn.hfm.io/fractals.html
- https://www.karlsims.com/julia.html