Abstract

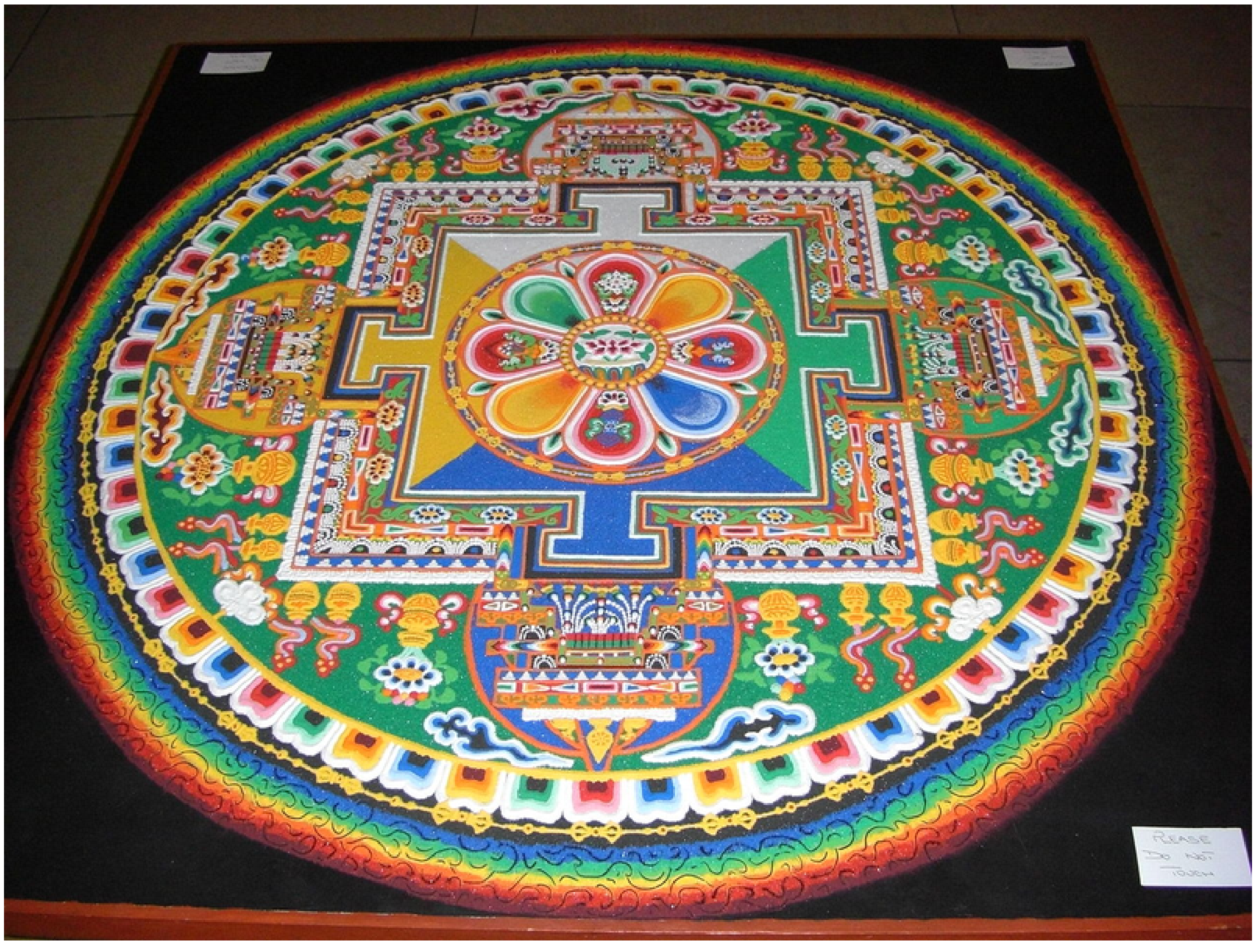
Using Haskell’s advanced type system to map the structures in Tidal Cycles to the underlying shapes of Mandala art and produce beautiful visualisations.

Mandalas are geometric designs, created with circles and repeated simple shapes. Weaved squares, concentric circles, intricate triangles and squiggly lines form the piece. They are common in south east asian art, showing up in temples and sand paintings alike. Today they are widely used, to the extent that you find them on shirts and household items.

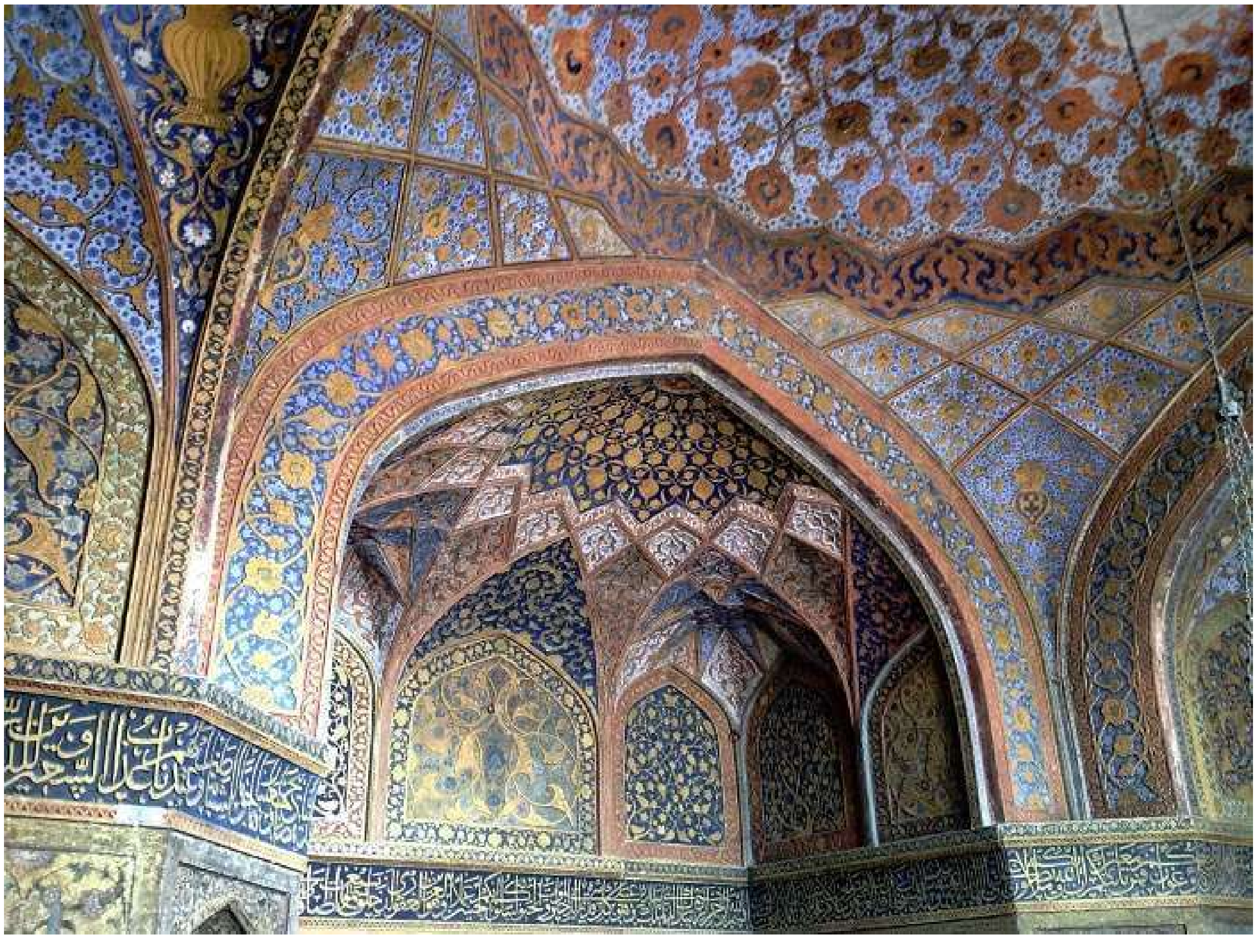
Tidal Cycles is a live coding environment for music production, created by Alex McLean. All the Tidal visualisations I saw were linear. Notes playing forward with time. But mandala art remains *in situ*. The shapes morph in the same place, accompanying the rhythms and cycles of the music. The periodic nature of music should reflect in its visuals as well. Thus came the project idea: Map the underlying structures of Tidal to mandala patterns.

BACKGROUND

From all that I found out about mandalas, they are general patterns. They are not really particular to India or any other country. Native American art has some mandalas, so does Tibetan sand art.



[Tibetan monks take days to painstakingly make these sand paintings. After a while, they let the tides wash over them.]



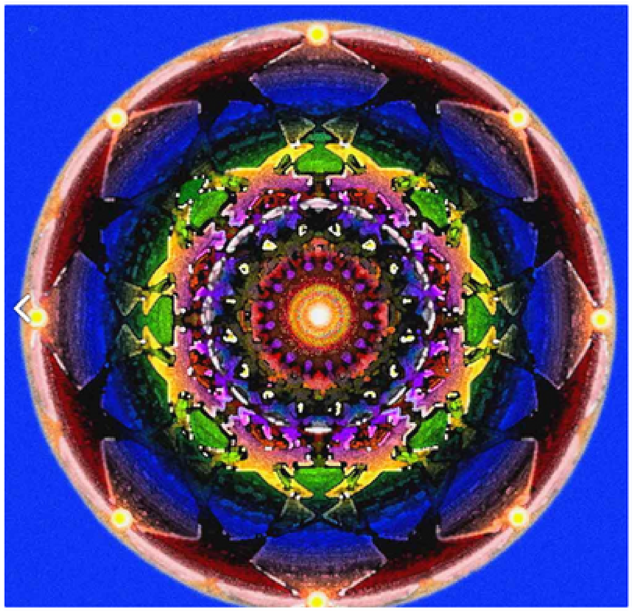
[Pattern finds its home in the white marbles that adorn the Taj Mahal]

Not unlike euclidean rhythms[cite], the underlying structures make the art universal. However, their mathematical symmetry, though obvious, remains undefinable. No simple code like euclidean rhythms that is concise. Instead, Mandala art in computers is formed usually through an exploratory process, going back as far as we have generated graphics. Speaking of computers generating graphics

FRACTALS

Some mandalas are fractal. It's an aspect of their underlying mathematics. Arthur C Clarke noted an odd coincidence when he wrote about mandalas -

> "[..] but indeed the Mandelbrot set does seem to contain an enormous number of mandalas."



[Mandalas and Fractals: The visual similarity is obvious. Bottom Left Picture illustrates the coast of britain[cite]]

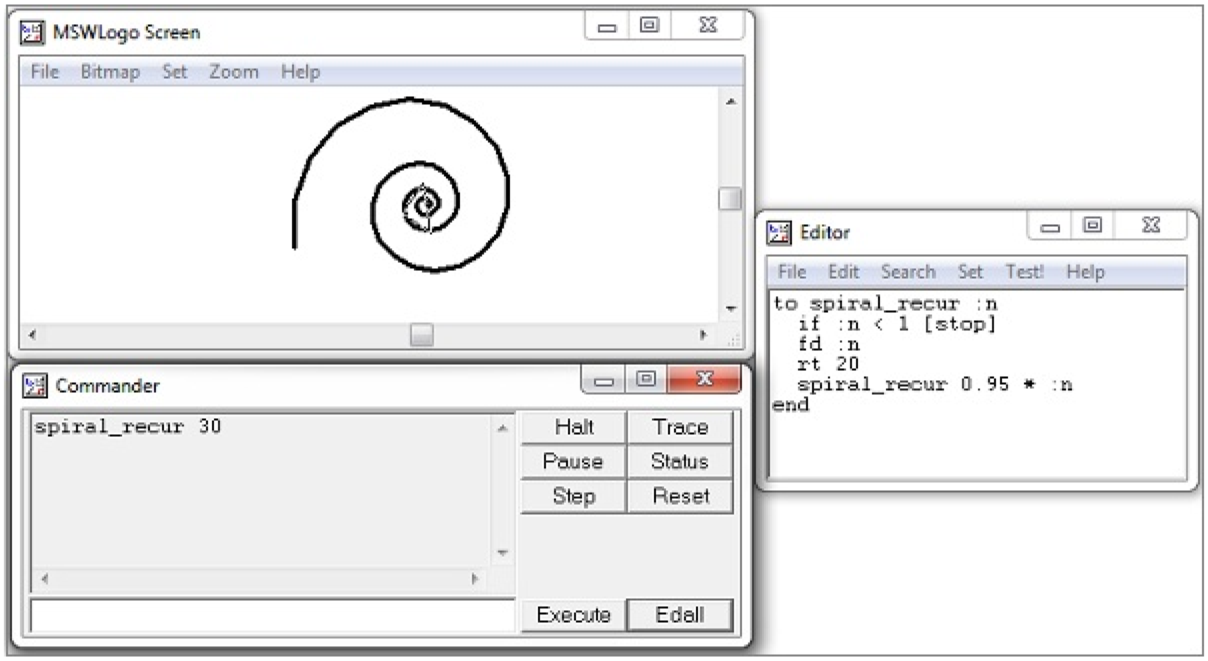
LANGUAGES

Languages like processing [cite] or libraries such as p5.js[cite] can produce mandala graphics. No doubt, the heavy workload should be handled by languages more suited to the task. Even Haskell animation libraries are built as thin wrappers on top of OpenGL(a C animation library).

Haskell is particularly well suited to representing these abstract patterns. Mapping its type system with simple shapes could lead to varying results. How could it be used to map Mandalas onto the existing structures of Tidal Cycles?

TURTLE GRAPHICS

Turtle Graphics are a simple graphics system. Typical commands include move forwards, move left, right and so on. Some of you might be familiar with it from Python's Turtle graphics library. For the old school ones Microsoft's Visual Logo might ring a bell.



```
spiral_recur 30
```

```
to spiral_recur :n
  if :n < 1 [stop]
  fd :n
  rt 20
  spiral_recur 0.95 * :n
end
```

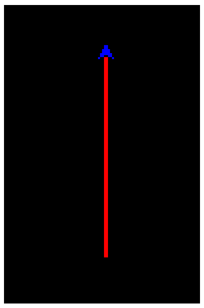
[MSWLogo Interface]

Turtle Graphics are useful in this case because, from fairly small lines of code, you start seeing strange patterns emerge. I explored animation libraries like Gloss, Reanimate and WorldTurtle. WorldTurtle seemed most suited to the task.

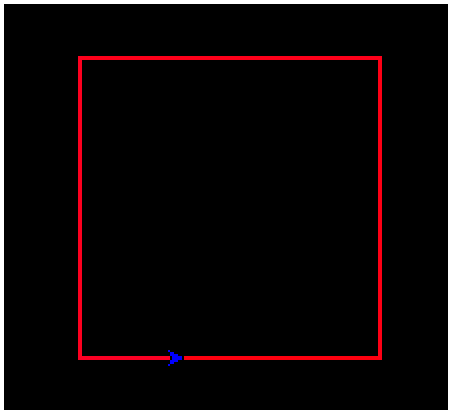
We integrated a basic Turtle Notation within Tidal's parser. The system needed to be portable to other libraries, so we created an intermediate notation. Tidal’s parser could now produce Turtle commands that animated the screen. Understanding the basics of monadic parsers [1] proved useful. Thus began the patterns.

Basic Patterns

“f” : pattern, moves forward with time



`“f r”`: Moves forward in the first half of the cycle, and then moves by 90 degrees in the second.

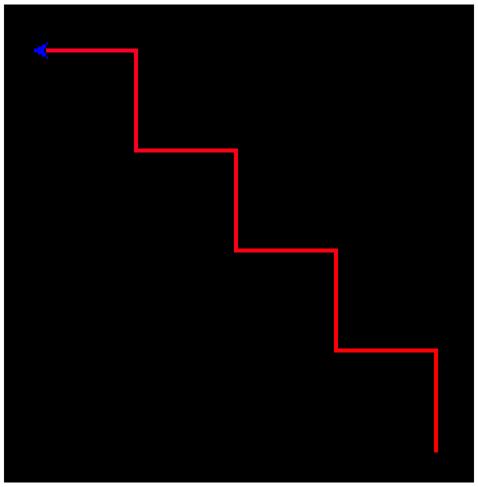


Mini Notation Magic

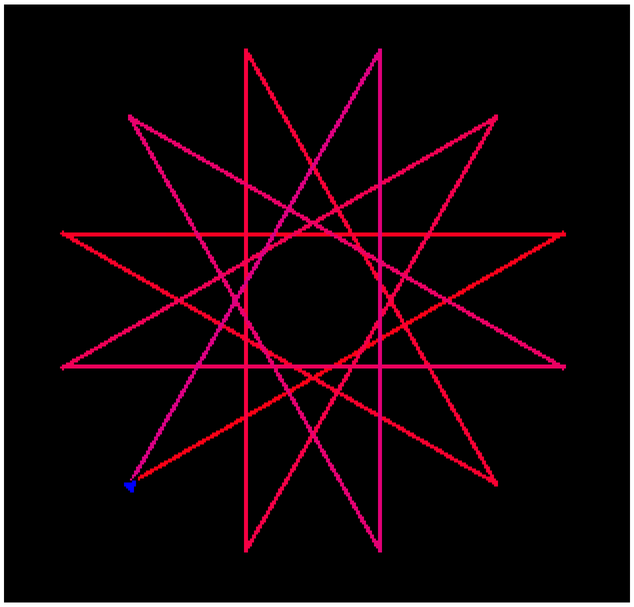
Tidal’s Mini-notation is used for writing patterns of various sorts (notes, samples, parameters).

Internally, the mini-notation is parsed as a *shortcut* for a function. You could otherwise write it using longer *function compositions*.

`“f <l r>”`: Alternatively moves left and right in each cycle’s second half.



`“f [f f l]”`: Starts forming Mandala like patterns.



But there was a problem.

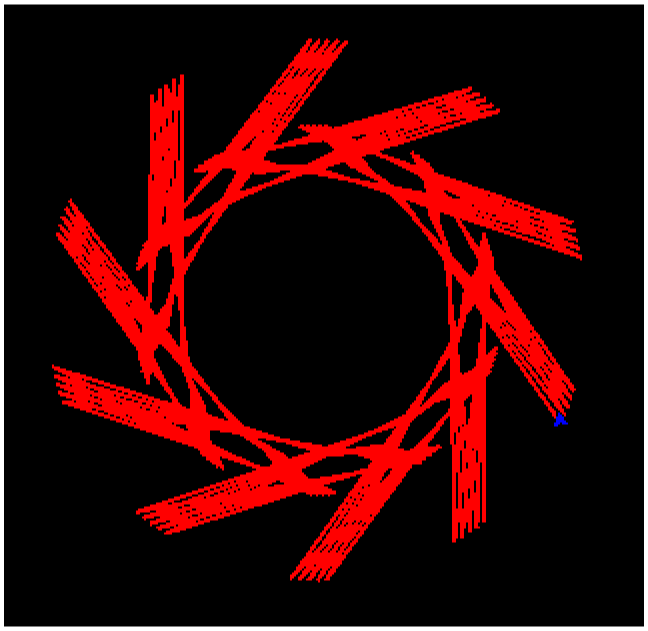
The system is not real time. WorldTurlte's API does not give low level access to the time at which the pattern is produced. This leads to graphics that are only theoretically in sync with music. Gloss, on top of which WorldTurtle is built, does provide access to time.

There was also the problem of changing patterns in real time. But, by storing patterns in mutable, shared variables, we could handle this with threads. This is a work in progress.

Meanwhile, Some more patterns:

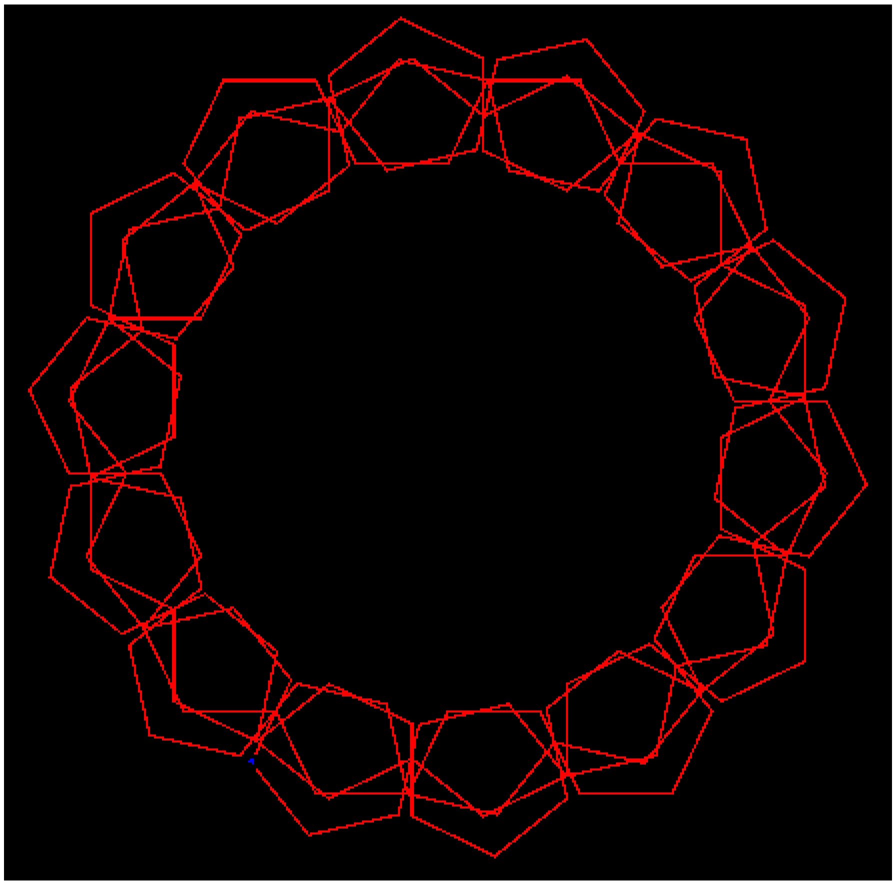
Ninja Star

`"f l l [f r r f l l f r r] f l l"`: Mininotation magic at hand, again.



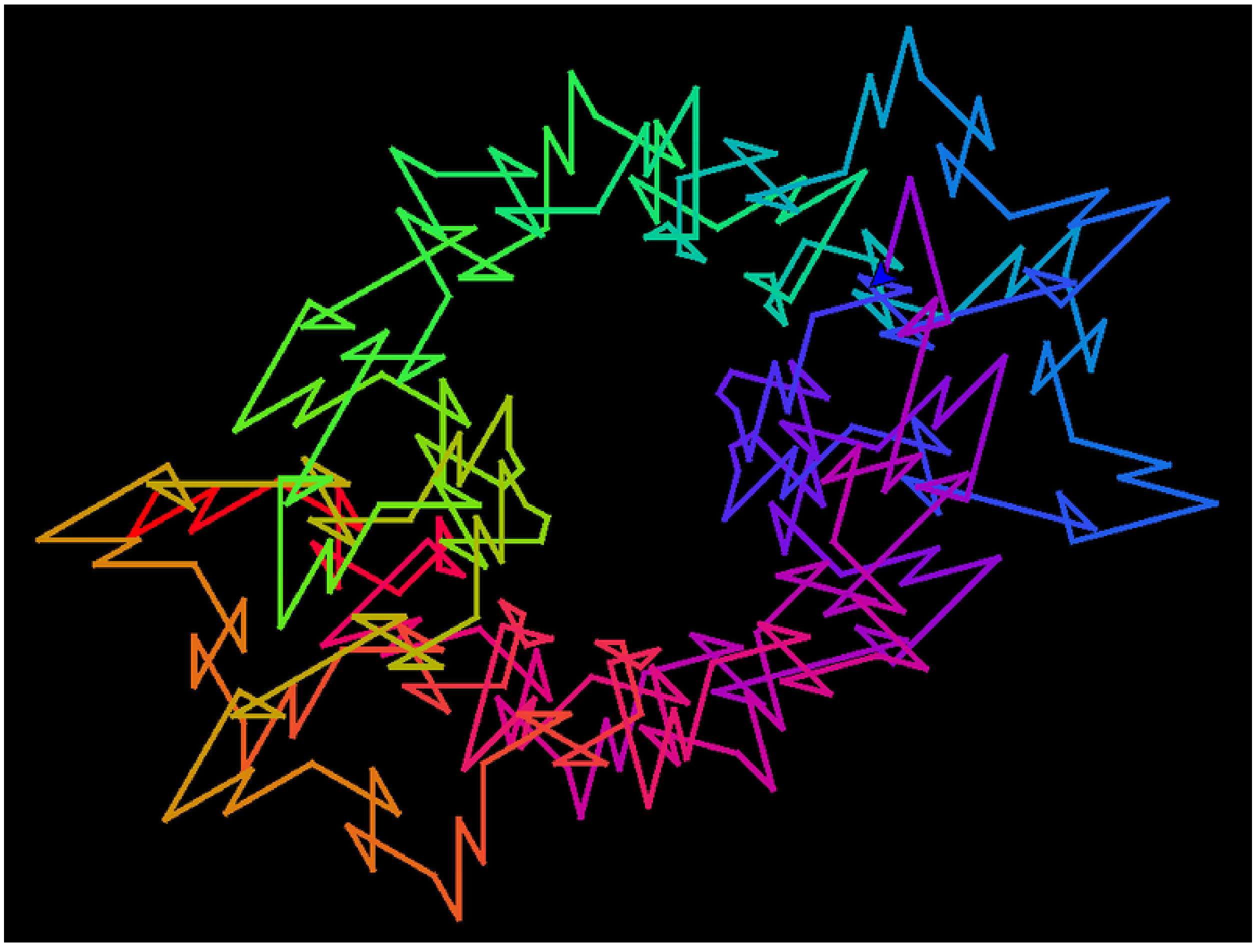
Honeycomb

```
"f <l r> f <r l r>"
```



Demonic

```
append (slowSqueeze "1 3 1" ("[l f, f r]")
```

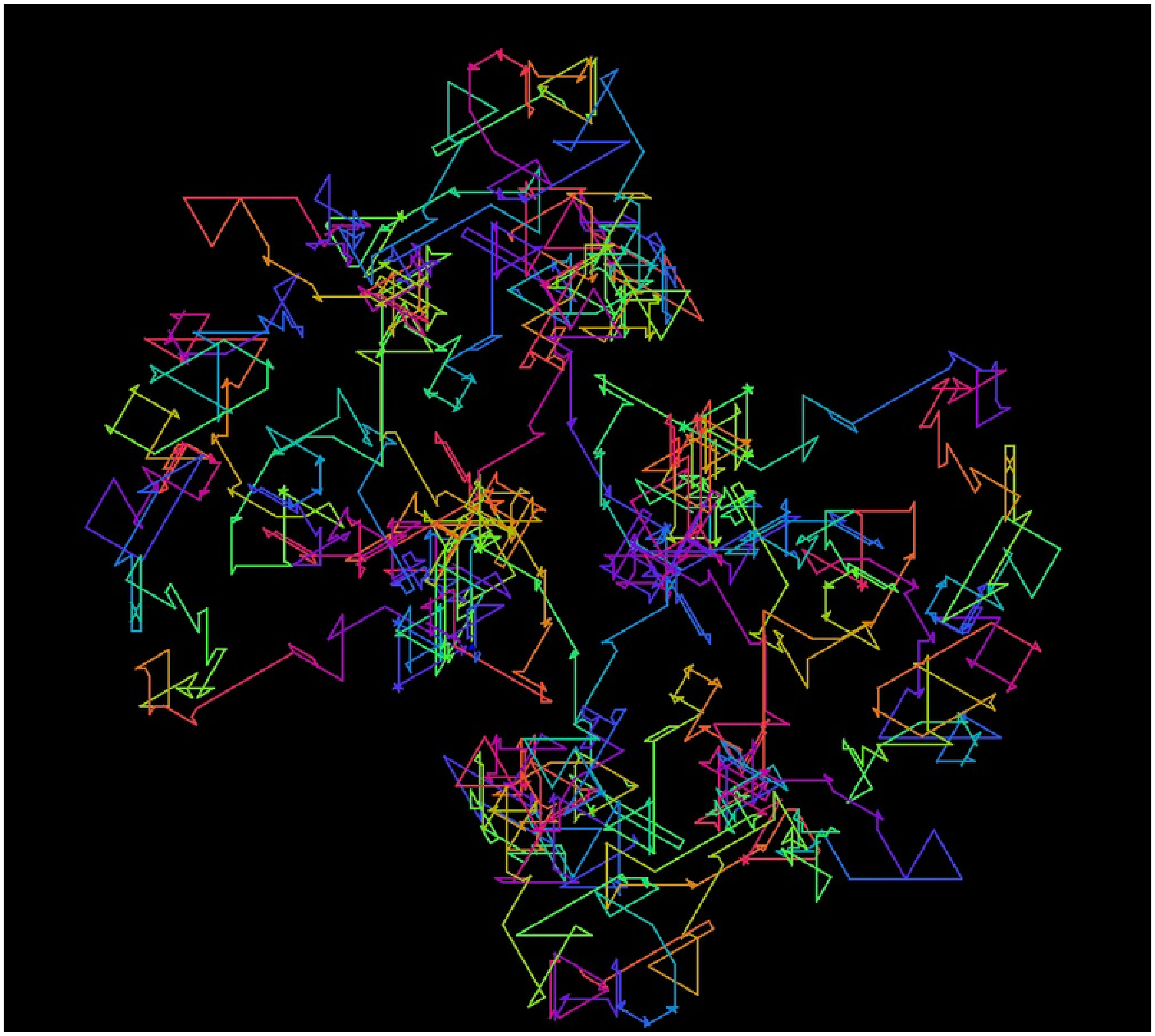


ChaosMap

`slow “1 1 2 3 5 8” “f l l”`:Everything is patternable.

At its base, `“f l l”` on its own produces a simple triangle.

This will slow down each cycle by the first pattern. Slow the first and second parts of the cycle by 1, third part by 2 and so on.



[A friend who likes physics said it looks like Brownian motion.]

You can find more patterns here.

ANIMATION AND TIME

This talk by Conal Elliot on Functional Reactive Animation[undefined] specifies what graphics systems do. They abstract the pixels away. Keep continuous space to work at a higher level. This allows for better composition too. You can scale and morph images without too much difficulty. The task is to use the same methods for time.

Regardless, some things I have learnt in this time: The rich variety of mandala art in many continents; L-systems[undefined] that produce tree-like structures using grammars; and music theory: tones, scales, chord progressions and their mathematical underpinnings.

There is the intuition we have that fields of knowledge are interlinked. That these patterns are present in many areas. But you can't work on intuition alone. So how do you confirm it?

Well, you can see it. As you watch the pattern form chaotic shapes on a screen, the connection is confirmed. These patterns still have a long way to go. An FFI could allow JavaScript libraries to produce the animations instead. The new version of Tidal could lead to a new world of possibilities.

However, the current system does show the structure of Tidal. The ChaosMap pattern, after going haywire in all directions, comes back to its original point. Seemingly random, until the very end when the pattern is visible. It showcases the underlying mathematical beauty at work. This was the central goal to accomplish.