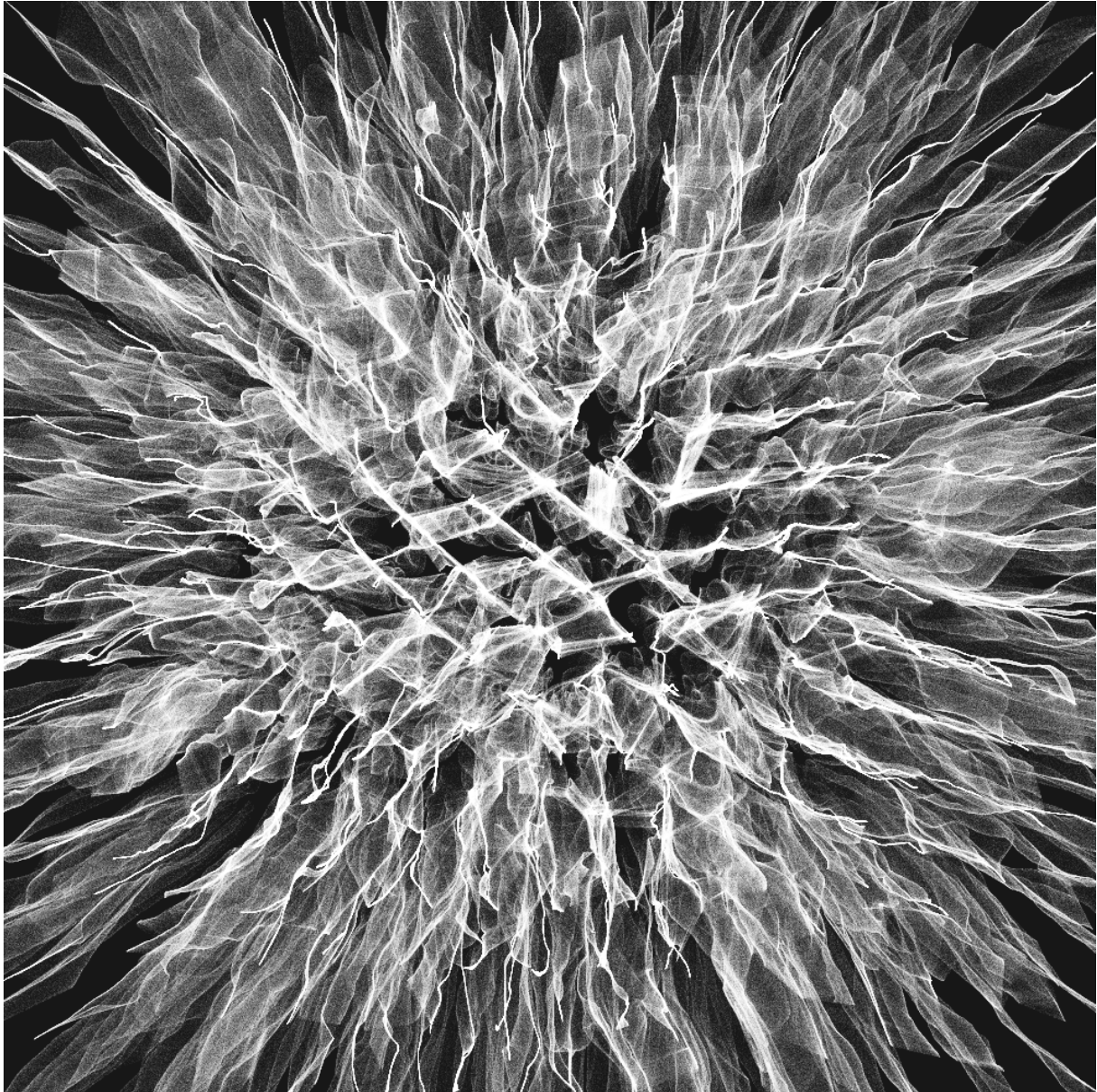

SNEK-A System for Making Generative Systems

About

This library is specifically written to be useful for a broad range of ways in which I create art using various generative algorithms.



In short [snek](#) is four things:

1. A simple (graph) data structure for working with vertices and edges The structure is named [snek](#); the name is explained below. This structure is combined with a programming pattern

About the Name

A while back someone on Twitter suggested that if Python 3 was named “snek” it would avoid naming confusion. I found that amusing at the time, and picked `snek` as the placeholder name for this project. I’ve been looking for a better name, but I haven’t found one yet.

Alterations

The pattern depends on the concept of `alterations`. In short: an `alteration` is a change that will be applied to the structure at the end of a given context. `alterations` are further described in <https://inconvergent.net/2017/snek-is-not-an-acronym/>.

I have also written about things related to `snek` at

- <https://inconvergent.net/2017/a-propensity-for-mistakes/> (indirectly about `snek`)
- <https://inconvergent.net/2017/a-method-for-mistakes/>
- <https://inconvergent.net/2017/arbitrary-alterations/>
- <https://inconvergent.net/2017/grains-of-sand/>

Here is an example of manipulating a `snek` instance called `snk` using `alterations`. Alteration constructors are postfixed with `?`.

```
1 ; context start
2 (snek:with (snk)
3   ; iterate vertices
4   (snek:itr-verts (snk v)
5     ; move alteration
6     (snek:move-vert? v (rnd:in-circ 1d0))
7     ; w will be an arbitrary
8     ; vertex in snk
9     (snek:with-rnd-vert (snk w)
10      ; join v and w if they are closer than d
11      (if (< (snek:edge-length snk v w) d)
12        ; join vertices alteration
13        (snek:add-edge? v w))))
14 ; context end
15 ; alterations have been applied
```

You can also manipulate the state directly. These functions are postfixed with `!`. Eg. `(snek:move-vert! ...)`.

Examples

There are some examples included. All examples are in the `examples` folder.

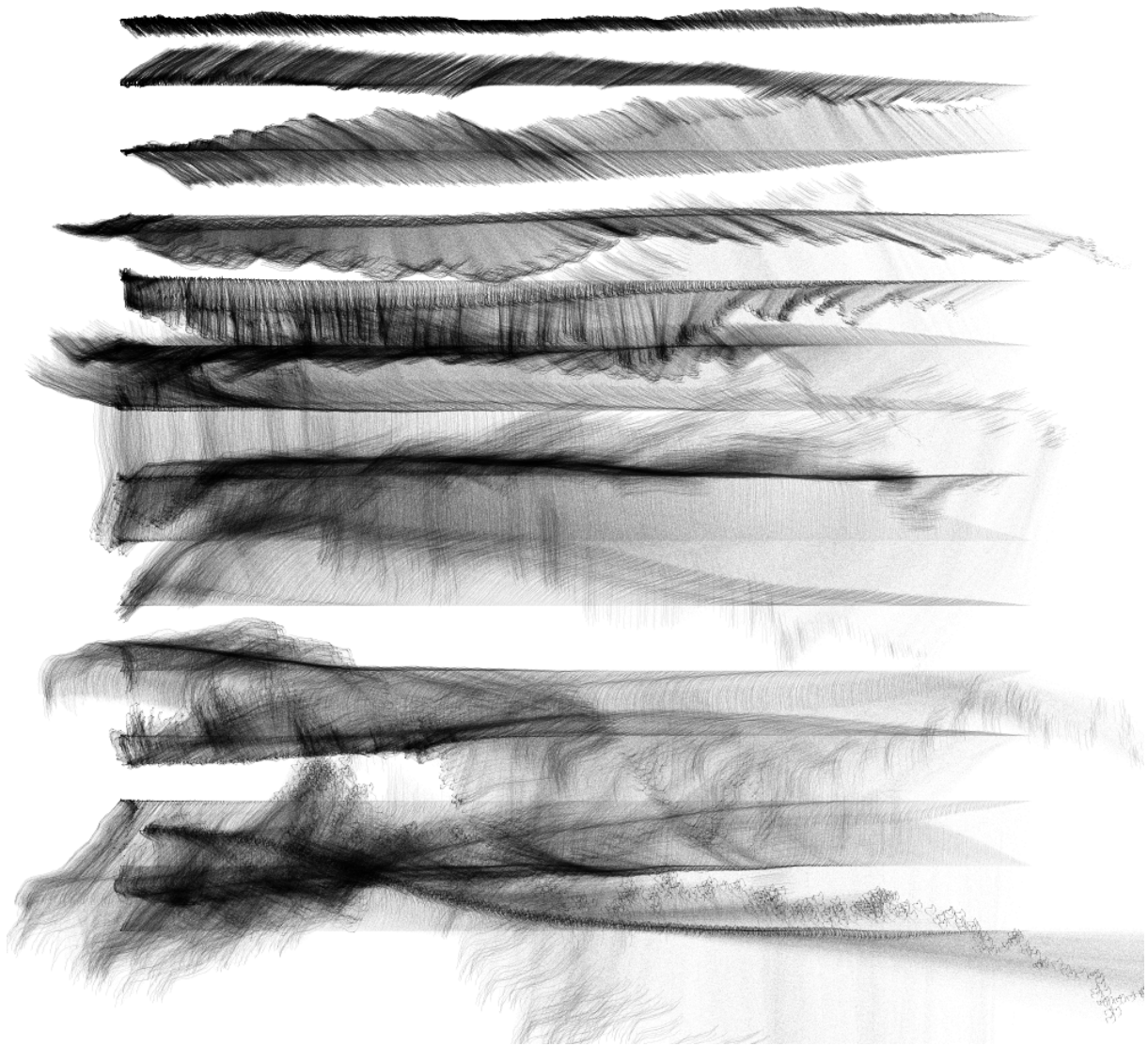
If you don't provide a filename (with full or relative path) as the first argument, the resulting file will be named `./tmp.png`.

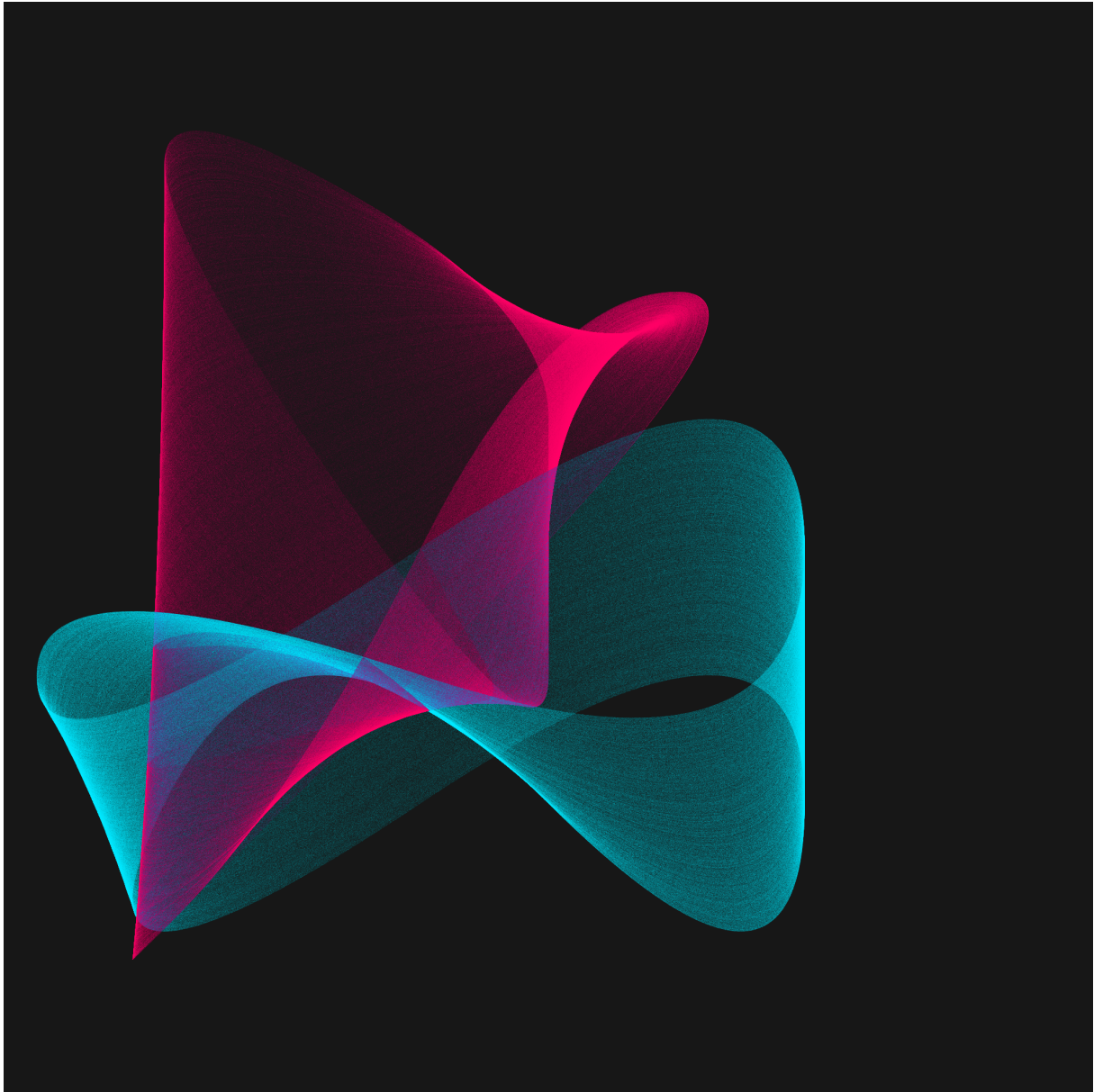
Custom alterations

You can define your own arbitrary alterations. There is an example of this in `ex/custom-alt.lisp`. I have also written about it here: <https://inconvergent.net/2017/arbitrary-alterations/>

Usage

I use snek for most of the work that I post online (<https://twitter.com/inconvergent>). Both for generating raster images as well as vector images for plotter drawings.





Here are some plotted examples:

- <https://inconvergent.net/2017/spline-script-plots/>
- <https://inconvergent.net/mechanical-plotter-drawings/>
- <https://inconvergent.net/mechanical-plotter-drawings/3/>
- <https://inconvergent.net/mechanical-plotter-drawings/5/>

Dependencies

This code requires `libpng-dev`, `Quicklisp`, `zpng`, `cl-svg` and `cl-png`. The path to quicklisp must be set in `src/load`. `zpng`, `cl-svg` and `cl-png` are automatically installed via `quicklisp`.

- <https://www.quicklisp.org/beta/>
- <http://www.xach.com/lisp/zpng/>

Tests

There are some tests included, see the `test` folder.

Stability, Changes and Versioning

This code is highly experimental on my part. It is likely to change with no warning or explanation. I will keep a note of the version number in `src/load.lisp`.

Thanks

I would like to thank:

- <https://twitter.com/RainerJoswig>
- <https://twitter.com/paulg>
- <https://twitter.com/jackrusher>

Who have provided me with useful hints and code feedback.