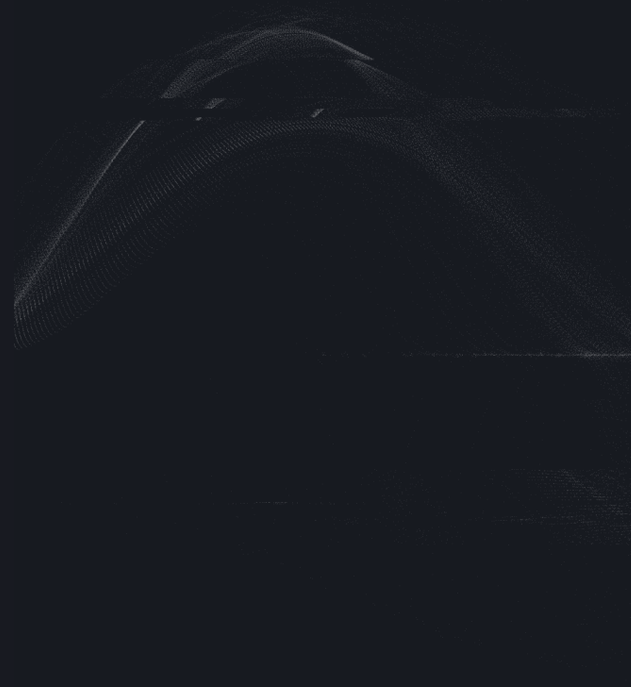
INSTITUTE OF SONOLOGY

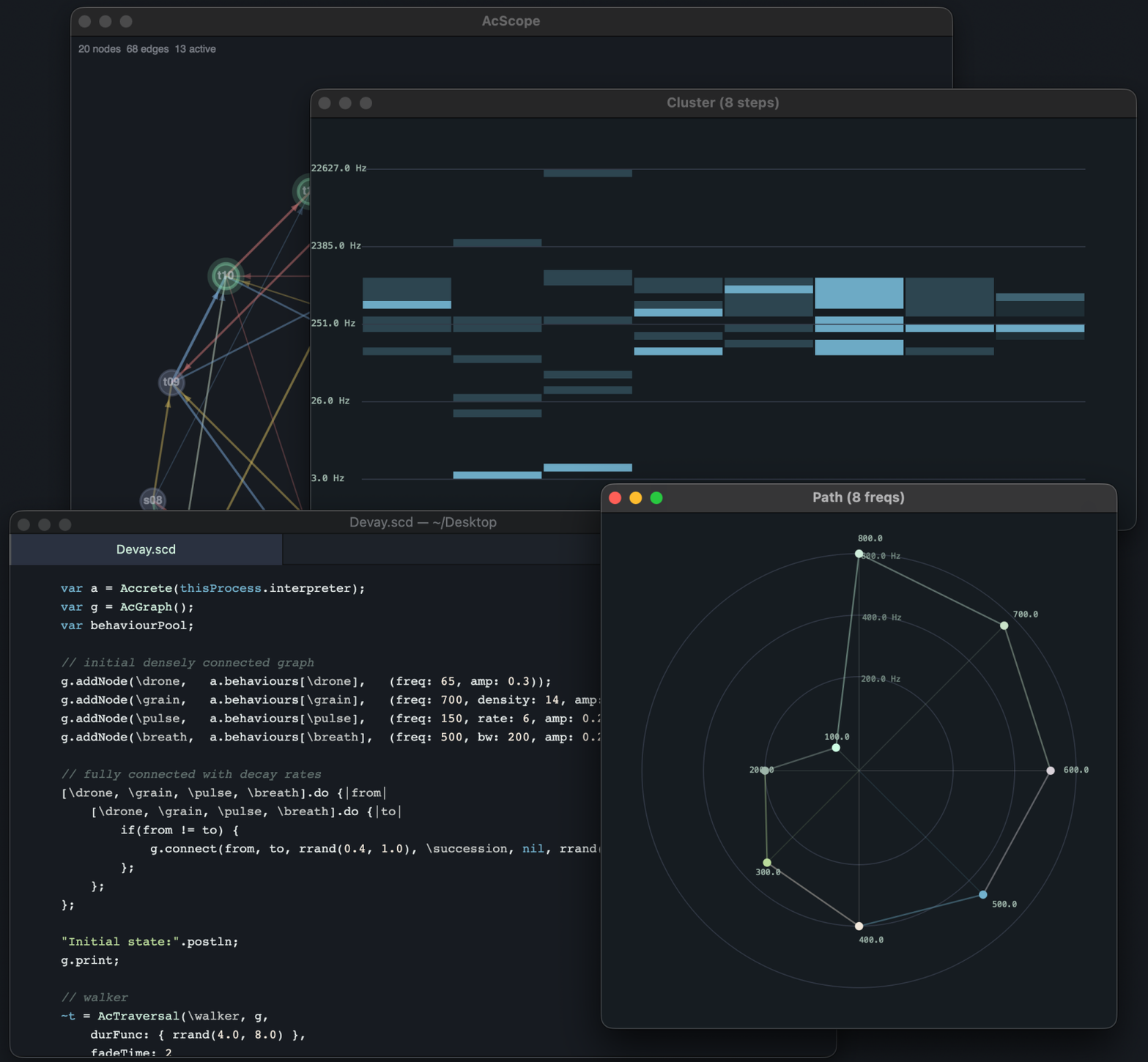
Royal Conservatoire The Hague

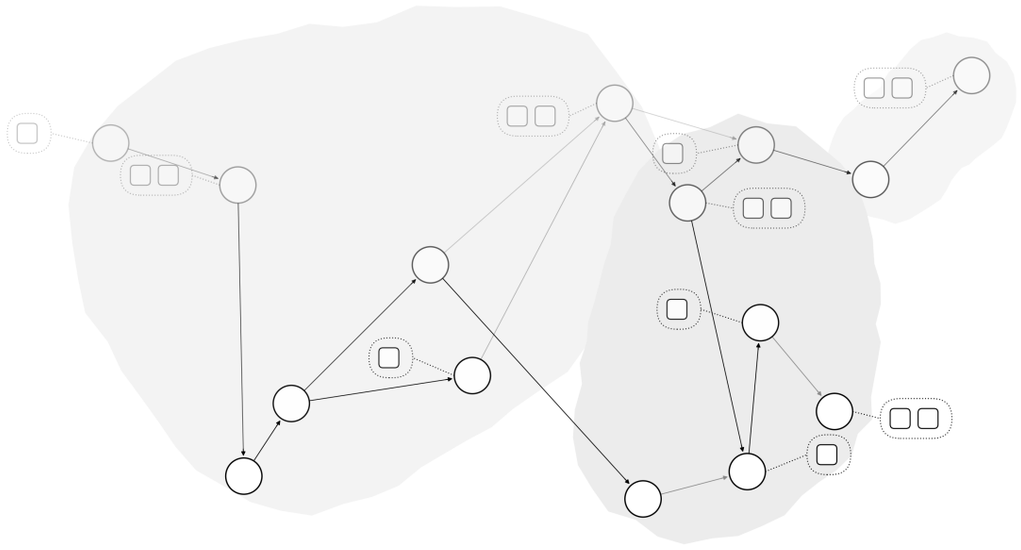


Tools, Sound and Structure

16.10.2026

http://www.bjarni-gunnarsson.net


AcScope
20 nodes 68 edges 13 active
Cluster (8 steps)
22627.0 Hz
2385.0 Hz
251.0 Hz
26.0 Hz
3.0 Hz
t10
t09
s08
Devay.scd — ~/Desktop
Devay.scd
var a = Accrete(thisProcess.interpreter);
var g = AcGraph();
var behaviourPool;
// initial densely connected graph
g.addNode(\drone, a.behaviours[\drone], (freq: 65, amp: 0.3));
g.addNode(\grain, a.behaviours[\grain], (freq: 700, density: 14, amp
g.addNode(\pulse, a.behaviours[\pulse], (freq: 150, rate: 6, amp: 0.
g.addNode(\breath, a.behaviours[\breath], (freq: 500, bw: 200, amp: 0.
// fully connected with decay rates
[\drone, \grain, \pulse, \breath].do {|from|
[\drone, \grain, \pulse, \breath].do {|to|
if(from != to) {
g.connect(from, to, rrand(0.4, 1.0), \succession, nil, rrand
};
};
};
"Initial state:".postln;
g.print;
// walker
~t = AcTraversal(\walker, g,
durFunc: { rrand(4.0, 8.0) },
fadeTime: 2
Path (8 freqs)
800.0
800.0 Hz
700.0
400.0 Hz
200.0 Hz
100.0
200.0
600.0
300.0
500.0
400.0



The rendering of statistical information into graphical form gives it a simplicity and legibility that hides every aspect of the original interpretative framework on which the statistical data were constructed. The graphical force conceals what the statistician knows very well — that no "data" pre-exist their parameterization. Data are capta, taken not given, constructed as an interpretation of the phenomenal world, not inherent in it.

Johanna Drucker. Humanities Approaches to Graphical Display. Digital Humanities Quarterly 5:1 (2011)

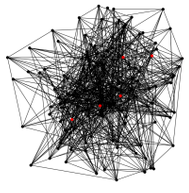
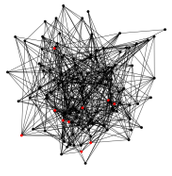
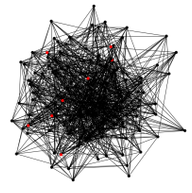
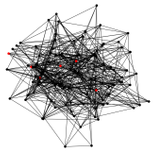
Selective Retention

- **Tools and Interfaces**
  - Tools as **interfaces** to existing material, enabling through **interaction**
  - The act of implementation becomes a **method of discovery** and clarification
  - Work unfolds through **feedback** between systems, sound, and perception

Emergent Intention

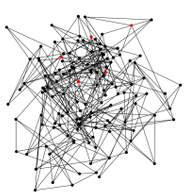
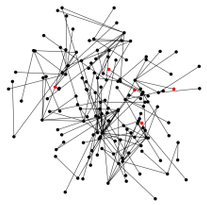
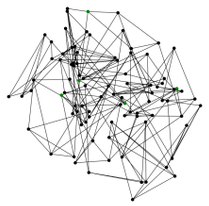
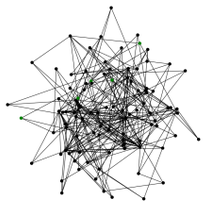
- **Intention** arises within the process rather than preceding it
  - Algorithms operate as **acting systems,** shaping both method and result

- **Relational Thinking**
  - Composition approached **bottom-up**, emphasizing **material interaction**
  - Focus on **local differences** and interdependencies rather than global design
  - Creative judgment and **intention emerge dynamically** through making

Relational Systems

- **Behaviour and Interaction**
  - Sound processes conceived as **behaviours** in continuous feedback loops
  - The composer becomes an **observer** of evolving algorithmic relations
  - Emphasis on **interaction** over control, where change drives form
- **Growth Models and ML**
  - **Graph-based structures** explored as compositional frameworks
  - **Preferential attachment** for clustering and emergent scales
  - Networks treated as **performative configurations**, not fixed designs
- **Causality**
  - Events arise through **connections** and **transitions** between nodes
  - **Causality** becomes **distributed** and temporal, shaped by sound itself
  - Intention as **relational**

Concerns

Convergence

- A practice can repeat itself through the means meant to open it
- Software biases outcomes: what is used gets repeated until it becomes a given

Generative Entrenchment

- Early technical choices gather dependents and become the hardest to change
- Model sits on top of material already made; material and its synthesis remain stable

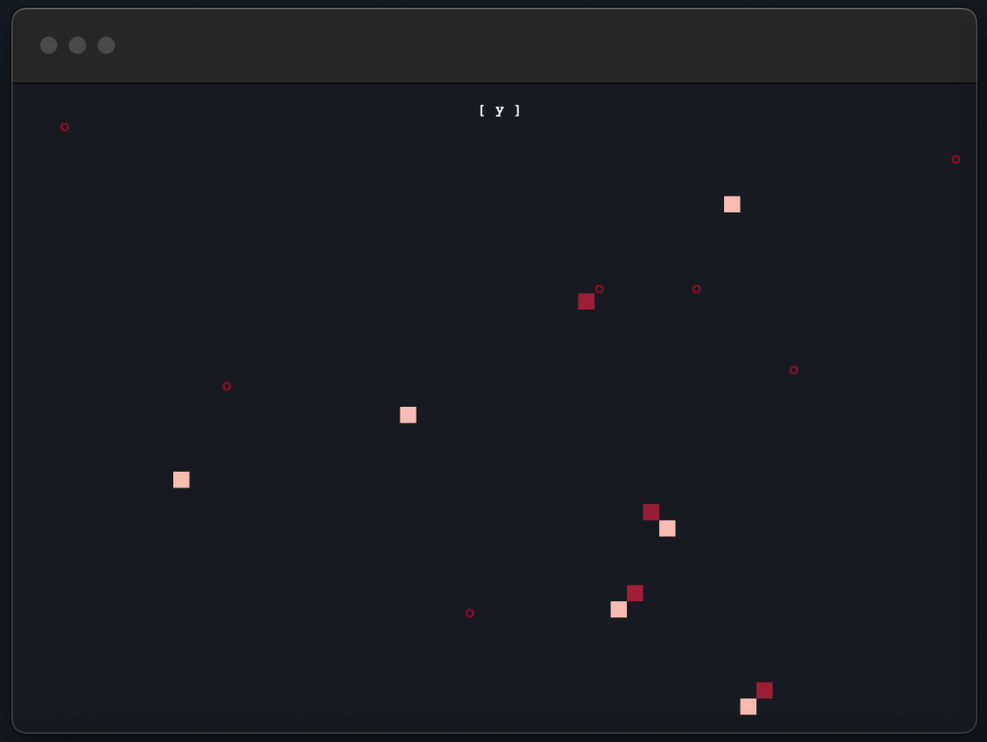
Systems and Scenarios

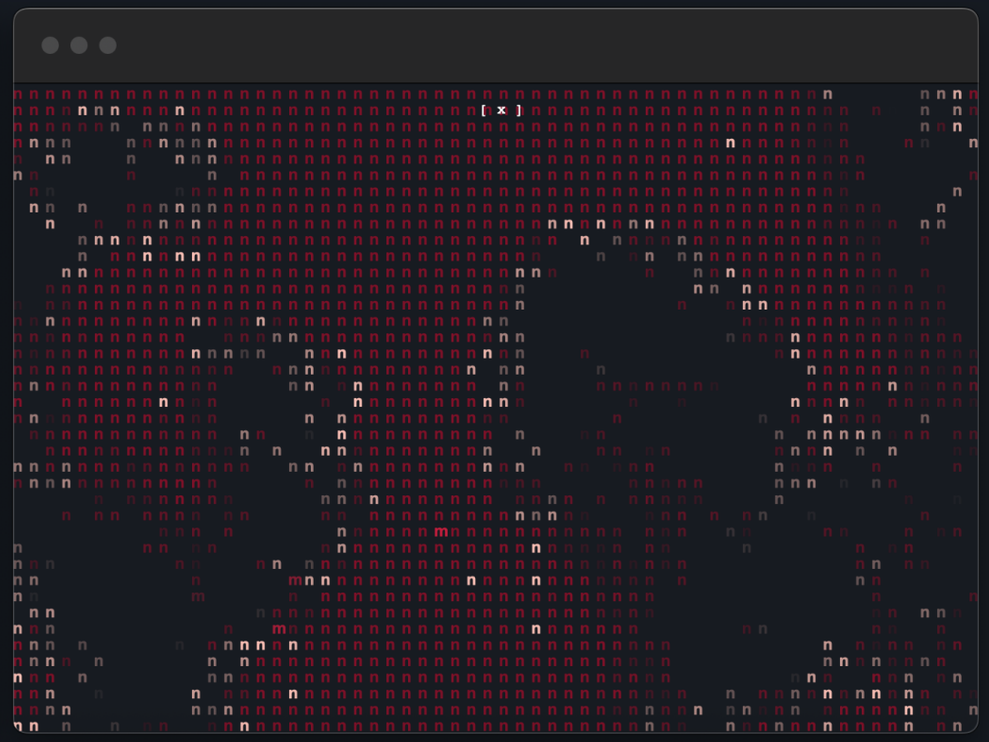
- Works conceived as systems, micro-worlds and processes to engage with
- Sounds chosen to fit the mechanism: continuous, steerable textures
- Music that could go on forever and could only stop

Questioning

- Keep projects experimental, speculative and reflective
- Analysis and learning of my own work as a route to original results

Blocking Behaviours





run2g.scd — ~/Works/Pieces/ICLC

__.scd

run2g.scd

```
NF(\swircon, {|fmod=20,ffrom=2000,fto=6000|
    var src = LPF.ar(Zircon.ar(), LFNoise1.ar(fmod).range(ffrom,fto));
    Ambspat.ar(8, src.sum, LFNoise1.ar(SinOsc.ar(0.01).range(0.01,0.1)).range(-1.0,1.0));
}).play;

)

~sim.bind(\swsc)

( // 07

~sim.focus("x", 15); ~sim.focus("y", 16);
~of.action.interrupt(\swsc); ~of.action.interrupt(\swosom); ~of.action.interrupt(\swircon);

NF(\brefm, {|freqMin=5,freqMax=10,mul=20,add=60,rate=0.5,modFreq=1100,index=0.3,amp=0.2,hpf=60|
    var freq = Demand.kr(Impulse.kr(rate), 0, Diwhite(freqMin, freqMax, inf).midicps);
    HPF.ar(PMOsc.ar(LFCub.kr([freq, freq/2], 0, mul, add), modFreq, index) * amp, hpf)
}).play;

NF(\brefoil, {|rate=2,trig=0.5,a=4,b=6,min=0.1,max=0.2,amp=0.3,freqmod=0.1,reps=4,hpf=20|
    var trigger, seq, freq, src, sd = SampleDur.ir;
    trigger = Impulse.kr(trig);
    seq = Dseq([
        Dbrown(rate * freqmod, rate, rate * freqmod, reps),
        Dwhite(rate * freqmod, rate, reps*2)], inf);
    freq = Demand.kr(trigger, 0, seq);
    src = HPF.ar(DemandEnvGen.ar(
        Dseq([min, min.neg, [max.neg, max]], inf),
        Dseq([sd * a, sd * b], inf)
    ) fold2(SinOsc.ar([freq, freq*0.51]), hpf, amp);
```

```
NF('breded')

NF('breded')

null                                    11:41:07.579


in file null

NF('cofoil')

NF('cofoil')

NF('cofoil')

an Interpreter
ICLC
## init brutes and bullies ##
MIDI Sources:
    MIDIEndPoint("IAC Driver", "BG.Virtual")
MIDI Destinations:
    MIDIEndPoint("IAC Driver", "BG.Virtual")
## loading complete

###
###
### - synthesis starts -
###
###
```

scd/runs/run2g.scd 313:53 LF ⚠ 1 deprecation UTF-8 SuperCollider create

Ever-Present Change

_.scd — ~

_.scd

```
(

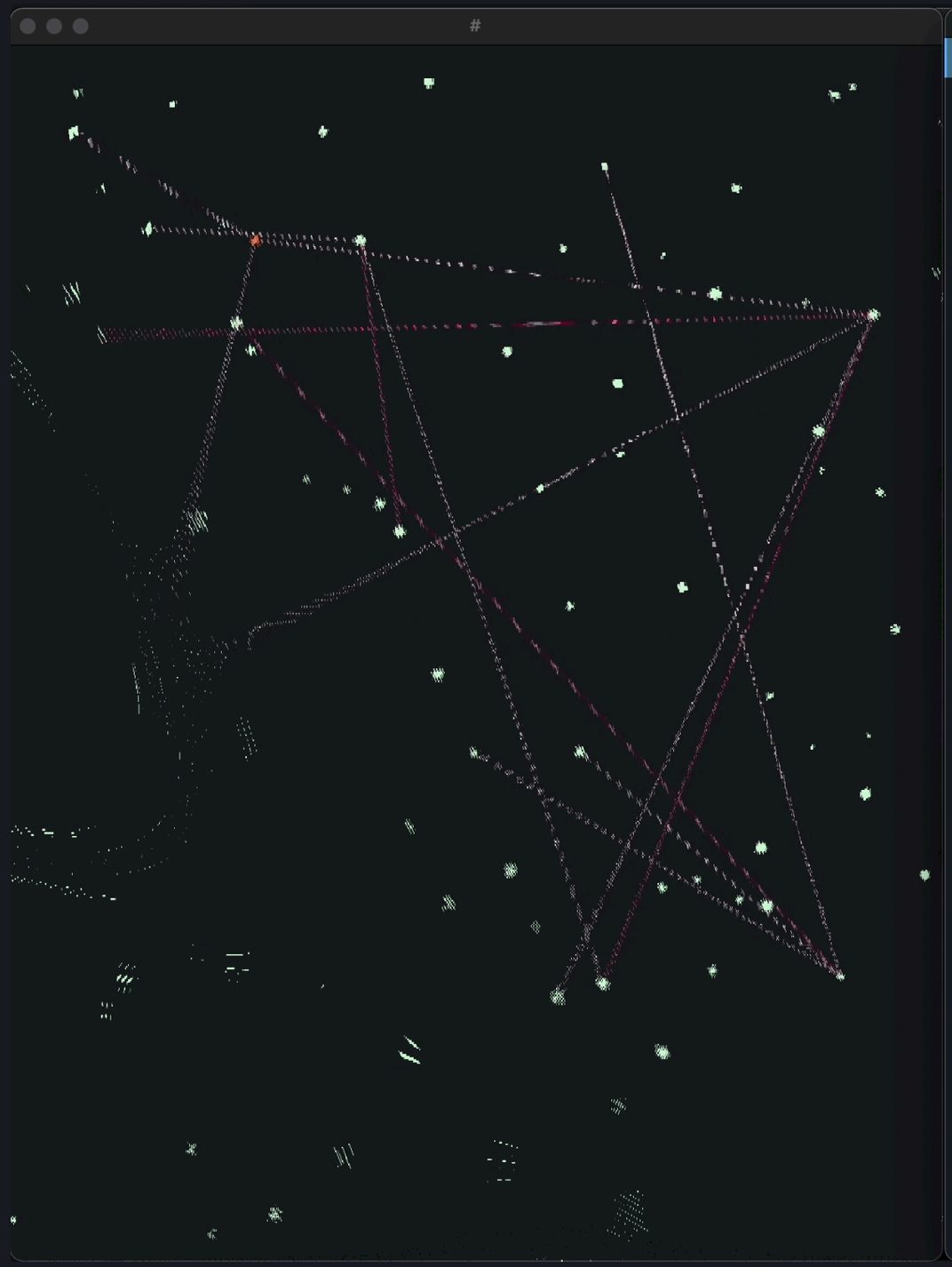
 NF(\pa, {
      |hifrom=4,hito=10,hirate=1,hifreq=200,hires=0.1,hiamp=250,
      lorate=20,
      lofreq=6000,lores=0.01,loamp=125|
      PanAz.ar(8,
      BBandPass.ar(Impulse.ar(
      LFNoise1.ar(hirate).range(hifrom,hito)) * 0.01, hifreq, hires, hiamp) +
      BBandPass.ar(Impulse.ar(lorate) * 0.1, lofreq, lores, loamp),
      LFNoise1.ar(Line.kr(4,10,25)).range(-1,1))
 }).play;

 )

a = EPCA.proc(\pa)

EPCA.attach
```

Streamlines



_.scd — ~

_.scd

```
Streamlines.behaviour(\stress);

Streamlines.behaviour(\drift);

Streamlines.interrupt([\slapoint,\slarolls,\slatacs,\slatics]);

(

 NF(\cranivite, {|rate=12,bfmin=400,bfmax=1000,amp=0.8,bw=12,freq=10,hpf=40|
     RLPF.ar(
         BPeakEQ.ar( HPF.ar(
         BBandStop.ar(LeastChange.ar(
             GravityGrid.ar(0, Array.fill(2, { freq * rrand(0.25,1.2) }) ),
             LoFM.ar(freq)), LFNoise1.ar(0.1).range(bfmin,bfmax), bw),
     hpf), 60, 2, -9) * amp
     , 8500, 0.75, 1.8)
     }).play;
)

(

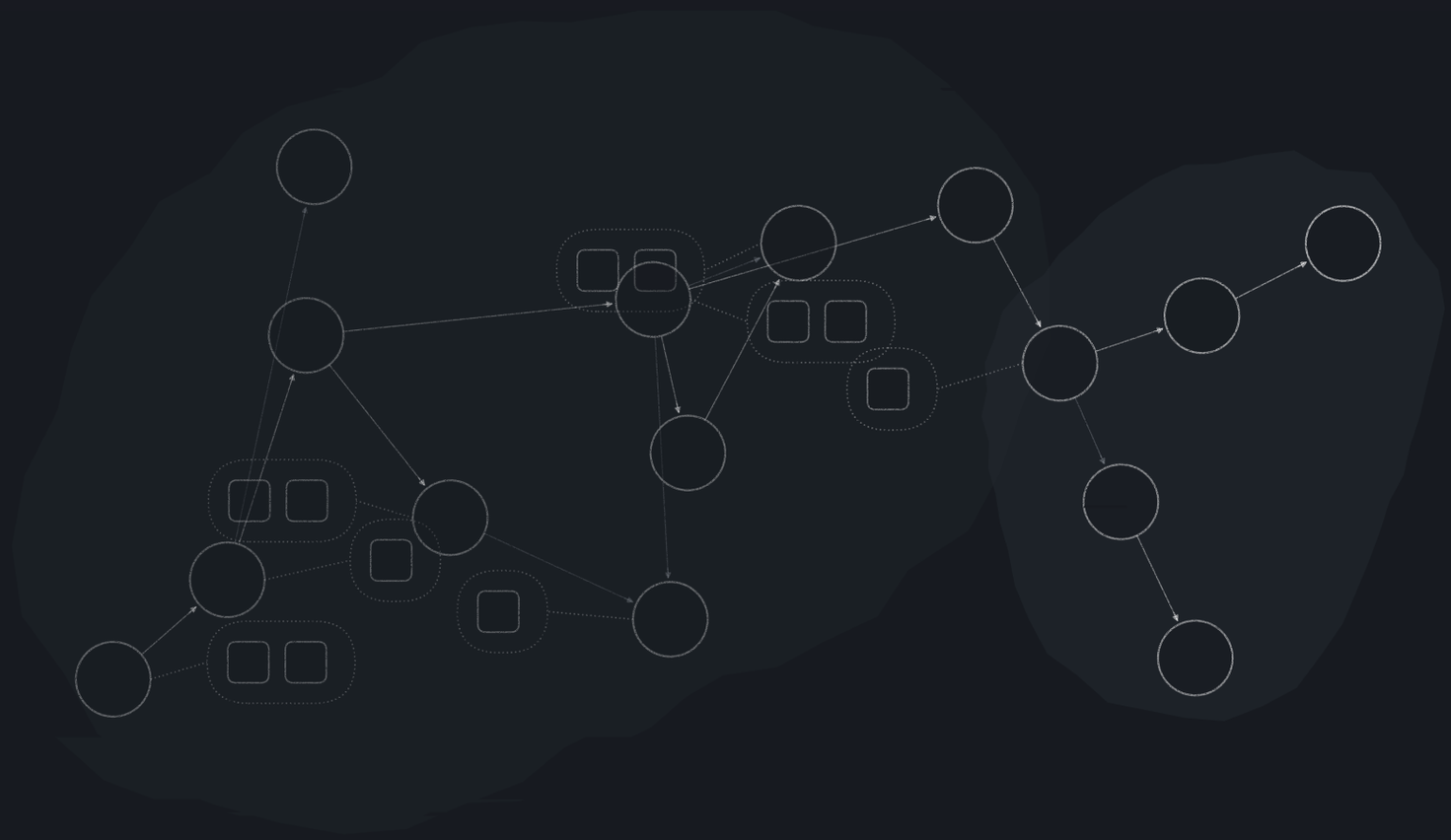
 NF(\crinop, {|modFreq=18, feedb=0.2, amp=0.1, ffrom=20, fto=40|
     SinOscFB.ar(LFNoise1.ar(0.01).range(ffrom, fto)*1.2, feedb).
 fold2(SinOsc.ar( modFreq / [1,1.2])) * amp
 }).play;

)

(

 NF(\crjonvmar, {|fspeed=10,fcenter=4000,wfmult=1,filtbw=12,lpf=1500,amp=0.3|
     var a = 1, b = 35, min=0.01, max=0.8;
     var sd = SampleDur.ir * wfmult;
     var size = 3, from = 1, to = 15;
     var mfreq = 10, mfrom = 1, mto = 28;
     var snd = DemandEnvGen.ar(
         Dseq([min, min.neg, [max.neg, max]], inf),
         Dbrown(sd * a, sd * b, sd*2);
     ).fold2(SinOsc.ar([0.01, 0.012])) * amp;
     snd = LPF.ar(snd, lpf);
     snd = snd.mod(SinOsc.ar(LFNoise0.ar(mfreq).range(mfrom, mto)));
     snd = HPF.ar(BBandStop.ar(snd,
     Lag.ar(LFNoise1.ar( [fspeed,fspeed*2]).range(fcenter * 0.2, fcenter * 2), 0.05), filtbw)).sum;
     2.collect{|i| BBandPass.ar(snd, 150 * (i + rrand(1,10)), 0.4, SinOsc.ar(0.1.rand).range(0.0,3.
 }).play;

)
```



Accrete

Accrete Public

Pin | Watch 0 | Fork 0 | Star 0

main | 1 Branch | 0 Tags | Go to file t | Add file | Code

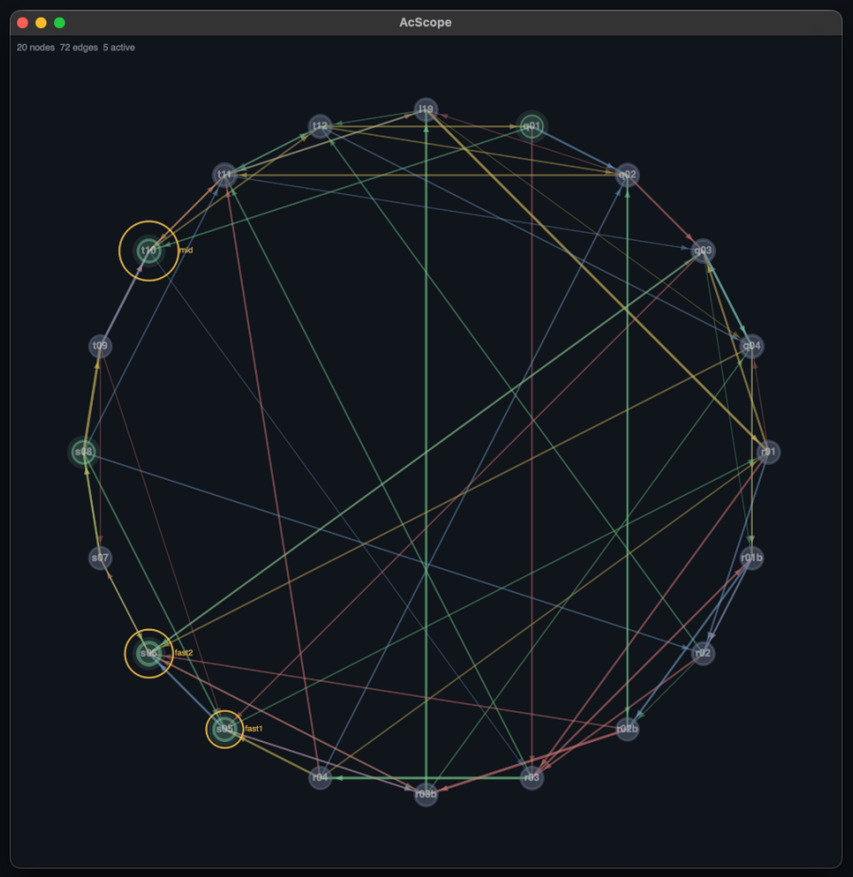
bjarnig updates — 2c1ec16 · 6 hours ago — 18 Commits

classes	process	18 hours ago
docs	additions	18 hours ago
examples	process	18 hours ago
src	updates	18 hours ago
.gitignore	latest	18 hours ago
README.md	mix	16 hours ago

README

Accrete

A graph-based system for SuperCollider.



About

Graph-based compositional engine for SuperCollider

Readme
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 1

bjarnig Bjarni Gunnarsson

Languages

SuperCollider 100.0%

https://github.com/bjarnig/Accrete

Accrete.scd

```
    Averse Network.
    Buffer-playback patterns from Averse converted into network behaviours.
    The walker moves through different event.
    Periodic collapses reshape the network.

*/

(
s.waitForBoot {
    var bufferPath = "/Users/bjarni/Library/Application Support/SuperCollider/Extensions/Accrete/examples/per
    var a, g;

    a = Accrete(thisProcess.interpreter);

    fork {
        a.loadAverse(thisProcess.interpreter, bufferPath);
        s.sync;

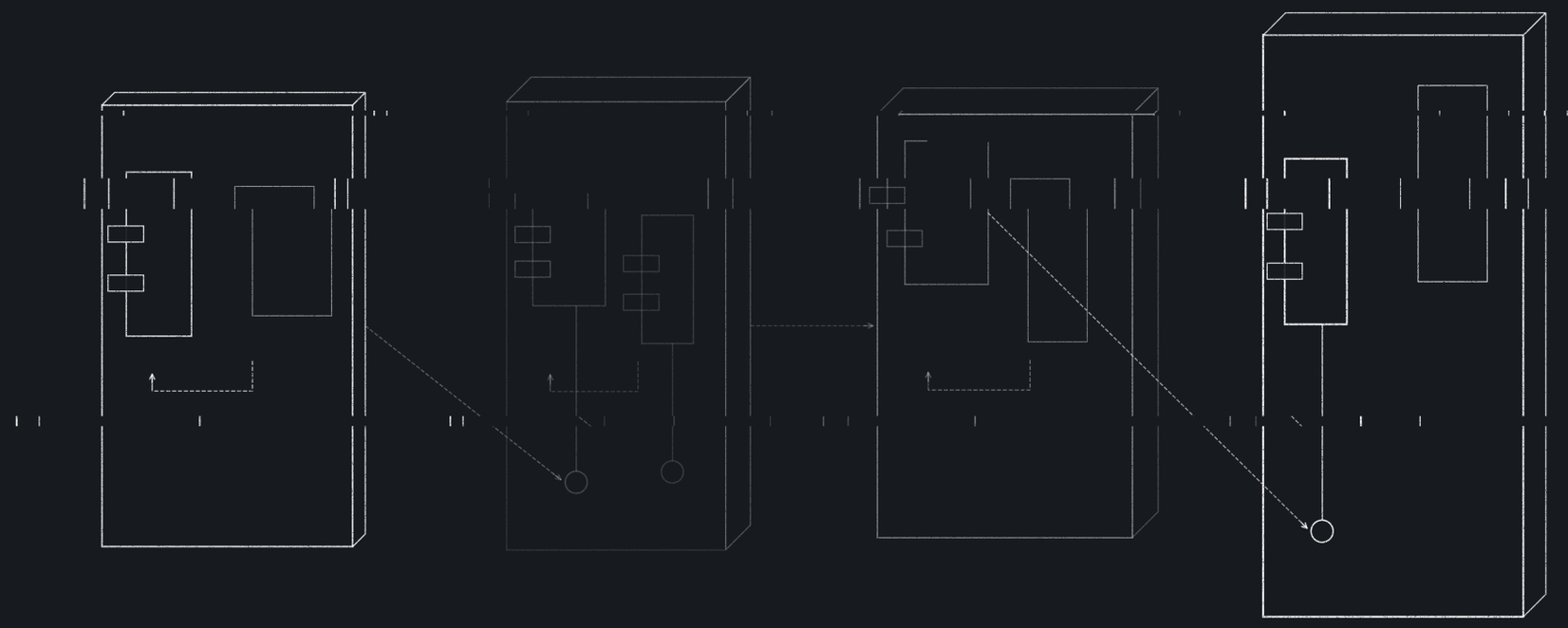
        g = AcGraph(2, 0);
        g.initRouting(\averse);

        g.addNodes(a.averseNames.collect {|name| [name, a.averse[name]] });

        AcGrowth.ring(g, a.averseNames, decayRate: rrand(0.01, 0.04));
        AcGrowth.crossLinks(g, 2, decayRate: rrand(0.02, 0.06));

        ~t1 = AcTraversal(\fast1, g,
            durFunc: { rrand(1.0, 3.0) }, fadeTime: 1, contrastBias: 0.7);
        ~t2 = AcTraversal(\fast2, g,
            durFunc: { rrand(1.0, 2.5) }, fadeTime: 1, contrastBias: 0.6,
            deactivateOnLeave: false);
        ~t3 = AcTraversal(\mid, g,
            durFunc: { rrand(1.5, 3.0) }, fadeTime: 1.5,
            deactivateOnLeave: false);
        ~t1.walk(\r01);
        ~t2.walk(\s08);
        ~t3.walk(\q01);

        ~evo = AcGrowth.evolve(g);
```



Ora

ORA

Ora takes its name from the Latin ōra, meaning border, rim, frontier, or edge. The piece unfolds along such boundaries, between sound and process, between tone and texture. All musical material is derived from a small set of sine waves that act as both source and constraint. These sets are stretched, multiplied, and transformed by algorithms, generating textures that continuously reorient themselves. Ora is part of an ongoing project exploring process-based composition through generative design, emphasizing continuity and sustained temporal processes.

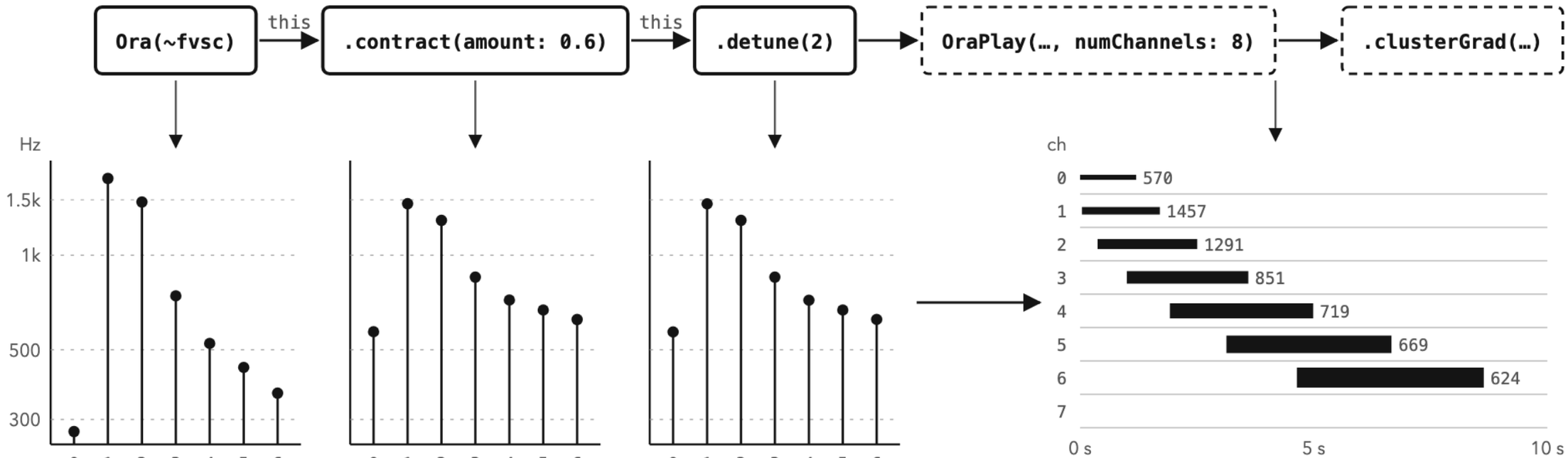
ORA

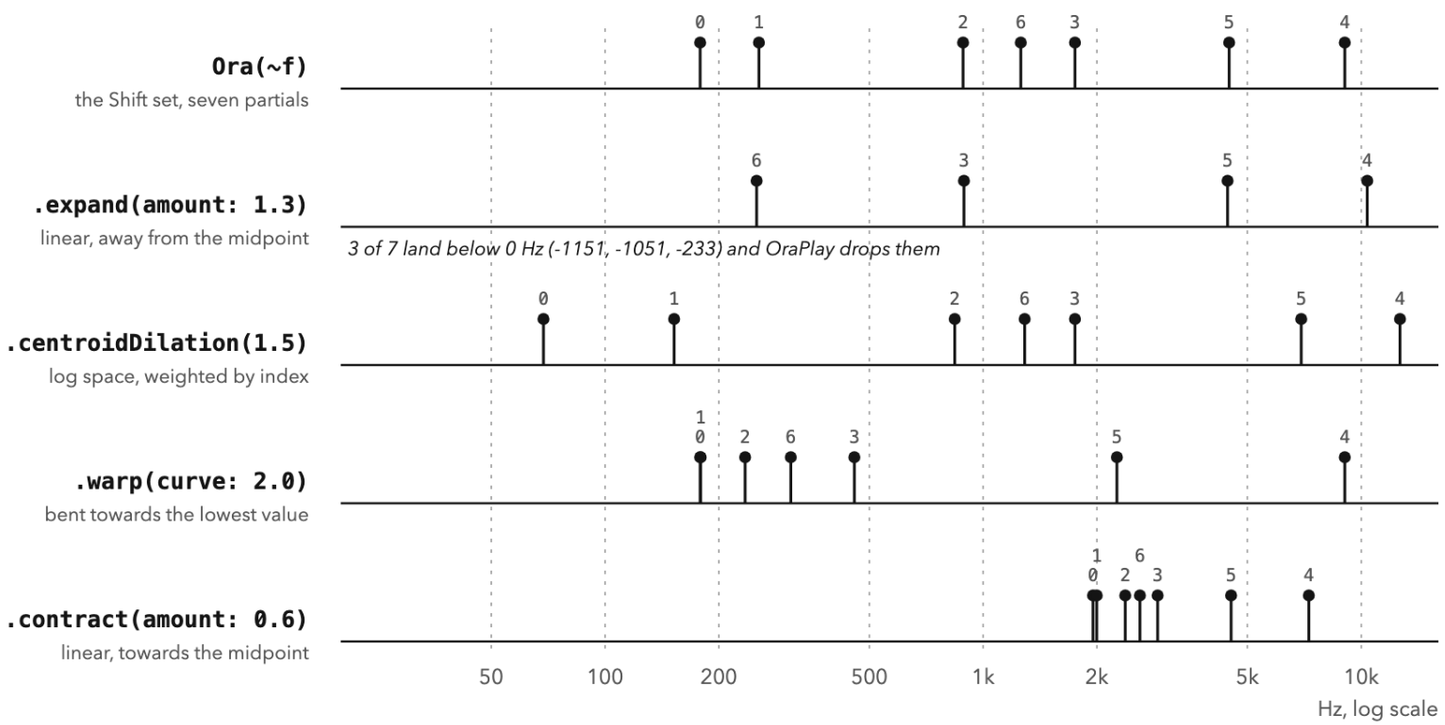
- Fixed media removes the live habits of switching, interrupting and holding
- Serial and stochastic operations lifted from the sample to the frequency-set
- The array of frequencies is the unit that operations compose on
- Material avoided until now: stable pitches, chords, arpeggios, permutations
- By accident: the electronic sound highlighted through changing pitches
- Every transformation returns this: the method chain is the compositional artefact
- Generation and transformation composed as one sequence
- Sets sounded through bandpass filters and resonators
- Plotting follows the development of the material inside the working loop
- Snapshot and feedback: a state that sounds right seeds the next chain
- A spatially distributed space with its own causality



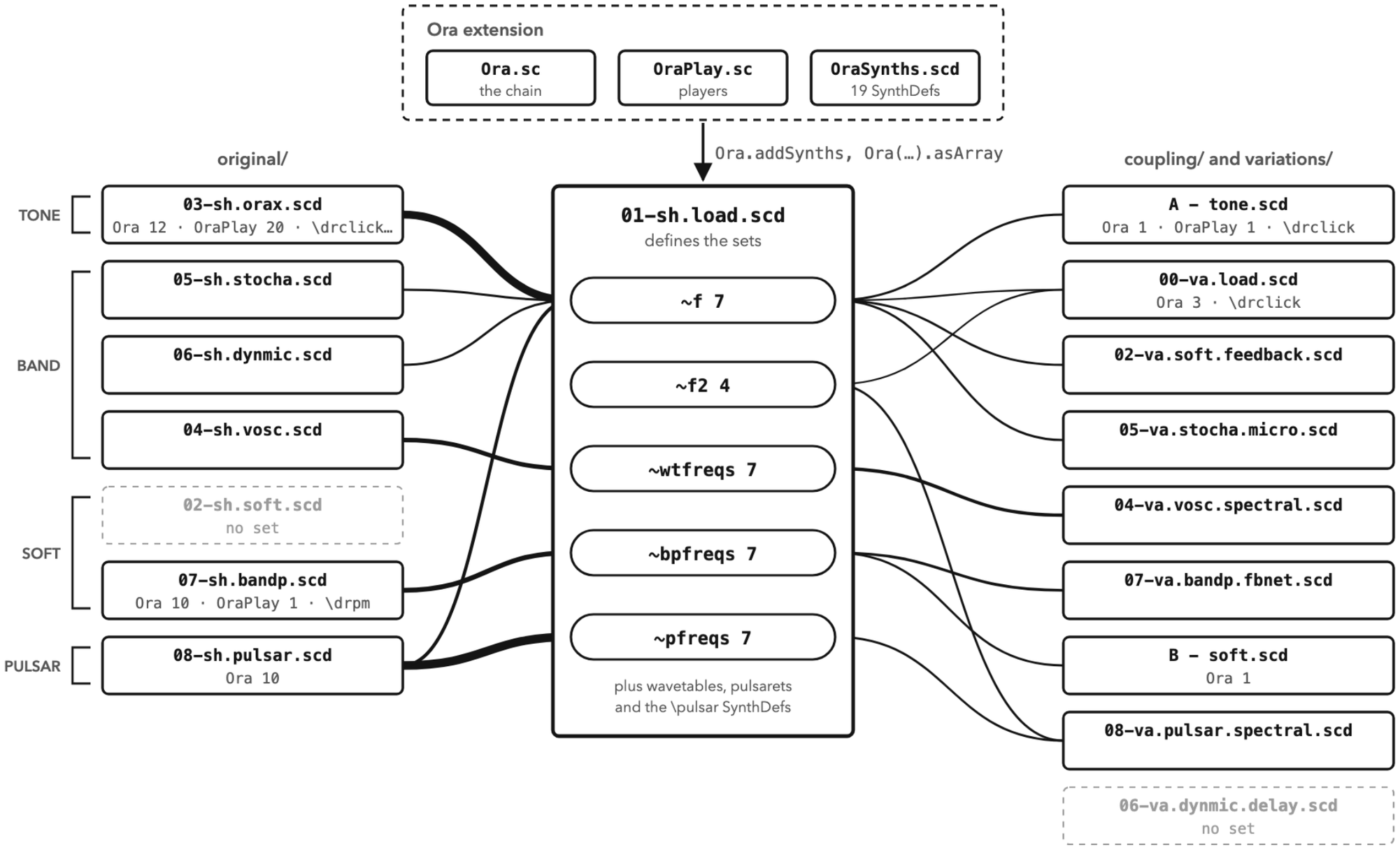
Ora - Transformations

each transformation returns the object, so a line of code is a chain of operations on one set





Ora - Couplings



- Coupling of electronic sound through the frequency arrays
- Different sets, their coupling and transformations as relational

ORA.scd — ~/_

ORA.scd

~/_/ORA.scd* 2:1 LF 1 deprecation UTF-8 SuperCollider 1 update

Ora Public

Unpin | Watch 0 | Fork 0 | Star 4

main | 1 Branch | 0 Tags | Go to file t | Add file | Code

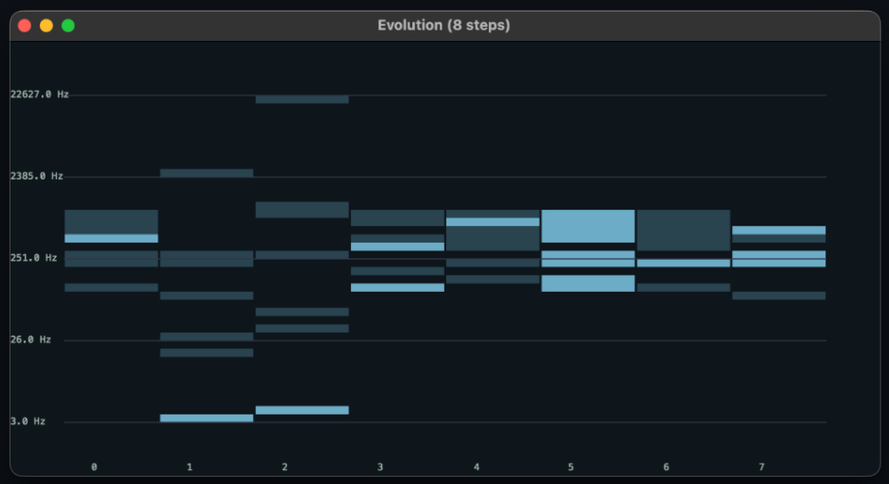
bjarnig guide 1cedd8c · 7 minutes ago 25 Commits

classes	edits	18 hours ago
docs	guide	7 minutes ago
piece	alter	17 hours ago
scd	guide	7 minutes ago
README.md	guide	7 minutes ago

README

Ora

A set of processes for generating, transforming, and sequencing frequency arrays.



Overview

- **Ora** - Transform frequency arrays using serial techniques and parametric operations
- **OraGen** - Generate dissonant and inharmonic pitch sets
- **OraPlay** - Play frequency arrays with diverse textural patterns
- **OraSeq** - Manage and visualize sequences of transformed frequency sets
- **OraRatios** - Build and manipulate frequency arrays from ratio relationships
- **OraSpear** - Import and extract frequencies from SPEAR analysis files

About

A set of processes for generating, transforming, and sequencing frequency arrays.

Readme
Activity
4 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 1

bjarnig Bjarni Gunnarsson

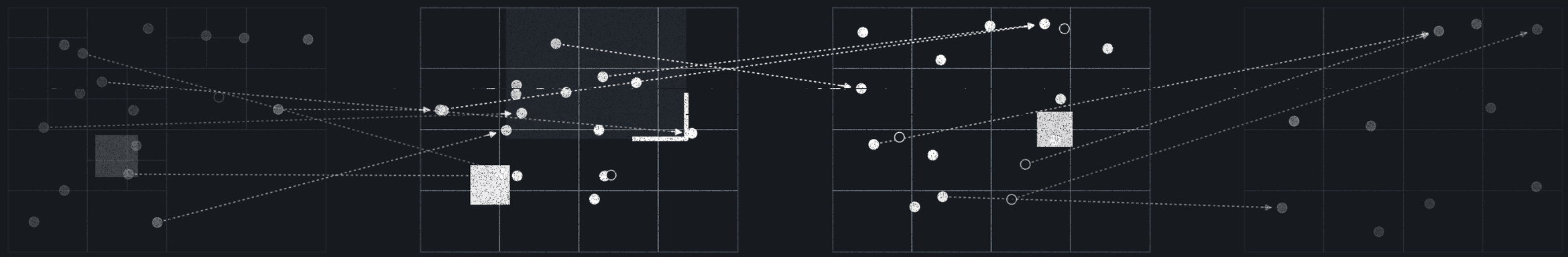
Languages

SuperCollider 100.0%

https://github.com/bjarnig/Ora



Ora



Genesis

Genesis

Genesis explores a set of sound synthesis procedures built from stochastic, chaotic and network-based functions. Each generator is a short program that produces continuously varying waveforms and places them within a predefined proportional structure. Koenig described electronically produced sound as "both a form-section and an acoustico-musical unit, a quantum", highlighting its role as both the part and the whole. The piece considers the continuity this idea implies and the tensions it introduces. It reflects on the scope of decisions, selections, and rules, and how these interact through the processes that make up the piece.

1 Structure

The strutural sections are constructed from a single unit through the ratios 4:5:6:7:9:11:13 taken in the order below.

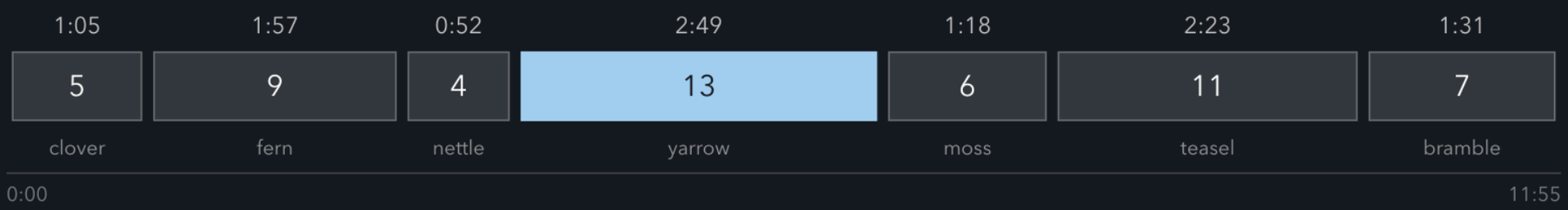
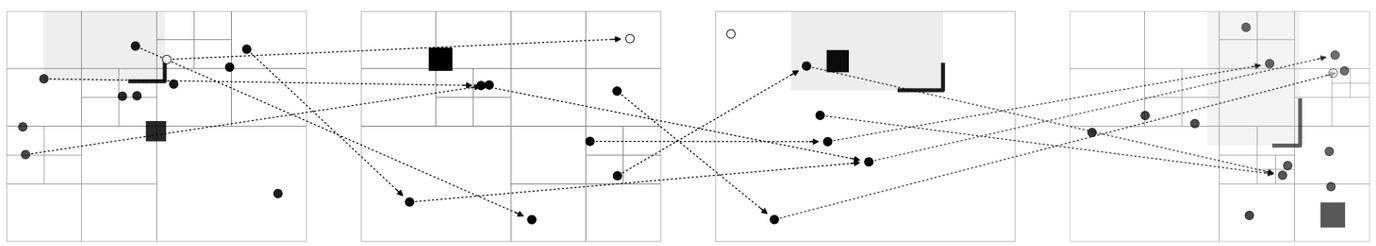


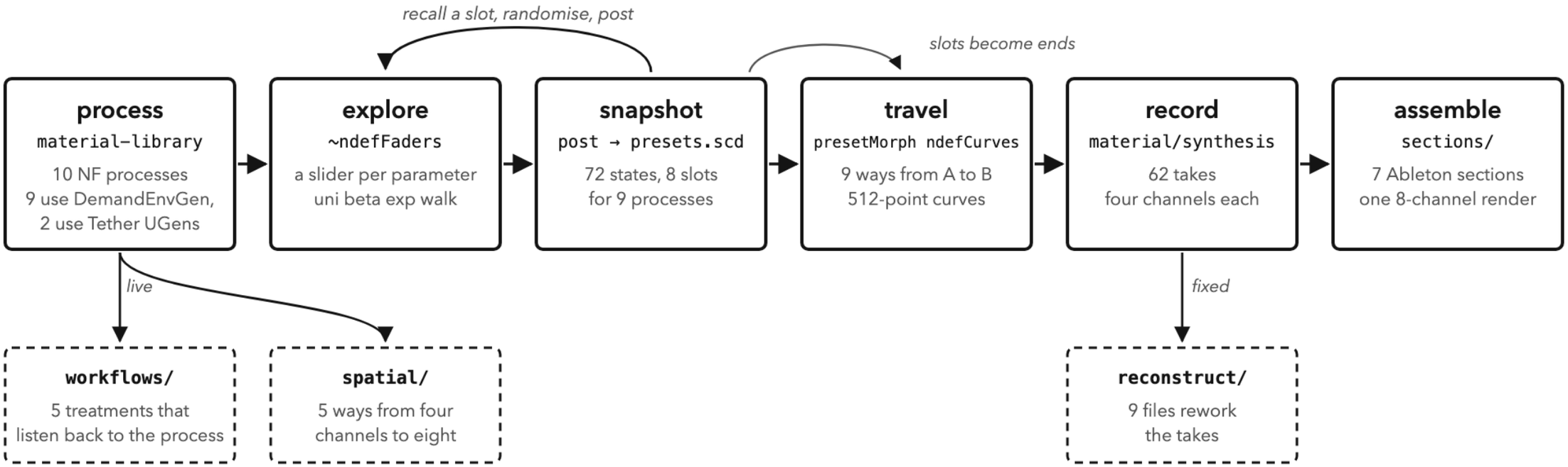
Fig. 1

Genesis

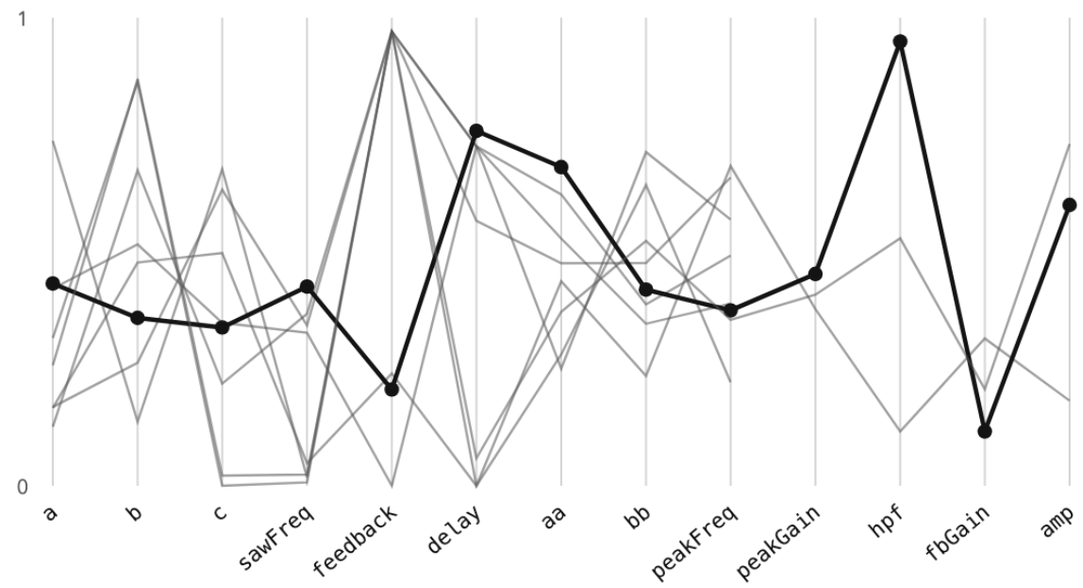
- Koenig: electronic sound is both a form-section and an acoustic unit
- Each section presents one generator as a constant stream
- Seven durations from one unit, decided before the sound
- Handcraft for articulation in time without real-time intervention
- Selection moves to the waveform and to the choice and placement of recorded takes
- SSP in Koenig's terms: amplitude and time lists assembled by selection principles
- Folding: three amplitudes reflected off a moving threshold
- One sample a segment: one control sweeps from tone to broadband noise
- Walks give the waveform a memory: at stiffness 1 it is chosen, at 0 it is found
- Tethers and bounds decide how far the walk may go
- Machine listening measured ten processes before anything was chosen

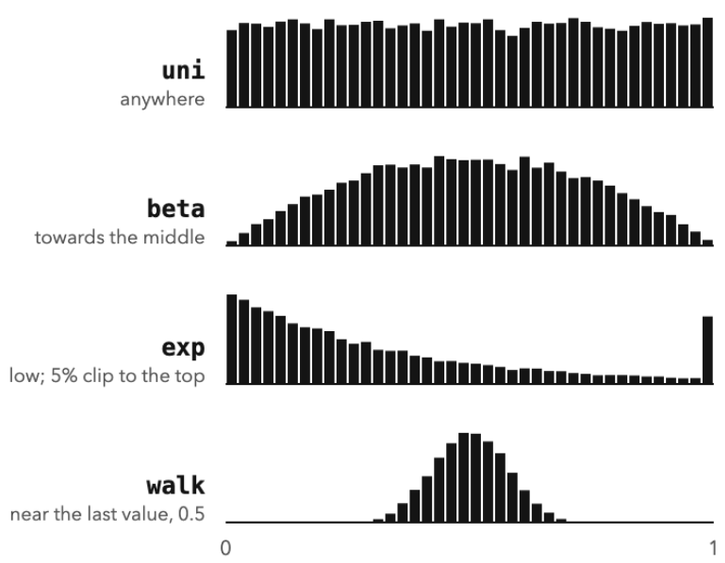


Genesis - Workflow



the eight \gorse snapshots, one line each, in fader space





dark line: slot 7. Gaps: 5 of 8 were posted before peakGain and fbGain existed

```
NF('gorse').set(\a, 0.4607, \b, 0.3907, \c, 0.22, \sawFreq, 190.78, \feedback, 0.99212, \delay, 0.0377, \aa, 23.71, \b …
```

post: slot 7 as one line of presets.scd, which every window reads back



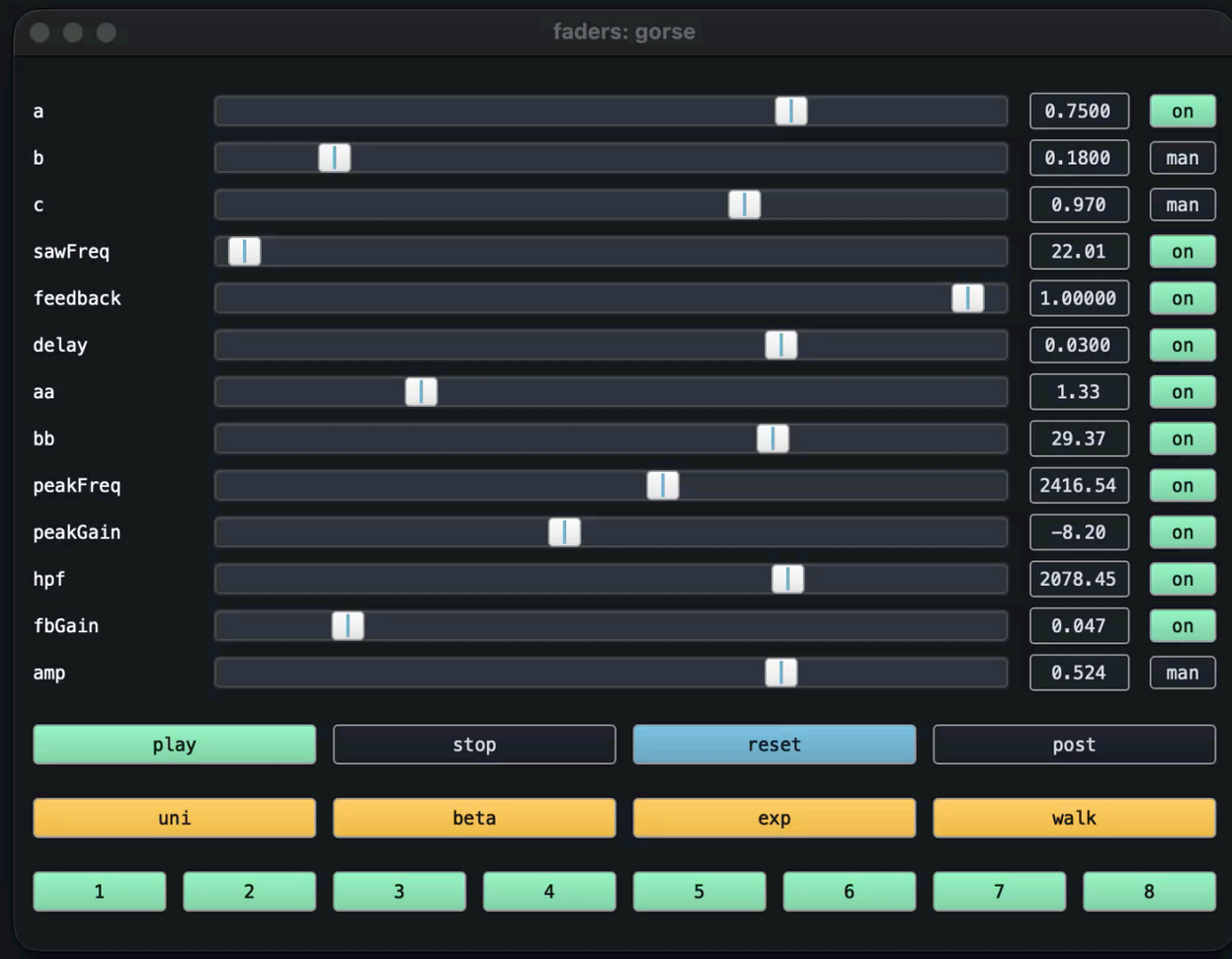
Genesis

Genesis - Tools



Genesis - States





- Each random draw (uniform, beta, exponential, a walk) informs where the interesting states in a parameter space might lie
- Deciding which faders the randomness moves and which stay under manual control forms part of the exploration
- A saved preset marks one state worth returning to among the countless others that passed
- The path between two states can be as audible as the states themselves, interpolation might becomes material

Genesis

README

Genesis

Genesis explores a set of sound synthesis procedures built from stochastic, chaotic and network-based functions and a set of tools for playing them: faders, presets, morphing between presets, and workflows that process or resonate the material.



Running it

Open `core/example.scd` and work down the blocks. The first one loads everything:

```
~here = thisProcess.nowExecutingPath.dirname +/+ "../";
(~here ++ "core/material-library.scd").load;   // ~material, ~gen, ~play, ~solo, ~free, ~loadec
(~here ++ "core/fader-specs.scd").load;        // ~specs
(~here ++ "core/ndef-faders.scd").load;        // ~ndefFaders
(~here ++ "core/preset-morph.scd").load;       // ~loadPresets, ~presetMorph
~presets = ~loadPresets.(~here ++ "core/presets.scd");
```

Then `~play.(\gorse)` to hear one, `~free.()` to put everything back to sleep.

The tools

Fader window: a slider per parameter, four randomize distributions, eight preset slots.

Morph window: travel from one preset to another, nine ways, over a duration.



README



Tether

Experimental SuperCollider UGens for digital synthesis.

UGens

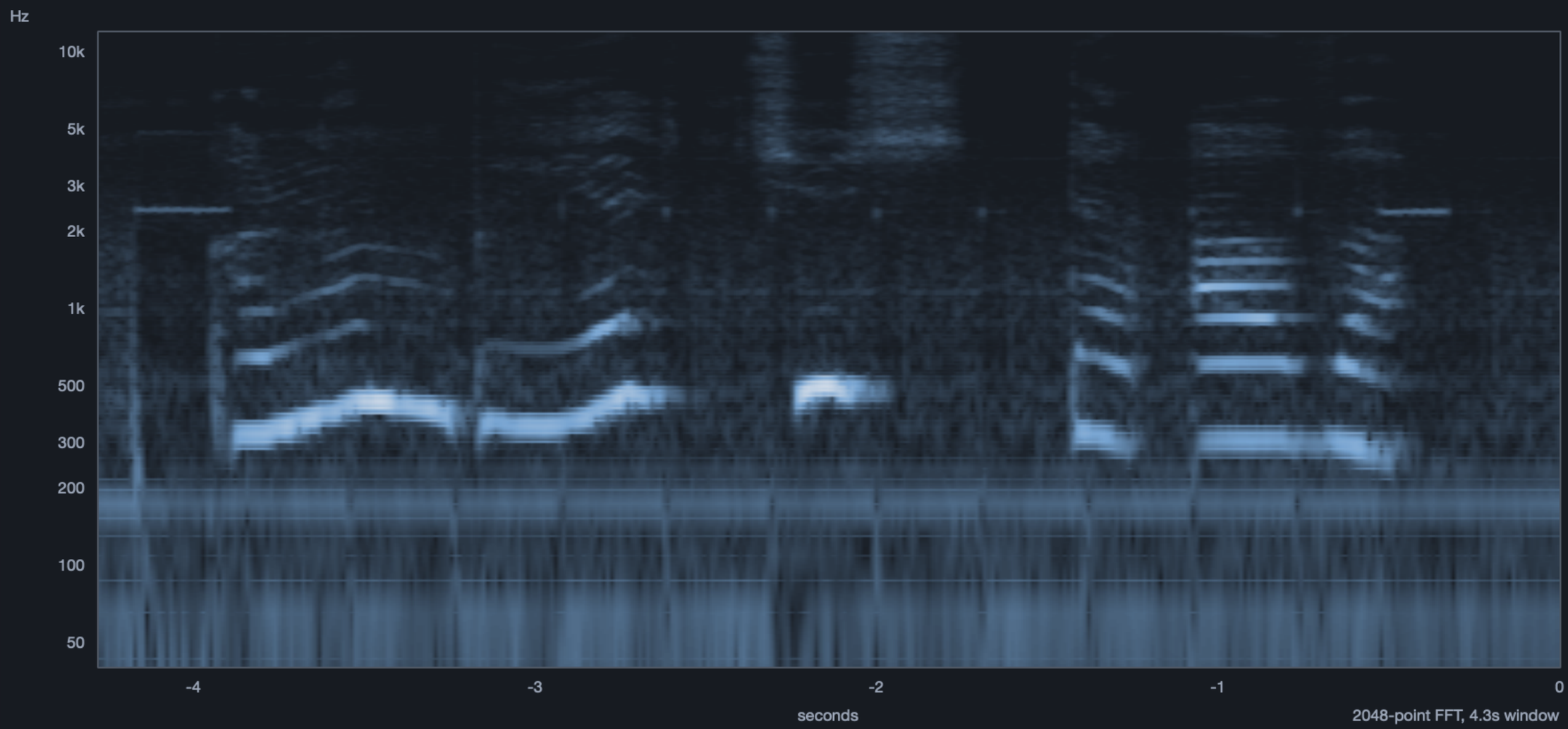
- **Gendreve** : mean-reverting breakpoint walk.
- **Diststoch** : free breakpoint walk; evolving distributions.
- **Cyclegen** : a frequency sequence sets pitch per cycle.
- **Chaosgen** : breakpoint values driven by a logistic map.
- **Varperiod** : live control-point count for timbral density.
- **Fracflight** : mean-reverting walk with fractional motion.
- **Grainstoch** : an emission pitch carrying walking grains.
- **Probstoch** : sparse grains on a fixed-pitch waveform.
- **Limcycle** : relaxation oscillator with stochastic forcing.
- **Oscnet** : coupled-oscillator network with selectable topology.
- **Ecostoch** : predator-prey (Lotka-Volterra) population synthesis.
- **Scangen** : scanned synthesis on a dynamic mass-spring lattice.
- **Lifestoch** : ALife birth-death breakpoint population.

https://github.com/bjarnig/Tether

https://github.com/bjarnig/Genesis

Listening and Analysis

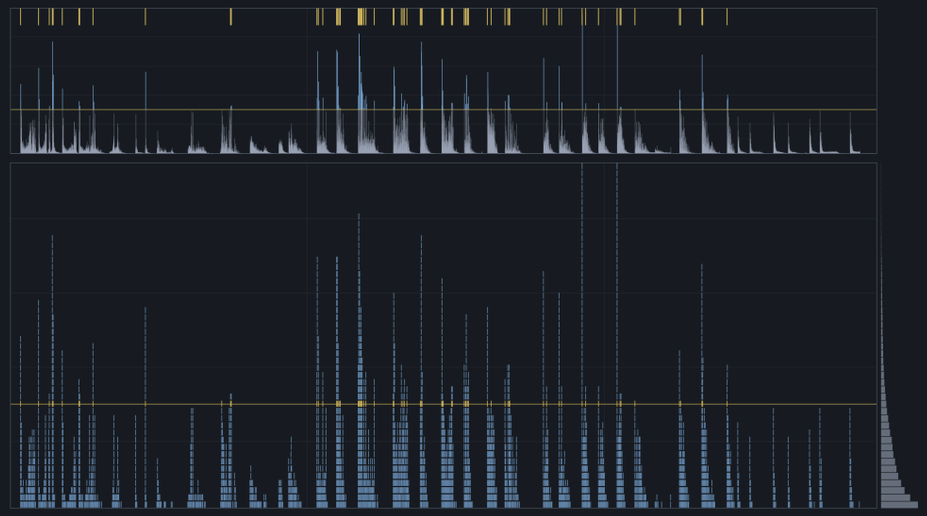
PlotLib

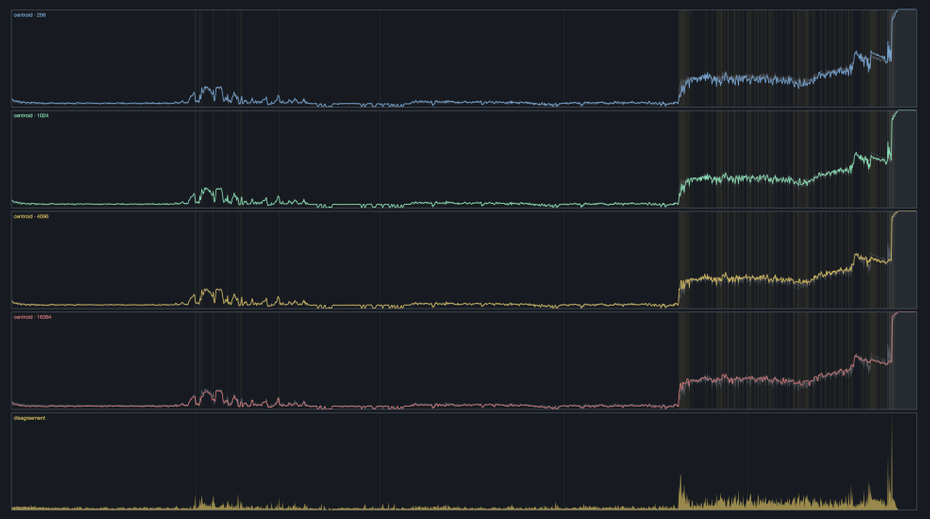


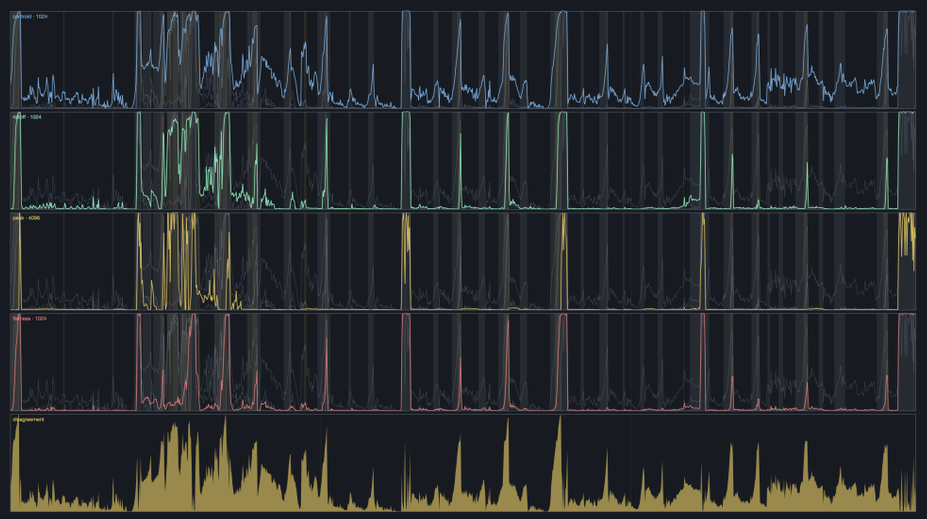
https://github.com/bjarnig/PlotLib

- Analysers observe a different focus in the same sound, and the differences between them highlights areas to explore
- An onset threshold might be less a measurement than an act that decides what exists in the material
- The structure of a work may depend as much on the chosen features as on the sound itself.
- Listening shared with a machine could widen perspective, or slowly direct it towards what the tools are able to register

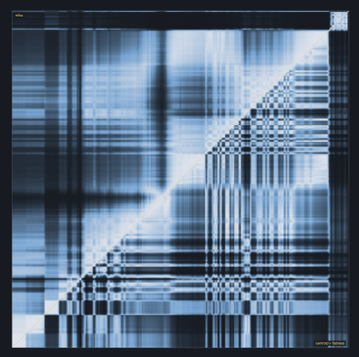
PlotLib

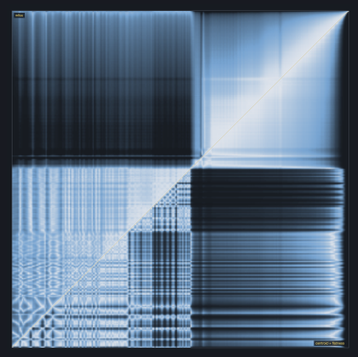


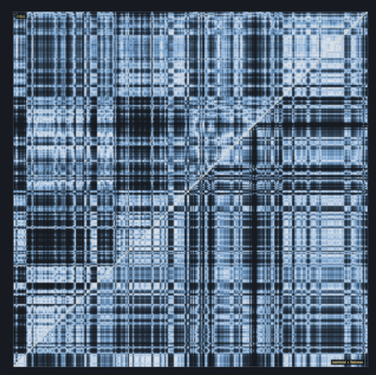

centroid - 256
centroid - 1024
centroid - 4096
centroid - 16384
disagreement

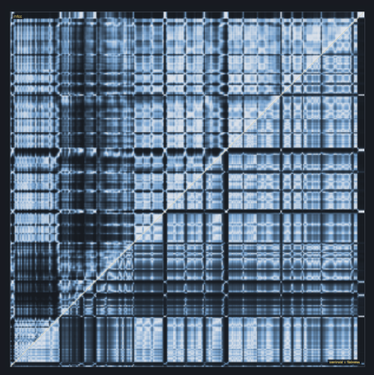

centroid - 1024
rolloff - 1024
peaks - 4096
flatness - 1024
disagreement


512 point FFT
40 mel bands
A-weighted
centroid

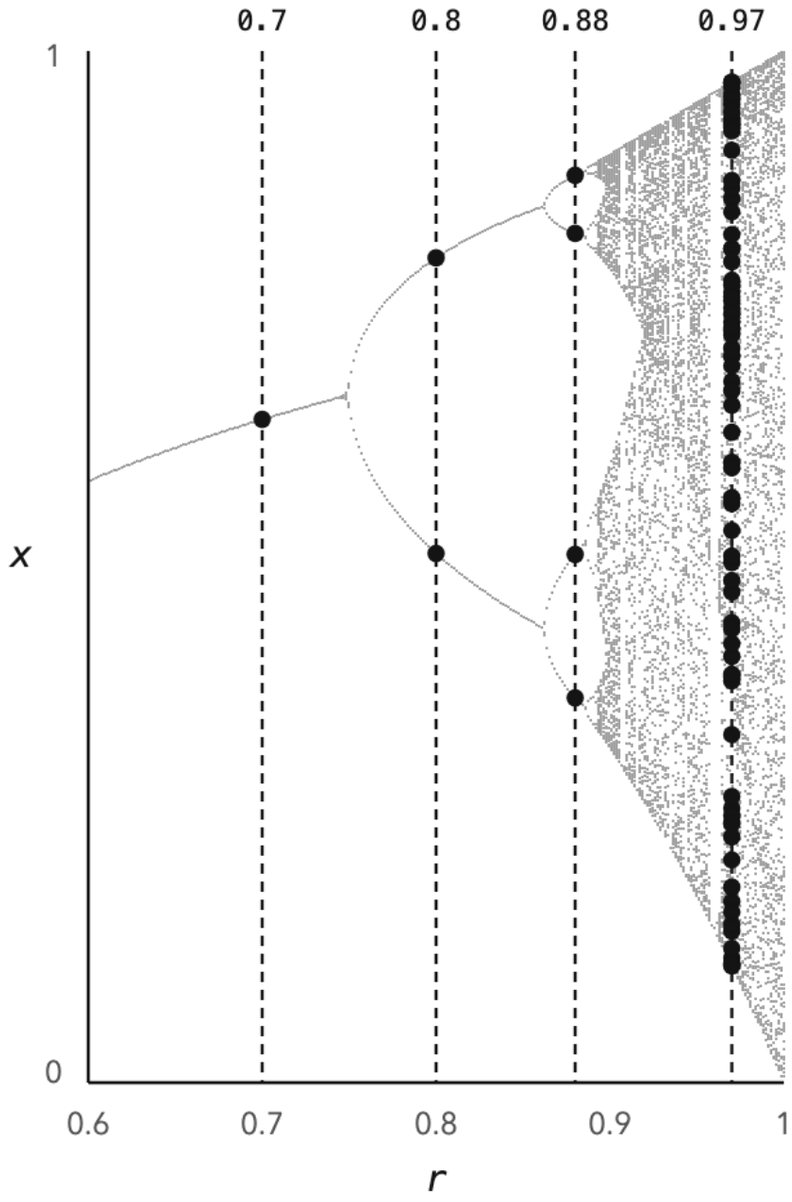

mfcc
centroid vs flatness

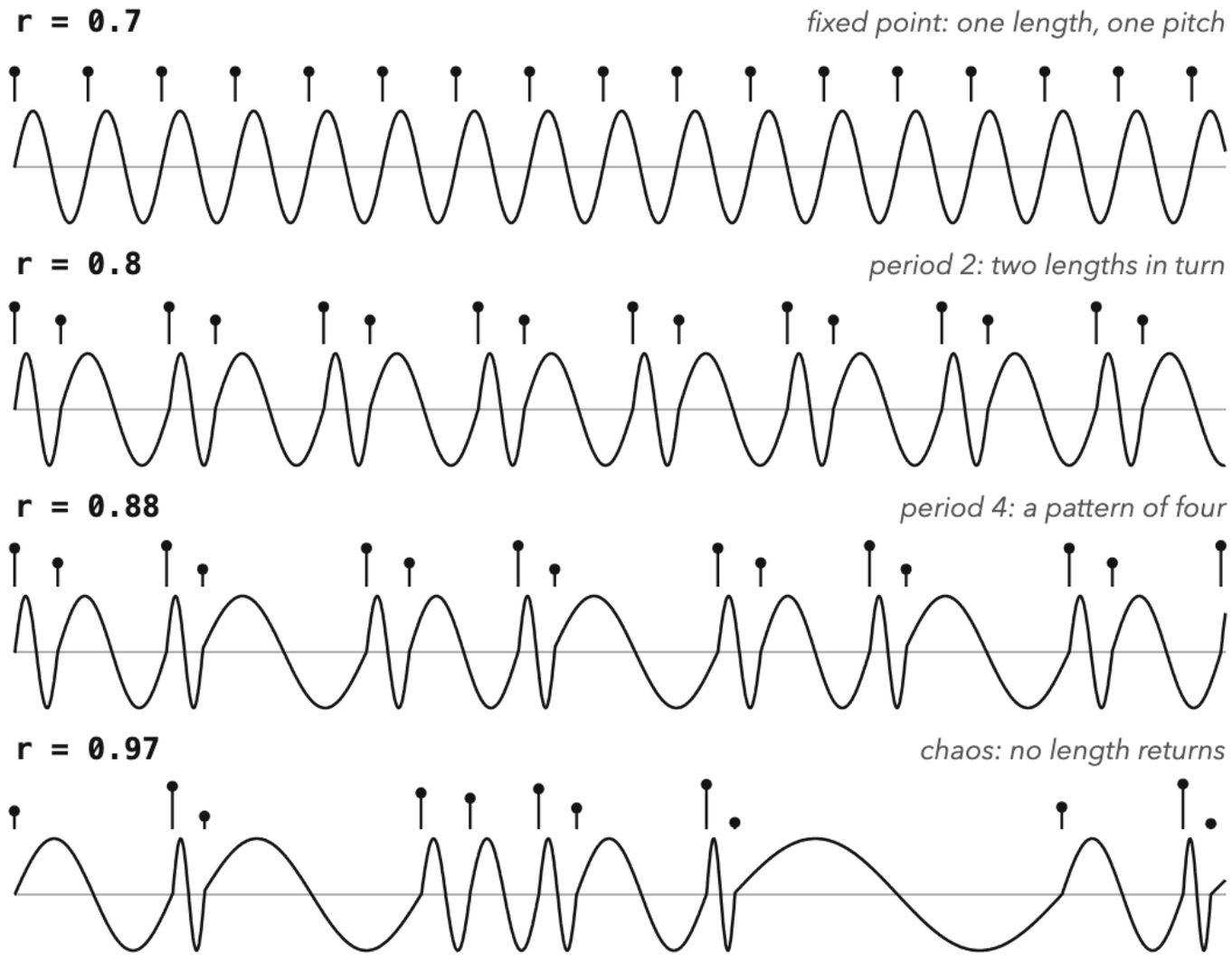

mfcc
centroid vs flatness

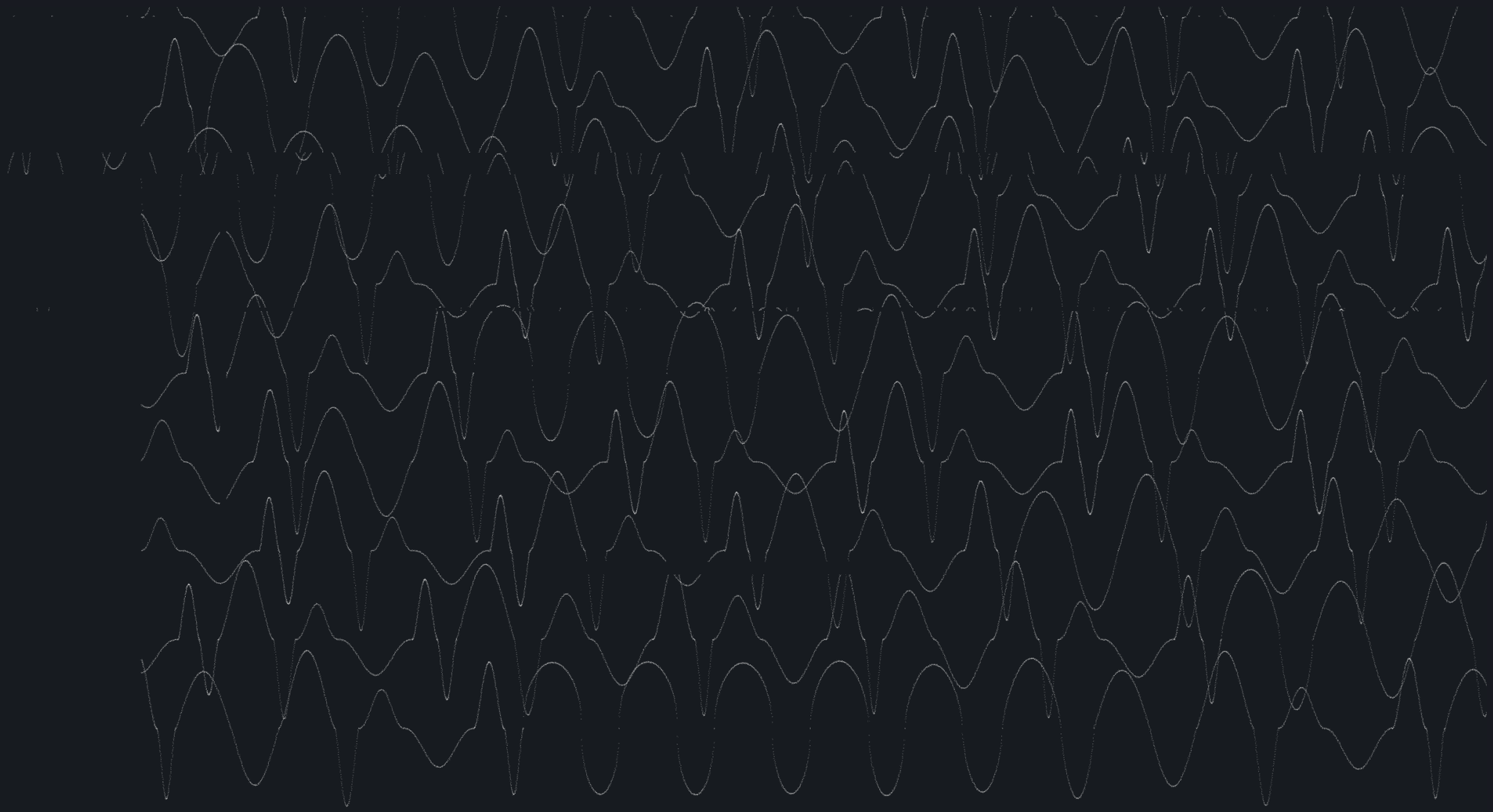

centroid vs flatness


centroid vs flatness

`WsMap.ar(r, 60, 1200): the logistic map`

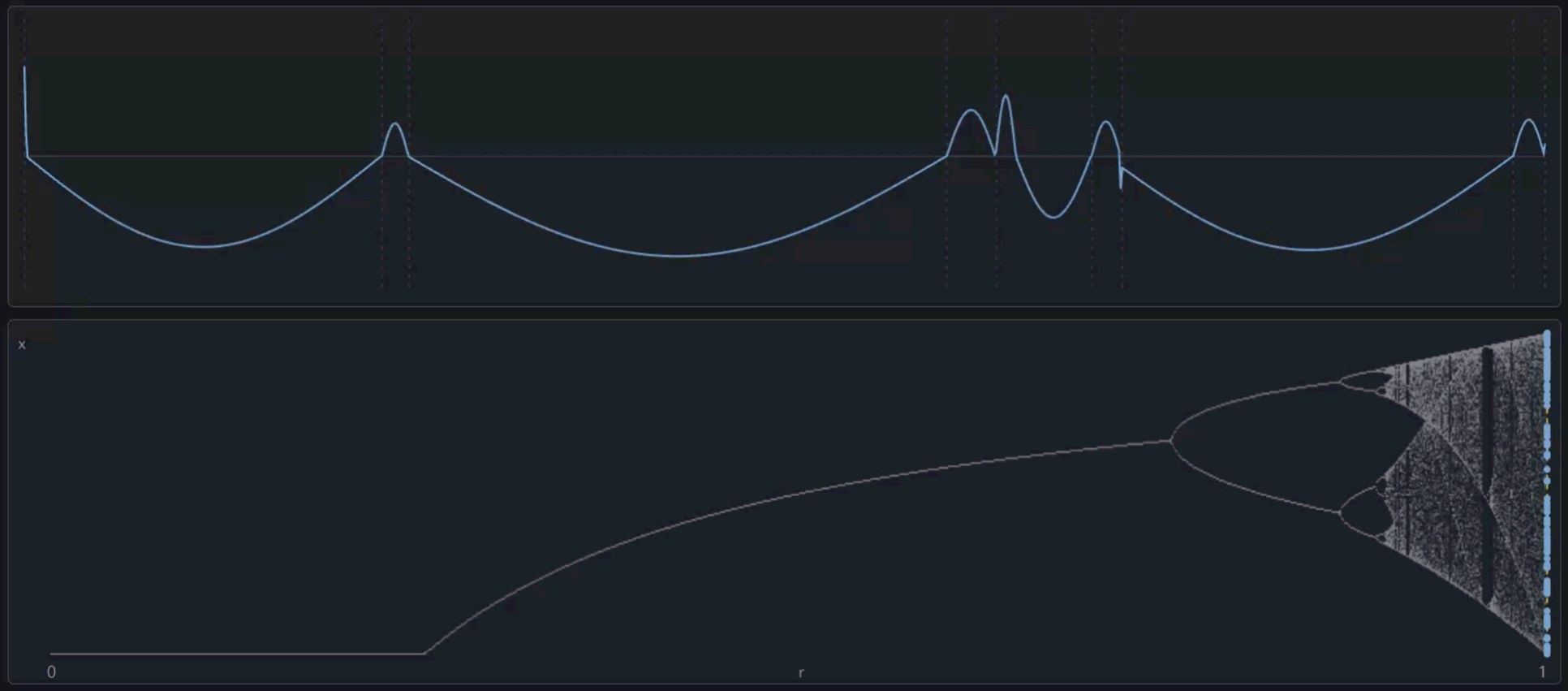






Crossings

Crossings: WsMap



state x — r — x 0.389 · waveset 240 Hz · no period up to 16: chaotic or quasi-periodic

play | presets: | fixed point | period 2 | period 4 | chaos | WsMap 5 | WsMap 8 | WsMap 13 | WsMap 14 | WsMap 20 | WsMap 22 | WsMap 24 | WsMap 25 | WsMap 28

map

map	logistic	
r		1.000
hyst		0.000

wavesets

lo, Hz		40
hi, Hz		4000
shape		0.000
skewDepth		2.000
ampDepth		0.800
balance		0.000
gain		0.150

https://bjarni-gunnarsson.net/works/docs/crossings

Crossings Public

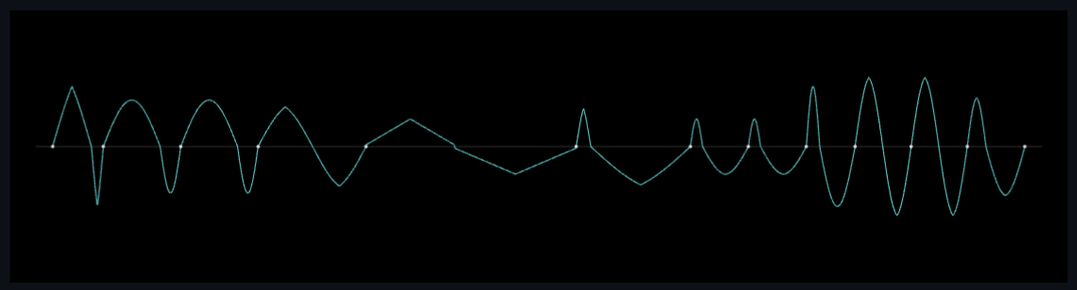
Pin | Watch 0 | Fork 0 | Star 1

main | 1 Branch | 1 Tag | Go to file T | Add file | Code

bjarnig Update README.md — b7d262c · now — 15 Commits

.github/workflows	feat: add WsSync, WsInfo and WsGroup	yesterday
HelpSource/Classes	feat: add WsSync, WsInfo and WsGroup	yesterday
classes	feat: add WsSync, WsInfo and WsGroup	yesterday
cmake_modules	feat: add WsGen	yesterday
docs	docs: add banner generator	yesterday
plugins	feat: add WsSync, WsInfo and WsGroup	yesterday
tests	feat: add WsGen	yesterday
.gitignore	chore: stop tracking CLAUDE.md	yesterday
CMakeLists.txt	feat: add WsSync, WsInfo and WsGroup	yesterday
LICENSE	feat: add WsGen	yesterday
README.md	Update README.md	now
build.sh	feat: add WsGen	yesterday

README | GPL-3.0 license



Crossings

SuperCollider server plugins (UGens) for waveset synthesis. A waveset is the span of signal between alternate zero crossings. Crossings synthesizes the wavesets themselves.

Every parameter is read once per waveset, at the zero crossing. Demand-rate inputs (`Dseq`, `Dbrown`, `Dwhite`, ...) advance one value per waveset, and each waveset can have its own length, shape, peak and skew. Pitch, timbre and rhythm come from the same sequence of decisions.

UGens

- `WsGen` : wavesets with per-waveset length, peak, shape (sine, triangle, square, spike, morphable), skew, repeats and glide.

About

No description, website, or topics provided.

Readme
GPL-3.0 license
Activity
1 star
0 watching
0 forks

Releases 1

v0.1.0 Latest
yesterday

Packages

No packages published
Publish your first package

Contributors 1

bjarnig Bjarni Gunnarsson

Languages

C++ 66.7% CMake 21.3%
SuperCollider 9.6% Shell 2.4%

https://github.com/bjarnig/Crossings

Material trace

The question then becomes: where is the memory of the stereotype kept, if not in the material trace of the stereotype in which the preexisting tool itself consists, repeated, duplicated by its "maker" and guiding the latter much more than being guided by him or her? In this sense, the archaic cortex and equipment are codetermined in a structural coupling of a particular sort.

Bernard Stiegler, Technics and Time 1 (1998)

