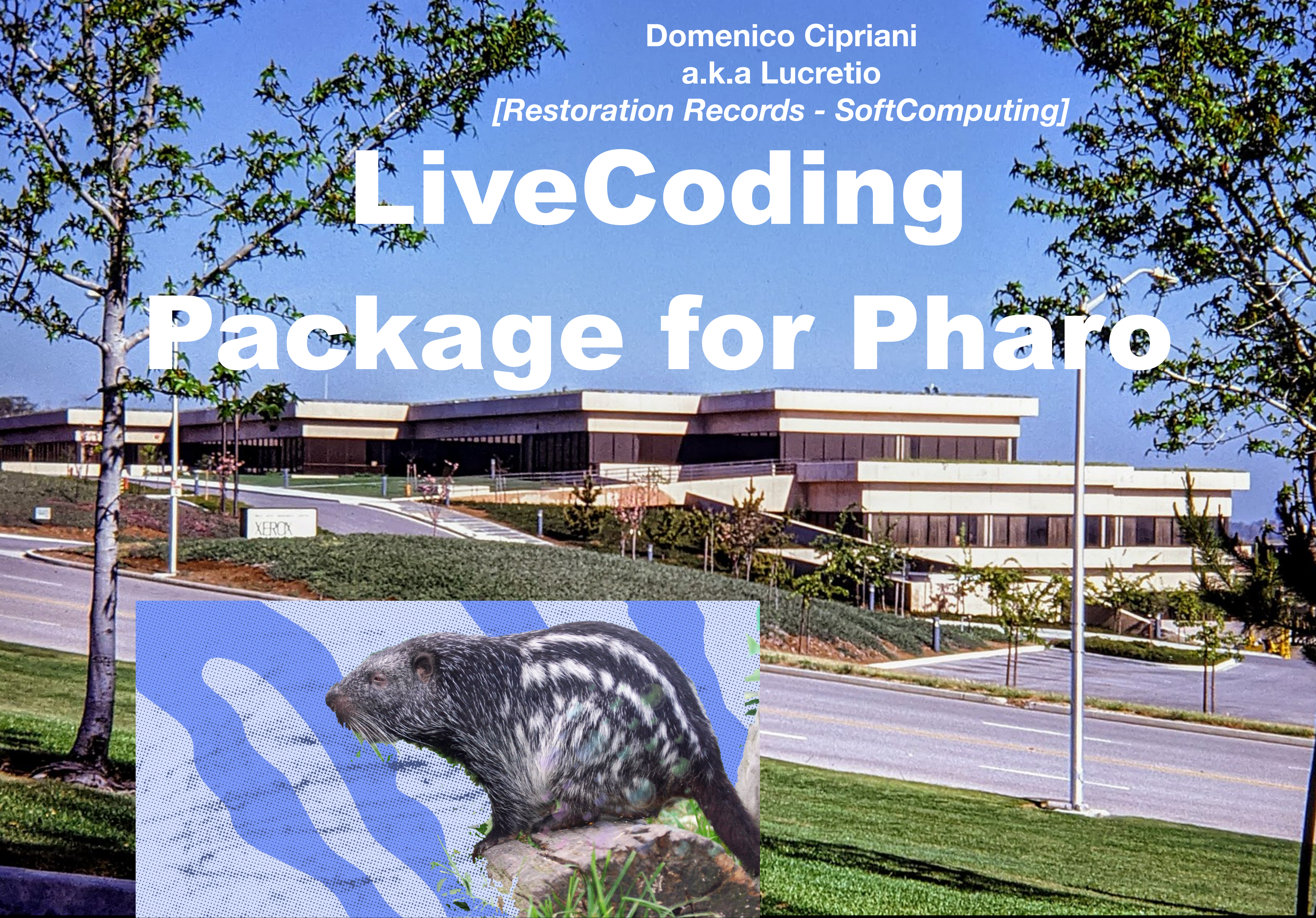


Domenico Cipriani
a.k.a Lucretio
[Restoration Records - SoftComputing]

LiveCoding Package for Pharo

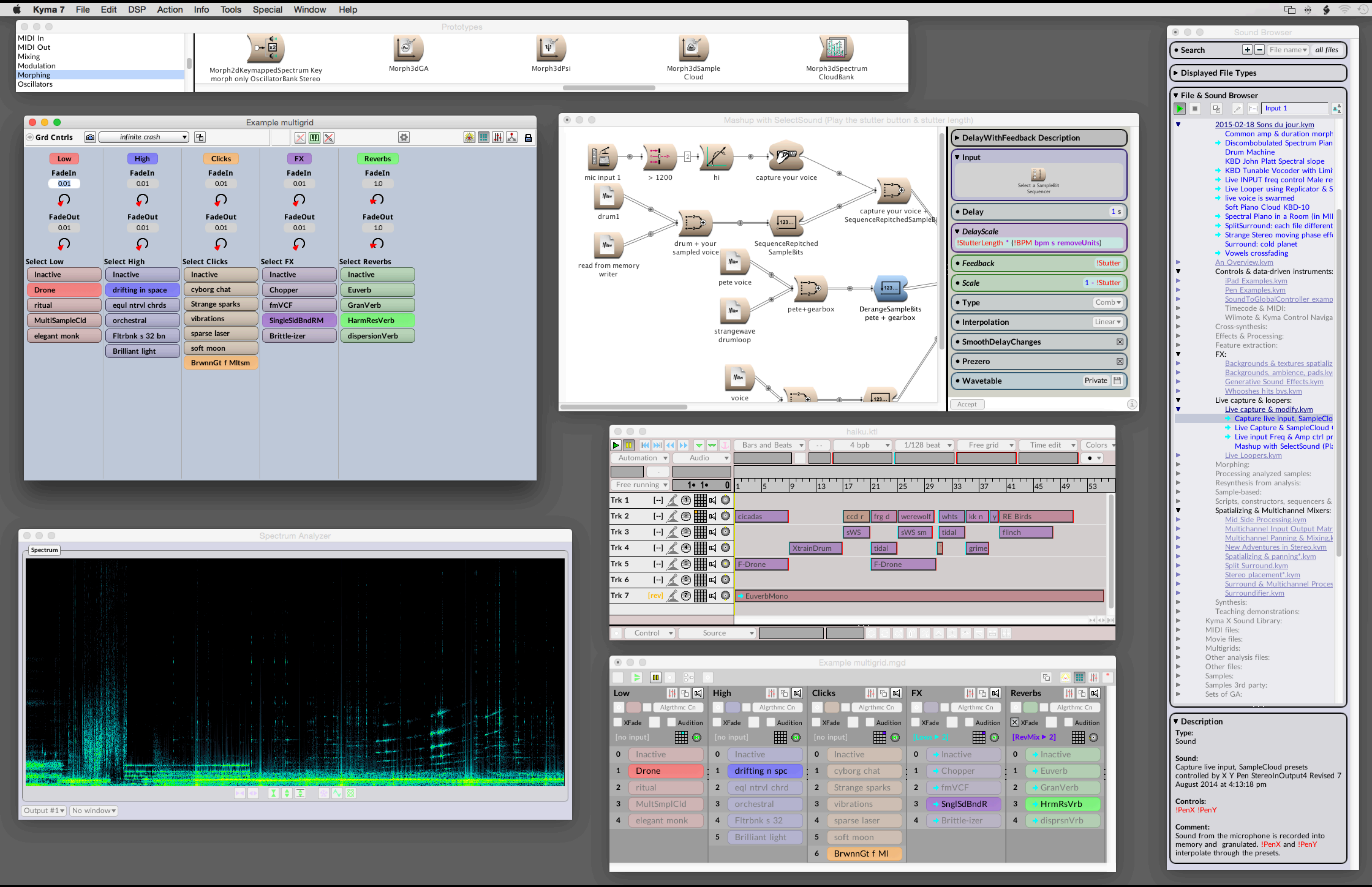
https://github.com/lucretiomsp/PharoLiveCoding

Symbolic Sound Kyma

- Music programming language and IDE written in Smalltalk created by Carla Scaletti and Kurt J. Hebel at Urbana Champaign, Illinois.

- The Smalltalk code is compiled on an external DSP called Paca(rana)

Kyma 7



Description

Type:
Sound

Sound:
Capture live input, SampleCloud presets controlled by X Y Pen StereoInOutput4 Revised 7 August 2014 at 4:13:18 pm

Controls:
!PenX !PenY

Comment:
Sound from the microphone is recorded into memory and granulated. !PenX and !PenY interpolate through the presets.

2 principi: economia e trasparenza

- “L’unico principio primario in ogni azione umana, é il dispendio del minimo sforzo per portare a termine un compito” (George Zipf)

- “L’iconicitá é la relazione di somiglianza tra i due aspetti di un segno: la sua forma e il suo significato. Un segno iconico é un segno che in qualche modo assomiglia al suo significato” (Meir)

- *La sintassi di Smalltalk può stare su una cartolina, mentre la sua semantica può essere letta come un pidgin English ed é pensata per i bambini*

Principi della programmazione orientata agli oggetti (Object Oriented Programming / OOP)

- Incapsulamento
- Astrazione
- Ereditarietà
- Polimorfismo (late binding)

On-the-fly programming music (or Live Coding)

- **Increasingly popular creative practice for audio-visual creation.**
- Typically, the process of writing source code is made visible by projecting the computer screen in the audience space, with ways of visualising the code an area of active research.
- **The figure of the live coder is who performs the act of live coding, "usually artists who want to learn the code, and coders who want to express themselves, or in terms of Wang & Cook the "programmer/performer/composer"**
- **TOPLAP (The (Temporary|Transnational|Terrestrial|Transdimensional) Organisation for the (Promotion|Proliferation|Permanence|Purity) of Live (Algorithm|Audio|Art|Artistic) Programming) is an informal organization formed in February 2004 to bring together the various communities that had formed around live coding environments.**
- **On-the-fly promotes live coding practice since 2020. This is a project co-funded by the Creative European program and run in Hangar, ZKM, Ljudmila and Creative Code Utrecht**

- “A programming language is a tool” (Bjarne Stroustrup)
- “If the only tool you have is an hammer, everything looks like a nail” (Abraham Maslow)
- “The most disastrous thing that you can ever learn is your first programming language” (Alan Kay)

Why Pharo?

- For Smalltalk!
- Arrays are at the core of electronic music, their manipulation in Pharo is extremely powerful.
- Because the Playground is the perfect environment for Live Coding.
- Because there are not other pure Object Oriented languages for Live Coding.
- For its expressiveness and reflectiveness.
- Because new methods and classes are created easily and always available to the system (i.e. no headers, no extra dependence, no tedious file management).

The LiveCoding Package

- To write music on-the-fly with Pharo
- Also for studio composition: a new kind of musical score
- Pharo acts as an arranger, another program generates the sound (Kyma, PureData, MaxMSP, ChucK, SuperCollider, and so on.
- Based on the OpenSoundControl at the moment, but MIDI implementation on the pipeline

PRINCIPLES

- **Iconicity** ➡ Written code should resemble what we hear

```
16 upbeats
```

- **Economy** ➡ The less we type, the better

```
#(60 63 67) + 16
```

- **Polysemy** ➡ Many ways to do the same thing

```
16 randomsFrom: #(50 54 57)
16 randomNotes: (50 54 47)
```

Economy and transparency

- “The only primary principle of every human action, including verbal communication, is the expenditure of the least amount of effort to accomplish a task. (George Zipf)
- “Iconicty is the relation of similarity between to aspects of a sign: its form and its meaning. An iconic sign is a sign that in some way resembles its meaning.”(Meir)
- *Smalltalk syntax fits on (half) a postcard, its semantic can be read as Pidgin English and it was thought to be easy to understand for children.*

OpenSoundControl OSC

- Developed by Adrian Freed and Matt Wright at CNMAT at the end of the 90s. Firs specification published in 2002.
- Flexible, fast and accurate alternative to the MIDI standard..
- Independent from the transport mechanism, OSC packets are typically sent and received thru UDP Sockets.
- ***Server/Client*** architecture The ***server*** sends the packets, the ***client*** receives them.
- An **OSC message** consists of an ***OSC Address*** Pattern, followed by an ***OSC Type Tag*** String, followed by zero or more ***OSC Arguments***
  (for example: /frequency,f 0.3).
- At the core of the ***LiveCoding Package for Pharo.***

OpenSoundControl OSC

- **The LiveCoding package simplify the creation and dispatching of OSC messages**

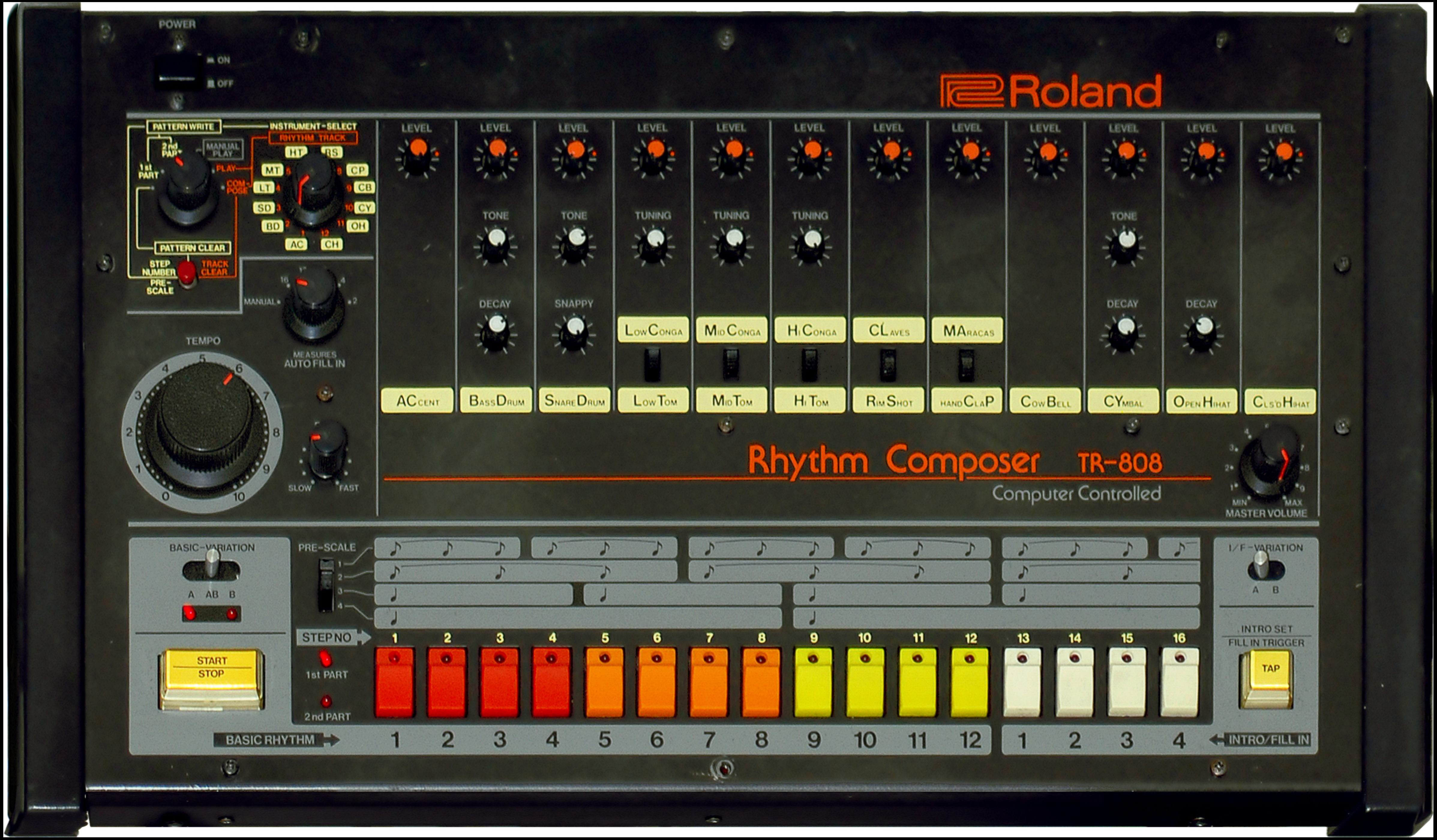
aNumber toLocal: aString. ▶ Send to the local host the message:

‘/aString, f aNumber’

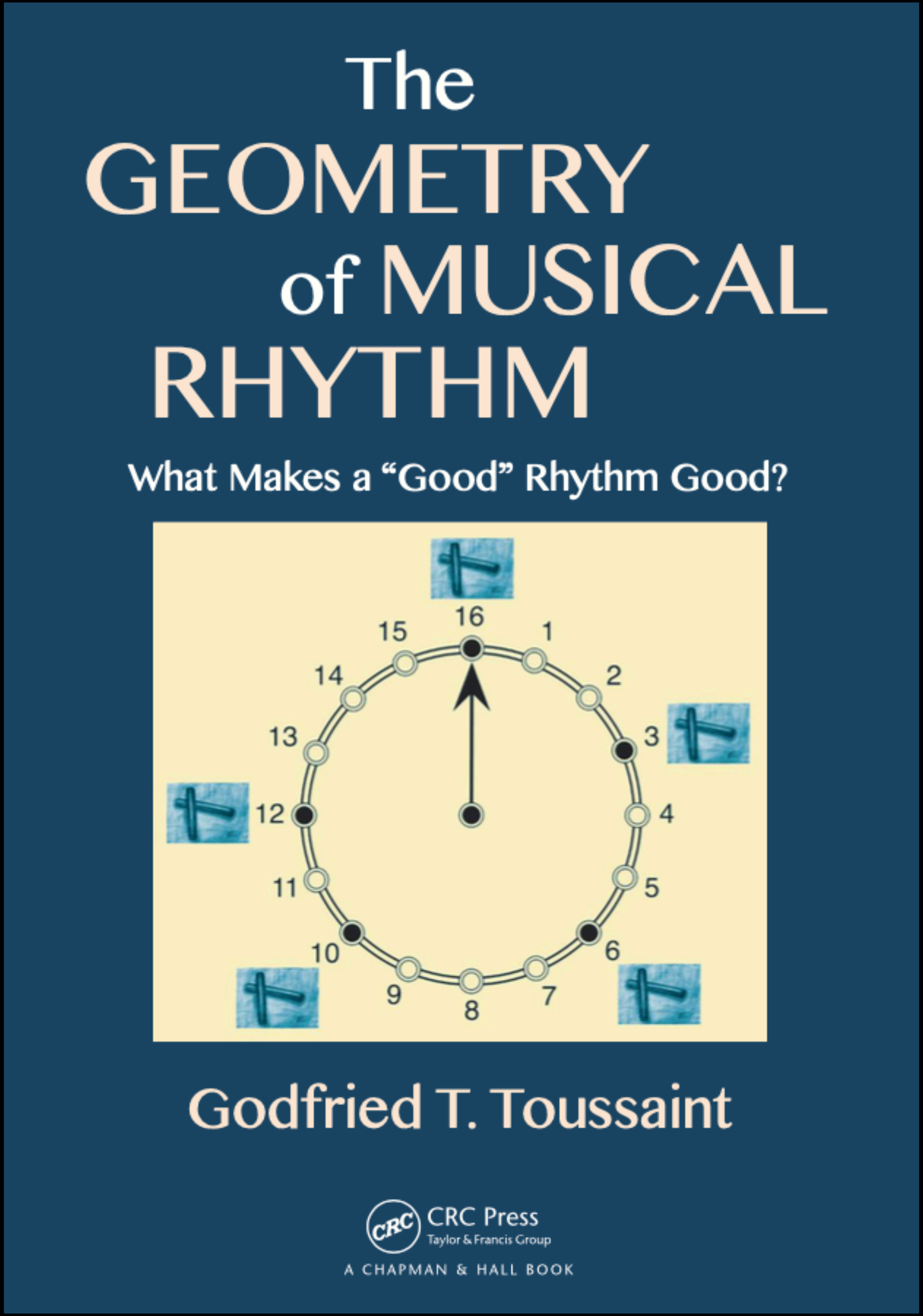
aNumber toKyma: aString. ▶ Send to the Paca(rana) the message

’/aString, f aNumber’

Step Sequencers

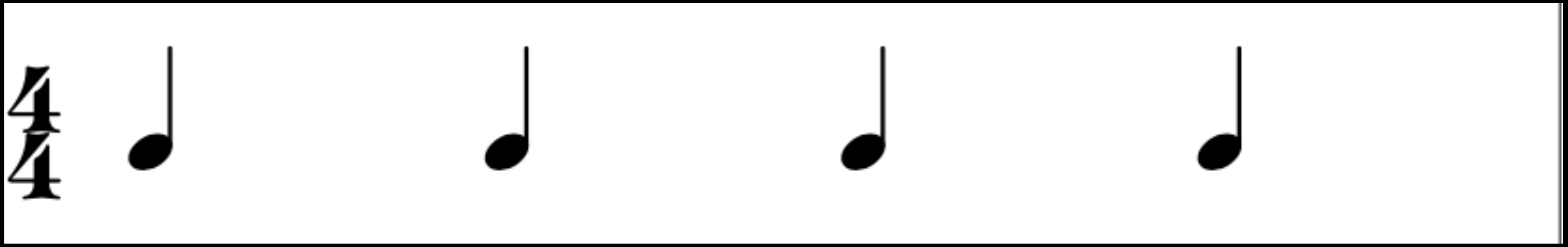


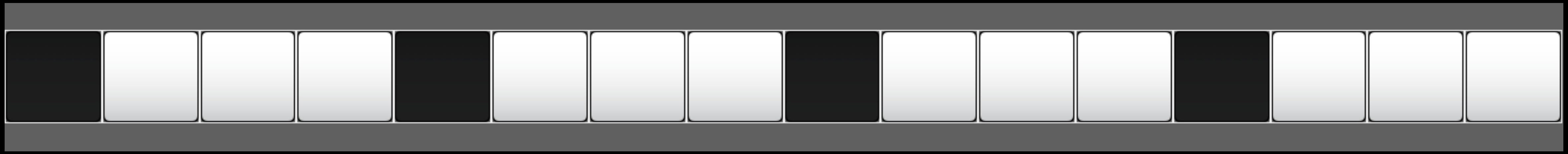
On a Step Sequencer, a step can be active (1) or not active (0).



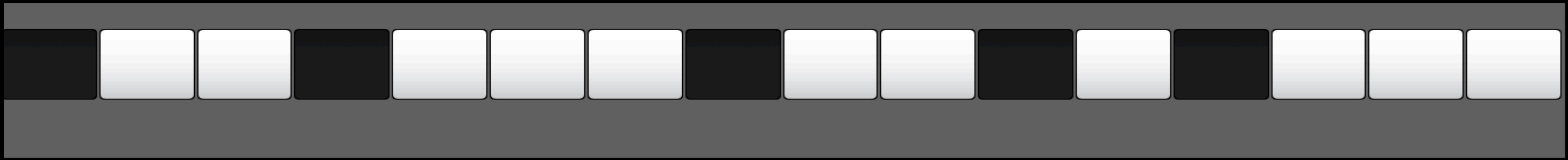
The LiveCoding package also contains a collection of Euclidean world rhythms and musical scales.

A rhythm can be represented as an array of 0s and 1s, where a 1 represents a trig.

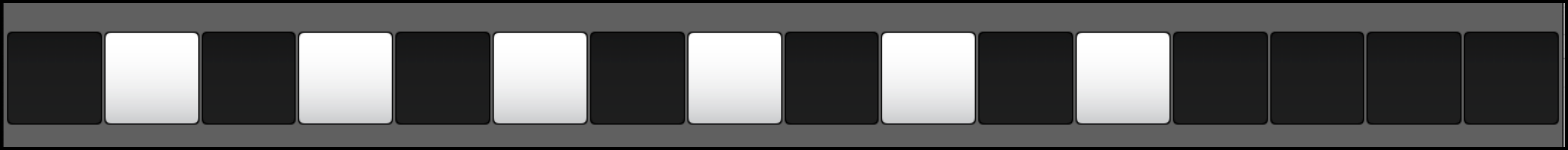




- **BINARY:** 1000100010001000
- **HEXADECIMAL:** 8888
- **SMALLTALK:** #(1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0)
- **LIVECODINGTALK:** 16 downbeats
- **LIVECODINGTALK:** ‘8888’ pattern



- **BINARY:** 1001 0001 0010 1000
- **HEXADECIMAL:** 9128
- **SMALLTALK:** #(1 0 0 1 0 0 0 1 0 0 1 0 1 0 0 0)
- **LIVECODINGTALK:** 16 rumba
- **LIVECODINGTALK:** '9128' pattern



- **BINARY:** 1010 1010 1010 1111
- **HEXADECIMAL:** AAAF
- **SMALLTALK:** #(1 0 1 0 1 0 1 0 1 0 1 0 1 1 1)
- **LIVECODINGTALK:** 12 quavers, 4 semiquavers
- **LIVECODINGTALK:** 'AAAF' pattern

Process - Performance - Sequence - Rhythm

- aNumberOfSteps sequenceFor: aPerformance rate: aStepDuration
- aNumberOfSteps localSequenceFor: aPerformance rate: aStepDuration

- **Performance is a subclass of Dictionary.**
- **I**t contains association of **Symbols** and **Arrays** or **Sequences**

- **Rhythm is** *a convenience* **subclass of Array**

- **Sequencer is a subclass of Array** made of Trigs, NoteNumbers and, Durations.
- It is created sending the message *asSeq* to an instance of the ***Rhythm***

- The **Process** forked at timingPriority check if the *value* of the *key* in the performance is a **Sequencer** or not.

- If the *value is a Sequencer, 3 OSC messages are sento to the client:*

- appending 'Gate', 'Note', 'Duration' to the key asString

In the Live Coding package you can create Arrays of Gates or of numbers by sending messages to integers, *for example:*

- 16 zeros
- 64 randoms
- 32 randomsInt: 88
- 16 quavers
- 128 randomWalksOn: (Scale sakura root: 48 octaves: 2)

Bars, Bytes, Beats, Nibbles, Steps, Bits

- 4 bits = 1 Nibble / 8 bit = 1 Byte.
- 1 Bar, 16 Steps (1/16th quantisation).
- If every step corrisponds to a Bit of Information, in a Bar we fino 16 Bits, i.e. 2 Bytes
- In every Bar there are 4 Beats, so in each Beat there are 4 Bits, i.e. 1 Nibble.
- Every Hexadecimal symbol represents a Nibble.

0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	Upbeat
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	Downbeat
9	1	0	0	1	
A	1	0	1	0	Quavers
B	1	0	1	1	
C	1	1	0	0	
D	1	1	0	1	
E	1	1	1	0	
F	1	1	1	1	Semiquavers