Domenico Cipriani

LiveCoding Package for Pharo

https://github.com/lucretiomsp/PharoLiveCoding

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 (TemporarylTransnationallTerrestriallTransdimensional) Organisation for the (PromotionlProliferationlPermanencelPurity) of Live (AlgorithmlAudiolArtlArtistic) 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**

- “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).

Symbolic Sound Kyma

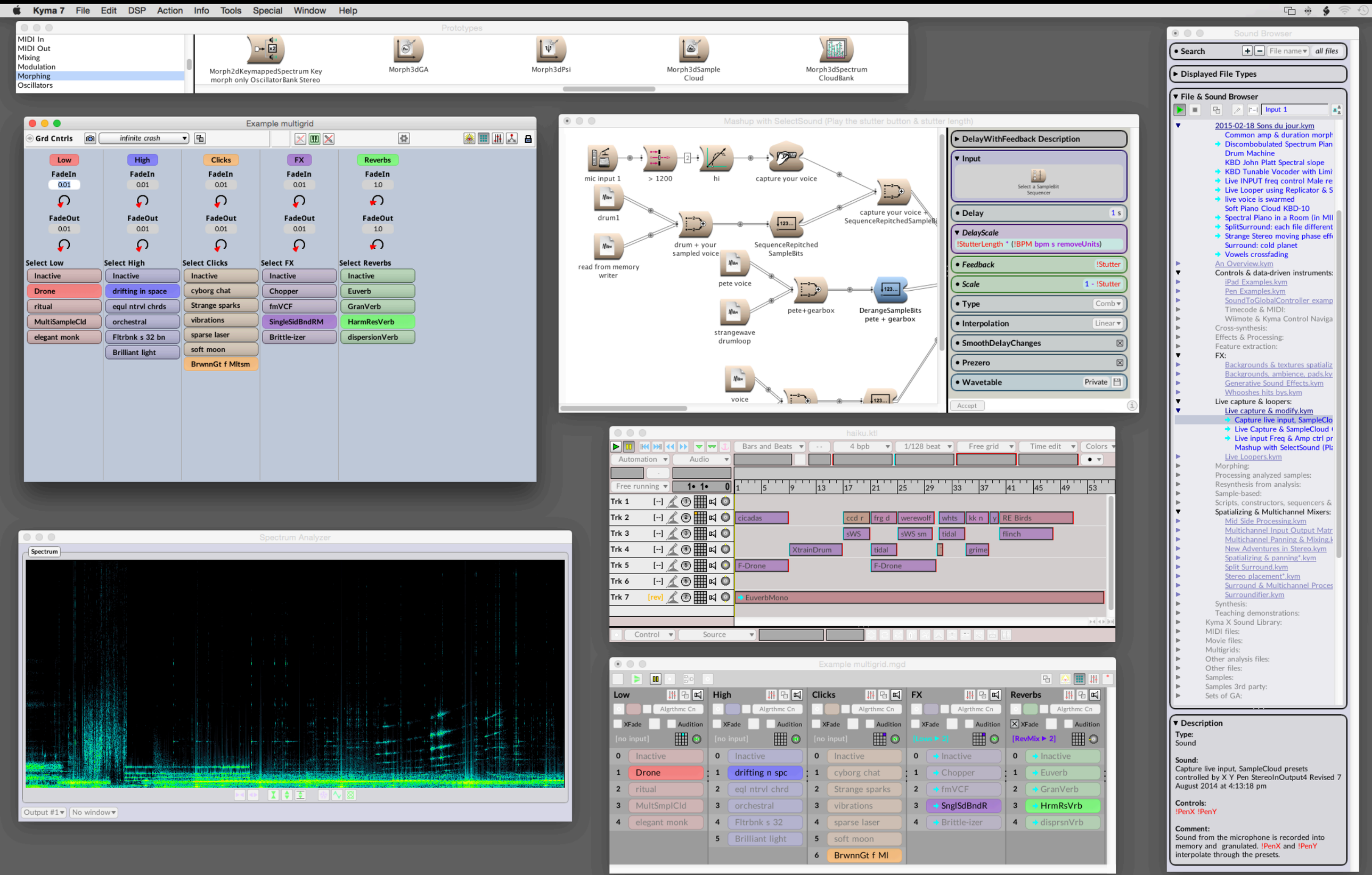
- **Music programming language and IDE written in Smalltalk created by Carla Scaletti and Kurt J. Hebel at Urbana Champaign, Illinois.**

- ***Most probably :) its developed on Objectworks\Smalltalk, Release 4.1 of 15 April 1992***

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

- **“The Holy Grail of sound design”**

Kyma 7


Kyma 7 File Edit DSP Action Info Tools Special Window Help
Prototypes
MIDI In
MIDI Out
Mixing
Modulation
Morphing
Oscillators
Morph2dKeymappedSpectrum Key morph only OscillatorBank Stereo
Morph3dGA
Morph3dPsi
Morph3dSample Cloud
Morph3dSpectrum CloudBank
Example multigrid
Grd Cntrls
infinite crash
Low
High
Clicks
FX
Reverbs
FadeIn
FadeOut
Select Low
Inactive
Drone
ritual
MultiSampleCld
elegant monk
Select High
Inactive
drifting in space
equl ntrvl chrds
orchestral
Fltrbnk s 32 bn
Brilliant light
Select Clicks
Inactive
cyborg chat
Strange sparks
vibrations
sparse laser
soft moon
BrwnnGt f Mltsm
Select FX
Inactive
Chopper
fmVCF
SingleSidBndRM
Brittle-izer
Select Reverbs
Inactive
Euverb
GranVerb
HarmResVerb
dispersionVerb
Mashup with SelectSound (Play the stutter button & stutter length)
mic input 1
> 1200
hi
capture your voice
drum1
drum + your sampled voice
SequenceRepitched SampleBits
capture your voice + SequenceRepitchedSampleBi
read from memory writer
pete voice
pete+gearbox
DerangeSampleBits pete + gearbox
strangewave drumloop
voice
DelayWithFeedback Description
Input
Delay
1 s
DelayScale
!StutterLength * (!BPM bpm s removeUnits)
Feedback
!Stutter
Scale
1 - !Stutter
Type
Comb
Interpolation
Linear
SmoothDelayChanges
Prezero
Wavetable
Private
Accept
Sound Browser
Search
File name
all files
Displayed File Types
File & Sound Browser
Input 1
2015-02-18 Sons du jour.kym
Common amp & duration morph
Discombobulated Spectrum Pian
Drum Machine
KBD John Platt Spectral slope
KBD Tunable Vocoder with Limi
Live INPUT freq control Male re
Live Looper using Replicator & S
live voice is swarmed
Soft Piano Cloud KBD-10
Spectral Piano in a Room (in MI
SplitSurround: each file different
Strange Stereo moving phase effe
Surround: cold planet
Vowels crossfading
An Overview.kym
Controls & data-driven instruments:
iPad_Examples.kym
Pen_Examples.kym
SoundToGlobalController examp
Timecode & MIDI:
Wiimote & Kyma Control Naviga
Cross-synthesis:
Effects & Processing:
Feature extraction:
FX:
Backgrounds & textures spatializ
Backgrounds, ambience, pads.ky
Generative Sound Effects.kym
Whooshes hits bys.kym
Live capture & loopers:
Live capture & modify.kym
Capture live input, SampleClo
Live Capture & SampleCloud
Live input Freq & Amp ctrl pr
Mashup with SelectSound (Pla
Live Loopers.kym
Morphing:
Processing analyzed samples:
Resynthesis from analysis:
Sample-based:
Scripts, constructors, sequencers &
Spatializing & Multichannel Mixers:
Mid Side Processing.kym
Multichannel Input Output Matr
Multichannel Panning & Mixing.k
New Adventures in Stereo.kym
Spatializing & panning*.kym
Split Surround.kym
Stereo placement*.kym
Surround & Multichannel Proces
Surroundifier.kym
Synthesis:
Teaching demonstrations:
Kyma X Sound Library:
MIDI files:
Movie files:
Multigrids:
Other analysis files:
Other files:
Samples:
Samples 3rd party:
Sets of GA:
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.
haiku.ktl
Bars and Beats
4 bpb
1/128 beat
Free grid
Time edit
Colors
Automation
Audio
Free running
Trk 1
Trk 2
Trk 3
Trk 4
Trk 5
Trk 6
Trk 7
cicadas
ccd r
frg d
werewolf
whts
kk n
RE Birds
sWS
sWS sm
tidal
flinch
XtrainDrum
grime
F-Drone
EuverbMono
Control
Source
Spectrum Analyzer
Spectrum
Output #1
No window
Example multigrid.mgd
Low
High
Clicks
FX
Reverbs
Algrthmc Cn
XFade
Audition
[no input]
[RevMix ▶ 2]
Inactive
Drone
ritual
MultSmplCld
elegant monk
drifting n spc
eql ntrvl chrd
orchestral
Fltrbnk s 32
Brilliant light
cyborg chat
Strange sparks
vibrations
sparse laser
soft moon
BrwnnGt f Ml
Chopper
fmVCF
SnglSdBndR
Brittle-izer
Euverb
GranVerb
HrmRsVrb
disprsnVrb

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)

What is the LiveCoding Package?

- A front-end to write real time scores for a real-time audio synthesis backend (via OSC) or a MIDI device

- The sound generators created on the backend must adhere to the LiveCoding Package syntax (!InstrumentGate ! / InstrumentNote / !InstrumentDuration / !InstrumentExtra1)

- A DSL to write quickly and easily rhythmical patterns and melodies

- An ethnomusical quick-guide

Performance: a subclass of Dictionary that will contains instances of the Sequencer class or arrays

Sequencer

Gates

Notes

Durations

Extra1 (optional)

Extra2 (optional)

noteIndex

Process - Performance - Sequence - Rhythm

- aPerformance playKymasequenceAt: aStepDuration rate: aNumberOfSteps
- aPerformance playLocalSequenceAt: aStepDuration rate: aNumberOfSteps

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

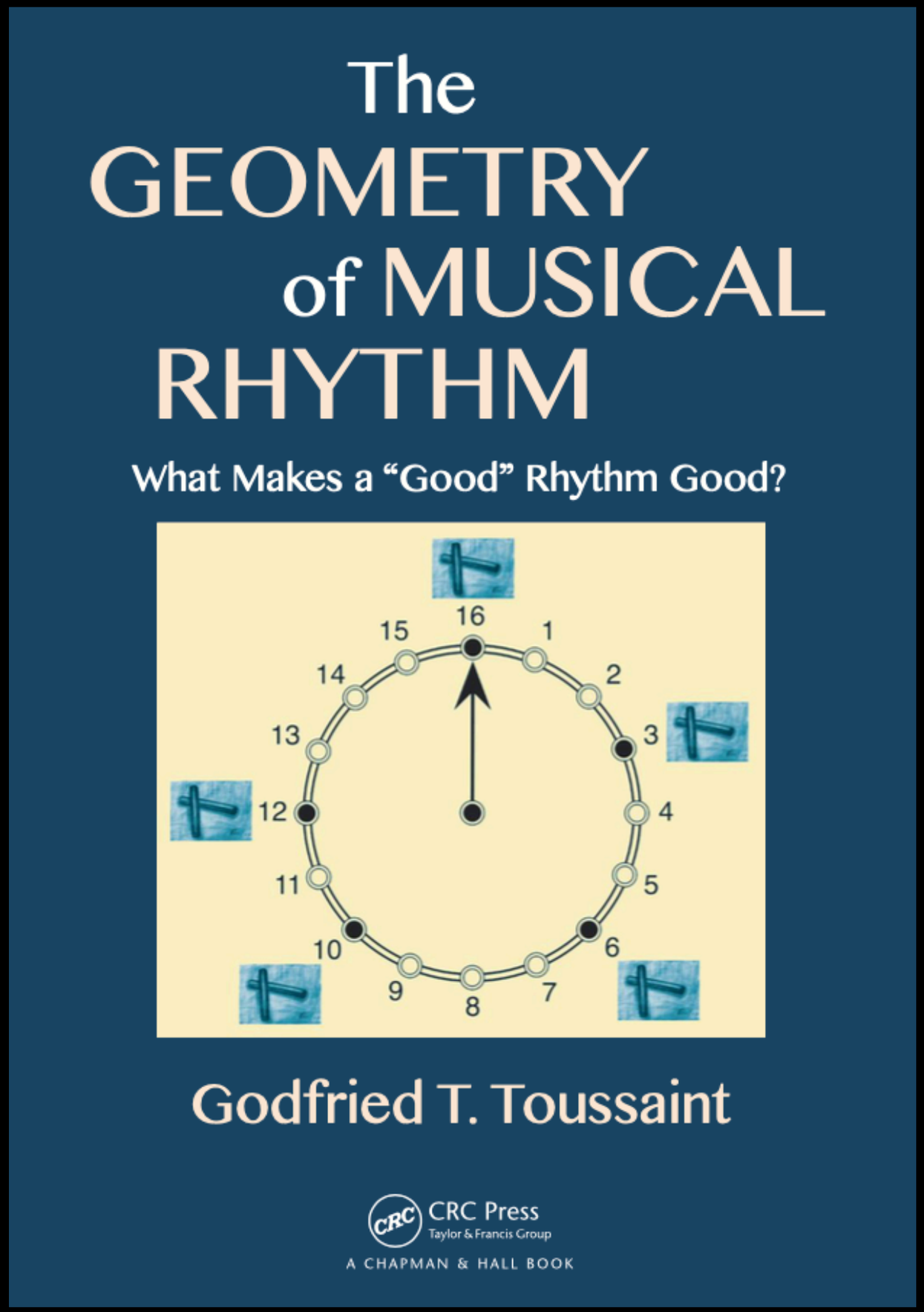
- **Rhythm is** *a convenience* **subclass of Array**
- **Sequencer** is an Object whose main instance variables are: **Gates**, NoteNumbers, Durations, Extra1, Extra2.

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 it is a **Sequencer**, the *noteIndex* of the Sequencer is incremented at every *stepDuration* and reset to 1 every time it reaches the size of *gates*

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

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



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

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)

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)
```

OpenSoundControl OSC

- Developed by Adrian Free and Matt Wright at CNMAT at the end of the 90s. Firs specification published in 2002.

- Flexile, fast and accurate alternative to the MIDI standard..

- Indipendent from the transport mechanism, OSC packets are typically sent and received thru UDP Sockets.

- ***Server/Client*** architecture The ***server*** sends the package, 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**

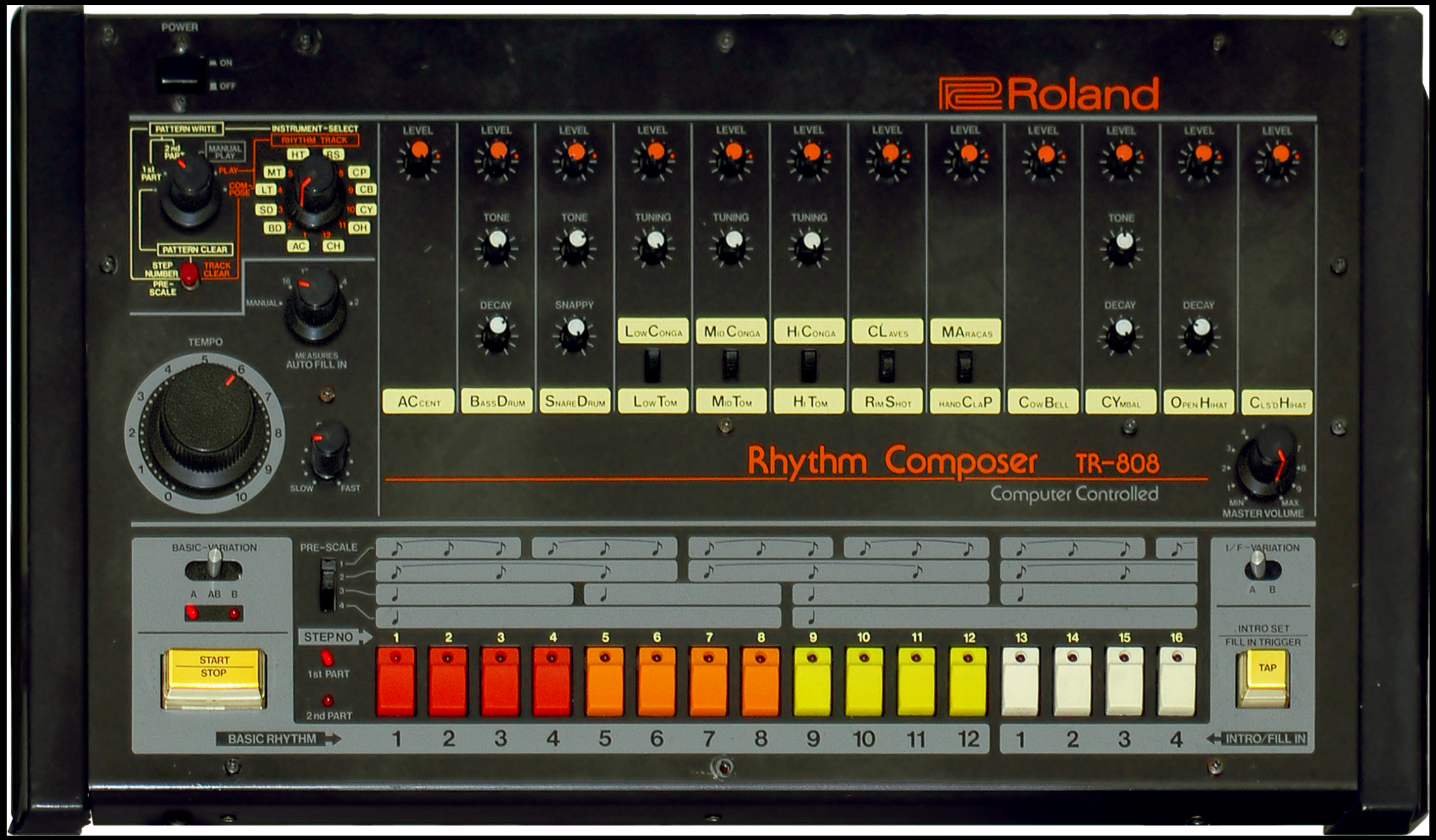
aNumber toLocal: aString. ▶ Send to the local host the message:

‘/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).

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

- **16 zeros**
- **64 randoms**
- **32 rumba**
- **16 quavers**

Scales!

- Sending a message with the name of a scale to the **Scale** class returns an array with the intervals of that scale. For example, Scale sakura returns #(0 1 5 7 8)

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

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.

MIDI integration

- In the spring of 2022, the **Pharo** team in Lille helped with the development of MIDI compatibility for the LiveCoding Package.

- A graduating student, **Antoine Delaby**, tutored by **Santiago Bragagnolo,** ported the MIDI output functionality of the C **PortMidi** library by the means of the *Unified ForeignFunctionInterface* provide by Pharo
- It is now possible to send out MIDI *noteOn* and *noteOff* messages from **Pharo** to external MIDI hardware connected to the host computer.
- The ***MIDISender*** object provided by the LiveCoding package is a convenience object modelled from the ***MIDIOut*** object of the ***ChucK*** programming language

aMIDISender playNote: aNoteNumber onChannel: aMIDIChannel duration: aDurationInSeconds

- aPerformance playMIDISequenceAt: aStepDuration rate: aNumberOfSteps on: aMIDISender

As supercollider frontend

- **Super Collider** is an environment and programming language originally released in 1996 by James McCartney for real-time audio synthesis and algorithmic composition
- It combines the object-oriented structure of **Smalltalk**, with a C-like syntax and features from functional programming language
- **SuperCollider**'s sound generation is bundled into an optimised command-line executable (named ***scsynth***) that is usually controlled within the **SuperCollider** language (***sclang***), bu that can be controlled via **OpenSoundControl**
- A full **Pharo** front-end for the ***scsynth*** server is being developed, to enable the user to generate synthesiser and effect and control the with **Pharo syntax**

Future Challenges

- **Tests and DebugDriverDevelopment!**
- Documentation, user guide, tutorials
- Optimisation!
- **MIDI input management.**
- Real polyrhythms (thanks Kasper for the complaint)
- **Graphical User Interface** easy creation
- Porting of **PortAudio** C library.
- An audio server running inside **Pharo**, on the model of **SuperCollider** ***scServer***.
- A full programming language inside **Pharo** capable not only of algorithmic composition but also of sound production and sound synthesis - communicating via OSC messages to the internal audio server.