PROGRAMMING MUSIC AND SYNTHESIZERS ON-THE-FLY WITH PHARO

Domenico Cipriani - ADC25

AGENDA

1. Foundations [45 minutes]
2. Rhythm and Melodies [75 minutes]
3. Sound Design with Phausto [75 minutes]
4. Performance [20 minutes]
5. Visual Interface and Control [25 minutes]

PART 1 : FOUNDATIONS

Domenico Cipriani - ADC25

What is live coding?

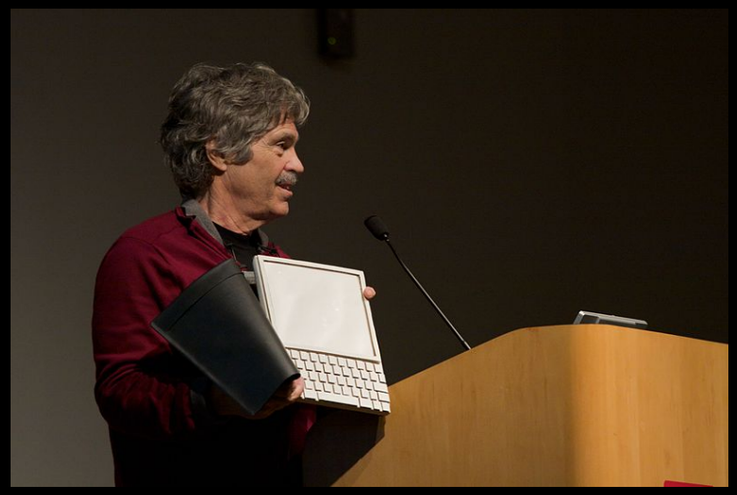
- **Live coding means writing and modifying code in real time to create music, visuals, or performances.**
- **The code itself becomes part of the live performance, often projected for the audience to see.**
- **It's about improvisation and creativity — composing and changing sounds or visuals on the fly.**
- Common in **electronic music and audiovisual art**, using languages like **SuperCollider, TidalCycles, or Sonic Pi.**
- Focuses on **process over perfection** — the act of coding *is* the art.

Pharo, Coypu, Phausto

PHARO	Free, open-source, general-purpose language + cross-platform ID
COYPU	API and Domain Specific Language for programming music on-the-fly
PHAUSTO	Library and API for DSP programming that enables sound generation within Pharo

Smalltalk

- A pure **Object-Oriented** Programming language developed at the *Learning Research Group* at Xerox PARC in the 1970s by Alan **Kay**, **Dan Ingalls**, **Adele Goldberg**, **Ted Kaeheler**.
- Designed for educational use following principles of *Constructionism*.
- Deeply influenced by ***Simula***, developed by **Ole-Johan Dahl** and **Kristen Nygaard** in the 1960s at the Norwegian Computing Center in Oslo.
- Not just a language, also an IDE users can inspect and modify.
- Programs written in Smalltalk are compiled into byte code and interpreted by a virtual machine.
- Smalltalk has influenced *Objective C*, *Ruby, SuperCollider*.



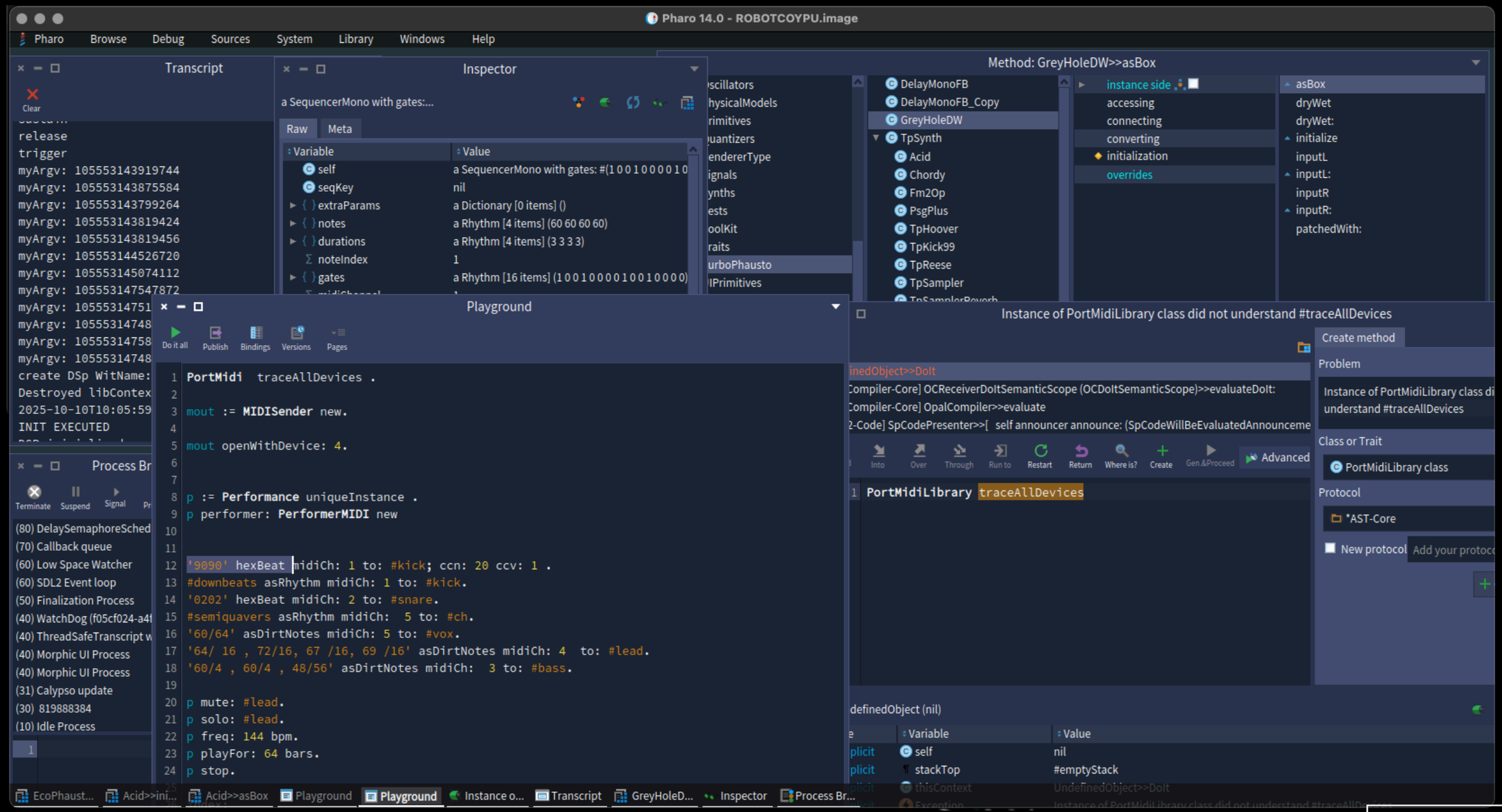
Install Pharo

Download the Pharo Launcher to download a Pharo Image:

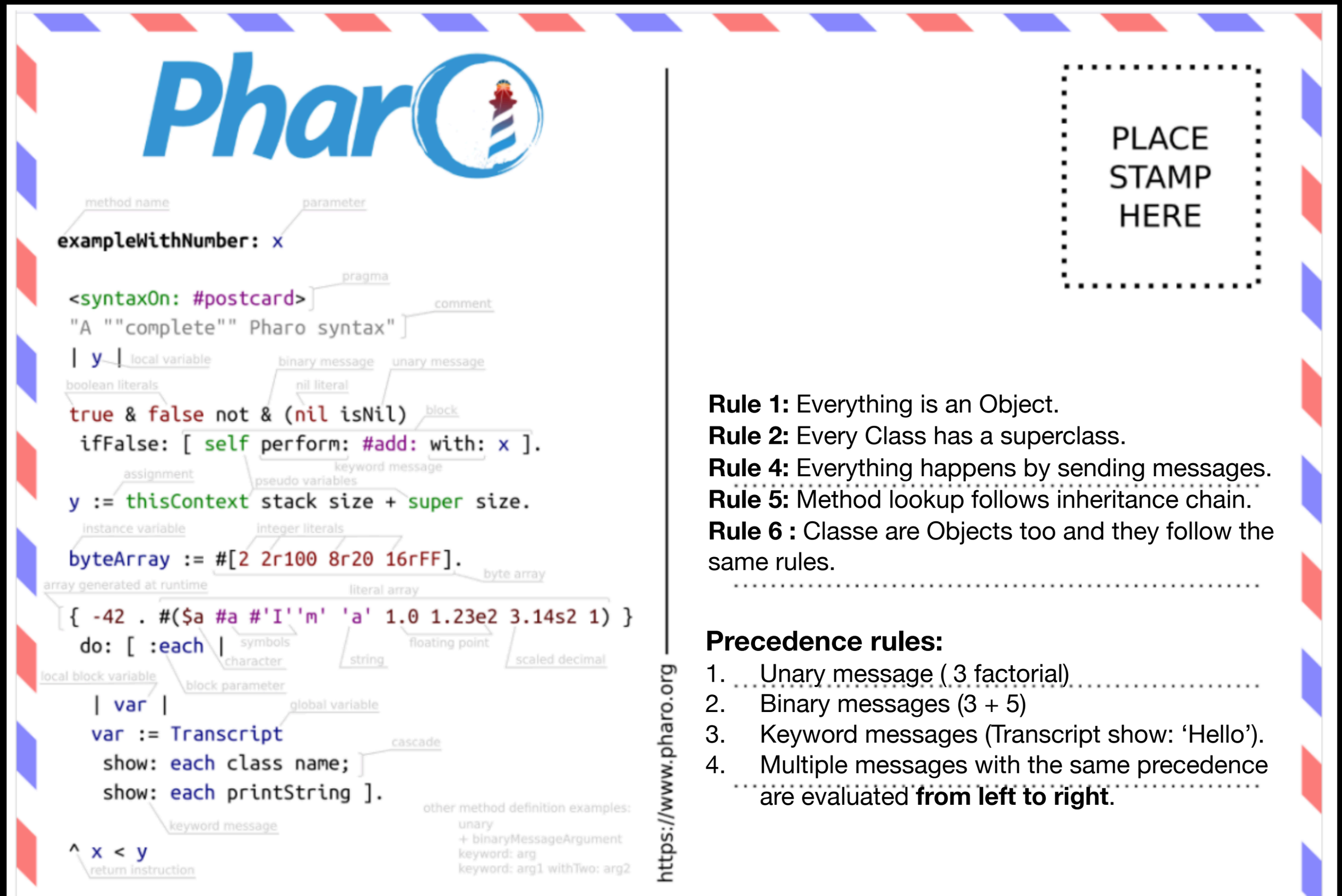
https://pharo.org/download

A Pharo image is an object space + Pharo Core Libraries + the virtual machine

The Pharo IDE



Pharo syntax on a postcard



Installing Coypu and Phausto

Go to the *Coypu* GitHub repositories:

https://github.com/lucretiomsp/Coypu

Copy the *Metacello* script into your *Playground*.

```
Metacello new
    baseline: 'Coypu';
    repository: 'github://lucretiomsp/coypu:master';
    load
```

Select all the text and evaluate (CMD/CTRL + D).
Coypu* already comes together with *Phausto

PART 1 : FOUNDATIONS

Domenico Cipriani - ADC25

Coypu's anatomy

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 downbeats.
#downbeats asRhythm.
'60 , ~ , ~ , ~ , 60 , ~ , ~ , ~ , 60 , ~ , ~ , ~ , 60 , ~ , ~ , ~' asDirtNotes.
#(1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0) asSeq.
```

Connecting with audio servers

Creating rhythms

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

BINARY	100010001000
HEXADECIMAL	8888
PHARO	#(1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0)
COYPU	#downbeats asRhythm
COYPU	16 downbeats
COYPU	‘8888’ hexBeat

Creating rhythms

- #(1 0 0 1 0 0 0 1 0 0 1 0 1 0 0 0)
- #rumba asRhythm
- ‘9128’ hexBeat

- #(1 0 1 0 1 0 1 0 1 0 1 0 1 1 1)
- 12 quavers, 4 semiquavers
- ‘AAAF’ hexBeat

The ‘musky’ notation

PART 3 : SOUND DESIGN WITH PHAUSTO

Domenico Cipriani - ADC25

Start your engine

Let’s build a synth

Drive Phausto with Coypu

Build your own library

PART 4 : PERFORMANCE

PART 5 :
VISUAL INTERFACE

Domenico Cipriani - ADC25