

PROGRAMMING MUSIC AND SYNTHESIZERS ON-THE-FLY WITH PHARO



Domenico Cipriani - ADC25

Inria



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

Inria



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.

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

The screenshot displays the Pharo 14.0 IDE interface, titled "Pharo 14.0 - ROBOTCOYPU.image". The interface is divided into several panes:

- Transcript:** Shows a list of messages sent to objects, including "myArgv:" and "create DSP WitName: Destroyed libContext".
- Inspector:** Displays the state of a "SequencerMono" object, showing variables like "self", "seqKey", "extraParams", "notes", "durations", "noteIndex", and "gates".
- Playground:** Contains a code editor with a MIDI-related script. The script includes comments and code for creating a MIDI sender, opening a device, and sending MIDI notes.
- Process Browser:** Lists running processes, including "DelaySemaphoreSched", "Callback queue", "Low Space Watcher", "SDL2 Event loop", "Finalization Process", "WatchDog", "ThreadSafeTranscript", "Morphic UI Process", "Calypso update", and "Idle Process".
- Method Browser:** Shows the "GreyHoleDW" method, with a list of methods like "accessing", "connecting", "converting", "initialization", and "overrides".
- Exception Viewer:** Displays an error message: "Instance of PortMidiLibrary class did not understand #traceAllDevices".

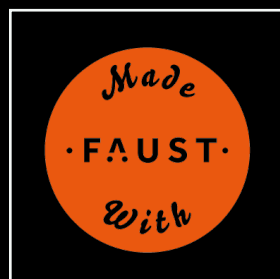
Domenico Cipriani - ADC25

Inria



Pharo syntax on a postcard

- Everything is an Object.
- Every object is an instance of a class.
- Every class has a superclass.
- Object only communicates between them by sending messages
- Methods look-up follows the inheritance chain.
- The class Object is at the beginning at the inheritance chain and defines almost 400 methods.



Pharo syntax on a postcard

- POSTCARD IMAGE



Installing Coypu and Phausto

Go to the *Coypu* GitHub repositories:

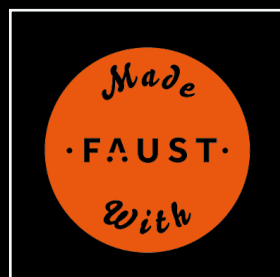
<https://github.com/lucretiomsp/Coypu>

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

Inria



Coypu's anatomy



Connecting with audio servers



Creating rhythms

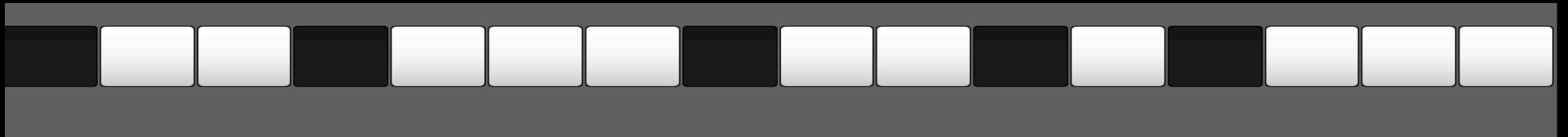


A rhythm can be represented as an array of 0s and 1s, where each 1 represents a trigger.



- **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
- **LIVECODINGTALK:** `'9128'` hexBeat



- `#(1 0 1 0 1 0 1 0 1 0 1 0 1 1 1)`
- `12` quavers, `4` semiquavers
- `'AAAF'` hexBeat

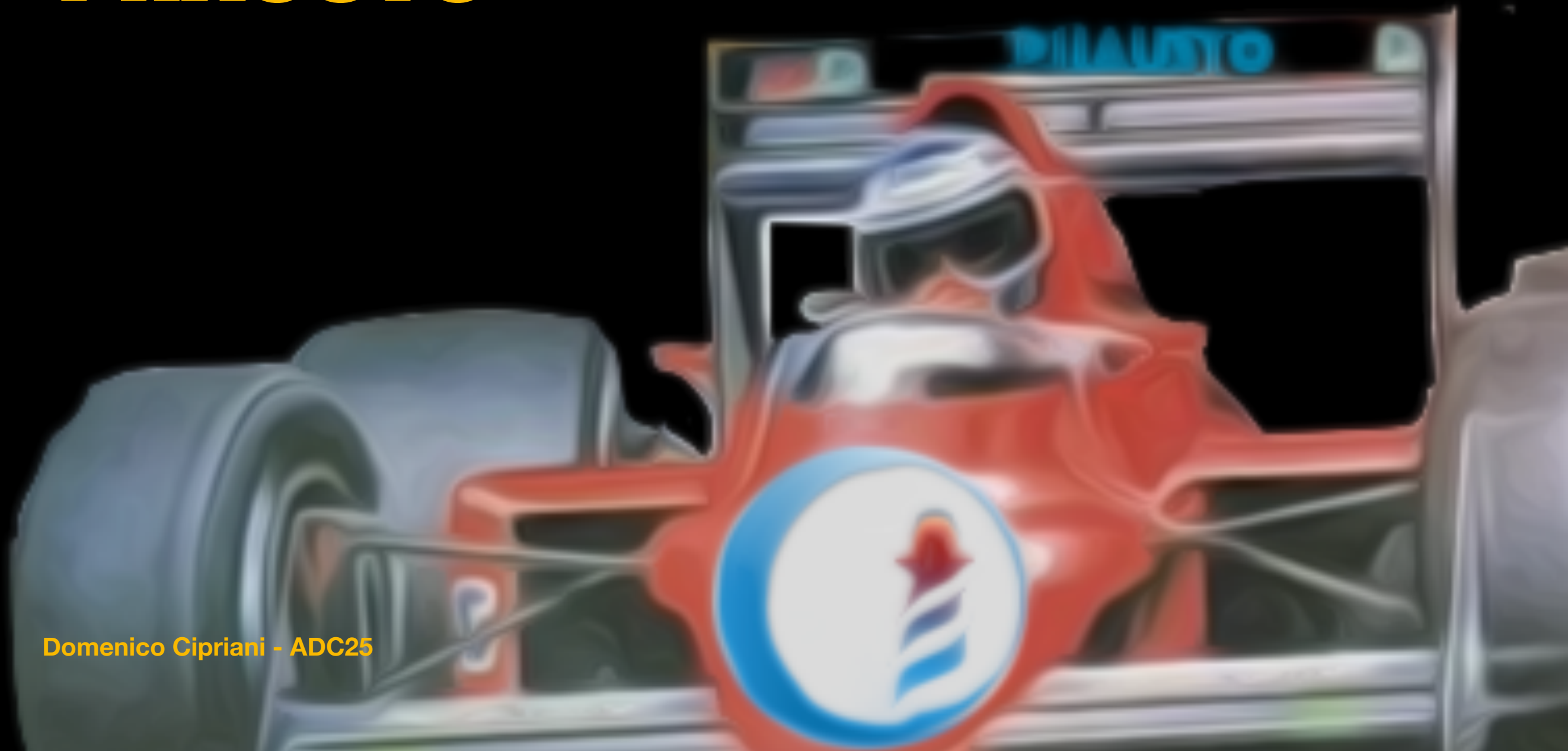
The 'musky' notation

Domenico Cipriani - ADC25

Inria



PART 3: SOUND DESIGN WITH PHAUSTO



Domenico Cipriani - ADC25

Start your engine

Domenico Cipriani - ADC25

Inria



Let's build a synth



Drive Phausto with Coypu

Domenico Cipriani - ADC25

Inria



Build your own library

Domenico Cipriani - ADC25

Inria



PART 4: PERFORMANCE



Domenico Cipriani - ADC25

Inria



PART 5: VISUAL INTERFACE

