FoxDot >> Coding music

Part 1: Introduction for beginners

Tutor: Jens Meisner
Computerkuenstler und Mediengestalter

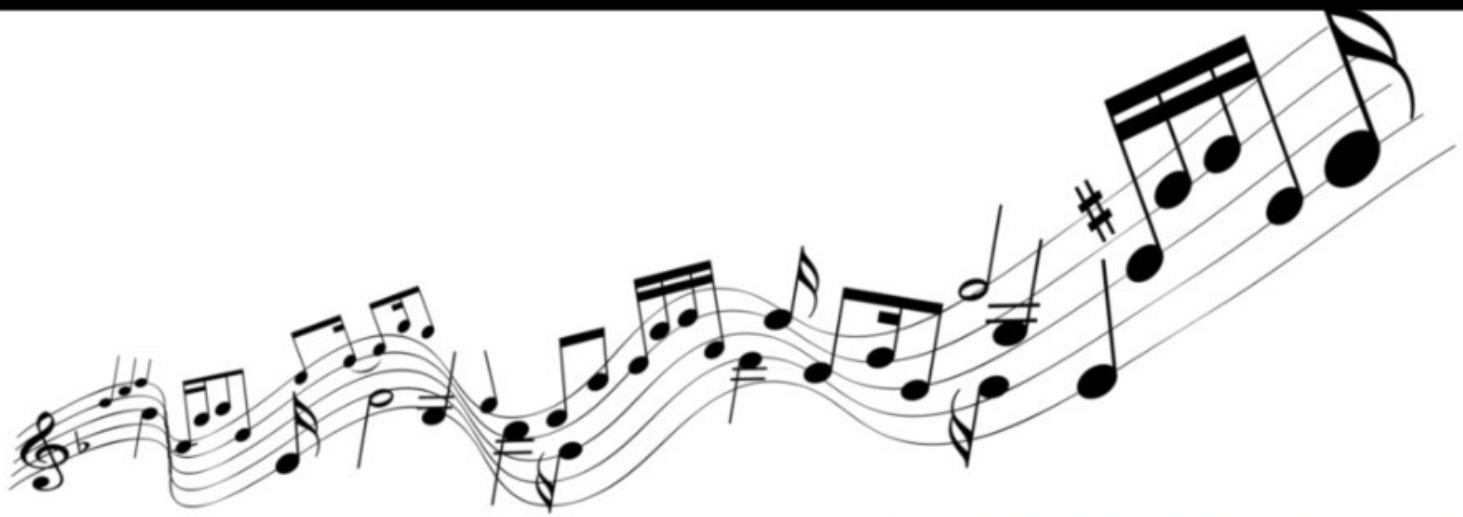
“Live Coding is a new direction in electronic music and video, and is getting somewhere interesting...

Live Coders exposes and rewire the innards of software while it generates improvised music.”
- toplap.org -

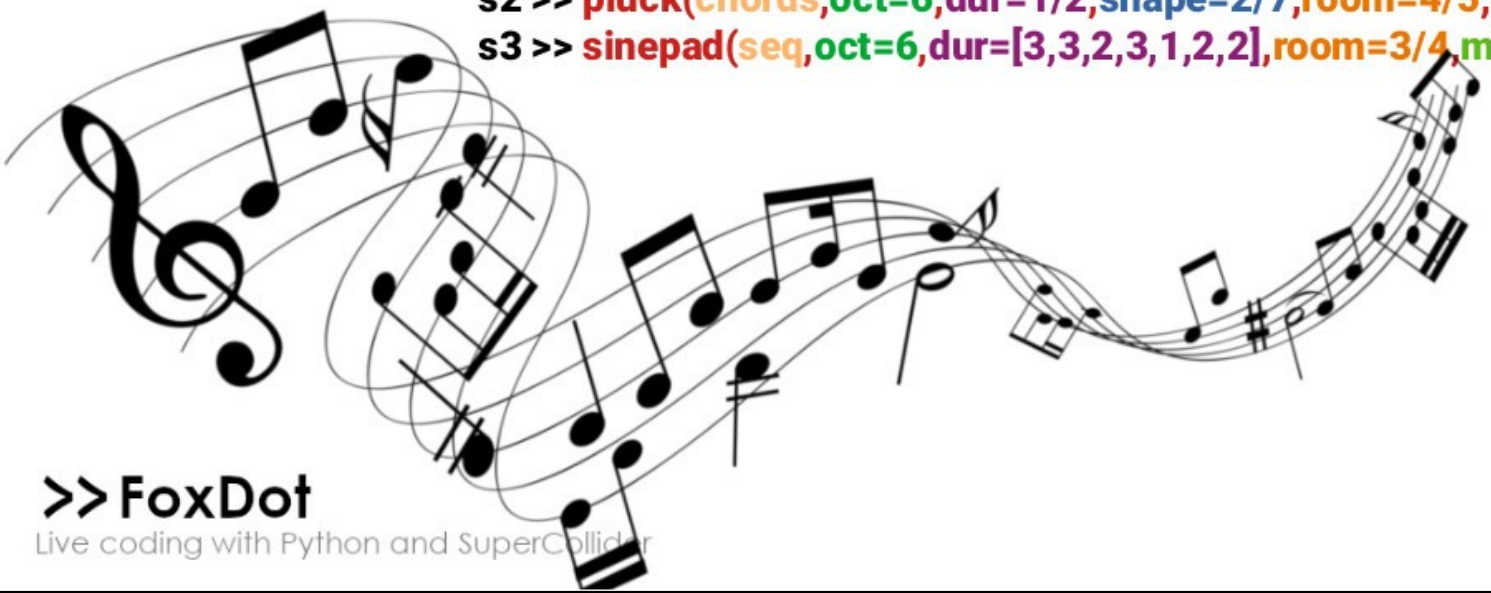
>>FoxDot
Live coding with Python and SuperCollider

An Introduction >>

Why using code for music ?



```
s1 >> klank(bassline,oct=4,dur=[6,4,2,2,2],shape=2/5,amplify=3/5,amp=var([0,1],8))
s2 >> pluck(chords,oct=6,dur=1/2,shape=2/7,room=4/5,mix=1/2,amplify=1/2,amp=1)
s3 >> sinepad(seq,oct=6,dur=[3,3,2,3,1,2,2],room=3/4,mix=1/2,amplify=4/5,amp=1)
```

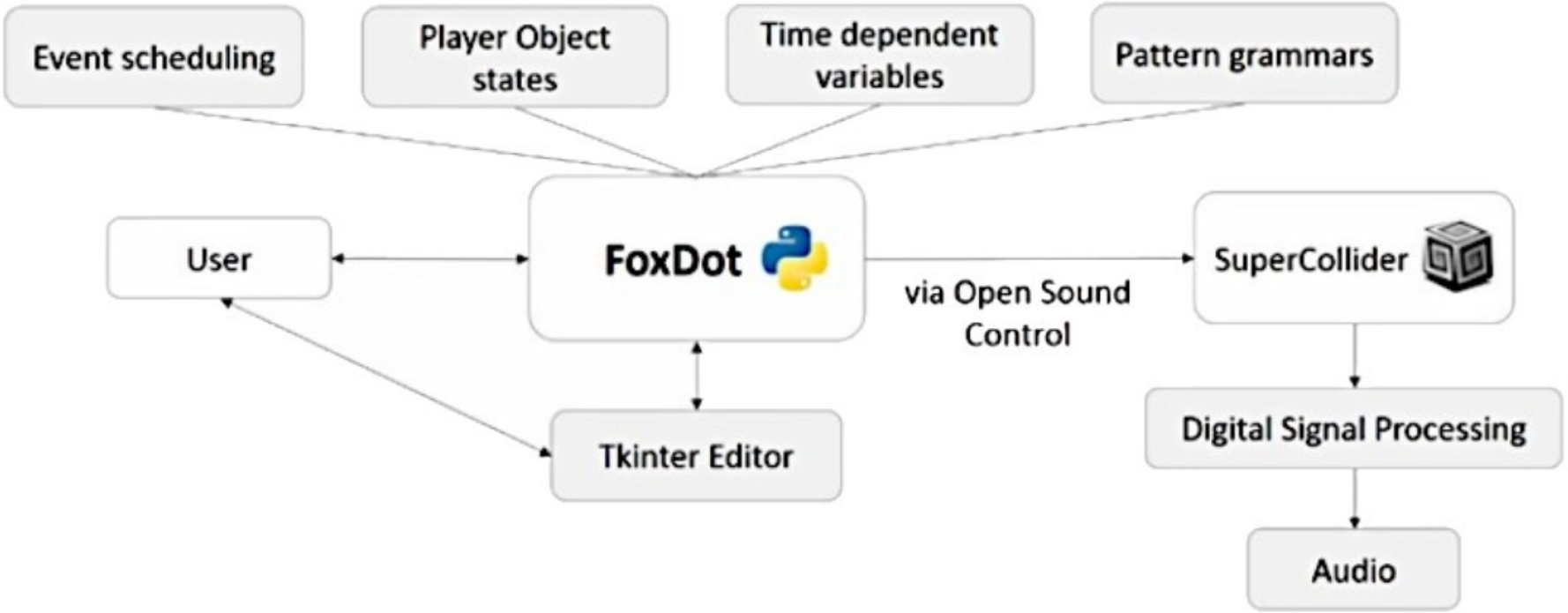


>> FoxDot

Live coding with Python and SuperCollider

An Introduction >>

What is FoxDot?


Event scheduling
Player Object states
Time dependent variables
Pattern grammars
User
FoxDot
via Open Sound Control
SuperCollider
Tkinter Editor
Digital Signal Processing
Audio

Troop - Ryan@localhost:57890

File Edit Code Constraints Help

```
1 # Whos here to dance?
2
3 d1 >> play("x", dur=PDur([2,2,3,5,7],8), sus=1, pan=PStep(7, P+(-1,1))) + 2
4
5 b1 >> bass(dur=16, sus=4, room=1, mix=1, shape=0.5, lpf=linvar([1000,8000],12)) + (0, PRand
  ([6,7,9]))
6
7 l1 >> play("shroom", rate=PRand([5,10]), shape=.1).every(6, "stutter", 4, pan=[-1,1])
8
9 l2 >> sitar(PWalk(), oct=[2,3], dur=[4,2,2], sus=[1,2,3,4,5], slide=0.2) + [1,3,5]
10 Ryan
11 z1 >> soprano(l2.degree + (2,[4,6,7]), dur=[8,4,2,1], oct=4, sus=2, blur=4, amp=)
12                                                                          loz
13 Scale.default=("mixolydian")
14
```

```
loz : z1 >> soprano(l2.degree + (2,[4,6,7]), dur=[8,4,2,1], oct=7, sus=2
, blur=4)
loz : z1 >> soprano(l2.degree + (2,[4,6,7]), dur=[8,4,2,1], oct=5, sus=2
, blur=4)
Ryan : d1 >> play("x", dur=PDur([2,2,3,5,7],8), sus=1, pan=PStep(7, P+(-
1,1))) + 2
loz : z1 >> soprano(l2.degree + (2,[4,6,7]), dur=[8,4,2,1], oct=4, sus=2
, blur=4)
```

>> FoxDot

Live coding with Python and SuperCollider

Getting Started With FoxDot >>

If you want to know more about a function or a class,

type in help + the Python object in brackets

help(object)

In example, if you want know more about the “pluck” instrument, type in:

help(pluck)

Getting Started With FoxDot >>

Player Objects

“pluck” is an Python object, that represents a synthesizer definition

in SuperCollider called SynthDef.

To see all available SynthDefs, type in:

print(SynthDefs)

>> Before we start, some basics >>

- To execute a single line, press ALT + ENTER
- To execute an entire block of lines, press CTRL + ENTER
- If you want to stop all sound playing, press CTRL + .
- If you want to stop a single line, add .stop() at the end, and press ALT + ENTER:

  To create a working instrument, you need name it first. You only can use one letter with a following number: Try it and all above with an instrument of the list

s1 >> pluck()

Getting Started With FoxDot >>

Patterns

Getting Started With FoxDot >>

TimeVars

Getting Started With FoxDot >>

“play” and “loop” Players

- Beside synthesizer instruments in SuperCollider called SynthDefs, there are 2 other Players to be used in FoxDot
- “play” and “loop” are actually audio sample player
- “play” uses letters and characters in brackets to call samples, that are part of the FoxDot basic installation
- e.g.:

b1 >> play("x-o-")

Use {} [] () <> like so: **b1 >> play("x-o[-{ox}]")**

{} - Random, [] - All-in-one step, () - In turn, <> - Simultaneously

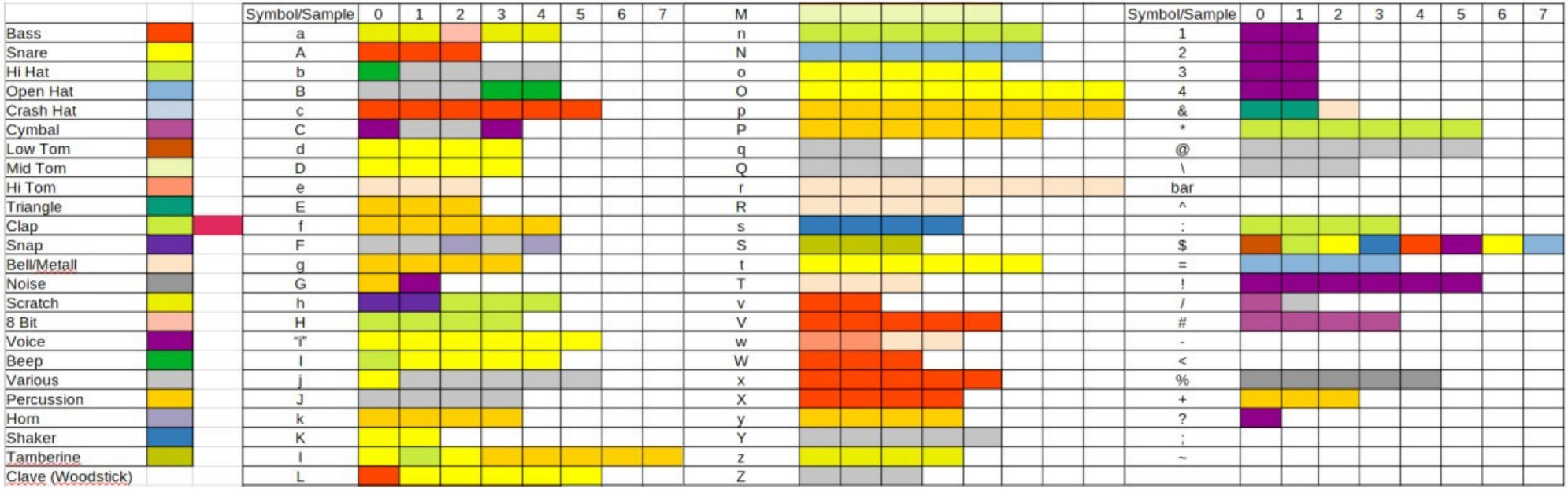
- There are many variables, that can be add to the “play” Player by adding it behind the quotes separated by a comma, like so:

b1 >> play("<x-o-><..+.+.[.+]>", pan=(-1, 1))

- One of this variables is “sample”.
- Each character relates to a folder of files arranged in alphabetical order. To select a different file, use the sample keyword:

b1 >> play("x-o-", sample=1)

- The following list shows a loose categorization of currently available samples



>> FoxDot

Live coding with Python and SuperCollider

- The “loop” Player can be used to implement your own samples, like a singing voice, an analog instrument, or you have your own drum samples

- The variable dur is used to set a duration of the loop

l1 >> loop("path/to/my/file.wav", dur=32)

- You can put files in a special folder located in FoxDot/snd/_loop_ which can be opened by going to “Help & Settings” and then “Open Samples Folder” from the FoxDot menu.

- You don’t need to supply the full path (or extension) for files in this folder:

l1 >> loop("my_file", dur=4)

First, A Little Bit Music Theory >>

Song Structure

- Common Structures for a song is as followed:

  Intro (4 Bars) - **Verse** (8 -16 Bars) + **Prechorus** (Optional) - **Chorus**(8 - 16 Bars) - **2nd Verse** (8 – 16 Bars) + **Prechorus** (Optional) - **Chorus** (8-16 Bars)

- The following is an example of a song structure in common electronic music:

Intro	Break	Buildup	Drop	Break	Buildup	Drop	Outro
16 Bars	16 Bars	4/8/16 Bars	16 Bars	16 Bars	4/8/16 Bars	16 Bars	16 Bars

First, A Little Bit Music Theory >>

Chords and Notes

Chords

Melodies – From Chords To Melody

Melodies – From Melody To Chords

Melodies – Add a counter melody (Arpeggio)

Melodies – From Chord Progression to Bass lines

Melodies – From Bass lines to Chord Progression

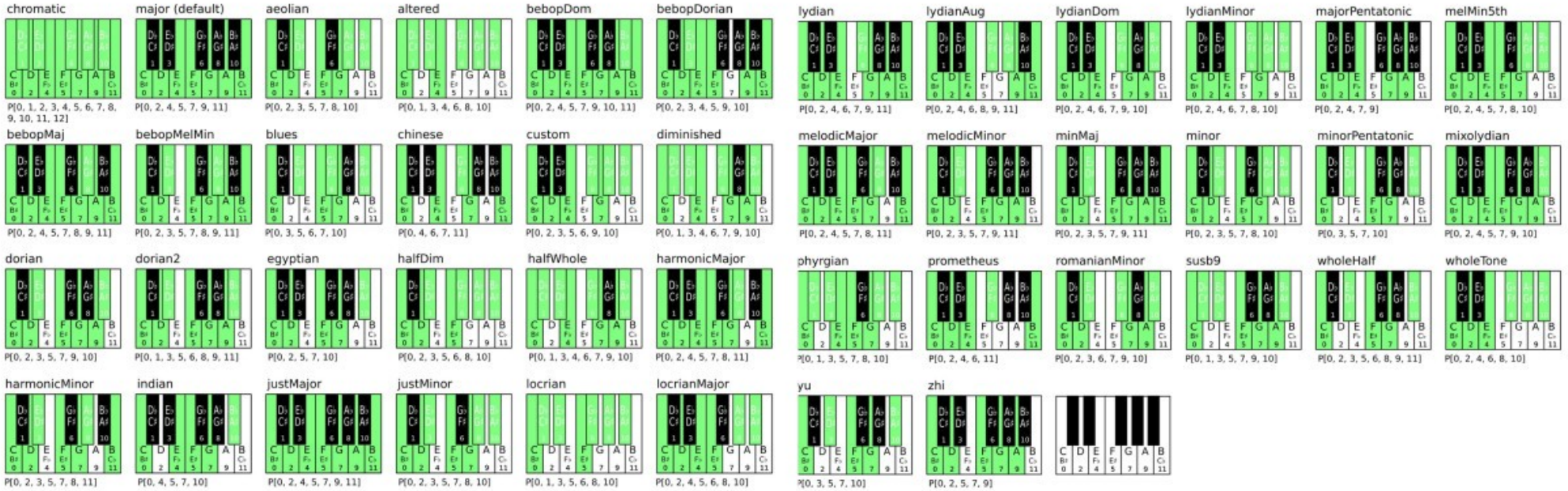
First, A Little Bit Music Theory >>

Scales and Modes

Using Scales

Using Modes

- The schematics shows all modal Scales available by name in FoxDot



>>FoxDot

Live coding with Python and SuperCollider

First, A Little Bit Music Theory >>

BeatBox

Live Jam >> Let’s make music together

>> FoxDot

Live coding with Python and SuperCollider

Live Jam >>

Connect and get Troop started

Live Jam >>

Connect to the Wifi Network

SSID: **Wifi4DS**
Password: **25M4Ei7[89**

- Open a terminal, execute “python /Path/To/Script/run-client.py”, fill in the form and press “Ok”