Programming beeps and boops

Ein Livecoding-Workshop

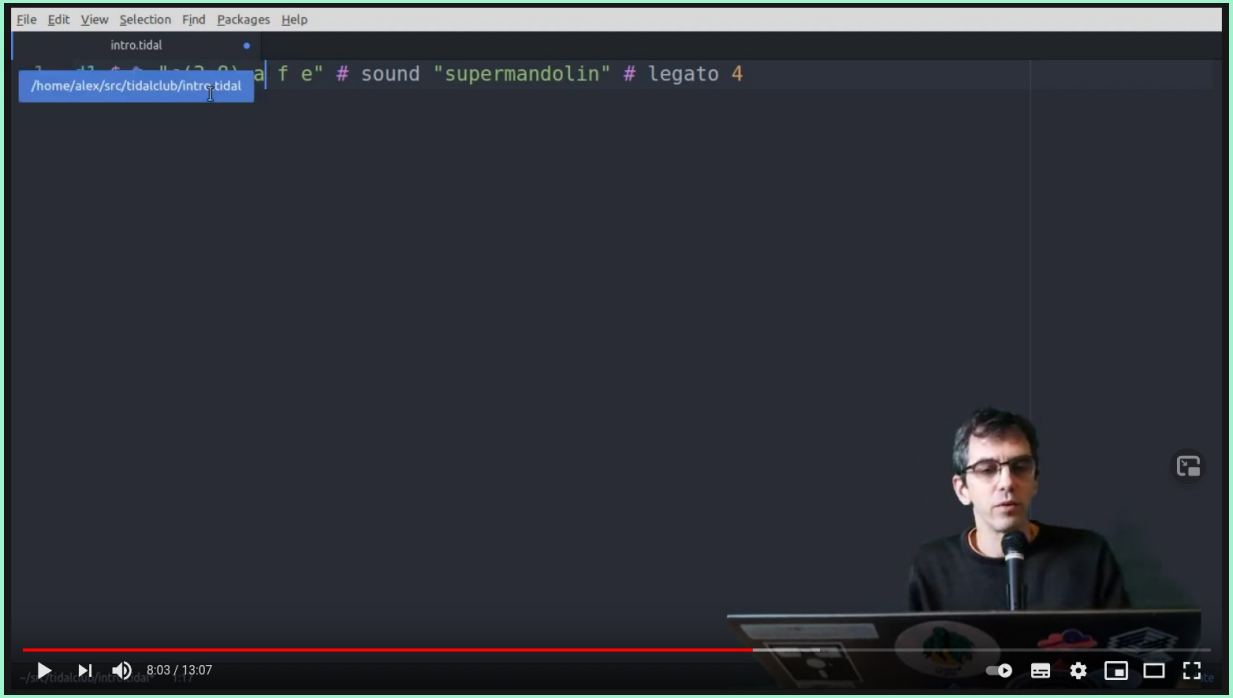
Dortmund, April 2021

lislis <mail@lislis.de>

Was machen wir heute?

Livecoding | Algorave

Musik live coden



https://www.youtube.com/watch?v=-QY2x6aZzqc

Installation

TidalCycles

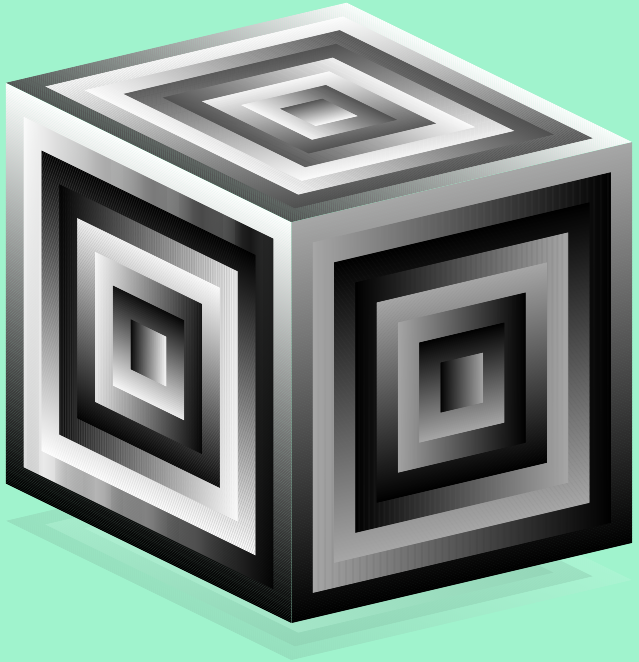
https://tidalcycles.org/Installation

- SuperCollider (SuperDirt)
- Haskell
- Tidal
- Atom

SuperCollider

https://supercollider.github.io/

- Real-time Audio Synthesis
- Seit 1996 in Entwicklung!



Haskell

https://www.haskell.org/

Funktionale Programmier-sprache, die wir nicht lernen müssen \o/

Tidal

https://tidalcycles.org/

- Pattern-Engine für musikalische Komposition
- Redet mit SuperDirt (Synthesiser)

Atom

https://atom.io/

Texteditor mit
Tidal-Plugin

Installation testen

- SuperCollider starten
- Im Hauptfeld folgendes eingeben und evaluieren (`Shift+Enter`):

  ```
  SuperDirt.start
  ```

- Atom und eine neue Datei erstellen und als **test.tidal** speichern

Hello beep!

- In **test.tidal** schreiben:

  d1 $ sound "bd"

  hush

- Zum starten die erste Zeile evaluieren
- Zum stoppen die zweite Zeile evaluieren

Probieren geht über studieren

Sounds machen

- Grundlegendes Format für einen Sound

```
d1 $ sound "drum"
```

- Sound stoppen

```
d1 $ silence
```

- Anderer Sound aus dem Sampleset

```
d1 $ sound "drum:1"
```

Sounds machen

- Grundlegendes Format für einen Sound

```
d1 $ sound "drum"
```

- Sound stoppen

```
d1 $ silence
```

- Anderer Sound aus dem Sampleset

```
d1 $ sound "drum:1"
```

Samples

- flick sid can metal future gabba sn mouth co gretsch mt arp h cp cr newnotes bass hc tabla bass0 hh bass1 bass2 oc bass3 ho odx diphone2 house off ht tink perc bd industrial pluck trump printshort jazz voodoo birds3 procshort blip drum jvbass psr wobble drumtraks koy rave bottle kurt latibro rm sax lighter lt arpy feel less stab ul
- SuperCollider menu: 'File > Open user support directory > downloaded-quarks > Dirt-Samples'

Sequences

- Schreibe mehrere Sounds in ein Pattern um eine Sequence zu erzeugen
- Vergleiche diese Sequences

```
d1 $ sound "bd hh sn hh"
```

```
d1 $ sound "bd bd hh bd sn bd hh bd"
```

Cycle

- Im Hintergrund läuft ein kontinuierlicher Timing-Loop, der Cycle
- Default ist 1 Cycle pro Sekunde
- Sounds in der Sequence werden gleichmäßig über den Cycle verteilt
- Du kannst die cps selst setzen

```
setcps 0.6
```

Mehr

- Du kannst mehrere Sequenzen mit d1 - d9 gleichzeitig spielen lassen
- Mit **hush** kannst du alle Sequenzen gleichzeitig stoppen
- Mit **solo** kannst nur einen bestimmten Channel spielen lassen. Mit **unsolo** wird es aufgehoben
- Mit **~** kannst du eine Pause angeben

Subsequence

- Mit `[]` kannst du Subsequenzen in einem Step definieren

```
d1 $ sound "bd [bd cp] bd bd"
```

- `[]` können beliebig verschachtelt werden

Multiply divide

- Mit `*` kannst du einen Step in der Sequence wiederholen (schneller spielen)
- Mit `/` wird der Step langsamer gespiel (weniger wiederholt)
- Beides geht auch mit Subpatterns []

Schedule across cycles

- Mit `<` und `>` können Patterns über mehrere cycles hinweg definiert werden

```
d1 $ sound "bd <sd cp arpy>"
```

Effekte

- In Tidal gibt es Effekte die verändern, wie etwas klingt
- Sie werden auch Control Patterns genannt
- Beispiel vowel

```
d1 $ sound "drum drum drum drum" # vowel "a"
```

Structure + Sound

- Vergleiche

```
d1 $ sound "drum drum drum drum" # vowel "a o e"
```

und

```
d1 $ vowel "a o ~ i" # sound "drum"
```

- Das Pattern auf der linken Seite gibt immer die Struktur vor, das Pattern rechts wird dann gematcht

Mehr Control Patterns

- **gain** verändert die Lautstärke

```
d1 $ sound "bd hh sn:1 hh sn:1 hh" # gain "1 0.7 0.5"
```

- **speed** verändert die Wiedergabegeschwindigkeit des Samples
- **up** erhöht den Pitch des Samples

Mehr mehr Control Patterns

- **pan** gibt uns Kontrolle über Stereoeffekte
- **shape** fügt Verzerrung hinzu
- Mehr mehr mehr in der Doku

https://tidalcycles.org/index.php/Category:Control_Functions

Continuous patterns

- **sine** is a continuous pattern following a sine curve from 0 to 1 and back

```
d1 $ sound "bd*32" # gain sine
```

- Es gibt auch **tri**, **saw** und **rand**

Experimentieren!

Was gibt es noch?

- Sonic Pi **https://sonic-pi.net/**
  - Set mit Sam Aaron
  - **https://www.youtube.com/watch?v=G1m0aX9Lpts**
- Overtone **https://overtone.github.io/**
  - Set mit Joseph Wilk
  - **https://www.youtube.com/watch?v=_S5FZ2CcLlc**

Misc

- Sound programmieren in Ruby allgemein **https://ruby-synth.fun/**
- Algorave Events **https://algorave.com/**
- Tidal artists **https://tidalcycles.org/Showcase**
- Ein Beispiel **https://cargocollective.com/tiemposdelruido/music**

Beep boop!

Danke!

Beispiele aus **https://tidalcycles.org/Basic_Patterns**

lislis <mail@lislis.de>