Elektronische Musik machen mit SuperCollider

modern_dragon

01. Juni 2024

Elektronische Musik machen mit SuperCollider

Appetithäppchen

Appetithäppchen

Auszug aus “Algorithmic Fantasy in A flat” (2023)

Wer bin ich & was mache ich

Wer bin ich & was mache ich

modern_dragon aka Camilla (sie/ihr // they/them)
musikalische Wurzeln in der klassischen Musik
seit 2018 Experimente mit SuperCollider

Organisatorisches

Organisatorisches

Material zum Kurs: https://the-emergent.de/gpn22.html

Was ist SuperCollider?

Was ist SuperCollider?

- “a platform for audio synthesis and algorithmic composition”
- eine Kombination aus
  - einer Programmiersprache (sclang),
  - einem Audio-Server (scsynth) und
  - einer IDE (scide)
- entwickelt ab 1996 von James McCartney
- Open Source (GPL 3.0) seit 2002
- interpretierte Programmiersprache
- objektorientiert

Warum mag ich SuperCollider?

- komplett mit Code kontrollierbar
- viele Interaktionsmöglichkeiten (z. B. MIDI, OSC, Arduino)
- sehr detaillierte Möglichkeiten zum Sound Design
- inspiriert einen ganz eigenen Zugang zu elektronischer Musik
- kreativ mit Zufall arbeiten
- vergleichsweise ressourcensparsam
- Lebendige Community von Menschen, die Plugins entwickeln

Die SC IDE

Aufbau und Hilfesystem

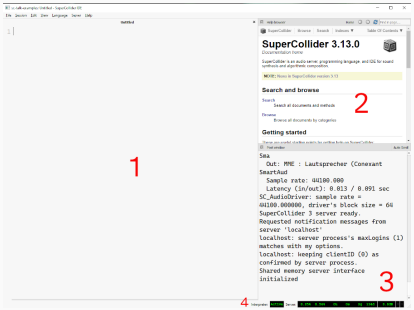

Figure 1: Screenshot: SC IDE

1: Arbeitsbereich 2: Hilfe-Browser 3: Post Window 4: Statuszeile

Werte rechts von “Server”

- durchschnittliche CPU-Auslastung
- Spitzen-CPU-Auslastung
- aktive UGens
- aktive Synths
- Gruppen
- geladene SynthDefs

Code ausführen

- Zeilenweise: Umschalt+Enter
- Blockweise: Strg+Enter bzw. Cmd+Enter
- Markierten Code: Umschalt+Enter

Hilfesystem

- Tastenkürzel **Strg+D** bzw. **Cmd+D**
- Hilfe-Dateien haben ein Inhaltsverzeichnis!
- leider sind nicht alle Teile der SuperCollider-Bibliothek dokumentiert

Hallo SuperCollider!

Safer Krachmachen

Bevor wir mit Klangexperimenten anfangen:

- SuperCollider hat keinen eingebauten Sicherheits-Mechanismen, die super laute oder unangenehme Klänge verhindern.
- Kann (im Extremfall!) Ohren und Equipment (Kopfhörer, Lautsprecher) beschädigen!

- im Zweifelsfall Kopfhörer absetzen
- und/oder Systemlautstärke herunterdrehen
- Server Meter im Auge behalten
- Mitdenken beim Sound Design
- “Not-Aus”-Tastenkürzel: **Strg+. bzw. Cmd+.**

Das “Hallo Welt” von SuperCollider

```
s.boot;
{SinOsc.ar(440, 0, 0.5)}.play;
```

Experimentierphase

Sucht euch aus den ersten paar Beispielen in der Datei beispiele.scd eins aus und spielt damit für die nächsten 5-10 Minuten herum.

Grundlagen der Syntax

Grundlagen der Syntax

- Codeblöcke in runden Klammern ()
- Code-Statement wird mit ; abgeschlossen
- Funktionen {}
- Arrays []
- Kommentare:
  - // einzeiliger Kommentar
  - /* mehrzeiliger Kommentar */

Variablen

- globale Variablen
  - einzelne Kleinbuchstaben `a-z`
  - traditionell reservierter Buchstabe `s` = lokaler Server
  - beliebig lang: mit vorangestellter Tilde
    - z. B. `~grundton`
    - muss mit einem Kleinbuchstaben anfangen

- lokale Variablen
  - innerhalb eines Codeblocks/einer Funktion
  - mit dem Schlüsselwort var deklarieren:
    - `var bsp = 10;`

Argumente

- Variablen einer Funktion, die zur Laufzeit “von außen” geändert werden können
- Schlüsselwort `arg`
- normalerweise ganz am Anfang der Funktion

Nützliche Visualisierungen

- Server Meter (s.meter)
- Oszilloskop (s.scope)
- FreqScope (s.freqscope)
- Server Tree (s.plotTree)
- .plot(): Wellenformen visualisieren

Funktionen, die Klänge erzeugen

Oszillatoren

zum Beispiel:

- `SinOsc.ar`;
- `LFTri.ar`;
- `Saw.ar`;
- `VarSaw.ar`;
- `Pulse.ar`;

Rauschen

- `WhiteNoise.ar`;
- `PinkNoise.ar`;
- `BrownNoise.ar`;

Filter

- `LPF.ar()`: Tiefpass
- `HPF.ar()`: Hochpass
- `BPF.ar()`: Bandpass
- `BRF.ar()`: Band Reject/Notch-Filter

Filter-“Explosionen” verhindern

- Cutoff-Frequenzen nahe 0 $\rightarrow$:((
- Abhilfe: .clip() begrenzt Eingabewerte auf einen zulässigen Wertebereich
- bei Frequenzen z.B. zwischen 20 und 20000 Hz

Parameter per Zufall oder Maus steuern

- Zufallsfunktion: `LFNoise2.kr`
- `MouseX.kr`
- `MouseY.kr`

Nützliche Methoden für Tonhöhen

- midicps <> cpsmidi
- midiratio <> ratiomidi
- Scale-Bibliothek

.ar, .kr – was ist das schon wieder?

- “Auflösung”, mit der die Funktion ausgeführt wird
- vergleichbar mit Bildauflösung!
- ar = “audio rate” (entspricht der Samplerate des Audioservers)
  - für Funktionen, die direkt Klang erzeugen
- kr = “control rate”
  - abhängig von Blockgröße = Zahl der Samples in einer Kontroll-Periode
  - (so viele Samples fasst der Server zu einem Block zusammen)
  - bei 48kHz = 750 mal pro Sekunde neu berechnet
  - für Funktionen, die nicht direkt Klang erzeugen
  - z. B. zum Modulieren von Parametern anderer Funktionen

Experimentierphase II

Experimentierphase II

Aufgabe: Nehmt euch einen der besprochenen Oszillatoren und experimentiert damit oder mit den Beispielen aus den Hilfedateien.

SynthDefs: Die Sound Design-“Blaupausen” in SuperCollider

Was ist eine SynthDef?

- “Konstruktionsanweisung” für einen Klang (oder Effekt)
- enthalten eine Funktion, die einen Klang erzeugt (oder verändert)
- Synth erzeugen, der sich einfach steuern lässt
- nützlich in Kombination mit Patterns

Hüllkurven aka Envelopes

- “Verlaufskurve” für einen Wert
- Typen von Hüllkurven
  - perkussiv: feste Dauer
  - mit unbegrenzter Dauer: z. B. ADSR, ASR
- Hüllkurven sind nicht nur für die Lautstärke gut!

- Envelopes werden in SuperCollider mit der Klasse `Env` erzeugt
- Aufbau:
  - levels: Lautstärken der Eckpunkte
  - times: Dauer der Segmente
  - curves: Form der Kurve: 0 ist linear
- `Done.freeSelf`: entfernt den Synth vom Server, wenn die Hüllkurve ihr Ende erreicht hat
- Nützliche Kurzschreibweisen: “vorgefertigte” Kurven, z.B.
  - Env.perc(): perkussiv
  - Env.adsr(): ADSR (attack, decay, sustain, release)

Komposition mit Patterns

Was sind Patterns?

- Bibliothek von Klassen zum Sequencing von Klangereignissen
- fangen immer mit P an: Pbind, Pseq, Prand, Pwhite etc.
- `Pbind` ist eine Art Container
- Werte immer paarweise nach dem Muster `\parameter, \wert`

Patterns für Werte-Sequenzen

- `Pseq`
- `Prand` und `Pxrand`
- `Pwhite`
- `Pseries`
- `Pgeom`
- Wiederholungsverhalten je nach Pattern unterschiedlich
- `inf` für unendliche Wiederholung
- kürzester Stream (= Wertefolge) bestimmt die Dauer

default keys: Standard'vokabular' für Patterns

```
\scale
\octave
\dur
\legato
\strum
\mtranspose
\ctranspose
```

Experimentierphase III

(Hier ist Zeit für eure eigenen Ideen!)

Zum Weitermachen

Material zum Weiterlernen

- leider sehr wenig auf Deutsch :(
- Tutorials in der SC-Dokumentation
- Bruno Ruviaro: A Gentle Introduction to SuperCollider
- Youtube-Kanal von Eli Fieldsteel
- Forum: scsynth

Vielen Dank!

Vielen Dank!

Fragen? Komische Fehlermeldungen? mail@camilla-kutzner.de

@draco@sonomu.club