



Musik Programmieren

Kurzanleitung Strudel

Benjamin Schmid

Mit Strudel kann Musik „programmiert“ werden. Mittels Code werden Perkussion und Instrumente beschrieben und Soundeffekte angewandt.

Kapitel 1: Einführung

Auf <https://strudel.cc> finden Sie Strudel, mit dem wir Musik programmieren werden. Auf <https://smbe.ch/strudel> finden Sie einige Beispiele, was man mit Strudel erstellen kann.

Der Hauptbereich von Strudel ist der Codeeditor auf der linken Seite, in dem Sie die Musik programmieren. Auf der rechten Seite haben Sie verschiedene Tabs, um verschiedene Infos nachzuschlagen: patterns für Ihre programmierte Musik, sounds für die verfügbaren Instrumente, und reference für die verfügbaren Funktionen, die Sie aufrufen können. Die komplette Dokumentation können Sie aufrufen, indem Sie auf `learn` klicken.

Um die Bedienung ausprobieren zu können, können Sie den folgenden Code kopieren:

```
1 | s("hh!4, bd ~@3, ~@2 sd ~")
```

Um den Code abzuspielen können Sie entweder oben rechts auf `play` klicken, oder Sie drücken `Strg + Enter`. Nachdem Sie etwas geändert haben, müssen Sie auf `update` klicken oder erneut `Strg + Enter` drücken. Um die Musik zu stoppen, klicken Sie auf `stop` oder drücken Sie `Strg + Punkt` oder `Alt + Punkt`.

Kapitel 2: Workshop

Arbeiten Sie nun den Workshop durch, der auf <https://smbe.ch/strudel> verlinkt ist. Dieser sollte alles Wichtige erklären. In Kapitel 3 finden Sie die wichtigsten Befehle auch nochmals zusammengefasst.

Nutzen Sie auch die verlinkten Beispiele um zu sehen, wie diese umgesetzt sind. Spielen Sie gerne etwas herum und versuchen Sie, diese zu modifizieren.

Kapitel 3: Cheat Sheet

3.1. Variablen

Die Ergebnisse eines Funktionsaufrufs können in einer Variable gespeichert werden, um dies mehrfach verwenden zu können.

```
1 const akkorde = chord("<Am C D F Am E Am E>").voicing()
2 $: akkorde.sound("piano")
3 $: akkorde.sound("gm_trumpet")
```

3.2. Instrumentenspuren

- `$`: am Zeilenanfang erstellt neue Instrumentenspur.
- `_``$`: schaltet die Spur stumm.
- `S``$`: schaltet die Spur auf Solo (alle anderen Spuren werden stumm geschaltet).

Alternativ können auch Namen vergeben werden:

```
1 drums: s("hh*4")
2 piano: note("c e g b").sound("piano")
```

3.3. Mini-Notation

- `"c d e f"`: Die verfügbare Zeit (normalerweise ein Cycle) wird auf die Elemente gleichmässig verteilt.
- `"[c d e f]*2"`: Das Pattern wird schneller abgespielt.
- `"[c d e f]/2"`: Das Pattern wird langsamer abgespielt.
- `"<c d e f>"`: Das Pattern wird langsamer abgespielt, sodass jedes Element die gesamte verfügbare Zeit einnimmt. Wenn dies als Teil-Pattern verwendet wird, wechseln sich die einzelnen Elemente bei jedem Durchlauf ab.
- `"c [d e] f"`: Das Pattern in Klammern ist verschachtelt und zählt für das äussere Pattern wie ein einzelnes Element. In diesem Beispiel würden sie doppelt so schnell abgespielt.
- `"c, e"`: Parallel abspielen.
- `"c@2"`: Element verlängern.
- `"c!2"`: Element mehrfach wiederholen (gleich wie `"c c"`).
- `"c?0.1"`: Element mit angegebener Wahrscheinlichkeit zufällig auslassen.
- `"c | d"`: Element zufällig auswählen.

- `"bd(3,8,0)":` Euklidischer Rhythmus.
- `~:` Pause.

Die verschiedenen Elemente der Mini-Notation können beliebig kombiniert werden. Die meisten Funktionen akzeptieren neben den normalen Argumenten auch solche Mini-Notationen. So kann z.B. das Instrument gewechselt werden.

```
1 | chord("<Am C D F Am E Am E>").voicing().sound("<piano gm_trumpet>")
```

3.4. Perkussion

- Standard Perkussion: `s("hh*4")`.
- Drum Sounds auswählen: `s("hh*4").bank("RolandTR909")`.

bd	bass drum	sd	snare drum	rim	rimshot
hh	hihat	oh	open hihat	lt	low tom
mt	middle tom	ht	high tom	rd	ride cymbal
cr	crash cymbal	cp	clap	sh	shakers
cb	cowbell	tb	tambourine		

3.5. Melodie

- Notennamen: `note("c eb g# b")`.
- Tonart: `n("0 2 4 <[6,8] [7,9]>").scale("C:minor")`.
- Akkorde: `chord("<Am C D F Am E Am E>").voicing()`.

Ein paar verfügbare Tonarten:

major	minor	pentatonic
dorian	mixolydian	minor:pentatonic
melodic:minor	harmonic:minor	blues

3.6. Instrument

- Instrument wählen mit `.sound("piano")`.
- Verfügbare Instrumente im rechten Panel unter `sounds`, dann `samples` und `synths`.

Ein paar nützliche Instrumente:

piano	steinway	sawtooth
supersaw	triangle	sine
gm_acoustic_bass	gm_acoustic_guitar_nylon	gm_alto_sax
gm_baritone_sax	gm_brass_section	gm_cello
gm_church_organ	gm_clarinet	gm_electric_guitar_clean
gm_english_horn	gm_flute	gm_french_horn
gm_fx_crystal	gm_harpsichord	gm_muted_trumpet

gm_oboe	gm_pan_flute	gm_slap_bass_1
gm_steel_drums	gm_tenor_sax	gm_trombone
gm_trumpet	gm_tuba	gm_tubular_bells
gm_violin	gm_xylophone	

3.7. Effekte

- `.gain(.5)`: Lautstärke.
- `.struct("x x ~ x")`: Wendet eine Struktur auf das Pattern an. So kann z.B. eine einzelne Note aufgeteilt werden:

```
1 | chord("<Am C D F Am E Am E>").voicing().struct("x x ~ x")
```

- `.delay(.5)`: Echo.
- `.delaytime(.5)`: Zeitverzögerung des Echos.
- `.room(.5)`: Reverb (wie in einer Höhle).
- `.lpf(2000)`: Low-pass filter, schneidet hohe Frequenzen ab.
- `.hpf(2000)`: High-pass filter, schneidet tiefe Frequenzen ab.
- `.attack(.5)`: Wie lange es geht, bis der Ton voll da ist.
- `.decay(.5)`: Wie lange es geht, bis das initiale Anspielen des Tons vorbei ist.
- `.sustain(.5)`: Lautstärke, auf der der Ton nach dem initialen Anspielen gehalten wird.
- `.release(.5)`: Wie lange der Ton noch nachhallt.

3.8. Diverses

- `setcpm(120)`: Tempo einstellen (Cycles per Minute).
- `stack(...)`: Mehrere Elemente gleichzeitig abspielen.
- `arrange([2, ...], [3, ...])`: Mehrere Elemente nacheinander abspielen, wobei die jeweiligen Elemente für die gegebene Anzahl Cycles beibehalten werden.

3.9. Visuelles Feedback

- `._pianoroll()`: Zeigt gespielte Noten.
- `._scope()`: Oszilloskop.
- `._spectrum()`: Spektrogramm.