Programmier deinen Song

Erste Schritte mit der Musik-Software Sonic Pi

Sonic Pi ist eine freie Software, mit der man spielerisch aus Programmcode Soundeffekte, Beats, Melodien und ganze Songs kreieren kann. Das macht Spaß und bietet für Kinder und Erwachsene einen idealen Einstieg in die Welt der Programmierung.

Von Pit Noack

Es gibt viele Möglichkeiten, Musik zu machen: Man kann Gitarre oder Synthesizer spielen oder mit digitalen Tonstudios Musik produzieren. Sonic Pi geht einen anderen Weg: Das Programm erzeugt Klänge aus Codes, die sich zu vollständigen Liedern zusammensetzen lassen. Das ist einfacher, als es sich anhört.

Sonic Pi wurde von seinem Erfinder Sam Aaron ursprünglich entwickelt, um Kindern das Programmieren beizubringen. Es ist aber alles andere als ein Spielzeug: Die intelligente, erweiterbare Struktur macht das Programm auch für Profis interessant.

Im diesem Artikel geht es um die Grundlagen von Sonic Pi. Alle Code-Beispiele sind eigenständig lauffähig und zeigen die Möglichkeiten der Software. Auf fehlerhaften Code und Vertipper weist Sonic Pi mit Fehlermeldungen hin. In einer der nächsten c't-Ausgaben wird ein zweiter Artikel zeigen, wie man einen Beat mit Sonic Pi schreibt.

Samples und Synths

Sonic Pi bietet zwei Möglichkeiten, Klänge zu erzeugen: mit Samples oder mit Synths. Bei der ersten Art spielt man vorproduzierte Klänge ab. Sonic Pi bringt 129 solcher Klänge von Schlagzeug- bis Klavier-Sounds mit – die Auswahl lässt sich mit Klängen aus anderen Quellen erweitern. Tippen Sie folgenden Befehl in den Puffer-Bereich, um ein Sample zu starten:

```
sample :bd_haus
```

Drücken Sie den Run-Knopf und das Programm spielt eine knackige Techno-Bassdrum ab. Auch wenn nur ein Ton erklingt, ist das bereits ein erstes, vollständiges Sonic-Pi-Programm. Die Liste mit allen eingebauten Samples spuckt Sonic Pi aus, indem Sie `sample` und ein Leerzeichen eintippen oder im Hilfe-Fenster unter „Samples“ nachschauen. Die vorinstallierten Klangschnipsel sind in insgesamt elf Gruppen aufgeteilt. Die Abkürzungen vor dem

ersten Unterstrich bezeichnen die Gruppen: `ambi` steht für flächige Klänge, `bd` für Bassdrums, `drum` für weitere Schlagzeugsounds, `elec` für elektronische Klänge und `loop` für rhythmische Fragmente. Spielen Sie unterschiedliche Samples ab, um die Klangvielfalt zu erforschen. Interessante Kandidaten sind zum Beispiel `:guit_e_fifths`, `:bass_voxy_c`, `:loop_amen`, `:ambi_lunar_land`. Mit der Funktion `sample` lassen sich selbst aufgenommene Klänge einbinden – näheres dazu finden Sie im Tutorial unter 3.6.

Sonic-Pi-Programme bestehen überwiegend aus Funktionsaufrufen. `sample` ist eine solche Funktion. Sie erwartet als zusätzliche Information den Namen des Klanges, der abgespielt werden soll. Solche zusätzlichen Informationen nennt man Argumente.

Die zweite Art der Klangerzeugung verwendet synthetische Klänge. Synths berechnen Klänge in Echtzeit, statt sie aus dem Speicher abzuspielen. Mit `play` spielt man einen Synth:

```
play 64
```

Die Zahl 64 steht dabei für die Tonhöhe. Stellen Sie sich eine Klaviertastatur mit 128 Tasten vor: Die für die tiefste Note ganz linke Taste hat die Nummer 0, die ganz rechte Taste mit der höchsten Note die Nummer 127. Alternativ zu diesen Zahlen kann man die Tonhöhe auch mit dem Notennamen bezeichnen.

```
play :e4
```

In diesem Fall steht das „e“ für die gleichnamige Note, und die „4“ für die vierte Oktave. Sonic Pi bietet viele Synths an, die man mit `use_synth` auswählt. Diese Wahl ist so lange aktiv, bis ein neuer Synth ausgewählt wird.

```
use_synth :piano
play 30
```

Probieren Sie unterschiedliche Tonhöhen (Zahlen zwischen 0–127) und unterschiedliche Synths aus, etwa `:chiplead` (klingt wie ein 8-Bit-Computer aus den 80er-Jahren) und `:tb303` (eine Simulation der legendären Roland TB-303).

Klangfolgen komponieren

Eine Schwalbe macht noch keinen Sommer, und ein einzelner Ton ist noch kein Song. Wie erzeugt man Abfolgen von Klängen, sprich: Melodien und Rhythmen? Die bloße Aneinanderreihung von `sample`- und `play`-Befehlen genügt nicht, da diese Klänge gleichzeitig abgespielt werden. Der Befehl `sleep` lässt den Computer bis zur Ausführung der nächsten Codezeile warten. So bringt man Sonic Pi dazu, die Klänge nacheinander abzuspielen:

```
sample :bd_haus
sleep 1
sample :drum_snare_hard
```

Die Zahl hinter `sleep` steht für die Dauer der Pause in Schlägen; in diesem Fall steht `1` für eine Sekunde. Das liegt daran, dass die voreingestellte Geschwindigkeit von Sonic Pi 60 bpm (bpm = beats per minute, Schläge pro Minute) beträgt. Verdoppelt man das Tempo mit dem Befehl `use_bpm 120` auf 120 Schläge pro Minute, dauert ein Schlag eine halbe Sekunde, dann sorgt `sleep 1` für eine Unterbrechung von einer halben Sekunde:

```
use_bpm 120
sample :bd_haus
sleep 1
sample :drum_snare_hard
```

Mit `sample`, `play` und `sleep` lassen sich bereits ganze Stücke schreiben. Eine gute Übung besteht darin, eine einfache Melodie auszuwählen und nachzubauen.

Klangtüftler

Sonic Pi bietet viele Optionen, mit denen sich Klänge verändern lassen. Grundlegende, für Synths und Samples gültige Optionen sind etwa `amp` und `pan`. Die Abkürzung amp steht für amplification, übersetzt: Verstärkung. Der minimale Wert ist 0, also Stille, nach oben hin gibt es keine Grenze. Zu hohe Lautstärken führen aber zu Verzerrungen. Der voreingestellte Wert beträgt 1.

```
sample :ambi_piano, amp: 0.5
sleep 2
sample :ambi_piano, amp: 1.5
```

Die Abkürzung `pan` steht für Panorama. Diese Option verschiebt den Klang im Stereobild. `pan: -1` positioniert den Klang ganz links, `pan: 0` in der Mitte, `pan: 1` ganz rechts. Im folgenden Beispiel wandert ein Trommelschlag von links nach rechts:

```
sample :drum_tom_lo_hard, pan: -1
sleep 2
sample :drum_tom_lo_hard, pan: -0.5
sleep 2
sample :drum_tom_lo_hard, pan: 0.5
sleep 2
sample :drum_tom_lo_hard, pan: 1
```

Verändert man bei Samples die Abspielgeschwindigkeit, führt das oft zu überraschenden Klangeffekten. Das erreicht man mit der Option `rate`. Das folgende Beispiel spielt das Sample zuerst in normaler Geschwindigkeit ab, danach noch einmal in halber:

```
sample :guit_em9
sleep 4
sample :guit_em9, rate: 0.5
```

Man kennt das Phänomen vom Plattenspieler: Die Veränderung der Abspielgeschwindigkeit verändert auch die Tonhöhe. Es gilt: Eine Verdopplung der Abspielgeschwindigkeit (`rate: 2`) schiebt die Tonhöhe um eine Oktave nach oben; das führt zum Micky-Maus-Effekt; eine Halbierung (`rate: 0.5`) bewirkt das Gegenteil und schiebt den Klang eine Oktave nach unten. Ein negatives Vorzeichen kehrt die Abspielrichtung um, das Sample wird rückwärts abgespielt. So wird ein Sample in halber Geschwindigkeit, rückwärts abgespielt:

```
sample :guit_em9, rate: -0.5
```

Experimentieren Sie mit unterschiedlichen Samples und Geschwindigkeiten.

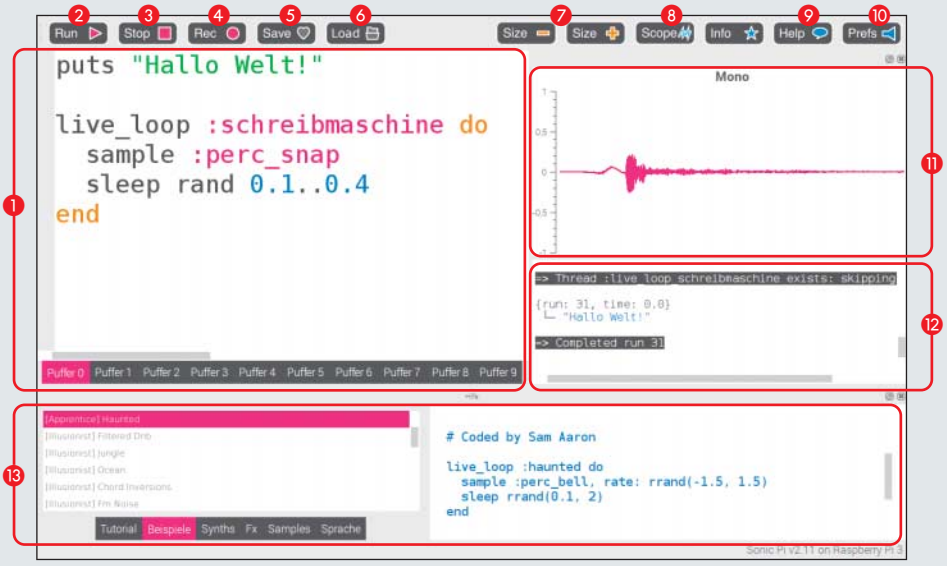
Der c’t-Tipp für Kinder und Eltern

Erste Schritte mit Sonic Pi

- Sonic Pi, PC mit Windows, macOS oder Linux sowie Kopfhörer oder Lautsprecher
- sicherer Umgang mit Tastatur und Maus, Englischkenntnisse von Vorteil
- Einfache Rhythmen und Melodien entstehen innerhalb weniger Minuten, für ganze Songs sollte man deutlich mehr Zeit einplanen.
- Kinder ab etwa zehn Jahren starten mit Eltern oder älteren Geschwistern. Jugendliche ab circa 14 Jahren legen alleine los.
- keine

Mit Code musizieren

Die Bedienoberfläche von Sonic Pi



1. Buffer -Fenster: hier wird der Code hineingeschrieben
2. Code im ausgewählten Buffer starten
3. Ausführung des Codes beenden
4. Aufnahme der von Sonic Pi erzeugten Musik starten / beenden
5. Code im aktuell ausgewählten Buffer in einer Textdatei speichern
6. Code aus einer Textdatei in den aktuell ausgewählten Buffer laden
7. Schriftgröße des Codes verändern
8. Fenster für die Wellenformdarstellung öffnen / schließen
9. Hilfe-Fenster öffnen / schließen
10. Fenster für Einstellungen öffnen / schließen
11. Anzeige der Wellenform
12. Protokoll-Fenster: Zeigt Nachrichten des aktuell laufenden Programms an
13. Hilfe-Fenster

Besonders interessant klingen solche Manipulationen bei längeren Samples, die mit `guit`, `ambi` oder `loop` beginnen.

Optionen, die den Lautstärkeverlauf beeinflussen, bieten weiteren Spielraum. Die wichtigsten Begriffe dabei sind `attack` und `release`: Ersterer bestimmt die Dauer, die ein Klang braucht, um seine volle Lautstärke zu erreichen. Mit `release` bezeichnet man die Ausklingdauer. Dieses Verhalten ist wesentlich für die Charakteristik aller Klänge. Ein Xylophon, mit einem Filzschlägel gespielt, klingt „weich“, besitzt also eine längere Attack. Bei der Verwendung eines Holzschlägels klingt das Xylophon härter, perkussiver (kürzere Attack). Beim Klavier bewirkt die Betätigung des rechten Pedals die Anhebung der Dämpfer, was zu einer längeren Ausklingzeit führt, also zu längerem Release.

Ein Beispiel für einen kurzen, perkussiver Klang in Sonic Pi:

```
play 64, release: 0.1
```

Ein Klang, der langsam anschwillt und langsam ausklingt:

```
play 64, attack: 4, release: 4
```

Die volle Bandbreite der Klangmöglichkeiten von Synths lässt sich erst mit den Optionen richtig ausschöpfen. Der Synth `blade` (benannt nach dem Filmklassiker Blade Runner) etwa eignet sich gut, um elektronische Streicherklänge mit langen Ein- und Ausklingphasen zu erzeugen:

```
use_synth :blade
play 50, attack: 2, release: 4
```

Mit `attack` und `release` lässt sich schon viel machen. Es gibt noch weitere Optionen, die den Lautstärkeverlauf verändern, wie `sustain` und `decay`. Näheres hierzu finden Sie im eingebauten Tutorial unter Punkt 2.4.

Viele Optionen lassen sich ebenso auf Synths wie auf Samples anwenden (etwa `amp` und `pan`). Die Option `rate` ist nur bei Samples sinnvoll. Zudem gibt es Optionen, die nur auf bestimmte Synths anwendbar sind. Die Option `cutoff` funktioniert nur bei solchen Synths, die einen Tiefpassfilter besitzen. Dazu zählen `tb303` und `blade`. `cutoff` steuert die Frequenz des Filters, ein hoher Wert sorgt für einen hellen Klang, ein niedriger Wert klingt tief und dunkel. Erlaubt sind Werte zwischen 0 und 130:

```
use_synth :tb303
play 30, cutoff: 50
sleep 1
play 30, cutoff: 80
sleep 1
play 30, cutoff: 110
```

Play it again, Sam

Programmierer mögen es bequem. Wenn ein Stück Code acht Mal ausgeführt werden soll, dann ist es umständlich, den Code acht Mal hinzuschreiben. Sonic Pi bietet die Möglichkeit, Codestücke zu wiederholen:

```
8.times do
  sample :bd_haus
  sleep 0.25
  sample :drum_snare_hard
  sleep 0.25
end
```

Code, der zwischen einem `do` und einem `end` steht, nennt man einen Block. `8.times` sorgt dafür, dass dieser acht Mal ausgeführt wird.

Die Wiederholung mithilfe von Blöcken macht den Code übersichtlicher, und Änderungen müssen nur noch an einer Stelle vorgenommen werden.

Zufallsmusik

Computer arbeiten im Idealfall streng logisch und kennen keinen Zufall. Diesen können sie aber simulieren. Das kann man sich beim Komponieren zunutze machen.

Die Funktion `rand_i` erzeugt Zufallszahlen. „Rand“ steht für „random“ (beliebig, zufällig), „i“ steht für „integer“ (ganzzahlig). `rand_i 30..90` erzeugt ganze Zufallszahlen zwischen 30 und 90.

Das folgende Beispiel spielt acht Noten mit einer zufälligen Tonhöhe zwischen 30 und 90. `play` erwartet als Argument eine Zahl. Wann immer im Code eine Zahl erwartet wird, kann man auch einen Ausdruck einsetzen, der eine Zahl erzeugt. In diesem Fall gibt die Funktion `rand_i` eine solche Zahl zurück.

Bild: Marko Löhmus

Sonic-Pi-Erfinder Sam Aaron spielt mit der Programmiersprache auch live, hier auf der Java-Konferenz GeekOUT 2016 in Tallinn.

```
use_synth :piano
8.times do
  play rand_i 30..90
  sleep 0.25
end
```

Wenn Sie diesen Code wiederholt starten, sind jedes Mal exakt dieselben Noten zu hören. Das hat einen guten Grund: Wenn Sie anderen Sonic Pi-Nutzern Code zuschicken, können sie sich darauf verlassen, dass das Ergebnis überall exakt gleich klingt, trotz der zufälligen Werte.

Mit der Funktion `use_random_seed` lässt sich der Zufallsgenerator auf einen anderen Startpunkt setzen und wird andere Zufallszahlen erzeugen:

```
use_random_seed 672
use_synth :piano
8.times do
  play rand_i 30..90
  sleep 0.25
end
```

Wählen Sie zwei beliebige Zahlen und setzen Sie diese abwechselnd hinter `use_random_seed`. Sie werden hören, dass gleiche Werte für den Seed gleiche Zahlenfolgen erzeugen. Wenn man Gleitkommazahlen statt ganzer Zahlen benötigt, wählt man die Funktion `rand`. Im folgenden Beispiel wird `sleep` mit einer Zufallszahl zwischen 0.1 und 0.5 gefüttert:

```
use_synth :piano
8.times do
  play rand_i 30..90
  sleep rand 0.1..0.5
end
```

Zwei weitere Zufallsfunktionen sind nützlich: `choose` wählt aus mehreren vorgegebenen Werten einen zufälligen aus. Im folgenden Beispiel wird bei jedem Durchgang eines von drei vorgegebenen Samples gespielt und eine Pause von einem viertel, einem halben oder einem ganzen Schlag eingebaut:

```
64.times do
  sample [:bd_haus, :sn_dub, ↵
  ↳:elec_blup].choose
  sleep [0.125, 0.25, 0.5].choose
end
```

Die Funktion `one_in` (einer von) verwenden Sie, wenn eine Aktion mit einer bestimmten Wahrscheinlichkeit ablaufen soll. Man nutzt es in Kombination mit `if` (wenn, falls). Im folgenden Beispiel wird im Schnitt jedes fünfte Mal das Sample abgespielt:

```
128.times do
  if one_in 5
    sample :drum_cymbal_closed
  end
  sleep 0.1
end
```

Klangverfremdung mit fx

Die Funktion `use_fx` (fx ist die englische Abkürzung für Effekt) bietet eine weitere Möglichkeit, Klänge zu verändern. Häufig genutzte Effekte sind etwa `:echo`, `:reverb` (Hall), `:distortion` (Verzerrung). Zum Vergleich, so klingt ein Schlag auf eine Snare Drum ohne Effekt:

```
sample :drum_snare_soft
```

Snareschlag mit Echo:

```
with_fx :echo do
  sample :drum_snare_soft
end
```

Auch die Effekte lassen sich mit Optionen bearbeiten. Bei `echo` kann man mit `:phase` die Dauer der Verzögerung in Schlägen verändern:

```
with_fx :echo, phase: 0.5 do
  sample :drum_snare_soft
end
sleep 2
with_fx :echo, phase: 0.05 do
  sample :drum_snare_soft
end
```

Es lassen sich auch Effekte ineinander verschachteln. Das nächste Beispiel versieht einen Snare-Schlag erst mit dem Hall eines großen Raumes und manipuliert den Klang dann zusätzlich mit einem Flanger. Das ist ein beliebter psychedelischer Klangeffekt:

```
with_fx :flanger do
  with_fx :reverb, room: 0.99 do
    sample :drum_snare_soft
  end
end
```

Der Hilfebereich listet unter „FX“ alle verfügbaren Effekte und deren Optionen in englischer Sprache auf.

Live Loops

Mit Sonic Pi kann man nicht nur zu Hause Beats und Melodien basteln. Mit viel Übung lässt sich die Software live auf der Bühne spielen. Sam Aaron, Entwickler von Sonic Pi, bringt mit dieser Form des Live Codings auf Festivals Menschen zum Tanzen.

Dreh- und Angelpunkt des Live Codings ist die Funktion `live_loop`, die auch mit Blöcken arbeitet. Im Beispiel wird ein Live Loop mit dem Namen `mein_beat` erzeugt. Den Namen des Live Loop muss man wählen. Der Code in dem Block wird bis zum Beenden des Programms wiederholt:

```
live_loop :mein_beat do
  sample :bd_haus
  sleep 0.5
  sample :drum_snare_hard
  sleep 0.5
end
```

Die Besonderheit des Live Loop besteht darin, dass es möglich ist, den Code zu verändern, während er läuft. Diese Eigenschaft ist eine Grundvoraussetzung, um mit Sonic Pi live zu musizieren, da man nicht die Musik unterbrechen muss, um den Code zu ändern.

Probieren Sie es aus: Starten Sie das Live-Loop-Beispiel. Verändern Sie den Code, und drücken Sie erneut den Run-Knopf. Schalten Sie einzelne Codezeilen durch Vorausstellen eines `#` ein und aus. Experimentieren Sie mit mehreren Live Loops, die parallel laufen. Beachten Sie dabei, dass verschiedene Live Loops auch verschiedene Namen haben müssen.

ring, tick und look

Sonic Pi kann sich Folgen von Werten merken. Das ist sehr nützlich, denn Melodien sind nichts anderes als Folgen von Tonhöhen, Tonlängen und Pausenzeiten. Sie werden in sogenannten Ringen gespeichert. Der Name „Ring“ kommt daher, dass, wenn das Ende einer Folge erreicht wurde, die Folge von vorne beginnt.

Das folgende Beispiel erzeugt zuerst einen Ring mit den Werten 60, 55, 65 und speichert ihn unter dem Namen `noten`. Mit `noten[0]` erhält man den ersten Wert des Ringes, also 60. `noten[1]` ergibt 55, und `noten[2]` ist 65. Mit `noten[3]` landet man dann wieder bei dem ersten Wert, also 60.

```
noten = ring 60, 55, 65
play noten[0] # 60
sleep 1
play noten[1] # 55
sleep 1
play noten[2] # 65
sleep 1
play noten[3] # 60
sleep 1
play noten[4] # 55
sleep 1
play noten[5] # 65
# und so weiter...
```

Ihren vollen Nutzen entfalten die Ringe erst, wenn man ein weiteres Hilfsmittel nutzt. Die Funktion `tick` arbeitet wie ein Zähler. Wenn das Programm startet, gibt der Aufruf von `tick` den Wert 0 zurück. Jedes weitere Mal, wenn `tick` im Code steht, wird der Wert um 1 erhöht. Beim ersten Mal gibt `tick` also den Wert 0 zurück, beim zweiten mal den Wert 1 und immer so weiter.

Im folgenden Beispiel wird `tick` verwendet, um damit in einem Live Loop die Werte im Ring der Reihe nach abzufragen und an `play` zu übergeben:

```
noten = ring 40, 45, 50, 55, 60
use_synth :piano
live_loop :ring_und_tick do
  play noten[tick]
  sleep 1
end
```

Man kann auch mehrere Ringe gleichzeitig verwenden. Das folgende Beispiel nutzt einen zweiten Ring für die Pausen zwischen den Noten. Da durch `noten[tick]` der Wert von `tick` schon erhöht wurde, wird für die Pausen `pausen[look]` verwendet. `look` gibt nämlich den Wert der tick-Variable zurück, ohne ihn zu erhöhen:

```
noten = ring 40, 45, 50, 55, 60
pausen = ring 0.25, 0.25, 0.5
use_synth :piano
live_loop :ring_und_tick do
  play noten[tick]
  sleep pausen[look]
end
```

Experimentieren Sie mit dem letzten Beispiel, indem Sie die Werte in den beiden Ringen verändern oder Werte hinzufügen. Sie können den Live Loop erweitern, indem Sie etwa ein weiteres Instrument oder einen Effekt hinzufügen, dessen Eigenschaften von weiteren Ringen gesteuert werden. Spannende Ergebnisse kann man erzielen, wenn man mit vielen Ringen unterschiedlicher Länge experimentiert.

Eigene Funktionen schreiben

Bis jetzt haben Sie Funktionen benutzt, die Sonic Pi mitbringt. Man kann mit `define` aber auch eigene Funktionen bauen. Eine Funktion braucht einen Namen, das ist im folgenden Beispiel `elektro_specht`. Es folgt ein Block, der den Code enthält, der ausgeführt werden soll, wenn die Funktion benutzt wird:

```
define :elektro_specht do
  8.times do
    sample :elec_mid_snare
    sleep 0.05
  end
end
```

Ist die Funktion einmal definiert, lässt sie sich genau so benutzen wie alle anderen: indem man ihren Namen eintippt. Viele eingebaute Funktionen, die Sie bisher benutzt haben, verwenden Argumente – etwa `sleep` die Dauer der Unterbrechung in Schlägen oder `sample` den Namen eines Klanges. Auch selbst geschriebene Funktionen können Argumente haben. Angenommen, Sie möchten die Anzahl der Schläge des elektrischen Spechtes mit einem Argument bestimmen, dann überlegen Sie sich einen sinnvollen Variablennamen (hier: `anzahl_schlaege`) und schreiben ihn zwischen zwei senkrechte Striche (Pipe-Zeichen) an den Beginn des Blocks. Damit dieser Variablenname wirksam wird, muss er noch an der entsprechenden Stelle im Block eingesetzt werden:

```
define :elektro_specht ↴
↳do |anzahl_schlaege|
  anzahl_schlaege.times do
    sample :elec_mid_snare
    sleep 0.05
  end
end
```

Die Verwendung der Funktion sieht dann so aus:

```
elektro_specht 16
sleep 1
elektro_specht 8
```

Will man mehrere Argumente verwenden, schreibt man sie durch Kommas getrennt hintereinander:

```
define :elektro_specht do ↴
↳|anzahl_schlaege, pausen_dauer|
  anzahl_schlaege.times do
    sample :elec_mid_snare
    sleep pausen_dauer
  end
end
elektro_specht 10, 0.1
sleep 1
elektro_specht 32, 0.01
```

Wenn Sie Gefallen an der Komposition von Musik mittels Code gefunden haben, gibt es jetzt viel zu experimentieren und zu entdecken. Einen tieferen Einblick in die Möglichkeiten Sonic Pis gibt der zweite Teil dieser Mini-Serie, der in einer der kommenden Ausgaben erscheint. *(mre@ct.de)* c't

Download Sonic Pi, Maschinennah Creative Coding: ct.de/yvrq