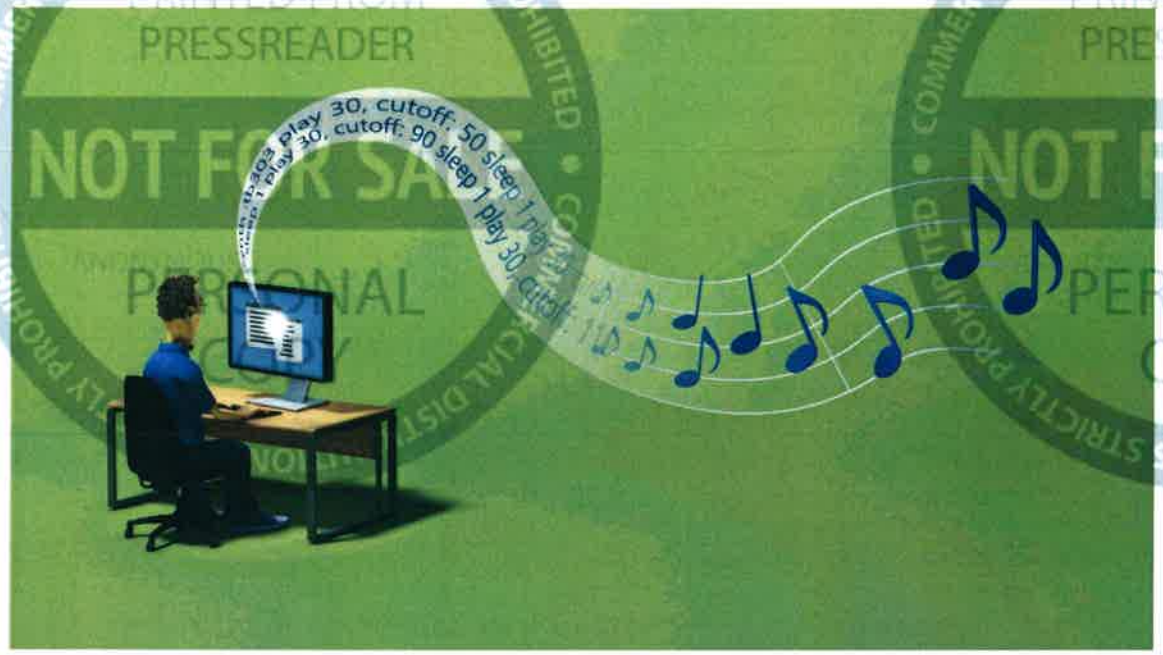


Programmeer je muziek

De eerste stappen met muzieksoftware Sonic Pi

Sonic Pi is gratis software waarmee je spelenderwijs met programmeercode geluidseffecten maakt, en beats, melodieën en complete songs kunt creëren. Dat is leuk om te doen en biedt een ideale instap in de wereld van programmeren.

Pit Noack

Er zijn veel mogelijkheden om muziek te maken. Je kunt gitaar of synthesizer spelen of met digitale geluidsstudio's muziek produceren. Sonic Pi bewandelt een andere weg: het programma maakt geluiden op basis van code, waarmee je complete nummers kunt samenstellen. Dat is makkelijker dan het misschien klinkt.

Sonic Pi werd door zijn uitvinder Sam Aaron oorspronkelijk ontwikkeld om kinderen te leren programmeren. Het is echter zeker geen speelgoed: de intelligente en uitbreidbare structuur maakt het programma ook geschikt voor professionals.

In dit artikel gaat het om de basisprincipes van Sonic Pi. Alle codevoorbeelden werken zelfstandig en laten de mogelijkheden van de software zien. Sonic Pi waarschuwt je bij typ- en codefouten. In een van de volgende nummers van c't zal een ander artikel laten zien hoe je een beat met Sonic Pi maakt.

In tegenstelling tot wat de naam doet vermoeden, is Sonic Pi er niet alleen voor de Raspberry Pi, maar ook voor Windows, Linux en macOS. Voor Windows is er een installeerbare versie, maar ook een portable variant.

Samples en synths

Sonic Pi biedt twee mogelijkheden om klanken voort te brengen: met samples of met synths. Bij de eerste manier speel je voorgeproduceerde geluiden af. Sonic Pi heeft 129 dergelijke geluiden, variërend van slagwerk- tot pianoklanken – die selectie is uit te breiden met geluiden uit andere bronnen. Typ het volgende commando in het bufferdeel in om een sample te starten:

```
sample :bd_haus
```

Druk op de Run-knop en het programma begint een techno-bassdrum af te spelen. Ook al klinkt er nu maar één geluid, dit is toch al een eerste volwaardig Sonic Pi-programma. De lijst met alle ingebouwde samples in Sonic Pi krijg je te zien door `sample` en

een spatie in te typen of via de Help-knop bij Samples te kijken. De voorgeïnstalleerde audiosamples zijn in elf groepen verdeeld. De afkortingen voor de eerste underscore geven de groep aan: `ambi` staat voor ambiente geluiden, `bd` voor bassdrum, `drum` voor andere slaginstrumenten, `elec` voor elektronische geluiden en `loop` voor ritmische fragmenten. Speel verschillende samples af om te ontdekken welke geluiden er allemaal zijn. Interessante voorbeelden zijn onder meer `:guit_e_fifths`, `:bass_voxy_c`, `:loop_amen` en `ambi_lunar_land`. Met de functie `sample` kun je zelf opgenomen geluiden toevoegen – meer daarover in de ingebouwde Tutorial bij punt 3.6 (Externe samples).

Sonic Pi-programma's bestaan voornamelijk uit functie-aanroepen. De functie `sample` is er daar een van. Als extra informatie moet je daar de naam van het geluid aan meegeven dat je afgespeeld wilt hebben. Dergelijke extra informatie heet een argument.

De tweede manier van geluidsproductie is met synthesizerklanken. Synths berekenen de geluiden realtime in plaats van een opgeslagen geluid af te spelen. Met play speel je een synthesizerklank af:

```
play 64
```

Het getal 64 staat daarbij voor de toonhoogte. Stel je een pianoklavier met 128 toetsen voor: de toets helemaal links met de laagste noot heeft nummer 0, de toets helemaal rechts met de hoogste noot nummer 127. Als alternatief voor die getallen kun je de toonhoogte ook met de nootnaam aangeven:

```
play :e4
```

In dit geval staat de `e` voor de betreffende noot en de `4` voor de vierde octaaf. Sonic Pi biedt veel synthesizerklanken, die selecteer je met `use_synth`. Die keuze is net zo lang actief tot je een nieuwe selecteert.

```
use_synth :piano
play 30
```

Probeer verschillende toonhoogten (getallen tussen 0-127) en synthesizerklanken uit, bijvoorbeeld `:chiplead` – die klinkt als een 8-bit computer uit de jaren 80 – en `:tb303`, een simulatie van de legendarische Roland TB-303.

Melodie componeren

Het genereren van klanken is natuurlijk nog maar het begin, daar heb je niet met een een heel nummer mee. Hoe zet je nu een aantal klanken achter elkaar, oftewel hoe maak je een melodie of ritme? Het simpel achter elkaar zetten van `sample`- en `play`-commando's is niet voldoende omdat die klanken dan tegelijkertijd afgespeeld worden. Het commando `sleep` laat de computer wachten met het uitvoeren van de volgende commandoregel. Daarmee kun je Sonic Pi de klanken na elkaar laten afspelen:

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

Het getal achter `sleep` staat voor de duur van de pauze tussen de slagen. In dit geval staat 1 voor een seconde. Dat komt doordat het standaard ingestelde tempo bij Sonic Pi 60 bpm (beats per minute) is. Als je het tempo verdubbelt met het commando `use_bpm 120` naar 120 slagen per minuut, dan duurt een slag een halve seconde en dan betekent `sleep 1` een pauze van een halve seconde. Voor hetzelfde resultaat krijg je dan:

```
use_bpm 120
sample :bd_haus
sleep 2
sample :drum_snare_hard
```

Met `sample`, `play` en `sleep` kun je al hele muziekstukken schrijven. Probeer als oefening maar eens een populaire melodie na te maken.

Geluiden veranderen

Sonic Pi biedt veel opties om geluiden aan te passen. Belangrijke opties voor zowel synths als samples zijn bijvoorbeeld `amp` en `pan`. De afkorting amp staat voor amplification oftewel versterking. De minimale waarde is 0 (stilte dus), maar aan de bovenkant is er geen grens. Een te hoge versterking leidt echter tot vervorming. De standaard ingestelde waarde is 1.

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

De afkorting `pan` staat voor panorama. Die optie verschuift de klank in het stereobeeld van `pan -1` helemaal links via `pan 0` in het midden tot `pan 1` helemaal rechts. In het volgende voorbeeld loopt een trommelslag van links naar 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
```

Verander je de afspeelsnelheid bij de samples, dan leidt dat vaak tot verrassende geluidseffecten. Dat doe je met de optie `rate`. Het volgende voorbeeld speelt een sample eerst op de normale snelheid af, daarna met de halve snelheid:

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

Dat fenomeen ken je misschien van je platenspeler: het veranderen van de afspeelsnelheid verandert ook de toonhoogte. Een verdubbeling van de afspeelsnelheid (`rate: 2`) verschuift de toonhoogte naar een octaaf hoger. Dat leidt tot een Micky Mouse-effect. Het halveren (`rate: 0.5`) bewerkstelligt het tegenovergestelde en maakt de klank een octaaf lager. Een min voor het getal draait de afspeelrichting om,

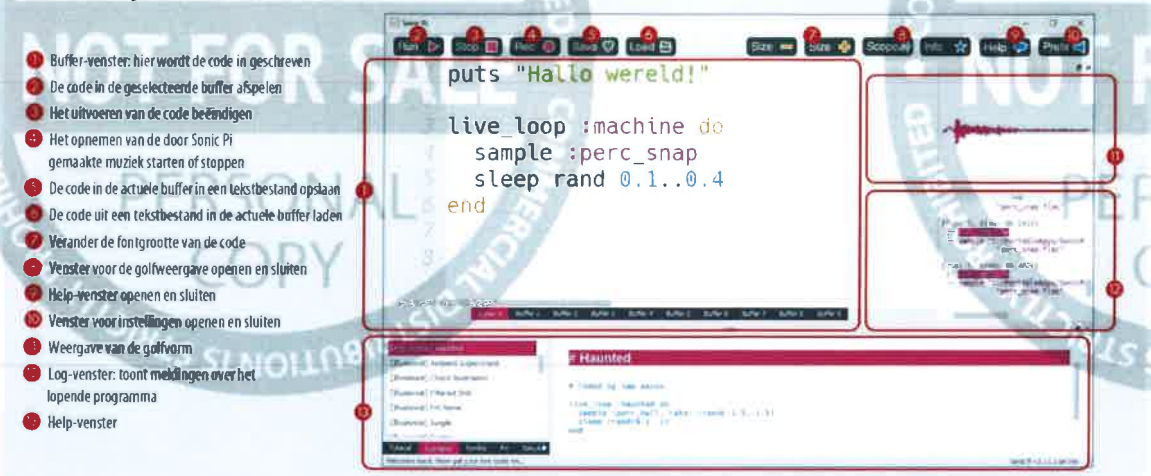
De c't-tip voor ouders en kinderen

De eerste stappen met Sonic Pi

- Sonic Pi, een pc met Windows, macOS of Linux en een hoofdtelefoon of luidspreker.
- Weten hoe je met het toetsenbord en de muis werkt, en een beetje Engels is ook een voordeel.
- Eenvoudige ritmes en melodieën heb je in een paar minuten gemaakt, voor hele nummers moet je duidelijk meer tijd uittrekken.
- Kinderen vanaf ongeveer tien jaar kunnen met ouders of oudere broers of zussen aan de slag. Kinderen vanaf een jaar of 14 kunnen zelf aan de slag.
- Geen.

Muziek maken met code

De bedieningsinterface van Sonic Pi



de sample wordt dan achteruit afgespeeld. Zo wordt een sample met halve snelheid achterwaarts afgespeeld:

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

Experimenteer wat met verschillende samples en snelheden. Dergelijke manipulaties klinken met name interessant bij langere samples die met `guit`, `ambi` of `loop` beginnen.

Opties die het volumeverloop beïnvloeden geven nog meer speelruimte. De belangrijkste termen daarbij zijn `attack` en `release`. De eerste bepaalt de duur die een klank nodig heeft om het volledige volume te bereiken, de tweede is de wegsterftijd van het geluid. Die waarden zijn essentieel voor de karakteristiek van een klank. Een xylofoon die met een vilten stok bespeeld wordt klinkt 'zacht' en heeft dan ook een langere attack. Met een houten stok klinkt een xylofoon harder en meer percussief, dus met een kortere attack. Bij een piano zorgt het rechter pedaal ervoor dat de dempers van de snaren gaan, wat ertoe leidt dat de snaren langer blijven doorklinken en dus tot een langere release.

Een voorbeeld voor een korte, percussieve klank in Sonic Pi:

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

Een klank die langzaam aanzwelt en langzaam uitdooft:

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

Het volledige scala aan klankmogelijkheden van synths is met de opties optimaal te benutten. De synth `blade` (genoemd naar de filmklassieker Blade Runner) is bijvoorbeeld goed geschikt om elektronische snaarklanken met lange aanzwel- en uitdooffases te genereren:

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

Met `attack` en `release` is al veel te doen. Er zijn nog andere opties die het volumeverloop kunnen veranderen, waaronder `sustain` en `decay`. Meer daarover staat in de ingebouwde Tutorial bij punt 2.4 (Geluidsduur met "Envelopes").

Veel opties zijn zowel op synthesizerklanken als samples toe te passen, zoals `amp` en `pan`. De optie `rate` is alleen bij samples zinvol. Bovendien zijn er opties die alleen op bepaalde synths toe te passen zijn. De optie `cutoff` werkt bijvoorbeeld alleen bij de synths die een laagdoorlaatfilter hebben. Dat zijn onder andere `tb303` en `blade`. Daarbij bepaalt `cutoff` de frequentie van het filter. Een hogere waarde zorgt een scherpere klank, een lagere waarde klinkt diep en donker. Je kunt waarden tussen 0 en 130 gebruiken:

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

Play it again, Sam

Programmeurs houden van makkelijk. Als een stuk code achtmaal uitgevoerd moet worden, dan is het te veel gedoe om dat acht keer in te typen. Sonic Pi biedt een mogelijkheid stukken code te herhalen:

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

Code die tussen en `do` en een `end` staat, noem je een blok. Met `8.times` zorg je ervoor dat dit acht keer uitgevoerd wordt.

Het herhalen met behulp van blokken maakt de code overzichtelijker, en wijzigingen aan de code hoeven dan maar één keer gedaan te worden.

Toevalsmuziek

Computers werken in het ideale geval puur logisch en kennen geen toeval. Dat kun je echter simuleren. Daar kun je bij het componeren je voordeel mee doen.

De functie `rand_i` maakt toevalsgetallen. Daarbij staat rand voor random, oftewel willekeurig. De i staat voor integer, dus

`rand_i 30..90` levert een geheel getal op tussen de 30 en 90.

Het volgende voorbeeld speelt acht tonen af met een willekeurige toonhoogte tussen de 30 en 90. De functie `play` verwacht een getal als argument. Altijd als er in de code een getal verwacht wordt, kun je ook een uitdrukking gebruiken die een getal oplevert. In dit geval levert de functie `rand_i` een dergelijk getal op.

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

Als je die code nog een keer laat lopen, krijg je echter steeds dezelfde tonen te horen. Daar is een goede reden voor: als je je code naar andere Sonic Pi-gebruikers stuurt, moet je er zeker van zijn dat het resultaat overal hetzelfde zal zijn, ondanks de willekeurige waarden.

Met de functie `use_random_seed` kun je de toevalsgenerator met een andere startwaarde laten beginnen. Dan worden iedere keer als je het programma start andere toevalsgetallen gegenereerd:

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

Kies twee willekeurige getallen en zet die afwisselend achter `use_random_seed`. Dan zul je horen dat dezelfde waarde voor de seed tot dezelfde tonen (getalvolgordes) zal leiden. Als je floatingpoint-getallen in plaats van gehele getallen nodig hebt, gebruik je de functie `rand`. In het volgende voorbeeld wordt `sleep` met een toevalsgetal tussen 0.1 en 0.5 gevoed:

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

Twee andere toevalsfuncties zijn te gebruiken: `choose` selecteert een willekeurig getal uit meerdere opgegeven waarden. In het volgende voorbeeld wordt bij elke doorloop een van drie opgegeven samples afgespeeld met een pauze van een kwart, halve of hele tel:

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

De functie `one_in` (een van) gebruik je als een actie met een bepaalde waarschijnlijkheid moet verlopen. Die gebruik je in combinatie met `if`. In het volgende voorbeeld wordt een sample gemiddeld eens in de vijf keer afgespeeld:

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

Geluidseffecten

De functie `use_fx` biedt een andere mogelijkheid klanken te bewerken. Vaak gebruikte effecten zijn bijvoorbeeld `:echo`, `:reverb`, en `:distortion`. Ter vergelijking: zo klinkt een slag op een snaredrum zonder effecten:

```
sample :drum_snare_soft
```

Dit is een snaredrum met echo:

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

Ook de effecten zijn met opties te bewerken. Bij `echo` kun je met `:phase` de duur van de vertraging in tellen veranderen:

```
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
```

Je kunt effecten ook in elkaar nesten. Het volgende voorbeeld voorziet een snareslag eerst met de reverb van een grote kamer en bewerkt het geluid dan nog met een flanger. Dat is een populair psychedelisch geluidseffect:

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

Het Help-venster van Sonic Pi toont bij 'Fx' alle beschikbare effecten en hun opties. Die beschrijvingen zijn nog wel in het Engels.

Live loops

Met Sonic Pi kun je thuis niet alleen beats en melodieën in elkaar knutselen, maar met wat oefening kun je met de software ook live op het podium spelen. Sam Aaron, de ontwikkelaar van Sonic Pi, krijgt met die manier van live-coding op festivals mensen aan het dansen.

Het centrale item van de live-coding is de functie `live_loop`, die ook met blokken werkt. In dit voorbeeld wordt een live-loop met de naam `mijn_beat` aangemaakt. Je moet een naam voor een live-loop opgeven. De code in het blok wordt tot het beëindigen van het programma herhaald:

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

Het speciale aan een een live-loop is dat het mogelijk is om de code te veranderen terwijl hij loopt. Die eigenschap is een basiseis om met Sonic Pi live muziek te kunnen maken omdat je de muziek niet kunt onderbreken om de code te veranderen.

Probeer het eens uit: start het live-loop voorbeeld. Verander de code en druk opnieuw op de Run-knop. Schakel de afzonderlijke regels code uit of in door er een # voor te zetten of die te verwijderen. Experimenteer eens met meerdere parallel lopende live-loops. Let er daarbij op dat de verschillende live-loops ook verschillende namen moeten hebben.

Ring, tick en loop

Sonic Pi kan series van waarden onthouden. Dat is heel handig, want melodieën zijn niets anders dan series van toonhoogtes, toonduren en pauzetijden. Die worden in zogeheten rings opgeslagen. De naam ring komt doordat een serie weer vooraan opnieuw begint als hij aan het einde is gekomen.

Het volgende voorbeeld maakt eerst een ring aan met de waarden 60, 55 en 65 en slaat die dan op onder de naam `tonen`. Met `tonen[0]` krijg je de eerste waarde van de ring, oftewel 60. Op die manier leidt `noten[1]` tot 55 en `noten[2]` tot 65.

Dan komt `noten[3]` weer uit bij de eerste waarde van 60.

```
tonen = ring 60, 55, 65
play tonen[0] # 60
sleep 1
play tonen[1] # 55
sleep 1
play tonen[2] # 65
sleep 1
play tonen[3] # 60
sleep 1
play tonen[4] # 55
sleep 1
play tonen[5] # 65
# en zo verder...
```

De meerwaarde van de rings zie je pas als je een ander hulpmiddel gebruikt. De functie `tick` werkt als een teller. Als het programma start, levert het aanroepen van `tick` een 0 op. Elke keer als je het weer aanroept, wordt de waarde dan met 1 verhoogd.

In het volgende voorbeeld wordt `tick` gebruikt om bij een live-loop de waarden in de ring op te vragen en aan `play` mee te geven:

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

Je kunt ook meerdere rings tegelijkertijd gebruiken. Het volgende voorbeeld gebruikt een tweede ring voor de pauzes tussen de tonen. Omdat door `tonen[tick]` de waarde van `tick` al verhoogd werd, wordt voor de pauzes `pauzes[look]` gebruikt omdat `look` de waarde van de `tick`-variabele terug levert zonder die te verhogen:

```
tonen = ring 40, 45, 50, 55, 60
pauzes = ring 0.25, 0.25, 0.5
use_synth :piano
live_loop :ring_en_tick do
   play tonen[tick]
   sleep pauzes[look]
end
```

Experimenteer wat met het laatste voorbeeld door de waarden in de beide rings te veranderen of waarden toe te voegen. Je kunt de live-loop uitbreiden door bijvoorbeeld een ander instrument of een effect toe te voegen waarvan de eigenschappen dan door andere rings aangestuurd kunnen worden. Je krijgt spannende resultaten als je met veel rings van verschillende lengtes experimenteert.

De uitvinder van Sonic Pi Sam Aaron speelt met de programmeertaal ook live, hier op de Java-conferentie GeekOUT2016 in Tallinn.

Eigen functies schrijven

Tot nu toe heb je functies gebruikt die Sonic Pi zelf heeft. Je kunt met `define` echter ook zelf eigen functies maken. Een functie moet een naam hebben, waarvoor in het volgende voorbeeld `electro_specht` gebruikt wordt. Dan volgt een blok met de code die uitgevoerd moet worden als de functie aangeroepen wordt:

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

Als de functie eenmaal gedefinieerd is, kun je die net zo gebruiken als alle andere functies door de naam in te typen. Veel ingebouwde functies die je tot nu toe gebruikt hebt, werken met argumenten – bijvoorbeeld bij `sleep` de duur van de onderbreking in tellen en bij `sample` de naam van een geluid. Ook zelf geschreven functies kunnen argumenten hebben. Als je het aantal slagen van de elektrische specht met een argument wilt bepalen, dan moet je een zinvolle variabelenaam bedenken (hier: `aantal_slagen`) en die tussen twee verticale strepen (pipe-teken) aan het begin van het blok zetten. Om de variabelenaam te werkzaam te laten zijn, moet je hem nog gebruiken op de juiste plek in het blok:

```
define :electro_specht do |aantal_
slagen|
   aantal_slagen.times do
      sample :elec_mid_snare
      sleep 0.05
   end
end
```

Je gebruikt de functie dan op de volgende manier:

```
electro_specht 16
sleep 1
electro_specht 8
```

Als je meerdere argumenten wilt gebruiken, schrijf je die er gescheiden door een komma achter:

```
define :electro_specht do |aantal_
slagen, pauze_duur|
   aantal_slagen.times do
      sample :elec_mid_snare
      sleep pauze_duur
   end
end
electro_specht 10, 0.1
sleep 1
electro_specht 32, 0.01
```

Als je de smaak van het componeren van muziek eenmaal te pakken hebt, is het zaak veel te experimenteren en te ontdekken. Een diepere duik in de mogelijkheden van Sonic Pi komt in het tweede deel van deze mini-serie in een van de volgende nummers van c't. (nkr) c't