COMPOSICIÓN ALGORÍTMICA Y LIVECODING

Alejandro Rodríguez Antolín (@AlexRoan3)
alex.rodriguez.antolin@gmail.com

Miguel Ángel Rodríguez Muiños (@mianromu)
mianromu@gmail.com

Materiales

- Presentación
  - https://leugimsan.es/esLibre/presentacion.pdf
- Código:
  - https://leugimsan.es/esLibre/codigo.txt

Unas definiciones….

- Composición Algorítmica
  - Composición musical usando algoritmos (programando).

- LiveCoding
  - Composición algorítmica en directo.

Sonic-Pi

- Pedagogía de la programación
- Sam Aaron (Universidad de Cambrigde, UK)
- Multiplataforma.
  - Linux, macOS, Windows
  - Desarrollado, inicialmente, para Raspberry Pi
    - SuperCollider + Ruby
- https://sonic-pi.net/


Sonic Pi
Archivo Editar Window
run stop rec save load
size size scope info help prefs
1 # Welcome to Sonic Pi
2
3
Scope
Registro
=> Has Sonic Pi made you smile?
We need *your* help to fund further development!
Sonic Pi is not financially supported by
any organisation.
Señales
|0| |1| |2| |3| |4| |5| |6| |7| |8| |9|
Ayuda
1 Bienvenido a Sonic Pi
1.1 Live Coding (Programación en vivo)
1.2 Explorando la interfaz
1.3 Aprender jugando
2 Sintetizadores
2.1 Tus primeros Sonidos
Tutorial Examples Synths Fx Samples Lang
Sonic Pi v3.2.2 on Win

Tocar una nota

```
play 65
```

Tocar una nota

Octava número	Nota número											
	C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B
–1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127				

- play 65 → Notación numérica MIDI
- play :F4 → Notación musical tradicional

Tocar un acorde

(versión MIDI)

```
play 60
play 64
play 67
```

Tocar un acorde

(versión notación musical)

```
play :C4
play :E4
play :G4
```

Tocar un acorde

(truco para los “de solfeo”)

```
play (chord :C4, :major)
```

** funciona también sin los paréntesis*

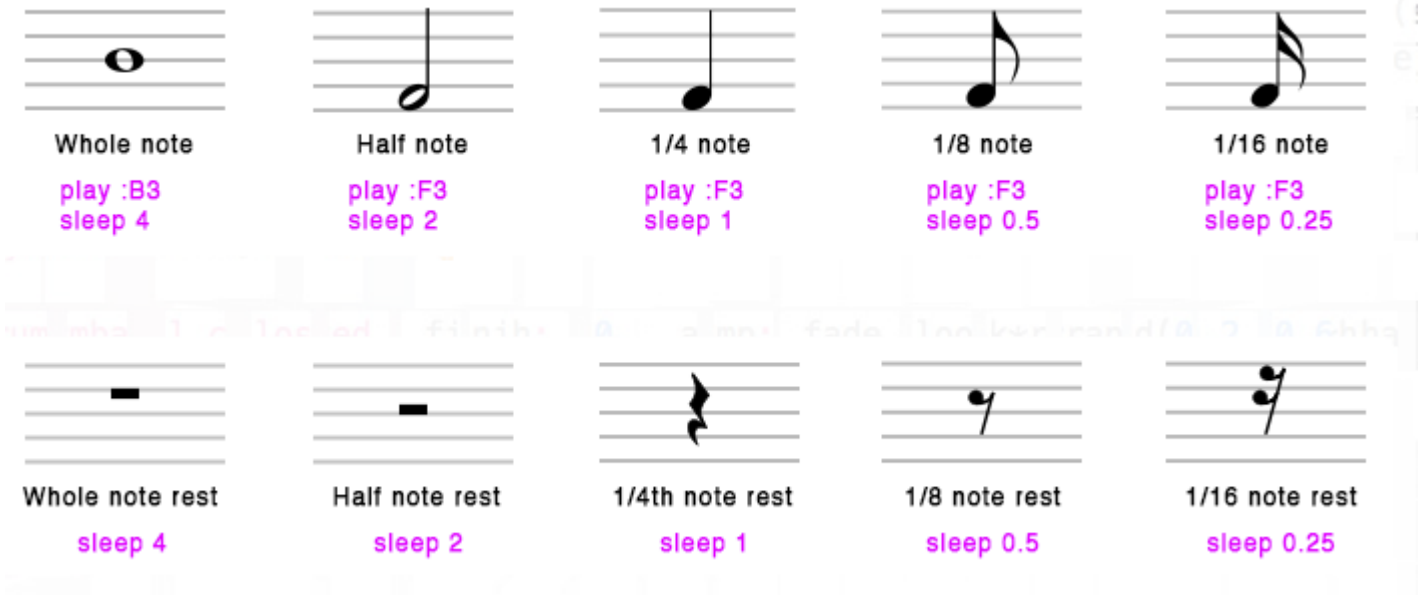
Tocar un arpegio o melodía

(uso de SLEEP)

```
play 60
sleep 1
play 64
sleep 1
play 67
```

Tocar un arpegio o melodía

(uso de SLEEP)



Simplificar la notación de un arpegio/melodía

```
play :c2
sleep 0.5
play :d2
sleep 0.25
play :e2
sleep 0.75
play :d2
sleep 0.5
```

```
play_pattern_timed [:c2, :d2, :e2, :d2], [0.5, 0.25, 0.75, 0.5]
```

BMP

Beats por minuto
(cantidad de notas por minuto)

```
use_bpm 60
play 60
sleep 1
play 64
sleep 1
play 67
sleep 1
```

```
use_bpm 120
play 60
sleep 1
play 64
sleep 1
play 67
```

Frère Jacques

(canción trad. francesa)

```
use_bpm 120
play_pattern_timed [:F4, :G4, :A4, :F4], [1, 1, 1, 1]
play_pattern_timed [:F4, :G4, :A4, :F4], [1, 1, 1, 1]
play_pattern_timed [:A4, :Bb4, :C5], [1, 1, 2]
play_pattern_timed [:A4, :Bb4, :C5], [1, 1, 2]
play_pattern_timed [:C5, :D5, :C5, :Bb4, :A4, :F4], [0.5, 0.5, 0.5, 0.5, 1, 1]
play_pattern_timed [:C5, :D5, :C5, :Bb4, :A4, :F4], [0.5, 0.5, 0.5, 0.5, 1, 1]
play_pattern_timed [:F4, :C4, :F5], [1, 1, 2]
play_pattern_timed [:F4, :C4, :F5], [1, 1, 2]
```

Volumen

- Amp: →Volumen [0 silencio, 1 normal, …]
  - play :c2, amp: 2

Balanceo

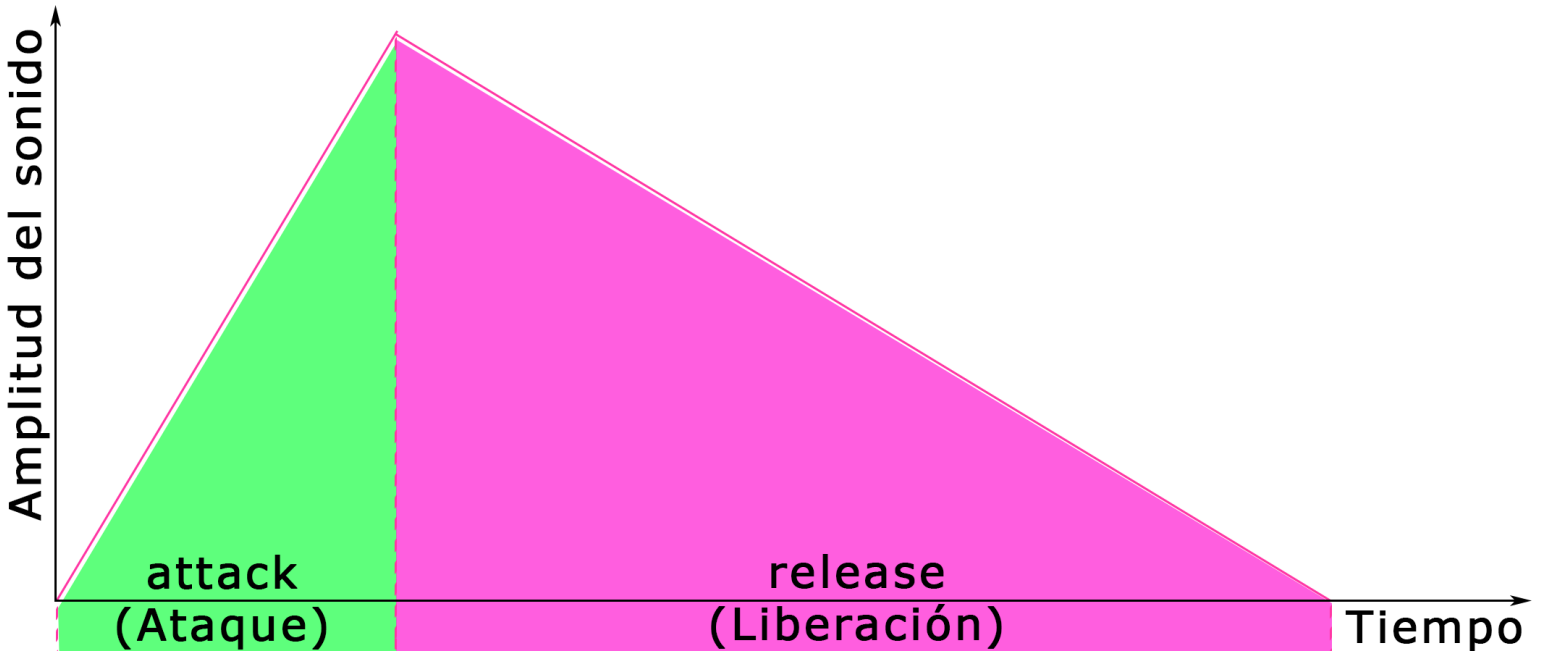
- Pan: → Balanceo Estéreo [-1 … 0 … 1]
  - play :c2, amp: 0.5, pan: -1 #izquierda
  - play :c2, amp: 0.5, pan: 0 #centro
  - play :c2, amp: 0.5, pan: 1 #derecha

Envolventes

- Describe cómo cambia un sonido a lo largo del tiempo.
  - Por ejemplo, una tecla de piano, al pulsarla y mantenerla pulsada, crea un sonido inicial casi inmediato cuyo volumen disminuye gradualmente hasta cero.

```
play 60, attack: 1, release: 3

play :c4, attack: 0, release: 0.2
```



Envolventes (II)

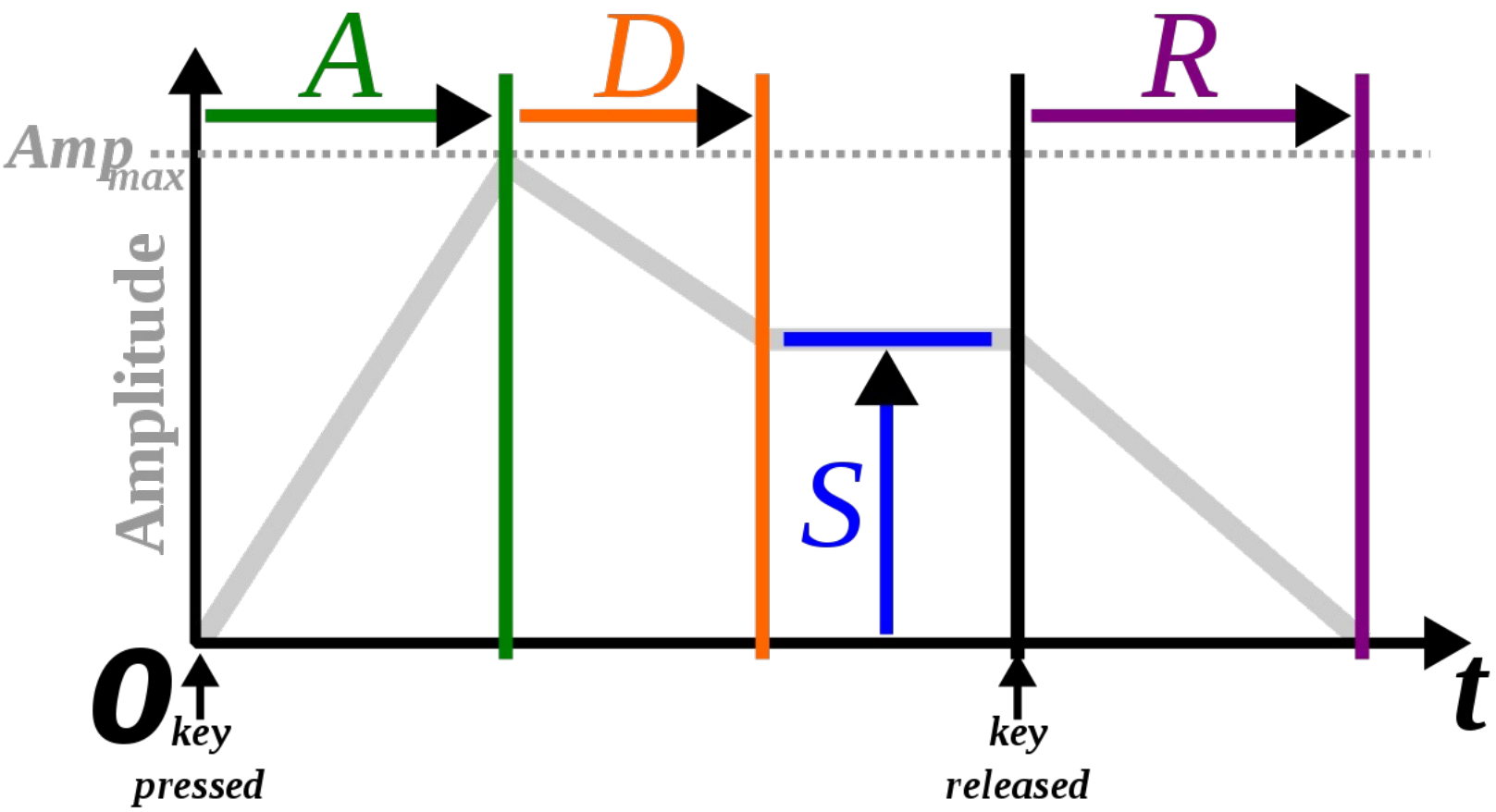
- Los generadores de envolventes, que permiten a los usuarios controlar las diferentes etapas de un sonido, son características comunes de sintetizadores, samplers y otros instrumentos musicales electrónicos.

- La forma más común de generador de envolvente se controla con cuatro parámetros: ATTACK (ataque), DECAY (decaimiento), SUSTAIN (sostenido) y RELEASE (liberación)

ADSR

- ATTACK: Es el tiempo que tarda la subida inicial del nivel desde cero hasta el pico, comenzando cuando se pulsa la tecla.
- DECAY: Es el tiempo que se tarda en bajar desde el nivel de ataque hasta el nivel de sustain designado.
- SUSTAIN: Es el nivel durante la secuencia principal de la duración del sonido, hasta que se suelta la tecla.
- RELEASE: Es el tiempo que tarda el nivel en decaer desde el nivel de sustain hasta cero después de soltar la tecla.

ADSR



Ejemplos ADSR

- SOLO ATTACK (tiempo hasta que sube el sonido a vol 1)

  play 65, attack: 0.5

- SOLO RELEASE (tiempo hasta que desaparece sonido)

  play 60, release: 2

- ATTACK Y RELEASE COMBINADOS

  play 60, attack: 0.7, release: 4

- ATTACK, RELEASE Y SUSTAIN (tiempo durante el que se mantiene sonido a 1 entre attack y release)

  play 60, attack: 0.3, sustain: 1, release: 1

Ejemplos ADSR (II)

- AÑADIR DECAY (fase entre attack y sustain), especificando volumen attack_level y sustain_level (volumen cuando empieza el release)

  play 60, attack: 0.1, attack_level: 1, decay: 0.2, sustain_level: 0.4, sustain: 1, release: 0.5

- AÑADIR DECAY LEVEL (volumen cuando acaba decay y empieza sustain)

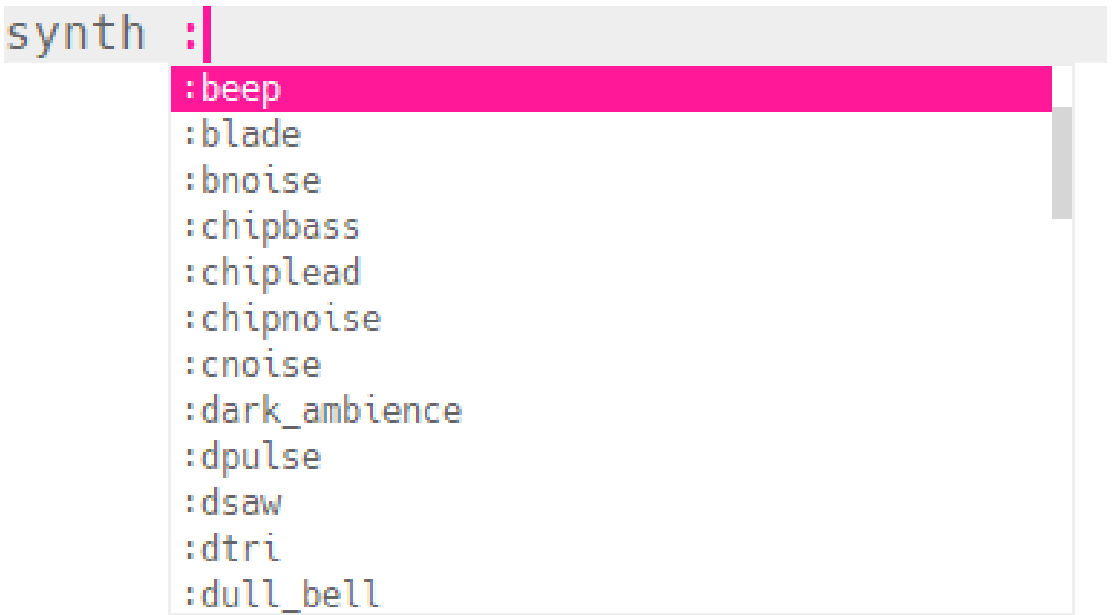
  play 60, attack: 0.1, attack_level: 1, decay: 0.2, decay_level: 0.3, sustain: 1, sustain_level: 0.4, release: 0.5

  * La duración total del sonido es la suma de las 4 fases (0.1 + 0.2 + 1 + 0.5)

Bueno, todo esto está muy bien...

pero Sonic-Pi no puede sonar más “pro”???

Sintetizadores



- use_synth :saw
  play 38

- use_synth :prophet
  play 38

Práctica

- Probar los ejemplos (notas, acordes, …) que hemos visto hasta ahora cambiando el Synth

- Probar estos Sintetizadores
  :dsaw
  :fm
  :prophet
  :pulse
  :saw
  :tb303

Un poco de “programación”...

Volvemos a Frère Jacques...

```
use_bpm 120
play_pattern_timed [:F4, :G4, :A4, :F4], [1, 1, 1, 1]
play_pattern_timed [:F4, :G4, :A4, :F4], [1, 1, 1, 1]
play_pattern_timed [:A4, :Bb4, :C5], [1, 1, 2]
play_pattern_timed [:A4, :Bb4, :C5], [1, 1, 2]
play_pattern_timed [:C5, :D5, :C5, :Bb4, :A4, :F4], [0.5, 0.5, 0.5, 0.5, 1, 1]
play_pattern_timed [:C5, :D5, :C5, :Bb4, :A4, :F4], [0.5, 0.5, 0.5, 0.5, 1, 1]
play_pattern_timed [:F4, :C4, :F5], [1, 1, 2]
play_pattern_timed [:F4, :C4, :F5], [1, 1, 2]
```

Bucles (finitos)

```
play_pattern_timed [:F4, :G4, :A4, :F4], [1, 1, 1, 1]
play_pattern_timed [:F4, :G4, :A4, :F4], [1, 1, 1, 1]
```

```
2.times do
  play_pattern_timed [:F4, :G4, :A4, :F4], [1, 1, 1, 1]
end
```

Práctica de bucles finitos

Frère Jacques

```
use_bpm 120

2.times do
  play_pattern_timed [:F4, :G4, :A4, :F4], [1, 1, 1, 1]
end

2.times do
  play_pattern_timed [:A4, :Bb4, :C5], [1, 1, 2]
end

2.times do
  play_pattern_timed [:C5, :D5, :C5, :Bb4, :A4, :F4], [0.5, 0.5, 0.5, 0.5, 1, 1]
end

2.times do
  play_pattern_timed [:F4, :C4, :F4], [1, 1, 2]
end
```

Bucles (infinitos)

```
loop do
  play 70
  sleep 0.5
end
```

Bucles (LiveCoding)

```
live_loop :nombre do
  play 65
  sleep 1
end
```

** hay que ponerle nombre*
** se puede modificar “en directo”*
** se pueden combinar varios, suenan a la vez*

Samples

** Sonic-Pi trae más de 150 samples*
** puedes usar samples externos*

Práctica de samples

:ambi_
:ambi_choir
:ambi_drone
:ambi_lunar_land
:bass_
:bd_haus
:drum_
:drum_cymbal_open
:elec_
:elec_beep
:elec_blup
:elec_flip
:elec_plip
:guit_
:loop_amen
:misc_
:perc_

Vamos a tocar la batería!

```
sample :drum
```

```
:drum_bass_hard
:drum_bass_soft
:drum_cowbell
:drum_cymbal_closed
:drum_cymbal_hard
:drum_cymbal_open
:drum_cymbal_pedal
:drum_cymbal_soft
:drum_heavy_kick
:drum_roll
:drum_snare_hard
:drum_snare_soft
```

Vamos a tocar la batería (II)

```
live_loop :bombo_caja do
  sample :drum_heavy_kick
  sleep 1
  sample :drum_snare_hard
  sleep 1
end
```

Vamos a tocar la batería (III)

```
live_loop :charles do
  sample :drum_cymbal_closed
  sleep 0.25
  sample :drum_cymbal_pedal
  sleep 0.5
end
```

Vamos a tocar la batería! (IV)

```
use_bpm 100

live_loop :bombo_caja do
  sample :drum_heavy_kick
  sleep 1
  sample :drum_snare_hard
  sleep 1
end

live_loop :charles do
  sample :drum_cymbal_closed
  sleep 0.25
  sample :drum_cymbal_pedal
  sleep 0.5
end
```

Ejemplos “pro” de batería

http://citme.music.asu.edu/resources-steam/remixsonicpicode/

** probar → “Complicated Drum Beat No Comments”*

Envolventes (y más) con samples….

VOLUMEN Y PANEO EN SAMPLES
sample :ambi_lunar_land, amp: 0.5
sample :loop_amen, pan: 1

VELOCIDAD DE REPRODUCCIÓN
sample :ambi_choir, rate: 1.5 (más rápido y agudo)
sample :ambi_choir, rate: 0.5 (más lento y grave)

REPRODUCIR AL REVÉS
sample :loop_amen, rate: -1

ENVOLVENTE ADSR CON SAMPLES
sample :loop_amen, attack: 1
sample :loop_amen, release: 0.75

EFECTO FADE IN / FADE OUT
```
sample :loop_amen, attack: 0.75, release: 0.75
```

EFECTOS EN CIMBAL: MÁS PERCUSIVO, MÁS ESTIRADO
```
sample :drum_cymbal_open, attack: 0.01, sustain: 0, release: 0.1
sample :drum_cymbal_open, attack: 0.01, sustain: 0.3, release: 0.1
```

COMBINACIÓN DE TODO LO ANTERIOR
```
sample :loop_amen, rate: 2, attack: 0.01, sustain: 0, release: 0.35
```

REPRODUCIR DESDE A HASTA B
```
sample :loop_amen, start: 0.5
sample :loop_amen, finish: 0.5
sample :loop_amen, start: 0.4, finish: 0.6
```

REPRODUCIR AL REVÉS (ALTERNATIVO)
```
sample :loop_amen, start: 0.6, finish: 0.4
```

COMBINANDO TODO
```
sample :loop_amen, start: 0.5, finish: 0.7, rate: 0.2
sample :loop_amen, start: 0.5, finish: 0.8, rate: -0.2, attack: 0.3, release: 1
```

Vamos a tocar un poco el bajo...

Sintetizadores

```
synth :
```

- :beep
- :blade
- :bnoise
- :chipbass
- :chiplead
- :chipnoise
- :cnoise
- :dark_ambience
- :dpulse
- :dsaw
- :dtri
- :dull_bell

```
use_synth :hoover
loop do
  play 60
  sleep 0.5
  play 67
  sleep 0.5
end
```

Creando una pista de bajo

```
live_loop :bajo do
  use_synth :fm
  play :c2, attack: 0, release: 0.25
  sleep 0.25
  play :c2, attack: 0, release: 0.3
  sleep 2
  play :e2
  sleep 0.75
  play :f2
  sleep 1
end
```

Nuestro primer Backing Track

```
use_bpm 100

live_loop :bombo_caja do
  sample :drum_heavy_kick
  sleep 1
  sample :drum_snare_hard
  sleep 1
end

live_loop :charles do
  sample :drum_cymbal_closed
  sleep 0.25
  sample :drum_cymbal_pedal
  sleep 0.5
end
```

```
live_loop :bajo do
  use_synth :fm
  play :c2, attack: 0, release: 0.25
  sleep 0.25
  play :c2, attack: 0, release: 0.3
  sleep 2
  play :e2
  sleep 0.75
  play :f2
  sleep 1
end
```

Funciones avanzadas

Aleatorización I:

La función “dado” (one_in)

```
live_loop :drums do
  if one_in(2)
    sample :drum_heavy_kick
    sleep 0.5
  else
    sample :drum_cymbal_closed
    sleep 0.25
  end
end
```

Aleatorización II:

El parámetro .tick

```
loop do
  play [:C4, :E4, :G4].tick
  sleep 1
end
```

** .tick → "elige" la siguiente opción de la lista*

Aleatorización III:
El parámetro .choose

```
loop do
  play [:c4, :e4, :g4].choose
  sleep 1
end
```

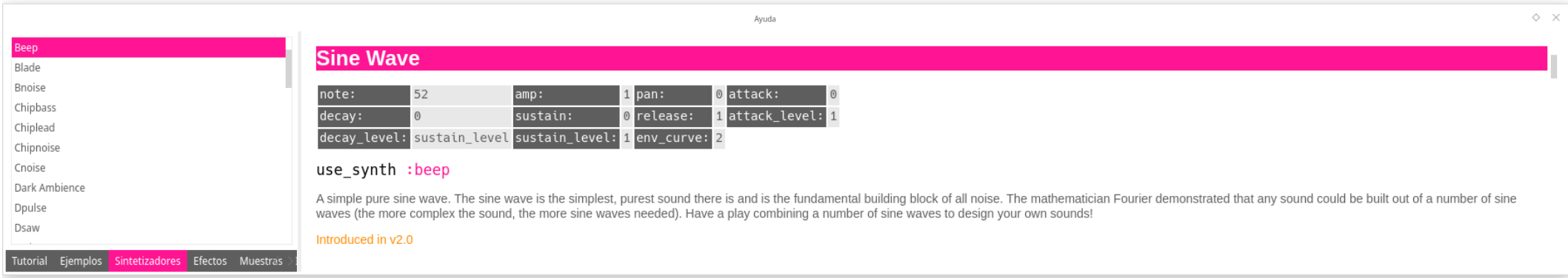
** .choose → "elige" una de las opciones de la lista*

Aleatorización III:
El parámetro .choose (II)

```
live_loop :sinte_bajo do
  use_synth :tb303
  use_octave [0,1].choose
  play (chord :C2, :major).choose, release: 0.125, cutoff: 100
  sleep [0.25, 0.125, 0.5].choose
end
```

** cutoff → elimina las frecuencias por encima de la indicada*

Uso de la ayuda de Sonic-Pi



Beep
Blade
Bnoise
Chipbass
Chiplead
Chipnoise
Cnoise
Dark Ambience
Dpulse
Dsaw
Dtri
Dull Bell
Fm
Gnoise
Growl
Hollow
Hoover
Kalimba
Mod Beep
Mod Dsaw
Mod Fm
Mod Pulse
Mod Saw
Mod Sine
Mod Tri
Noise
Piano
Pluck
Pnoise
Pretty Bell
Prophet
Pulse
Rodeo
Saw
Sine
Sound In
Sound In Stereo
Square
Subpulse
Supersaw
Tb303
Tech Saws
Tri
Zawa

TB-303 Emulation

note:	52	amp:	1	pan:	0	attack:	0
decay:	0	sustain:	0	release:	1	attack_level:	1
decay_level:	sustain_level	sustain_level:	1	env_curve:	2	cutoff:	120
cutoff_min:	30	cutoff_attack:	attack	cutoff_decay:	decay	cutoff_sustain:	sustain
cutoff_release:	release	cutoff_attack_level:	1	cutoff_decay_level:	cutoff_sustain_level	cutoff_sustain_level:	1
res:	0.9	wave:	0	pulse_width:	0.5		

```
use_synth :tb303
```

Emulation of the classic Roland TB-303 Bass Line synthesiser. Overdrive the res (i.e. use very large values) for that classic late 80s acid sound.

Introduced in v2.0

Options

note:

Note to play. Either a MIDI number or a symbol representing a note. For example: `30`, `52`, `:C`, `:C2`, `:Eb4`, or `:Ds3`

Default: 52
Must be zero or greater
May be changed whilst playing
Has slide options to shape changes

amp:

The amplitude of the sound. Typically a value between 0 and 1. Higher amplitudes may be used, but won't make the sound louder, they will just reduce the quality of all the sounds currently being played (due to compression.)

Default: 1
Must be zero or greater
May be changed whilst playing
Has slide options to shape changes

pan:

Position of sound in stereo. With headphones on, this means how much of the sound is in the left ear, and how much is in the right ear. With a value of -1, the sound is completely in the left ear, a value of 0 puts the sound equally in both ears and a value of 1 puts the sound in the right ear. Values in between -1 and 1 move the sound accordingly.

Default: 0
Must be a value between -1 and 1 inclusively
May be changed whilst playing
Has slide options to shape changes

attack:

Amount of time (in beats) for sound to reach full amplitude (attack_level). A short attack (i.e. 0.01) makes the initial part of the sound very percussive like a sharp tap. A longer attack (i.e 1) fades the sound in gently. Full length of sound is attack + decay + sustain + release.

Default: 0
Must be zero or greater
Can not be changed once set
Scaled with current BPM value

decay:

Amount of time (in beats) for the sound to move from full amplitude (attack_level) to the sustain amplitude (sustain_level).

Default: 0
Must be zero or greater
Can not be changed once set
Scaled with current BPM value

sustain:

Amount of time (in beats) for sound to remain at sustain level amplitude. Longer sustain values result in longer sounds. Full length of sound is attack + decay + sustain + release.

Default: 0
Must be zero or greater

Bueno, vale, …y qué otros parámetros interesantes me estoy perdiendo????

Más opciones de “play” (II)

```
loop do
  sample :drum_cymbal_closed, pan: rrand(-1, 1)
  sleep 0.5
end
```

** rrand( ) → elige aleatoriamente un valor*

Más opciones de “sample”

```
live_loop :prueba do
  sample :loop_amen
  sleep 2
end
```

** qué suena “raro” aquí???*

Más opciones de “sample” (beat_stretch)

```
live_loop :prueba do
  sample :loop_amen, beat_stretch:2
  sleep 2
end
```

** beat_strech → adapta el sample a la duración indicada*

Más opciones de “sample” (rate)

```
live_loop :prueba do
  sample :loop_amen, beat_stretch:2, rate: 2
  sleep 1
end
```

** rate → ajusta la velocidad de reproducción del sample*

Efectos (with_fx :)

```
live_loop :efecto do
  with_fx :reverb do
    play 50
    sleep 0.5
    sample :elec_plip
    sleep 0.5
    play 62
  end
end
```

** reverb, echo, distortion, flanger, autotuner, ...*

Autotuner
Band Eq
Bitcrusher
BPF
Compressor
Distortion
Echo
Eq
Flanger
Gverb
HPF
Ixi Techno
Krush
Level
LPF
Mono
NBPF
NHPF
NLPF
Normaliser
NRBPF
NRHPF
NRLPF
Octaver
Pan
Panslicer
Ping Pong
Pitch Shift
RBPF
Record
Reverb
RHPF
Ring Mod
RLPF
Slicer
Sound Out
Sound Out Stereo
Tanh
Tremolo
Vowel
Whammy
Wobble

Flanger

amp:	1	mix:	1	pre_mix:	1	pre_amp:	1
phase:	4	phase_offset:	0	wave:	4	invert_wave:	0
stereo_invert_wave:	0	delay:	5	max_delay:	20	depth:	5
decay:	2	feedback:	0	invert_flange:	0		

```
with_fx :flanger do
  play 50
end
```

Mix the incoming signal with a copy of itself which has a rate modulating faster and slower than the original. Creates a swirling/whooshing effect.

Introduced in v2.3

Options

amp:	The amplitude of the sound. Typically a value between 0 and 1. Higher amplitudes may be used, but won't make the sound louder, they will just reduce the quality of all the sounds currently being played (due to compression.) *Default: 1* *Must be zero or greater* *May be changed whilst playing* *Has slide options to shape changes*
mix:	The amount (percentage) of FX present in the resulting sound represented as a value between 0 and 1. For example, a mix of 0 means that only the original sound is heard, a mix of 1 means that only the FX is heard (typically the default) and a mix of 0.5 means that half the original and half of the FX is heard. *Default: 1* *Must be a value between 0 and 1 inclusively* *May be changed whilst playing* *Has slide options to shape changes*
pre_mix:	The amount (percentage) of the original signal that is fed into the internal FX system as a value between 0 and 1. With a pre_mix: of 0 the FX is completely bypassed unlike a mix: of 0 where the internal FX is still being fed the original signal but the output of the FX is ignored. The difference between the two is subtle but important and is evident when the FX has a residual component such as echo or reverb. When switching mix: from 0 to 1, the residual component of the FX's output from previous audio is present in the output signal. With pre_mix: there is no residual component of the previous audio in the output signal. *Default: 1* *Must be zero or greater* *May be changed whilst playing* *Has slide options to shape changes*
pre_amp:	Amplification applied to the input signal immediately before it is passed to the FX. *Default: 1* *Must be zero or greater* *May be changed whilst playing* *Has slide options to shape changes*
phase:	Phase duration in beats of flanger modulation. *Default: 4* *Must be greater than zero* *May be changed whilst playing* *Has slide options to shape changes* *Scaled with current BPM value*
phase_offset:	Initial modulation phase offset (a value between 0 and 1).

… y esto es sólo el principio!!!

LiveCoding con Sonic-Pi

https://www.youtube.com/watch?v=JiQHclg_648

Referencias

- La ayuda de Sonic-Pi
- https://rubentr.github.io/introduccion-sonic-pi/#/
- http://sonic-pi.mehackit.org/index_es.html
- https://magpi.raspberrypi.com/books/essentials-sonic-pi-v1
- https://thinking.is.ed.ac.uk/ada-lovelace-day/wp-content/uploads/sites/13/2015/10/Sonic_pi_workshop.pdf
- https://projects.raspberrypi.org/en/projects/getting-started-with-sonic-pi/

Gracias por vuestra atención!