List of TOPOS commands

Rhythm

- onbeat(...n: number[])
- oncount(beats: number[], meter: number)
- beat(n: number | number[] = 1, offset: number = 1)
- pulse(n: number | number[] = 1, offset: number = 1)
- bar(n: number | number[] = 1, offset: number = 1)
- dur(...list: numbers[]) // keeps the same value for a duration of n beats corresponding to the nth number of the list you provide - e.g. beat(0.5)::sound('sine').n([1,2].dur(1, 2))
- seq(expr: string, duration: number = 0.5)
- fullseq(expr: string, duration: number = 0.5)
- rhythm(divisor: number, pulses: number, length: number, rotate: number)
- bin(iterator: number, n: number) // iterator = $(1) then a number // $ is short hand for counter, plus the counter number
- binrhythm(divisor: number, n: number)
- prob(n: number)

Arrangement

- script([1,2,3,4].bar(8)) // Playing each script for 8 bars in succession
- script([1,2,3,4].dur(8, 2, 16, 4)) // You can also give a specific duration to each section using .dur
- flip(n: number, ratio: number = 50) // ratio is a percentage -
  - E.g. flip(4, 50) :: beat(1) :: snd('kick').out()
  - E.g. if (flip(4, 75)) { beat(1) :: snd('kick').out() } else { beat(.5) :: snd('snare').out() }
  - E.g. beat(0.5) :: snd(flip(2) ? 'bd' : 'hh').out()
- flipbar(n: number = 1)
- onbar(bars: number | number[], n: number)

Sounds

- beat(1) :: sound("bd").out() // The .out method can take an optional argument to send the sound to a numbered effect bus, from 0 to n
- The .n(number) method can be used to pick a sample from the currently selected sample folder // **on samples**
- The .n(number) method determines the number of harmonics in your waveform // **on synths**
- Use .orbit(n: number) to choose which output/orbit your sound will use
- .gain() // volume of the synth or sample (exponential)
- .vel(0 to 1) // multiplied with gain
- .db() // from -inf to +10
- .adsr(n, n, n, n) // attack, decay, sustain, release
- .ad(n, n) // attack, decay

- .lpf // cutoff freq for low pass filter
- .lpq(0 to 1) // resonance of low pass filter
- .hpf // cutoff freq for high pass filter
- .hpq(0 to 1) // resonance of high pass filter
- .bpf // cutoff freq for band pass filter
- .bpq(0 to 1) // resonance of band pass filter
- .ftype("12db" or "24db")
- .lpenv | .hpenv | .bpenv // frequency mod amount
- .lpa | .hpa | .bpa // attack time of filter env
- .lpd | .hpd |.bpd // decay time of filter env
- .lps | .hps | .bps // sustain level of filter env
- .lpr | .hpr | .bpr //release time of filter env
- .lpadsr |.hpadsr | .bpadsr //shorthand for above

Samples

- .begin(0 to 1) // sample start point
- .end(0 to 1) // sample end point
- .loopBegin(0 to 1) // loop section start
- .loopEnd(0 to 1) // loop section end
- .loop(0 or 1) // loop sample - 0 = off, 1 = on
- .stretch(n) // stretches audio playback rate over n beats
- .speed(n) // playback speed - negative values reverse the sample
- .cut(0 or 1) // cuts sample when another sample is played on the same orbit
- .clip(n) // multiply duration of sample with n
- .pan(0 to 1) // stereo position 0 = left, 1 = right
- .vib(n) // vibrato speed in Hz
- .vibmod(0 to n) // vibrato depth

Synths

- sound("sine", "triangle", "sawtooth", "square") //Classic oscillator shapes
- sound("brown", "pink", "white", "crackle") // Noise colours
- .freq(n) // frequency of the oscillator
- .note(n or note name) // note number or note name, e.g. 60, or C4
- .chord(chord name) // e.g. "C","Em7","Fmaj7","Emin"
- .chord(numbers) // e.g. chord(60,64,67)
- .invert(-n to n) // inverts chord depending on n
- .noise(n) // adds brown noise
- sound("wt_name") // anything using wt_ prefix will be treated as a wavetable from the AKWF pack. (See appendix at end of document)
- .fmi(1 to n) // fm index
- .fmh(1 to n) // fm ratio
- .fmwave("sine", "triangle", "sawtooth", "square") // fm waveform
- .fm("2:4") // shortcut for fmi and fmh
- .fmatk(n) // attack time for mod env

- .fmdec(n) // decay time for mod env
- .fmsus(n) // sustain level for mod env
- .fmrel(n) // release time for mod env

ZZFX synth

- sound(['z_sine', 'z_triangle', 'z_sawtooth', 'z_tan', 'z_noise)
- .zrand // randomisation factor
- .atk // attack time of env
- .dec // decay time of env
- .sus // sustain level of env
- .rel // release time of env
- .vol // volume
- .freq // frequency
- .note // note name
- .curve(0 to 3) // oscillator wave shaping
- .sld // pitch slide
- .dslide // other pitch slide
- .pj // pitch change after pitch jump time
- .pjt(n) // applies pitch jump after n
- .noise // adds noise
- .zcrush // bit crushing
- .zdelay // tiny delay
- .tremolo // amplitude tremolo
- .zmod // fm speed
- .duration // total sound duration (overrides env)

Speech synthesis

- e.g. const words = [“First Word”, “Second Word”]

onbeat(4) :: words.voice().pitch().lang(“fr”).volume().rate().speak()

Effects

- .room(0 to 1 // can go higher) // reverb level
- .size(0 to n) // room size of reverb
- .roomfade(n) // fade time in seconds
- .roomlp // low pass filter of reverb in Hz
- .roomdim // low pass freq at -60db in Hz
- .delay(0 to 1) // wet/dry mix
- .delayt(n) // delay time in milliseconds
- .delayfb(0 to 1) // delay feedback
- .phas(1 to n) // phaser speed
- .phasdepth(0 to 1) // amount of signal to go through phaser
- .phassweep(n) // freq sweep in Hz

- .phascenter(n to 1000) // center frequency
- .course() // sample rate
- .crush() // bit crushing - lower number is more extreme
- .shape(0 to 1) // wave shaper
- .vib() // as above
- .vibmod() // as above
- .cmp () // threshold value in dB
- .rt() // ratio
- .kn() // dB value for knee
- .cmpa(n) // attack time in seconds
- .cmpr(n) // release time in seconds

Midi

- midi_outputs() // this lists your midi inputs and outputs available
- midi_output("IAC driver") // enter the interface you want to use as a string
- midi(n, n, n) // note number, velocity, midi channel
- control_change({control: number, value: number, channel: number}) // passed as object
- program_change(program: number, channel: number)

Patterning things

- use .beat() to pattern other functions. Beat itself can be patterned.
- use .bar() to pattern other functions. Bar itself can be patterned.
- You can use .dur() on beat() and bar()
- .dur(...list: numbers[]) // keeps the same value for a duration of n beats corresponding to the nth number of the list you provide - e.g. beat(0.5)::sound('sine').n([1,2].dur(1, 2))
- .counter(name, limit?, step?) // returns the next value on the list based on counter value. The limit is optional and defaults to the length of the list. The step is optional and defaults to 1. Setting / changing limit will reset the counter
- $ is an alias for the above counter function

Pitch

- .pitch(n) // converts integer to pitch classes
- .semitones(n) // creates scale from semitone intervals
- .cents(n) // creates scale from cent intervals
- .ratios(n) // creates scale from ratios
- .edo(number, scale?: string|number[]) // creates scale from equal divisions of the octave
- .scale(scale: string, base note: number) // See here and here - can use these binary numbers or names as scales to use
- .scaleArp(scale: string, mask: number) // e.g. sound('sine').note([60].scaleArp("major", 3).beat(1)).out() // returns [0,2,4]
- You can use greek modal names (Ionian/Aeolian) or western names (Major/Minor)
- Microtonal scales can be used using Scala format or extended notation e.g.

- Young: 106. 198. 306.2 400.1 502. 604. 697.9 806.1 898.1 1004.1 1102. 1200.
- Wendy carlos: 17/16 9/8 6/5 5/4 4/3 11/8 3/2 13/8 5/3 7/4 15/8 2/1

Data Operations

- .palindrome() // concatenates a list with the same list in reverse
- .rand(index: number) // pick random element in given list
- .pick() // takes no arguments
- .degrade(amount: number) // removes *n*% of the list elements
- .repeat(amount: number) // repeat every list elements *n* times
- .repeatEven(amount: number) // repeat every pair element of the list *n* times
- .repeatOdd(amount: number) // repeat every odd element of the list *n* times
- .loop(index: number) // loop takes one argument, the index. It allows you to iterate over a list using an iterator such as a counter - e.g. sound('numbers').n([1,2,3,4,5,6].loop($(1, 5, 1))).out()
- .shuffle() // shuffles list
- .rotate(steps: number) // rotates list to the right
- .unique() // filter a list to remove repeated values
- .add() // add a given amount to every list element
- .sub() // subtract amount from every list element
- .mult() //multiply every list element by this number
- .div() // divide every list element by this number
- .counter(n, n, n) // counter name, limit, incrementor // $ is shorthand for this .counter()
- .drunk(n) // drunk walk counter
- .drunk_max(n) // sets max value of drunk counter
- .drunk_min(n) // sets min value of drunk counter
- .drunk_wrap(bool) // whether to wrap to 0 once upper limit is reached

Global Variables

- There is a global object that you can use to store and retrieve information. It is a simple key/value store. You can store any type of data in it
  - e.g. global.my_variable = 2 // you can use g for short
- This allows different scripts to share information
- Each script has its own iterator, accessed by the i variable

LFOs

- sine(freq: n, phase: n) // bipolar between -1 and 1 // freq in Hz
- usine(freq: n, phase: n) // unipolar between 0 and 1
- triangle // as above
- utriangle // as above
- saw // as above
- usaw // as above
- square(freq: n, duty: n) // duty cycle is 2nd argument

- usquare // as above
- noise(n) // returns random value between -1 and 1
- unoise(n) // returns random value between 0 and 1

Probabilities

- .r(min: number, max:number) // returns random number between min and max
- .ir(min: number, max:number) // returns random integer between min and max
- prob(n) // returns true n% of the time
- toss() // either returns true or false - coin toss
- seed(n | string) // sets seed of random number generator
- .odds(n: number, beats?: number) // returns true for every n (odds) (eg. 1/4 = 0.25) in given number of beats
- never(beats?: number): returns false
- almostNever(beats?: number): returns true 0.1% of the time in given number of beats
- rarely(beats?: number): returns true 1% of the time in given number of beats
- scarcely(beats?: number): returns true 10% of the time in given number of beats
- sometimes(beats?: number): returns true 50% of the time in given number of beats
- often(beats?: number): returns true 75% of the time in given number of beats
- frequently(beats?: number): returns true 90% of the time in given number of beats
- almostAlways(beats?: number): returns true 99% of the time in given number of beats
- always(beats?: number): returns true. Can be handy when switching between different probabilities
- once(): returns true once, then false until the code is force evaluated (Shift+Ctrl+Enter)
- Below is a list of probabilities:

• evenbar	• If the current bar is even	• .evenbar(s => s.note(58))
• even	• If the current beat is even	• .even(s => s.note(59))
• odd	• If the current beat is odd	• .odd(s => s.note(61))
• odds	• With a given probability	• .odds(0.3, s => s.note(62))
• never	• Never transforms the event	• .never(s => s.note(63))
• almostNever	• With a 2.5% probability.	• .almostNever(s => s.note(64))
• rarely	• With a 10% probability.	• .rarely(s => s.note(65))
• scarcely	• With a 25% probability.	• .scarcely(s => s.note(66))
• sometimes	• With a 50% probability.	• .sometimes(s => s.note(67))
• often	• With a 75% probability.	• .often(s => s.note(68))
• frequently	• With a 90% probability.	• .frequently(s => s.note(69))
• almostAlways	• With a 98.5% probability.	• .almostAlways(s => s.note(70))
• always	• Always transforms the Event.	• .always(s => s.note(71))

Register

- Use register(args) to register a chain you want to reuse e.g.
  - // Playing with extreme panning and playback rate
    - register('juxrev', n=>n.pan([0, 1]).speed([1, -1]))
  - // Using our new abstraction
    - beat(1)::sound('fhh').juxrev().out()
- Use to store presets etc
  - // Registering a specific synth architecture
    - register('sub', (n,x=4,y=80)=>n.ad(0, .25).fmi(x).pan([0, 1]).delay(0.5).delayt(1/8).delayfb(1/3).lpf(25+usine(1/3)*y).lpad(4, 0, .25))
  - // Using it with an arpeggio
    - rhythm(.25, [6, 8].beat(), 12)::sound('sine').note([0, 2, 4, 5].scale('minor', 50).beat(0.5)).sub(8).out()
- The chain can also be registered automatically for all events. This is useful if you want to add a specific effect to all your events e.g.
  - z0("h 9 ^ <7 5 3 1>").sound("sine").out()
  - z1("0 4 3 2").sound("sine").out()
  - all(x=>x.room(1).delay(rI(0,0.5)))
- .log() // print value to console - can use to see note, pitch, duration, octave etc

Scripts

- script(n) // calls script n e.g. script(1) or script(1,2,3,4)
- clear_script(n) // deletes numbered script
- copy_script(n, n) // copies script from script n to script n

Maths

- max(...values: number[]) // returns the maximum value of a list of numbers
- min(...values: number[]) // returns the minimum value of a list of numbers
- mean(...values: number[]) // returns the arithmetic mean of a list of number
- limit(value: number, min: number, max: number) // limits a value between a minimum and a maximum
- .linlin(inMin: number, inMax: number, outMin: number, outMax: number): scale linearly from one range to another.
- .linexp(inMin: number, inMax: number, outMin: number, outMax: number): scale a linear range to an exponential range.
- .explin(inMin: number, inMax: number, outMin: number, outMax: number): scale an exponential range to a linear range.
- .expexp(inMin: number, inMax: number, outMin: number, outMax: number): scale an exponential range to another exponential range.
- .lincurve(inMin: number, inMax: number, outMin: number, outMax: number, curve: number): scale a number from one range to another following a specific curve.
  - curve: number: 0 is linear, < 0 is concave, negatively curved, > 0 is convex, positively curved

- delay(ms: number, func: Function): void: Delays the execution of a function by a given number of milliseconds
- delayr(ms: number, nb: number, func: Function): void: Delays the execution of a function by a given number of milliseconds, repeated a given number of times

Ziffers

- Notation written in JS strings e.g.
  - z1('0.25 0 1 2 3 4 5 6 7 8 9')
  - z1('_ _ 0 {9 10 11} 4 {12 13 14}')
  - z1('^ 1/8 0 1 b2 3 4 _ 4 b5 4 3 b2 1 0')
- Space is used as separator

Syntax	Symbol	Description
• Pitches	• 0-9 {10 11 21}	• Numbers or escaped numbers in curly brackets
• Duration	• 0.25, 0.5	• Floating point numbers can also be used as durations
• Duration	• 1/4, 1/16	• Fractions can be used as durations
• Subdivision	• [1 [2 3]]	• Durations can be subdivided using square brackets
• Cycles	• 1 <2 4>	• Cycle values within the pattern
• Octave	• ^ _	• ^ for octave up and _ for octave down
• Accidentals	• # b	• Sharp and flats
• Rest	• r	• Rest / silences
• Repeat	• !1-9	• Repeat the item 1 to 9 times
• Chords	• [1-9]+ / [iv]+ / [AG] +name	• Multiple pitches grouped together, roman numerals or named chords
• Samples	• [a-z0-9_]+	• Samples can be used pitched or unpitched
• Index/ Channel	• [a-z0-9]+:[0-9]*	• Samples or midi channel can be changed using a colon

- Rhythms can be written as duration characters, fractions or decimal e.g.
  - z1('q 0 0 4 4 5 5 h4 q 3 3 2 2 1 1 h0')
  - z1('1/4 0 0 4 4 5 5 2/4 4
  - z1('0.25 5 1 2 6 0.125 3 8 0.5 4 1.0 0')
- All duration characters are listed below:

Character	Fraction	Duration	Name (US)	Name (UK)
• m..	• 14/1	• 14	• Double dotted maxima	• Double dotted Large
• m.	• 12/1	• 12	• Dotted maxima	• Dotted Large

• m	• 8/1	• 8	• Maxima	• Large
• l..	• 7/1	• 7	• Double dotted long note	• Double dotted longa
• l.	• 6/1	• 6	• Long dotted note	• Longa dotted
• l	• 4/1	• 4	• Long	• Longa
• d..	• 7/2	• 3.5	• Double dotted long note	• Double dotted breve
• d.	• 3/3	• 3	• Dotted whole note	• Double breve
• n	• 8/3	• 2.6666	• Triplet Long	• Triplet longa
• d	• 2/1	• 2	• Double whole note	• Breve
• w..	• 7/4	• 1.75	• Double dotted whole note	• Double dotted breve
• w.	• 3/2	• 1.5	• Dotted whole note	• Dotted breve
• k	• 4/3	• 1.3333	• Triplet double whole	• Triplet breve
• w	• 1/1	• 1	• Whole note	• Semibreve
• h..	• 7/8	• 0.875	• Double dotted half note	• Double dotted minim
• h.	• 3/4	• 0.75	• Dotted half note	• Dotted minim
• c	• 2/3	• 0.6666	• Triplet whole	• Triplet semibreve
• h	• 1/2	• 0.5	• Half note	• Minim
• p	• 1/3	• 0.3333	• Triplet half	• Triplet minim
• q..	• 7/16	• 0.4375	• Double dotted quarter note	• Double dotted crotchet
• q.	• 3/8	• 0.375	• Dotted quarter note	• Dotted crotchet
• q	• 1/4	• 0.25	• Quarter note	• Crotchet
• e..	• 7/32	• 0.2187	• Double dotted 8th	• Double dotted quaver
• e.	• 3/16	• 0.1875	• Dotted 8th	• Dotted quaver
• g	• 1/6	• 0.1666	• Triplet quarter	• Triplet crochet
• e	• 1/8	• 0.125	• 8th note	• Quaver
• s..	• 7/64	• 0.1093	• Double dotted 16th	• Double dotted semiquaver
• a	• 1/12	• 0.0833	• Triplet 8th	• Triplet quaver
• s.	• 3/32	• 0.0937	• Dotted 16th	• Dotted semiquaver
• s	• 1/16	• 0.0625	• 16th note	• Semiquaver
• t..	• 7/128	• 0.0546	• Double dotted 32th	• Double dotted demisemiquaver
• t.	• 3/64	• 0.0468	• Dotted 32th	• Dotted demisemiquaver
• f	• 1/24	• 0.0416	• Triplet 16th	• Triplet semiquaver
• t	• 1/32	• 0.0312	• 32th note	• Demisemiquaver

• u..	• 7/256	• 0.0273	• Double dotted 64th	• Double dotted hemidemisemiquaver
• u.	• 3/128	• 0.0234	• Dotted 64th	• Dotted hemidemisemiquaver
• x	• 1/48	• 0.0208	• Triplet 32th	• Triplet demi-semiquaver
• u	• 1/64	• 0.0156	• 64th note	• Hemidemisemiquaver
• o..	• 7/512	• 0.0136	• Double dotted 128th note	• Double dotted semihemidemisemiquaver
• y	• 1/96	• 0.0104	• Triplet 64th	• Triplet hemidemisemiquaver
• o.	• 3/256	• 0.0117	• Dotted 128th note	• Dotted semihemidemisemiquaver
• o	• 1/128	• 0.0078	• 128th note	• Semihemidemisemiquaver
• j	• 1/192	• 0.0052	• Triplet 128th	• Triplet semihemidemisemiquaver
• z	• 0/1	• 0	• No length	• No length

- Chords can be built by grouping pitches or using roman numeral notation, or by using named chords e.g.
  - z1('1.0 024 045 058 046 014')
  - z1('2/4 i vi ii v')
  - z1('0.25 Bmaj7!2 D7!2 _ Gmaj7!2 Bb7!2 ^ Ebmaj7!2')
  - z1('q Amin!2').key(["A2", "E2"].beat(4))
  - z1('i i v%-4 v%-2 vi%-5)
- .invert(n) // will invert chord
- .key(string) // will transpose chord
- You can also write the inversion in the ziffer pattern using & sign e.g. z1(Cmin Cmin%-1)
- .arpeggio(n) // will arpeggio over the chords
  - e.g. z1("i v%-1 vi%-1 iv%-2")
    .arpeggio(0,2,1,2)
    .noteLength(0.125)
    .sound("sine").out()
- Arpeggios can be written in the mini notation e.g. z1("(i v vi%-3 iv%-2)@(s 0 2 0 1 2 1 0 2)
- Functions can be chained on ziffer patterns e.g.
  - z1('0 1 2 3 4').key('G3').scale('minor').sound('sine').often(n => n.pitch+=3).out()
- Samples can be patterned using the sample names or using @-operator for assigning sample to a pitch. Sample index can be changed using the : operator e.g.
  - z1('bd [hh hh]').octave(-2).sound('sine')
  - z1('0@sax 3@sax 2@sax 6@sax').octave(-1).sound()
  - z1('e (0 2 6 3 5 -2)@sax (0 2 6 3 5 -2)@arp')
  - z1('e 1:2 4:3 6:2').octave(-1).sound("east").out()
- List operations e.g. z1("1/8 ((3 2 1)+(2 5)) 0 (2 3 5)-(1 2) // All arithmetic operators are supported
- Random numbers: (4,6) // random number between 4 and 6

- Variables are supported within the ziffer patterns e.g.
  - A=(0 2 3 (1,4)) Assign pre-evaluated list to a variable
  - B~(0 2 3 (1,4)) Assign list with operations to a variable
  - z1("s A=(0 (1,4)) B~(2 (3,8)) A B A B A")
  - z1("s A=(0 3) B=(3 8) C=(((A+B)+A)*B)
- .noteLength(n) // sets note length e.g. .noteLength(1/16,1/8)
- .at(index: number, ...args?: number[]) Get event(s) at given index
- .repeat(amount: number) Repeat the generated pattern without re-evaluating random patterns
- .keep() Keep the generated pattern without re-evaluating random patterns. Same as repeat(0).
- .shuffle() Shuffle the pattern
- .deal(amount: number): Shuffle the generated pattern and deal given number of elements
- .retrograde() Reverse the generated pattern
- .invert() Invert the generated pattern
- .rotate(amount: number) Rotate the generated pattern by given amount
- .between(start: number, end: number) Select a range of elements from the generated pattern
- .from(start: number) Select a range of elements from the start index to the end of the pattern
- .to(end: number) Select a range of elements from the beginning of the pattern to the end index
- .every(amount: number) Select every nth element from the pattern
- ziffer patterns can be synced to any event by using cue, sync, wait and listen methods
- .cue(string) // used on the main pattern. All functions below need .cue() on a pattern first
- .wait(string) // used on the receiving pattern. The receiving pattern will then start on the next cycle
- .sync(string) // used on the receiving pattern e.g.
  - onbar(1) :: cue("x")
  - onbar(4) :: cue("x")
  - z1("<0.25 0.125> 0 4 2 -2").sync("x").sound("sine").out()
  - z2("<0.25 0.125> 0 6 4 -4").sync("y").sound("sine").out()
- .listen(string) // receiving patterns listens to the cue message and plays one event from the pattern for every cue e.g.
  - beat(1.0) :: cue("boom")
  - z1("bd <hh hh:9>").listen("boom").sound().out()
- ziffer patterns can be synced to a beat using .notelength(0) or z in the ziffer pattern string e.g.
  - beat(.5) :: z1("z <bd sn> hh").sound().out()
  - beat(.5) :: z1("<bd sn> hh").notelength(0).sound().out()
- Numbered methods (z0-z16) are synced automatically to z0

Hydra

- beat(4) :: hydra.osc(3, 0.5, 2).out()
- beat(4) :: stop_hydra()
- hydra.setResolution(1024, 768)
- See Hydra documentation here

GIFs

- beat(0.25)::gif({
  url:v('gif')[$(1)%6], // Any URL will do!
  opacity: r(0.5, 1), // Opacity (0-1)
  size:"300px", // CSS size property
  center:false, // Centering on the screen?
  filter:'none', // CSS Filter
  dur: 2, // In beats (Topos unit)
  rotation: ir(1, 360), // Rotation (in degrees)
  posX: ir(1,1200), // CSS Horizontal Position
  posY: ir(1, 800), // CSS Vertical Position})

Text to Canvas

- Text can be drawn to canvas using the drawText() function. The function can take any unicode characters including emojis. The function can also be used to draw random characters from a given unicode range. Different filters can also be applied using the filter parameter. See filter here for more info
- drawText(text, fontSize, rotation, font, x, y) // Draws text to a canvas
- randomChar(number, min, max) // Returns a number of random characters from given unicode range
- emoji(size) // Returns a random emojis as text
- animals(size) // Returns a random animal emojis as text
- food(size) // Returns a random food emojis as text
- expression(size) // Returns a random expression emojis as text
- e.g. beat(0.5) && clear() && drawText(randomChar(10,1000,3000),30)

Appendix

Wavetables:

- _base (1)
- wt_stereo (undefined)
- wt_birds (undefined)
- wt_dbass (undefined)
- wt_distorted (undefined)
- wt_violin (undefined)
- wt_ebass (undefined)
- wt_oscchip (undefined)
- wt_flute (undefined)
- wt_bw_squ (undefined)
- wt_14 (undefined)
- wt_13 (undefined)
- wt_bw_sin (undefined)
- wt_vgame (undefined)
- wt_pluckalgo (undefined)
- wt_12 (undefined)
- wt_vgamebasic (undefined)
- wt_15 (undefined)
- wt_bw_sawgap (undefined)
- wt_symetric (undefined)
- wt_08 (undefined)
- wt_clarinett (undefined)
- wt_06 (undefined)
- wt_01 (undefined)
- wt_c604 (undefined)
- wt_epiano (undefined)
- wt_hdrawn (undefined)
- wt_bw_tri (undefined)
- wt_bw_perfectwaves (undefined)
- wt_07 (undefined)
- wt_bw_saw (undefined)
- wt_09 (undefined)
- wt_bw_sawbright (undefined)
- wt_sinharm (undefined)
- wt_linear (undefined)
- wt_bw_sawrounded (undefined)
- wt_eorgan (undefined)
- wt_theremin (undefined)
- wt_overtone (undefined)
- wt_raw (undefined)
- wt_altosax (undefined)
- wt_fmsynth (undefined)
- wt_10 (undefined)
- wt_17 (undefined)

- wt_bitreduced (undefined)
- wt_19 (undefined)
- wt_bw_squrounded (undefined)
- wt_18 (undefined)
- wt_20 (undefined)
- wt_16 (undefined)
- wt_11 (undefined)
- wt_oboe (undefined)
- wt_02 (undefined)
- wt_05 (undefined)
- wt_aguitar (undefined)
- wt_stringbox (undefined)
- wt_04 (undefined)
- wt_cello (undefined)
- wt_03 (undefined)
- wt_piano (undefined)
- wt_granular (undefined)
- wt_bw_blended (undefined)
- wt_eguitar (undefined)
- wt_hvoice (undefined)
- wt_clavinet (undefined)
- wt_snippets (undefined)

Drum Machines:

- AJKPercusyn_bd (1)
- AJKPercusyn_cb (2)
- AJKPercusyn_ht (1)
- AJKPercusyn_sd (1)
- AkaiLinn_bd (1)
- AkaiLinn_cb (1)
- AkaiLinn_cp (1)
- AkaiLinn_cr (1)
- AkaiLinn_hh (1)
- AkaiLinn_ht (1)
- AkaiLinn_lt (1)
- AkaiLinn_mt (1)
- AkaiLinn_oh (1)
- AkaiLinn_rd (1)
- AkaiLinn_sd (1)
- AkaiLinn_sh (1)
- AkaiLinn_tb (1)
- AkaiMPC60_bd (2)
- AkaiMPC60_cp (1)
- AkaiMPC60_cr (1)
- AkaiMPC60_hh (1)
- AkaiMPC60_ht (1)

- AkaiMPC60_lt (1)
- AkaiMPC60_misc (2)
- AkaiMPC60_mt (1)
- AkaiMPC60_oh (1)
- AkaiMPC60_perc (5)
- AkaiMPC60_rd (1)
- AkaiMPC60_rim (1)
- AkaiMPC60_sd (3)
- AkaiXR10_bd (10)
- AkaiXR10_cb (1)
- AkaiXR10_cp (1)
- AkaiXR10_cr (3)
- AkaiXR10_hh (2)
- AkaiXR10_ht (1)
- AkaiXR10_lt (2)
- AkaiXR10_misc (4)
- AkaiXR10_mt (2)
- AkaiXR10_oh (1)
- AkaiXR10_perc (15)
- AkaiXR10_rd (1)
- AkaiXR10_rim (2)
- AkaiXR10_sd (10)
- AkaiXR10_sh (1)
- AkaiXR10_tb (1)
- AlesisHR16_bd (1)
- AlesisHR16_cp (1)
- AlesisHR16_hh (1)
- AlesisHR16_ht (1)
- AlesisHR16_lt (1)
- AlesisHR16_oh (1)
- AlesisHR16_perc (8)
- AlesisHR16_rim (1)
- AlesisHR16_sd (1)
- AlesisHR16_sh (3)
- AlesisSR16_bd (13)
- AlesisSR16_cb (1)
- AlesisSR16_cp (1)
- AlesisSR16_cr (2)
- AlesisSR16_hh (3)
- AlesisSR16_misc (3)
- AlesisSR16_oh (4)
- AlesisSR16_perc (7)
- AlesisSR16_rd (3)
- AlesisSR16_rim (1)
- AlesisSR16_sd (12)
- AlesisSR16_sh (1)
- AlesisSR16_tb (1)
- BossDR110_bd (1)

- BossDR110_cp (1)
- BossDR110_cr (1)
- BossDR110_hh (1)
- BossDR110_oh (1)
- BossDR110_rd (1)
- BossDR110_sd (1)
- BossDR220_bd (1)
- BossDR220_cp (1)
- BossDR220_cr (1)
- BossDR220_hh (1)
- BossDR220_ht (1)
- BossDR220_lt (1)
- BossDR220_mt (1)
- BossDR220_oh (1)
- BossDR220_perc (1)
- BossDR220_rd (1)
- BossDR220_sd (1)
- BossDR55_bd (2)
- BossDR55_hh (2)
- BossDR55_rim (1)
- BossDR55_sd (8)
- BossDR550_bd (5)
- BossDR550_cb (2)
- BossDR550_cp (1)
- BossDR550_cr (1)
- BossDR550_hh (2)
- BossDR550_ht (3)
- BossDR550_lt (3)
- BossDR550_misc (3)
- BossDR550_mt (2)
- BossDR550_oh (2)
- BossDR550_perc (11)
- BossDR550_rd (2)
- BossDR550_rim (1)
- BossDR550_sd (6)
- BossDR550_sh (2)
- BossDR550_tb (1)
- CasioRZ1_bd (1)
- CasioRZ1_cb (1)
- CasioRZ1_cp (1)
- CasioRZ1_cr (1)
- CasioRZ1_hh (1)
- CasioRZ1_ht (1)
- CasioRZ1_lt (1)
- CasioRZ1_mt (1)
- CasioRZ1_rd (2)
- CasioRZ1_rim (1)
- CasioRZ1_sd (1)

- CasioSK1_bd (1)
- CasioSK1_hh (1)
- CasioSK1_ht (1)
- CasioSK1_mt (1)
- CasioSK1_oh (1)
- CasioSK1_sd (1)
- CasioVL1_bd (1)
- CasioVL1_hh (1)
- CasioVL1_sd (1)
- DoepferMS404_bd (2)
- DoepferMS404_hh (1)
- DoepferMS404_lt (1)
- DoepferMS404_oh (1)
- DoepferMS404_sd (1)
- EmuDrumulator_bd (1)
- EmuDrumulator_cb (1)
- EmuDrumulator_cp (1)
- EmuDrumulator_cr (1)
- EmuDrumulator_hh (1)
- EmuDrumulator_ht (1)
- EmuDrumulator_lt (1)
- EmuDrumulator_mt (1)
- EmuDrumulator_oh (1)
- EmuDrumulator_perc (1)
- EmuDrumulator_rim (1)
- EmuDrumulator_sd (1)
- EmuModular_bd (2)
- EmuModular_misc (1)
- EmuModular_perc (2)
- EmuSP12_bd (14)
- EmuSP12_cb (1)
- EmuSP12_cp (1)
- EmuSP12_cr (1)
- EmuSP12_hh (2)
- EmuSP12_ht (6)
- EmuSP12_lt (6)
- EmuSP12_misc (7)
- EmuSP12_mt (4)
- EmuSP12_oh (1)
- EmuSP12_perc (1)
- EmuSP12_rd (1)
- EmuSP12_rim (2)
- EmuSP12_sd (21)
- KorgDDM110_bd (1)
- KorgDDM110_cp (1)
- KorgDDM110_cr (1)
- KorgDDM110_hh (1)
- KorgDDM110_ht (2)

- KorgDDM110_lt (2)
- KorgDDM110_oh (1)
- KorgDDM110_rim (1)
- KorgDDM110_sd (1)
- KorgKPR77_bd (1)
- KorgKPR77_cp (1)
- KorgKPR77_hh (1)
- KorgKPR77_oh (1)
- KorgKPR77_sd (1)
- KorgKR55_bd (1)
- KorgKR55_cb (1)
- KorgKR55_cr (1)
- KorgKR55_hh (1)
- KorgKR55_ht (1)
- KorgKR55_oh (1)
- KorgKR55_perc (2)
- KorgKR55_rim (1)
- KorgKR55_sd (1)
- KorgKRZ_bd (1)
- KorgKRZ_cr (1)
- KorgKRZ_fx (2)
- KorgKRZ_hh (1)
- KorgKRZ_ht (1)
- KorgKRZ_lt (1)
- KorgKRZ_misc (1)
- KorgKRZ_oh (1)
- KorgKRZ_rd (1)
- KorgKRZ_sd (2)
- KorgM1_bd (3)
- KorgM1_cb (1)
- KorgM1_cp (1)
- KorgM1_cr (1)
- KorgM1_hh (2)
- KorgM1_ht (2)
- KorgM1_misc (16)
- KorgM1_mt (1)
- KorgM1_oh (2)
- KorgM1_perc (7)
- KorgM1_rd (1)
- KorgM1_rim (1)
- KorgM1_sd (4)
- KorgM1_sh (1)
- KorgM1_tb (1)
- KorgMinipops_bd (7)
- KorgMinipops_hh (4)
- KorgMinipops_misc (4)
- KorgMinipops_oh (4)
- KorgMinipops_sd (13)

- KorgPoly800_bd (4)
- KorgT3_bd (5)
- KorgT3_cp (1)
- KorgT3_hh (2)
- KorgT3_misc (4)
- KorgT3_oh (2)
- KorgT3_perc (4)
- KorgT3_rim (1)
- KorgT3_sd (5)
- KorgT3_sh (3)
- Linn9000_bd (1)
- Linn9000_cb (2)
- Linn9000_cr (2)
- Linn9000_hh (1)
- Linn9000_ht (2)
- Linn9000_lt (2)
- Linn9000_mt (1)
- Linn9000_oh (1)
- Linn9000_perc (3)
- Linn9000_rd (2)
- Linn9000_rim (1)
- Linn9000_sd (1)
- Linn9000_tb (1)
- LinnDrum_bd (1)
- LinnDrum_cb (1)
- LinnDrum_cp (1)
- LinnDrum_cr (1)
- LinnDrum_hh (3)
- LinnDrum_ht (2)
- LinnDrum_lt (2)
- LinnDrum_mt (1)
- LinnDrum_oh (1)
- LinnDrum_perc (6)
- LinnDrum_rd (1)
- LinnDrum_rim (3)
- LinnDrum_sd (3)
- LinnDrum_sh (1)
- LinnDrum_tb (1)
- LinnLM1_bd (4)
- LinnLM1_cb (1)
- LinnLM1_cp (1)
- LinnLM1_hh (1)
- LinnLM1_ht (1)
- LinnLM1_lt (1)
- LinnLM1_oh (1)
- LinnLM1_perc (3)
- LinnLM1_rim (1)
- LinnLM1_sd (1)

- LinnLM1_sh (1)
- LinnLM1_tb (1)
- LinnLM2_bd (4)
- LinnLM2_cb (1)
- LinnLM2_cp (1)
- LinnLM2_cr (1)
- LinnLM2_hh (2)
- LinnLM2_ht (1)
- LinnLM2_lt (1)
- LinnLM2_mt (1)
- LinnLM2_oh (2)
- LinnLM2_rd (1)
- LinnLM2_rim (2)
- LinnLM2_sd (4)
- LinnLM2_sh (1)
- LinnLM2_tb (1)
- MFB512_bd (1)
- MFB512_cp (1)
- MFB512_cr (1)
- MFB512_hh (1)
- MFB512_ht (1)
- MFB512_lt (1)
- MFB512_mt (1)
- MFB512_oh (1)
- MFB512_sd (1)
- MPC1000_bd (5)
- MPC1000_cp (1)
- MPC1000_hh (4)
- MPC1000_oh (1)
- MPC1000_perc (1)
- MPC1000_sd (4)
- MPC1000_sh (1)
- MoogConcertMateMG1_bd (3)
- MoogConcertMateMG1_sd (2)
- OberheimDMX_ (3)
- OberheimDMX_bd (3)
- OberheimDMX_cp (1)
- OberheimDMX_cr (1)
- OberheimDMX_hh (1)
- OberheimDMX_ht (1)
- OberheimDMX_lt (1)
- OberheimDMX_mt (1)
- OberheimDMX_oh (1)
- OberheimDMX_rd (1)
- OberheimDMX_rim (1)
- OberheimDMX_sd (3)
- OberheimDMX_sh (1)
- OberheimDMX_tb (1)

- RhodesPolaris_bd (4)
- RhodesPolaris_misc (4)
- RhodesPolaris_sd (4)
- RhythmAce_bd (3)
- RhythmAce_hh (1)
- RhythmAce_ht (1)
- RhythmAce_lt (1)
- RhythmAce_oh (1)
- RhythmAce_perc (6)
- RhythmAce_sd (3)
- RolandCompurhythm1000_bd (1)
- RolandCompurhythm1000_cb (1)
- RolandCompurhythm1000_cp (1)
- RolandCompurhythm1000_cr (1)
- RolandCompurhythm1000_hh (1)
- RolandCompurhythm1000_ht (1)
- RolandCompurhythm1000_lt (1)
- RolandCompurhythm1000_mt (1)
- RolandCompurhythm1000_oh (1)
- RolandCompurhythm1000_perc (3)
- RolandCompurhythm1000_rd (1)
- RolandCompurhythm1000_rim (1)
- RolandCompurhythm1000_sd (1)
- RolandCompurhythm78_bd (1)
- RolandCompurhythm78_cb (1)
- RolandCompurhythm78_hh (2)
- RolandCompurhythm78_misc (4)
- RolandCompurhythm78_oh (2)
- RolandCompurhythm78_perc (8)
- RolandCompurhythm78_sd (1)
- RolandCompurhythm78_tb (1)
- RolandCompurhythm8000_bd (1)
- RolandCompurhythm8000_cb (1)
- RolandCompurhythm8000_cp (1)
- RolandCompurhythm8000_cr (1)
- RolandCompurhythm8000_hh (1)
- RolandCompurhythm8000_ht (1)
- RolandCompurhythm8000_lt (1)
- RolandCompurhythm8000_mt (1)
- RolandCompurhythm8000_oh (1)
- RolandCompurhythm8000_perc (2)
- RolandCompurhythm8000_rim (1)
- RolandCompurhythm8000_sd (1)
- RolandD110_bd (1)
- RolandD110_cb (2)
- RolandD110_cr (1)
- RolandD110_hh (1)
- RolandD110_lt (1)

- RolandD110_oh (2)
- RolandD110_perc (3)
- RolandD110_rd (1)
- RolandD110_rim (1)
- RolandD110_sd (3)
- RolandD110_sh (1)
- RolandD110_tb (1)
- RolandD70_bd (4)
- RolandD70_cb (1)
- RolandD70_cp (1)
- RolandD70_cr (1)
- RolandD70_hh (1)
- RolandD70_lt (1)
- RolandD70_mt (1)
- RolandD70_oh (1)
- RolandD70_perc (1)
- RolandD70_rd (1)
- RolandD70_rim (1)
- RolandD70_sd (5)
- RolandD70_sh (1)
- RolandDDR30_bd (8)
- RolandDDR30_ht (4)
- RolandDDR30_lt (4)
- RolandDDR30_sd (8)
- RolandJD990_bd (10)
- RolandJD990_cb (1)
- RolandJD990_cp (1)
- RolandJD990_cr (1)
- RolandJD990_hh (4)
- RolandJD990_ht (1)
- RolandJD990_lt (5)
- RolandJD990_misc (12)
- RolandJD990_mt (2)
- RolandJD990_oh (2)
- RolandJD990_perc (6)
- RolandJD990_rd (1)
- RolandJD990_sd (15)
- RolandJD990_tb (1)
- RolandMC202_bd (5)
- RolandMC202_ht (3)
- RolandMC202_perc (1)
- RolandMC303_bd (16)
- RolandMC303_cb (2)
- RolandMC303_cp (8)
- RolandMC303_fx (2)
- RolandMC303_hh (6)
- RolandMC303_ht (5)
- RolandMC303_lt (5)

- RolandMC303_misc (8)
- RolandMC303_mt (6)
- RolandMC303_oh (5)
- RolandMC303_perc (39)
- RolandMC303_rd (2)
- RolandMC303_rim (6)
- RolandMC303_sd (26)
- RolandMC303_sh (7)
- RolandMC303_tb (5)
- RolandMT32_bd (1)
- RolandMT32_cb (1)
- RolandMT32_cp (1)
- RolandMT32_cr (1)
- RolandMT32_hh (1)
- RolandMT32_ht (1)
- RolandMT32_lt (1)
- RolandMT32_mt (1)
- RolandMT32_oh (2)
- RolandMT32_perc (13)
- RolandMT32_rd (1)
- RolandMT32_rim (1)
- RolandMT32_sd (2)
- RolandMT32_sh (2)
- RolandMT32_tb (1)
- RolandR8_bd (7)
- RolandR8_cb (1)
- RolandR8_cp (1)
- RolandR8_cr (1)
- RolandR8_hh (2)
- RolandR8_ht (4)
- RolandR8_lt (4)
- RolandR8_mt (4)
- RolandR8_oh (1)
- RolandR8_perc (8)
- RolandR8_rd (2)
- RolandR8_rim (2)
- RolandR8_sd (12)
- RolandR8_sh (2)
- RolandR8_tb (1)
- RolandS50_bd (4)
- RolandS50_cb (1)
- RolandS50_cp (1)
- RolandS50_cr (2)
- RolandS50_ht (1)
- RolandS50_lt (2)
- RolandS50_misc (6)
- RolandS50_mt (1)
- RolandS50_oh (1)

- RolandS50_perc (14)
- RolandS50_rd (1)
- RolandS50_sd (3)
- RolandS50_sh (4)
- RolandS50_tb (2)
- RolandSH09_bd (43)
- RolandSystem100_bd (15)
- RolandSystem100_hh (2)
- RolandSystem100_misc (2)
- RolandSystem100_oh (3)
- RolandSystem100_perc (19)
- RolandSystem100_sd (21)
- RolandTR505_bd (1)
- RolandTR505_cb (2)
- RolandTR505_cp (1)
- RolandTR505_cr (1)
- RolandTR505_hh (1)
- RolandTR505_ht (1)
- RolandTR505_lt (1)
- RolandTR505_mt (1)
- RolandTR505_oh (1)
- RolandTR505_perc (3)
- RolandTR505_rd (1)
- RolandTR505_rim (1)
- RolandTR505_sd (1)
- RolandTR606_bd (1)
- RolandTR606_cr (1)
- RolandTR606_hh (1)
- RolandTR606_ht (1)
- RolandTR606_lt (1)
- RolandTR606_oh (1)
- RolandTR606_sd (1)
- RolandTR626_bd (2)
- RolandTR626_cb (1)
- RolandTR626_cp (1)
- RolandTR626_cr (2)
- RolandTR626_hh (1)
- RolandTR626_ht (2)
- RolandTR626_lt (2)
- RolandTR626_mt (2)
- RolandTR626_oh (1)
- RolandTR626_perc (8)
- RolandTR626_rd (2)
- RolandTR626_rim (1)
- RolandTR626_sd (3)
- RolandTR626_sh (1)
- RolandTR626_tb (1)
- RolandTR707_bd (2)

- RolandTR707_cb (1)
- RolandTR707_cp (1)
- RolandTR707_cr (1)
- RolandTR707_hh (1)
- RolandTR707_ht (1)
- RolandTR707_lt (1)
- RolandTR707_mt (1)
- RolandTR707_oh (1)
- RolandTR707_rim (1)
- RolandTR707_sd (2)
- RolandTR707_tb (1)
- RolandTR727_perc (10)
- RolandTR727_sh (2)
- RolandTR808_bd (25)
- RolandTR808_cb (2)
- RolandTR808_cp (5)
- RolandTR808_cr (25)
- RolandTR808_hh (1)
- RolandTR808_ht (5)
- RolandTR808_lt (5)
- RolandTR808_mt (5)
- RolandTR808_oh (5)
- RolandTR808_perc (16)
- RolandTR808_rim (1)
- RolandTR808_sd (25)
- RolandTR808_sh (2)
- RolandTR909_bd (4)
- RolandTR909_cp (5)
- RolandTR909_cr (5)
- RolandTR909_hh (4)
- RolandTR909_ht (9)
- RolandTR909_lt (9)
- RolandTR909_mt (9)
- RolandTR909_oh (5)
- RolandTR909_rd (5)
- RolandTR909_rim (3)
- RolandTR909_sd (16)
- SakataDPM48_bd (3)
- SakataDPM48_cp (1)
- SakataDPM48_cr (1)
- SakataDPM48_hh (2)
- SakataDPM48_ht (1)
- SakataDPM48_lt (2)
- SakataDPM48_mt (1)
- SakataDPM48_oh (1)
- SakataDPM48_perc (2)
- SakataDPM48_rd (1)
- SakataDPM48_rim (1)

- SakataDPM48_sd (2)
- SakataDPM48_sh (2)
- SequentialCircuitsDrumtracks_bd (1)
- SequentialCircuitsDrumtracks_cb (1)
- SequentialCircuitsDrumtracks_cp (1)
- SequentialCircuitsDrumtracks_cr (1)
- SequentialCircuitsDrumtracks_hh (1)
- SequentialCircuitsDrumtracks_ht (1)
- SequentialCircuitsDrumtracks_oh (1)
- SequentialCircuitsDrumtracks_rd (1)
- SequentialCircuitsDrumtracks_rim (1)
- SequentialCircuitsDrumtracks_sd (1)
- SequentialCircuitsDrumtracks_sh (1)
- SequentialCircuitsDrumtracks_tb (1)
- SequentialCircuitsTom_bd (1)
- SequentialCircuitsTom_cp (1)
- SequentialCircuitsTom_cr (1)
- SequentialCircuitsTom_hh (1)
- SequentialCircuitsTom_ht (2)
- SequentialCircuitsTom_oh (1)
- SequentialCircuitsTom_sd (1)
- SergeModular_bd (1)
- SergeModular_misc (1)
- SergeModular_perc (5)
- SimmonsSDS400_ht (3)
- SimmonsSDS400_lt (6)
- SimmonsSDS400_mt (8)
- SimmonsSDS400_sd (3)
- SimmonsSDS5_bd (12)
- SimmonsSDS5_hh (5)
- SimmonsSDS5_ht (3)
- SimmonsSDS5_lt (8)
- SimmonsSDS5_mt (6)
- SimmonsSDS5_oh (2)
- SimmonsSDS5_rim (7)
- SimmonsSDS5_sd (21)
- SoundmastersR88_bd (1)
- SoundmastersR88_cr (1)
- SoundmastersR88_hh (1)
- SoundmastersR88_oh (1)
- SoundmastersR88_sd (2)
- UnivoxMicroRhythmer12_bd (1)
- UnivoxMicroRhythmer12_hh (1)
- UnivoxMicroRhythmer12_oh (1)
- UnivoxMicroRhythmer12_sd (1)
- ViscoSpaceDrum_bd (11)
- ViscoSpaceDrum_cb (1)
- ViscoSpaceDrum_hh (6)

- ViscoSpaceDrum_ht (7)
- ViscoSpaceDrum_lt (2)
- ViscoSpaceDrum_misc (2)
- ViscoSpaceDrum_mt (2)
- ViscoSpaceDrum_oh (3)
- ViscoSpaceDrum_perc (2)
- ViscoSpaceDrum_rim (1)
- ViscoSpaceDrum_sd (3)
- XdrumLM8953_bd (3)
- XdrumLM8953_cr (1)
- XdrumLM8953_hh (2)
- XdrumLM8953_ht (2)
- XdrumLM8953_lt (2)
- XdrumLM8953_mt (2)
- XdrumLM8953_oh (1)
- XdrumLM8953_rd (1)
- XdrumLM8953_rim (2)
- XdrumLM8953_sd (5)
- XdrumLM8953_tb (1)
- YamahaRM50_bd (103)
- YamahaRM50_cb (6)
- YamahaRM50_cp (2)
- YamahaRM50_cr (22)
- YamahaRM50_hh (18)
- YamahaRM50_ht (25)
- YamahaRM50_lt (49)
- YamahaRM50_misc (28)
- YamahaRM50_mt (34)
- YamahaRM50_oh (12)
- YamahaRM50_perc (56)
- YamahaRM50_rd (13)
- YamahaRM50_sd (108)
- YamahaRM50_sh (6)
- YamahaRM50_tb (3)
- YamahaRX21_bd (1)
- YamahaRX21_cp (1)
- YamahaRX21_cr (1)
- YamahaRX21_hh (1)
- YamahaRX21_ht (1)
- YamahaRX21_lt (1)
- YamahaRX21_mt (1)
- YamahaRX21_oh (1)
- YamahaRX21_sd (1)
- YamahaRX5_bd (2)
- YamahaRX5_cb (1)
- YamahaRX5_fx (1)
- YamahaRX5_hh (1)
- YamahaRX5_lt (1)

- YamahaRX5_oh (1)
- YamahaRX5_rim (1)
- YamahaRX5_sd (3)
- YamahaRX5_sh (1)
- YamahaRX5_tb (1)
- YamahaRY30_bd (13)
- YamahaRY30_cb (2)
- YamahaRY30_cp (1)
- YamahaRY30_cr (2)
- YamahaRY30_hh (4)
- YamahaRY30_ht (3)
- YamahaRY30_lt (3)
- YamahaRY30_misc (8)
- YamahaRY30_mt (2)
- YamahaRY30_oh (4)
- YamahaRY30_perc (13)
- YamahaRY30_rd (3)
- YamahaRY30_rim (2)
- YamahaRY30_sd (21)
- YamahaRY30_sh (2)
- YamahaRY30_tb (1)
- YamahaTG33_bd (4)
- YamahaTG33_cb (3)
- YamahaTG33_cp (1)
- YamahaTG33_cr (3)
- YamahaTG33_fx (1)
- YamahaTG33_ht (2)
- YamahaTG33_lt (2)
- YamahaTG33_misc (10)
- YamahaTG33_mt (2)
- YamahaTG33_oh (1)
- YamahaTG33_perc (12)
- YamahaTG33_rd (2)
- YamahaTG33_rim (1)
- YamahaTG33_sd (5)
- YamahaTG33_sh (1)
- YamahaTG33_tb (1)

Foxdot Samples

- fclap (6)
- fdarkkick (5)
- fbeep (5)
- ftriangle (3)
- fthree (2)
- fyeah (6)
- fsnap (6)

- fnoisekick (3)
- fcow (3)
- fdistkick (3)
- fscrap (7)
- ftoms (5)
- fclaps (5)
- fnoise (6)
- fhang (6)
- fhardhit (7)
- fqlower (2)
- fone (2)
- flazer (3)
- fohh (4)
- ffour (2)
- fplus (3)
- fsub (2)
- frevnoise (3)
- fjungle (6)
- ftwo (2)
- fsoftsnare (5)
- fssnare (5)
- frocksnare (5)
- fhits (5)
- ftabla (8)
- fslower (4)
- fscratch (4)
- fwood (4)
- ftablas (6)
- foh (6)
- fsnare (4)
- fmisc (2)
- fdub (4)
- fcowbell (3)
- fxupper (4)
- fkit (8)
- fconga (0)
- fglitch (6)
- fstab (9)
- ftamborine (3)
- fwoosh (5)
- frim (6)
- fcan (4)
- ftom (4)
- fhh (22)
- fnoises (5)
- froomy (4)
- fcrash (4)
- fpercus (9)

- fhardkick (5)
- fikea (18)
- fperc (5)
- fbins (4)
- fbleep (5)
- fride (4)
- fhit (4)
- fdot (4)

Amiga Samples

- STA3 (114)
- STA4 (108)
- STB2 (83)
- STB5 (36)
- STB4 (98)
- STB3 (89)
- STA5 (106)
- STA2 (110)
- ST92 (62)
- ST66 (93)
- ST59 (122)
- ST61 (128)
- ST95 (58)
- ST57 (81)
- ST68 (47)
- ST50 (97)
- ST04 (102)
- ST03 (131)
- ST35 (66)
- ST32 (105)
- ST51 (86)
- ST56 (84)
- ST69 (64)
- ST94 (121)
- ST60 (95)
- ST67 (71)
- ST93 (105)
- ST58 (104)
- ST33 (57)
- ST34 (78)
- ST02 (121)
- ST05 (100)
- ST27 (98)
- ST18 (99)
- ST20 (58)
- ST16 (85)

- ST29 (85)
- ST11 (114)
- ST45 (80)
- ST89 (119)
- ST42 (116)
- ST80 (70)
- ST74 (118)
- ST73 (90)
- ST87 (93)
- ST10 (47)
- ST17 (87)
- ST28 (33)
- ST21 (81)
- ST26 (81)
- ST19 (34)
- ST86 (87)
- ST72 (63)
- ST75 (141)
- ST81 (111)
- ST88 (95)
- ST43 (92)
- ST44 (71)
- STA7 (44)
- STA0 (124)
- STA9 (74)
- STB6 (26)
- STB1 (76)
- STB0 (113)
- STA8 (97)
- STA1 (123)
- STA6 (118)
- ST38 (66)
- ST07 (111)
- ST31 (78)
- ST09 (98)
- ST36 (72)
- ST62 (98)
- ST96 (113)
- ST91 (107)
- ST65 (108)
- ST53 (95)
- ST98 (100)
- ST54 (73)
- ST08 (105)
- ST37 (80)
- ST30 (123)
- ST39 (89)
- ST06 (70)

- ST01 (126)
- ST55 (104)
- ST52 (115)
- ST99 (93)
- ST64 (128)
- ST90 (101)
- ST97 (102)
- ST63 (73)
- ST41 (81)
- ST79 (112)
- ST46 (65)
- ST70 (103)
- ST84 (111)
- ST48 (49)
- ST83 (124)
- ST77 (61)
- ST23 (84)
- ST24 (128)
- ST12 (103)
- ST15 (85)
- ST49 (95)
- ST76 (91)
- ST82 (104)
- ST85 (113)
- ST71 (70)
- ST78 (84)
- ST47 (82)
- ST40 (117)
- ST14 (120)
- ST13 (106)
- ST25 (62)
- ST22 (103)

Amen Breaks

- amen1 (20)
- amen2 (40)
- amen3 (20)

Tidal Cycles Samples

- 808 (6)
- 909 (1)
- 808bd (25)
- 808cy (25)
- 808hc (5)

- 808ht (5)
- 808lc (5)
- 808lt (5)
- 808mc (5)
- 808mt (5)
- 808oh (5)
- 808sd (25)
- ab (12)
- ade (10)
- ades2 (9)
- ades3 (7)
- ades4 (6)
- alex (2)
- alphabet (26)
- amencutup (32)
- armora (7)
- arp (2)
- arpy (11)
- auto (11)
- baa (7)
- baa2 (7)
- bass0 (3)
- bass1 (30)
- bassdm (24)
- bassfoo (3)
- battles (2)
- bd (24)
- bend (4)
- bev (2)
- bin (2)
- birds (10)
- birds3 (19)
- bleep (13)
- blip (2)
- blue (2)
- bottle (13)
- breaks125 (2)
- breaks152 (1)
- breaks157 (1)
- breaks165 (1)
- breath (1)
- bubble (8)
- can (14)
- casio (3)
- cb (1)
- cc (6)
- chin (4)
- circus (3)

- clak (2)
- click (4)
- clubkick (5)
- co (4)
- coins (1)
- control (2)
- cosmicg (15)
- cp (2)
- cr (6)
- crow (4)
- d (4)
- db (13)
- diphone (38)
- diphone2 (12)
- dist (16)
- dork2 (4)
- dorkbot (2)
- dr (42)
- dr2 (6)
- dr55 (4)
- dr_few (8)
- drum (6)
- drumtraks (13)
- e (8)
- east (9)
- electro1 (13)
- em2 (6)
- erk (1)
- f (1)
- feel (7)
- feelfx (8)
- fest (1)
- fire (1)
- flick (17)
- fm (17)
- foo (27)
- future (17)
- gab (10)
- gabba (4)
- gabbaloud (4)
- gabbalouder (4)
- glasstap (3)
- glitch (8)
- glitch2 (8)
- gretsch (24)
- gtr (3)
- h (7)
- hand (17)

- hardcore (12)
- hardkick (6)
- haw (6)
- hc (6)
- hh (13)
- hh27 (13)
- hit (6)
- hmm (1)
- ho (6)
- hoover (6)
- house (8)
- ht (16)
- if (5)
- ifdrums (3)
- incoming (8)
- industrial (32)
- insect (3)
- invaders (18)
- jazz (8)
- jungbass (20)
- jungle (13)
- juno (12)
- jvbass (13)
- kicklinn (1)
- koy (2)
- kurt (7)
- latibro (8)
- led (1)
- less (4)
- lighter (33)
- linnhats (6)
- lt (16)
- made (7)
- made2 (1)
- mash (2)
- mash2 (4)
- metal (10)
- miniyeah (4)
- monsterb (6)
- moog (7)
- mouth (15)
- mp3 (4)
- msg (9)
- mt (16)
- mute (28)
- newnotes (15)
- noise (1)
- noise2 (8)

- notes (15)
- numbers (9)
- oc (4)
- odx (15)
- off (1)
- outdoor (6)
- padlong (1)
- pebbles (1)
- peri (15)
- pluck (17)
- popkick (10)
- print (11)
- proc (2)
- procshort (8)
- psr (30)
- rave (8)
- rave2 (4)
- ravemono (2)
- realclaps (4)
- reverbkick (1)
- rm (2)
- rs (1)
- sax (22)
- sd (2)
- seawolf (3)
- sequential (8)
- sf (18)
- sheffield (1)
- short (5)
- sid (12)
- sitar (8)
- sn (52)
- space (18)
- speakspell (12)
- speech (7)
- speechless (10)
- speedupdown (9)
- stomp (10)
- subroc3d (11)
- sugar (2)
- sundance (6)
- tabla (26)
- tabla2 (46)
- tablex (3)
- tacscan (22)
- tech (13)
- techno (7)
- tink (5)

- tok (4)
- toys (13)
- trump (11)
- ul (10)
- ulgab (5)
- uxay (3)
- v (6)
- voodoo (5)
- wind (10)
- wobble (1)
- world (3)
- xmas (1)
- yeah (31)

Juliette's Voice Samples

- juj (138)