CDP Composers' Desktop Project

CDP SYNTH Functions

(with Command Line Usage)

Functions to SYNTHESISE (Generate) soundfiles

SYNTH **CHORD**
Generate a chord with a simple waveform

CLICKNEW
Make clicktrack using times listed in textfile

SYNTH **CLICKS**
Create a click track from tempo, meter and barring data

IMPULSE
Create a stream of impulses

MULTIOSC
Nested FM-style oscillations

MULTISYNTH
Synthesize several sound-streams from a score

NEWSYNTH
Synthesisze complex spectra

NEWSCALES
Synthesise a series of short tones with defined frequency and timbre

SYNTH **NOISE**
Generate noise

PULSER SYNTH
Iterate synthesized wave-packets defined by partials data

SYNTH **SILENCE**
Make silent soundfile

SYNTH **SPECTRA**
Generate both channels of a stereo spectral band

SYNFILT
Noise filtered by time-varying filterbank, with time-variable Q

SYNSPLINE
Synthesise waveforms by smoothly joining randomly generated points

TS OSCIL
Create sound from time-series text data

TS TRACE
Create sound from time-series data treated as a pitch-trace

TSCONVERT
Convert input data to specified range and format

SYNTH **WAVE**
Generate simple waveforms

For more comprehensive synthesis facilities, users are recommended to explore the public domain program ***Csound***.

SYNTH CHORD – Generate a chord with a simple waveform

Usage

synth chord mode *outfile datafile sr chans dur* [**-a***amp*] [**-t***tabsize*]

Modes

1 *Datafile* contains a list of MIDI pitch values
2 *Datafile* contains a list of frequencies in Hz

Parameters

outfile – output file with synthesised chord
datafile – contains the MIDI pitch values or frequencies of the chord to be produced
sr – Sample Rate: may be 96000, 88200, 48000, 24000, 44100, 22050, 32000, or 16000
chans – number of channels (1-16)
dur – duration of the output sound, in seconds
-a*amp* – amplitude of the output sound (0.0 to 1.0). 1.0 is maximum amplitude, and the Default value.
-t*tabsize* – the size of the table storing the waveform. It defaults to 4096: the input value is always rounded to a multiple of 4.

Understanding the SYNTH CHORD Process

All notes of the chord produced begin and end at the same time. A pure sine tone is used to generate the chord.

Musical Applications

This program was written to provide a way of generating chordal sounds for the Tonal Harmony Workshop of the Musical Testbed in *Sound Loom*. It is not expected to be used as sonic source material, but given the versatility of the CDP software, the unexpected could happen. For example, it could provide a way to create synthetic mechanical sounds, further developed with functions such as wavecycle distortion, downwards transposition etc.: e.g., to make a background hum for a generator or spaceship.

End of SYNTH CHORD

CLICKNEW – Make clicktrack using times listed in textfile

Usage **clicknew clicks** *outfile clicktimes_datafile srate*

Parameters

outfile– output click track soundfile
clicktimes_datafile– text file of times
srate– sample rate of output soundfile
LEGAL SRATES are 16000, 22050, 32000, 44100, 48000, 88200 and 96000

Understanding the CLICKNEW Process

CLICKNEW produces a series of short clicks, at the times specified in the *clicktimes* data file. There is no variation in the length or volume of the clicks produced, and no implied metre or tempo. One drawback of the program could be the requirement to specify every single time. (An ancillary program to generate these for any given duration and tempo, could be a useful addition.)

The process may be compared with the more elaborate *synth clicks*.

Musical Applications

...

End of CLICKNEW

SYNTH CLICKS – Create a click track from tempo, meter and barring data

Usage

synth clicks mode *outname clickfile* [**-s***start* **-e***end* **-z***zero*]

Modes

1 *Start* and *end* are **times**
2 *Start* and *end* are **data line numbers**

Parameters

outname – output soundfile click track
clickfile – contains a numbered sequence of data lines followed by 2, 3 or 4 data items, separated by spaces. The click track goes from *start* to *end*, with the music (optionally) starting at data line zero. Format options start with a line number followed by 2 data items (first format option), OR a line number followed by 3 or 4 data items (second format option) – the 3 data item format omits specifying an accent pattern. See **Format of the Clickfile**, below.
[-s*start*] – (optional) start time of click track
[-e*end*] – (optional) end time of click track
[-z*zero*] – option to start music at line zero

Format of the Clickfile:

- **Time and Pauses format** – The following items are placed on each (numbered) data line:
  - *time* – (jump to) a time point, such as 141.52 sec. It must be later than the previous event. E.g., **1 TIME 141.52** (Note the line number. An equals sign is not used.)
  - *GP* – initiate a General Pause of specified duration, with or without an accent at the beginning: '0' for no accent and '1' for an accent). E.g., **2 GP 1 23.7** (Line 2, General Pause of 23.7 seconds, starting with an accent – heard as a click.)

    Example:

    ```
    1 TIME 10 [Line 1, time set to 10 seconds]
    2 GP 1 12 [Line 2, General Pause of 12 seconds, with an accent]
    ```

- **Fully specified Tempo and Barring** – The following items are placed on each (numbered) data line:
  1. *tempo* – in either of these two forms, with no spaces around the equals sign:
     - **1=144** – '1' signifies a crotchet (quarter-note) and '144' signifies the tempo MM=144. Thus '0.5' signifies a quaver (eighth-note) and '1.5' a dotted crotchet, etc.
     - **0.5=144to85.3** – this specifies a gradual tempo change from quaver=144 to quaver=85.3.
  2. *barring* – this takes the form 4:4 or 6:8 or 7:16 etc., with no spaces around the colon (':').
  3. *count* – this is the integer (whole- number) number of bars in this format.
  4. [*accent pattern*] – optional: may be omitted. The format for this uses numbers and dots, as follows:
     - **1.....** – a strong beat followed by 5 weak beats
     - **1..1..** – a strong beat (start) followed by a secondary beat (at 4), weak inbetween
     - **100100** – strong and secondary beats, but *no* intermediate beats

     Example: (Explanations in square brackets are *not* part of the *clickfile*)

     ```
     1 1=60 3:4 4 1.1 [Line 1, MM crotchet=60,
     3 crotchets per bar (4/4), 4 bars, with a main accent on
     the first beat and a secondary accent on the third beat]
     2 0.5=90 6:8 4 100100 [Line 2, MM quaver=90,
     6 quavers per bar (6/8), 4 bars, with a main accent on the
     first beat and a secondary accent on the fourth beat, and no
     intermediate accents]
     3 0.5=90to60 6:8 4 1..1.. [Line 3, MM changes
     from quaver=90 to quaver=60, 4 bars, with accents on first and
     fourth as in data line 1 and weaker intermediate beats]
     ```

     Accentuation defaults: If the accent pattern is omitted, default patterns are used. These comprise:

     a. an accent on the start of each bar plus secondary accents every 3rd beat in: 6:8, 9:8, 12:8. 15:8 etc. OR 6:4, 6:16, 6:32, 9:4, 9:16, etc.
     b. an accent on the start of each bar, with *no* secondary accents in *all other meters.*

Understanding the SYNTH CLICKS Function

SYNTH CLICKS makes a click track from *start* to *end*, the music starting at dataline 1, unless 'start at dataline 0' is selected.

Provision is made for jumps to time points, General Pauses, tempo, acceleration and deceleration (i.e., gradual change from one tempo to another), barring (i.e., meters) and accent patterns.

End of SYNTH CLICKS

IMPULSE – Create a stream of impulses

Usage

impulse impulse *outfile dur pitch chirp slope pkcnt level* [**-g***gap*] [**-s***srate*] [**-c***chans*]

Parameters

outfile – output soundfile
dur – duration of output sound. If set to zero, a single impulse is produced. (Range: 0 to 7200 secs.)
pitch – pitch of impulse-stream; determines stream-frequency. (Range: 0-127 MIDI)
chirp – glissing of impulse. Care, output may clip! (Range: 0 to 30)
slope – how quickly impulse rises and falls in level. Sharper slope gives brighter timbre. (Range 1 to 20)
pkcnt – number of peaks in impulse. (Range: 1 to 200)
pkcnt divided by stream-freq (*pitch*) gives the (unglissed) freq. of impulses.
level – overall output level. (Range: 0-1)
-g*gap* – relative size of any silent gap between impulses. (Range: -0.99 to +10.0)

If > zero, lowers apparent pitch of stream
If < zero, chirps overlap, raises apparent pitch of stream

-s*srate* – sample-rate of output file. (Default: 44100)
Legal sample rates are 16000, 22050, 32000, 44100, 48000, 88200 and 96000
chans – number of channels in output file. (Range: 1 to 8; default: 1)
chans >1 raises *pitch* to a harmonic of (hmc=chans)

***pitch, glis, slope, chirp, level* and *gap* can vary through time.**

Understanding the IMPULSE Process

IMPULSE synthesizes a pulse wave, a series of short pulses, which become a continuous pitch if fast enough (e.g. MIDI 24). To hear individual pulses, set *pitch* to a very low MIDI value (e.g. 20 or below). However, the MIDI pitch value is not the only one to determine the output pitch. *Chans* (number of output channels) doubles it for stereo and raises it by another octave for quad, etc. The optional gap-size (*gap*) lowers the pitch – larger gaps between pulses means lower frequency – while a gap-size of less than 0 raises it.

Three parameters affect the timbre of the pulses:

- *slope*, which determines the steepness of each impulse's rise and fall; higher value is brighter.
- *pkcnt*, which brightens the timbre and, when high enough, also raises the perceived pitch.
- *chirp* – adds buzz to the sound.

Musical Applications

...

End of IMPULSE

MULTIOSC – nested FM-style oscillations

Usage

multiosc multiosc 1 *outfile dur frq1 frq2 amp2 srate dovesplice*
multiosc multiosc 2 *outfile dur frq1 frq2 amp2 frq3 amp3 srate dovesplice*
multiosc multiosc 3 *outfile dur frq1 frq2 amp2 frq3 amp3 frq4 amp4 srate dovesplice*

Modes

1 Oscillation of oscillation
2 Oscillation of oscillation of oscillation
3 Oscillation of oscillation of oscillation of oscillation

Parameters

dur – duration of synthesized sound, or textfile output. (secs)
frq1 – driving frequency of output. (Range: 0.001-10000 Hz, time-variable)
frq2,3,4 – nested oscillations of table read (Range: 0.001-10000 Hz, time-variable).
amp2,3,4 – amplitude of suboscillations (Range 0 to 1).
srate – sample rate of synthesized sound.
Val: 96000, 88200, 48000, 24000, 44100, 22050, 32000, or 16000
dovesplice – length of splice of start and end of output sound (Range: 1-50 mS).

Parameters *frq1*, *frq2*, *frq3* and *frq4* can vary over time.

Understanding the MULTIOSC Process

MULTIOSC creates FM-style nested oscillations from a wavetable. In Mode 1 (oscillation of oscillation), a wavetable is frequency modulated – by *frq2*. In Mode 2 (oscillation of oscillation of oscillation), Mode 1's modulations are themselves modulated – by *frq3*. In Mode 3 (oscillation of oscillation of oscillation of oscillation), Mode 2's modulations are modulated – by *frq4*. The driving frequency (*frq1*) and all the other *frq*s can be time-varying.

However, the amplitude is not really time-variable (which would envelope the output) – that is, the program does give an output if (say) amp2 is a breakpoint file, but the result invariaby seems to be white noise.

Musical Applications

...

End of MULTIOSC

MULTISYNTH – Synthesize several sound-streams from a score

Usage

multisynth synth *outfile score MM* [**-j***jitter*] [**-o***ochans*] [**-b**]

Example commandline:

```
    multisynth synth synout.wav instr.txt 120
    instr.txt:
    cello 0 60 .7 6 8 62 .7 6 16 64 .7 6 24 65 .7 6 32 67 .9 12
```

Parameters

score – Text file with each line in the form:

> **Insname** followed by any number of sets-of-4 values representing
> **Time   Pitch   Loudness** and **Duration**
> e.g. (with two 4-sets) **trumpet 0 62 .7 3   8 64 .9 12**

- **Insname:** name of instrument – one of the following:
  flute   clarinet   trumpet   violin   cello   pianoRH   *and* pianoLH
  PianoLH (left-hand) notes must immediately follow pianoRH (right-hand) notes in the score file and both must be present.

- **Time:** Note onset time, measured in thirds-of-semiquavers from time 0 (integer). Times must progress along each line of sets-of-4.
  Cello, **violin** and **piano** (LH and RH) may play up to 4 different notes simultaneously, so long as the note-combinations are possible on the real instrument.

- **Pitch:** Pitch of the note, as a MIDI value (integer).
- **Loudness:** A positive number between 0 and 1.
- **Duration:** Duration of the note, measured in thirds-of-semiquavers (integer).
  Piano notes may overlay one another (sustained resonance), but other instruments may not:
  their duration values must not extend beyond the start of the following note.
  Simultaneously sounding notes must have the same duration.

MM – Metronome mark determining tempo of sound output (BPM).
-j*jitter* – Maximum random divergence from regular note placement (in mS; default 15 mS).
-o*ochans* – Number of output channels (2-8; default stereo).
The sounds in successive lines will be spaced in the output-space from line-1, at the far left, to the last-line, at far right.
A single-line score output will be placed at front centre.
-b – If *ochans* is greater than stereo, the loudspeaker array is assumed to be surround.
Use the "-b" flag to force a bounded array (with a left extreme and right extreme).
For surround, outputs numbered in order clockwise from front-centre.
For bounded, outputs numbered in order from far-left to far-right.

Understanding the MULTISYNTH Process

The process uses a special score format to mix a limited set of given sounds, optionally across multi-channel space. Once the score format has been grasped, the process is easy enough to run.

However, the use of third-of-a-semiquaver timings and an insistence on piano LH as well as RH suggest that this function may have been written for personal rather than general use. It also seems to ignore the fact that there are many free high-quality instrumental samples available nowadays and suitable ways of playing them.

Musical Applications

It might be helpful to run the example above with different instruments before deciding whether to use this function.

End of MULTISYNTH

NEWSCALES – Synthesise a series of short tones with defined frequency and timbre

Usage

newscales *outfile datafile spectrumfile* [*srate*]

Parameters

outfile – Generic name for output soundfiles: must not end with a number
Files are numbered <outname>0.wav, <outname>1.wav, etc.
datafile – Textfile of freq-amp pairs defining note-events in output files, for example:
220 0.5
275 0.5
330 0.5
440 0.5
spectrumfile – Textfile of harmonic-number & amplitude pairs, defining the spectrum.
EITHER one line for all notes (e.g. 1 1.0 2 0.5 3 0.4 4 0.3 5 0.2)
OR one line for each frequency listed in *datafile*. (First line corresponds to lowest note, etc.)
srate – Optional sample-rate (Values: 96000, 88200, 48000, 44100 or 32000 [not confirmed]).

Understanding the NEWSCALES Process

The process outputs a series of fixed-length (0.5 sec) tones, each defined by frequency and amplitude (in *datafile*) and by timbre (in *spectrumfile*). The spectrum can be different for each note, or the same for all notes.

Musical Applications

The function may have been written for test purposes (it appears to be unsupported in *Sound Loom*).

A more flexible solution with similar parameters would be a number of calls to **NEWSYNTH Mode 1**, which also defines an additive-synthesis spectrum for each output, but adds a DURATION parameter. NEWSYNTH Mode 2 also allows the sounds to be enveloped.

End of NEWSCALES

NEWSYNTH SYNTHESIS – Synthesize complex spectra

Usage

newsynth synthesis 1 *outfile spectrum srate dur frq*
newsynth synthesis 2 *outfile spectrum srate dur frq* [**-n***narrowing*] [**-c***centring*]
newsynth synthesis 3 *outfile spectrum srate dur frq chans maxrange rate*
[**-u***rise*] [**-d***fall*][**-f***steady*][**-s***splice*] [**-n***N*] [**-a**] [**-z**] [**-x**] [**-t***spacetype*] [**-r***rotspeed*]
[[**-m**] [**-j**] [**-e***from* **-E***time*] [**-c***to* **-C***time*]]
newsynth synthesis 4 *outfile srate dur frq atk ea dec ed atoh gtow* [**-f***flv*] [-r*rnd* | **-e**]
newsynth synthesis 5 *outfile srate dur frq damping k b*

Example command line to create complex spectra:

```
newsynth synthesis 1 outsndfile spectrum.txt 44100 5.0 110
```

Modes

1 Generate tones with any number of (possibly varying) partials
2 Generate wave-packet streams with any number of (possibly varying) partials
3 Multi-channel mode: partials spread over *N* octaves fade in and out randomly
4 Fractally arrayed spikes
5 Duffing damped oscillator

Parameters

outfile – output soundfile generated by the data
spectrum (**Modes 1-3**) – textfile of times, then a series of partial ratios and their relative levels.
Every line must have the same number of partials and levels.
On each line:

- TIME: the first entry on each line is a time. Times must start at zero and increase.
- PARTIAL-NUMBER: can be whole-number or fractional (e.g. 2.5); first one must be 1. Partial numbers must increase from entry to entry.
- LEVEL: values are relative to 1st value. Normal range -1 to 0 to +1; negative values invert phase
  Values >1 and <1 are also possible, as these are relative values

Example data line: 0.00 1 1.0 2 0.9 3 0.8 4 0.7 5 0.6 [etc.]

All Modes:
srate – sample rate of the synthesised sound
Possible values: 96000, 48000, 24000, 44100, 32000, 22050
dur – duration of the synthesised sound (Range: 0.04 - 7200 secs.)
frq – fundamental frequency of the output. (Range Modes 1-4: 0.001 to 10000 Hz; Mode 5: 1.0 to 200 Hz.)
frq can vary over time.

Mode 2 only:
-n*narrowing* – narrowing (>1) or broadening (<1) of the packet envelope (Range: 0 to 1000).
Values near zero may produce clicks (square-wave envelope).
Very high values with very high frequencies may produce click-impulses or silence.
-c*centring* – position of peak of packet envelope:

- **0** peak at the centre
- **-1** peak at the start
- **1** peak at the end

Mode 3 (multi-channel) only:
chans – number of output channels
maxrange – maximum range of transposition of the spectral components (in whole octaves)
rate – average time between changes to partial content of output (0.004-100 secs)
-u*rise* – time to expand to the maximum range (0-100 secs)
-d*fall* – time to return to the initial range, before the end (0-100 secs)
-f*steady* duration of steady-state sound at the end (0-3600 secs)
-s*splice* – splice-width for partial entry and exit (2-50 mS)
-n*N* – Same fixed number of partials chosen for each event (0-1000)
-t*spacetype* – type of output spatialisation:
0=random; other values only if *chans*=8:

1. Positions alternate between Left and Right sides, but are otherwise random.
2. Positions alternate between Front and Back, but are otherwise random.
3. Rotating clockwise or anticlockwise.
4. Random permutations of all 8 channels.
5. ... plus all possible pairs of channels.
6. ... plus all possible meaningful small and large triangles.
7. ... plus square, diamond and all-at-once.
   In types 4 to 7, all members of perm are used before next perm starts.
8. Alternate between all-left and all-right.
9. Alternate between all-front and all-back.
10. Alternate between all-square and all-diamond.
11. Rotate triangle formed by loudspeakers 2-apart clockwise.
12. Rotate triangle formed by loudspeakers 3-apart clockwise.
13. Rotate triangle formed by loudspeakers 2-apart anticlockwise.
14. Rotate triangle formed by loudspeakers 3-apart anticlockwise.

-r*rotspeed* – the rotation speed (for certain spatialisation types)
-a – activate an initial increase in the number of partials starting from only the fundamental
-z – activate a decrease in the number of partials as the sound moves from steady-state back to the fundamental
-x – (Xclusive): change all partials (as far as possible) from event to event
-m – (Move): distribute partials in space
-j – (Jump): all partials are assigned to the same location for any one event
-e*from* (Emerge): the sound emerges from channel ***from***, used with
-E*time* – (Emerge): the *time* from the beginning of the sound over which the sound emerges
-c*to* – (Converge): the sound converges to channel ***to***, used with
-C*time* – (Converge): the *time* from the end over which the sound converges

> **NB:** flags **-j**, **-e**, **-E**, **-c** and **-C** are only operational if **-m** is set. They work together as an optional set of parameters, but note the following:
> **NB: flags with NO parameters must be placed on the command line AFTER any flags WITH parameters.**

Mode 4 (fractal spikes) only:
Waveform consists of spikes distributed fractally over the wavelength.
atk – Length of spike attack (0-16 samples).
ea – Rise curve of attack (Range: 0.25 to 4.0; <1 rise slow-then-fast; >1 fast-then-slow)
dec – Length of spike decay (0-16 samples).
ed – Fall curve of decay (Range: 0.25 to 4.0; >1 fall fast-then-slow; <1 slow-then-fast)
Spike placement parameters:
atoh – Ratio of on-time (a) to 1/2-grouplength (see diagram below) in grouplength,
which determines relative length of "a" (ON) and "b" (OFF).
gtow – Ratio of grouplen to total-wavelength, which determines length of trailing silence (see diagram below).
Blocks "a" then subdivided in same proportions again, etc. iteratively.
Spikes are then placed at start of each "a" block in each subdivision.

-f*flv* – Alternate "a" blocks in waveform are assigned positive and negative values.
flv determines at which fractal level this +- switch takes place. (Range: 0-256)
Level 0 uses the large blocks in the uppermost (slowest) level.
NB Maximum level of *flv* will depend on various other parameters.
If *flv* is set too high it will be reset to the maximum fractal level achievable with the current parameters.
-r*rnd* – Subdivision places 2nd "a" segment randomly between the true pos and the grouplenth-end (Range: 0-1).
-e (**not with *rnd***) – Subdivision places 2nd "a" at END of grouplength.

frq, atoh, gtow and *rnd* can vary over time.

Mode 5 (duffing) only:
damping – Damping of forced oscillation (Range: 0.15 to 2)
damping can vary over time.
k, b – Coefficients determining the nature of the damping. (Ranges: k -10 to 10; b 20 to 50)

Understanding the NEWSYNTH Process

Modes 1 and 2 generate a waveform based on a pre-defined and possibly time-varying spectrum.
The partial number can be harmonic (e.g. 2) or inharmonic (e.g. 2.1), except for the fundamental, which must be 1.

Mode 2 further allows a narrowing of each wave-cycle envelope (too high a value produces silence), which ultimately reduces to clicks, and a centring of this envelope's peak on the start, middle or end of each cycle.

Mode 3 produces a multi-channel output, and its partial content changes by random transposition every *rate* seconds, within *maxrange*, with further parameters to control this change.

In **Mode 4**, the waveform consists of spikes distributed fractally over the wavelength and the output is invariably a buzzy tone, due to the added spikes and fractalisation.

Spike placement frame:

```
 _________wavelen__________
|                          |
 ___grouplen____
|               |
 1/2 grouplen

 _______
|       |
 _a_     _a_    |
|   |   |   |   |
|   |_b_|   |_b_|__________
                  trailing
                | silence  |
```

Each wavelength is divided into two portions – a group-length followed by a trailing silence. The group-length is subdivided into two ON-OFF blocks "a" and "b". The ratio between these is set by *atoh* (on-ratio), while *gtow* sets the ratio of the grouplength to the overall wavelength and hence determines the length of the trailing silence. The ON blocks are subdivided in the same proportions iteratively and spikes are placed at the start of each ON block in the subdivision. The optional *flv* (fractal level) determines the level at which positive and negative values are switched. Finally, the optional *rnd* randomizes the position of the second ON segment, in relation to the grouplength end, but must not be used with the **-e** flag, which places the second ON block at the grouplength end.

Mode 5 is an experimental Duffing oscillator. (In tests the *damping* parameter seemed to have no effect.)

N.B. Care should be taken, especially with middle to high fundamental frequencies, that no partial is specified which would be above the Nyquist limit (22050 Hz). In Mode 4, with a very restricted frequency range, a common error message is: "Cannot proceed with these parameters (frq possibly too high)".

Musical Applications

...

End of NEWSYNTH

SYNTH NOISE – Generate noise

Usage

synth noise *outfile sample_rate channels duration* [**-a***amplitude*] [**-f**]

Parameters

outfile – noise soundfile generated
sample_rate – sample rate of *outfile*

Standard sample rates supported:
96000, 88200, 48000, 44100, 32000, 24000, 22050, 16000, 11025, or 8000
rates below 8000 are not supported – a warning is issued for other rates

channels – number of channels in *outfile* (1-16)
duration – duration of *outfile* in seconds
-a*amplitude* – amplitude of *outfile* (0.0 < Range < 1.0 max & default)

Amplitude may vary over time

-f – write soundfile as 32-bit floats (Default: 16-bit shorts)

Understanding the SYNTH NOISE Process

This process creates a single type of white noise. Variants on this noise can be achieved with filtering.

Musical Applications

You are recommended not to just take the noise file as it comes. A bit of pre-processing will enhance results when making use of it for music composition. Here are a few ideas.

Noise can be used as a complex sound source. With a rapidly changing amplitude shape, it may be a useful source for various subtractive forms of sound transformation, such as HILITE TRACE or HILITE BLTR. It can add grit, such as by being a component in COMBINE INTERLEAVE (short bursts of sound alternating with other sonic material), or by being mixed at a discrete amplitude level with another sound. COMBINE MAX is an interesting way to mingle noise with another sound.

Noise can be very uniform, in spite of its complex waveform. So some work to make it more changeable might be in order before expecting very much to happen when subjecting the material to other transformations. Given amplitude variations, these may be enhanced with ENVEL WARP functions, and any spectral envelope present can be exaggerated with FOCUS EXAG.

End of SYNTH NOISE

PULSER SYNTH – Iterate synthesized wave-packets defined by partials data

Usage

pulser synth 1-3 *outfile partials-data dur pitch*
minrise maxrise minsus maxsus mindecay maxdecay speed scatter
[**-e***expr*] [**-E***expd*] [**-p***pscat*] [**-a***ascat*] [**-o***octav*] [**-b***bend*] [**-s***seed*] [**-S***srate*] [**-c***count*]

Modes

1 Packets all have the same spectrum
2 Spectrum changes through time from one packet to the next
3 Spectrum changes at random from one packet to the next

Parameters

outfile – output soundfile.
partials-data – text datafile; lines of format:
Mode 1: pairs of **pno level** (where **pno** = partial number)
Modes 2 & 3: **time pno1 level1 [pno2 level2 ...]**
Mode 2: times must begin at zero and increase. Mode 3: times are ignored.
All modes: **pno** range: 1-64; **level** range -1 to 1 (negative values invert the phase).
dur – duration of output stream.
pitch – MIDI pitch of packets. (Range 24 to 96; may vary over time.)
minrise,maxrise – min. and max. rise-time of packet envelope (range 0.002 to 0.2 secs). Risetime is set as a random value between the 2 limits.
minsus,maxsus – min. and max. sustain-time of packet envelope (range 0.0 to 0.2 secs). Sustain is set as a random value between the 2 limits.
mindecay,maxdecay – min. and max. decay-time of packet envelope (range 0.02 to 2 secs). Decaytime is set as a random value between the 2 limits.
speed – (average) time between packets in output. (Range 0.05 to 1 sec)
scatter – randomisation of speed. (Range: 0-1)

-e*expr* – rise slope. (Range: .25 to 4 – 1 linear; >1 steeper; <1 shallower)
-E*expd* – decay slope (Range: .25 to 4 – 1 linear: >1 steeper: <1 shallower)
-p*pscat* – random jitter of pitch of packets. (Range: 0 - 1 semitones)
-a*ascat* – random jitter of amplitude of packets. (Range: 0 – no jitter – to 1)
-o*octav* – amount of lower-octave reinforcement. (Range: 0 to 1)
-b*bend* – amount of upward pitchbend of packets. (Range: 0 to 1 semitones)
-s*seed* – same seed-number produces identical output on re-run. (Integer >=1)
-S*srate* – sampling rate for synthesis (44100 to 96000)
-c*count* – number of partials to use in synthesis. (Range: 1 to 64)
Fractional values: e.g. 2.4 uses partials 1 & 2 at specified level and partial 3 at 0.4 of specified level.
Zero (default) means ignore this parameter and use ALL partials.

All parameters except *dur*, *seed*, *srate* and *min/maxrise*, *min/maxsus* and *min/maxdecay* can vary over time.

Understanding the PULSER SYNTH Process

This third member of the PULSER group (see also PULSER PULSER and PULSER MULTI) generates its stream of wave-packets by synthesis. The spectrum is defined in a datafile of partials and can be fixed, time-varying, or change randomly.

Many parameters are similar to those of the other PULSER processes. The key one is *speed* – the speed of repetition – modifiable randomly by *scatter*; both of these can be time-varying.

The envelope is shaped by *minrise,maxrise* rise time, *minsus,maxsus* sustain time and *mindecay,maxdecay* decay time, the value in each case being set randomly between the min and max values. The steepness of the rise and decay slopes is set by the exponent values *expr* for rise and *expd*for decay.

The pitch (Mode 1) is time-variable, so can be precisely controlled. Various optional parameters enhance pitch, amplitude or spatial variation, scattering over stereo or multi-channel space (up to 8 channels).

Musical Applications

...

End of PULSER SYNTH

SYNTH SILENCE – Make a silent soundfile

Usage

synth silence *outfile sample_rate channels duration* [**-f**]

Parameters

> *outfile* – silent soundfile generated
> *sample_rate* – sample rate of *outfile*
>
> > Standard sample rates supported:
> > 96000, 88200, 48000, 44100, 32000, 24000, 22050, 16000, 11025, or 8000
> > rates below 8000 are not supported – a warning is issued for other rates
>
> *channels* – number of channels in *outfile* (1-16)
> *duration* – duration of *outfile* in seconds
> **-f** – write soundfile as 32-bit floats (Default: 16-bit shorts)

Understanding the SYNTH SILENCE Process

All the samples of a silent soundfile have zero amplitude.

Musical Applications

Silent soundfiles can be used to enforce pauses before, between, or after other sonic material. For example, a 2 second pause may be wanted between a list of soundfiles. This is done with the splicing function, SFEDIT JOIN, with a silent soundfile of 2 seconds duration spliced between each of the other soundfiles.

When specifying the sample rate, remember that it needs to be consistent with your other soundfiles so that it can be used in a mix.

Similarly, splicing with stereo soundfiles requires a stereo silent file.

End of SYNTH SILENCE

SYNTH SPECTRA – Generate both channels of a stereo spectral band

Usage

synth spectra *outfilename dur frq spread max-foc min-foc timevar* [**-p**]

Parameters

outfilename – generates output files named 'outfilename0' and 'outfilename1'
dur – duration of output sound, in seconds
frq – centre frequency of the band
spread – width of band in Hz (default) or as transposition ratio
min-foc – minimum degree to which the band energy is focused on (concentrated at) the centre frequency (Range: 0.001 to 1000)
max-foc – maximum degree to which the band energy is focused on (concentrated at) the centre frequency (Range: 0.001 to 1000)
timevar – degree to which the band varies over time (Range: 0 to 1)
-p – if this flag is set, *spread* is a transposition ratio

Understanding the SYNTH SPECTRA Process

SYNTH SPECTRA generates tones which can be spread across a bandwidth and be focused to different degrees on a specified pitch. This pitch focus may be made to wander around randomly within the band by the *timevary* parameter.

Focus (*min-* and *max-*) is how tightly the energy is focused around the frequency you specify.

- If **focus** is very high, you can get almost a pure tone (with a narrow bandwidth – *spread*, and *timevar* at 0).
- With a low **focus**, the band is broader, less focused.
- With different values, the **focus** wobbles, so the sound is somewhat coloured.

The *timevar* parameter can be used to move the point of focus in the frequency band.

- When *timevar* is set at 1, e.g., with a highly focused band, the point of focus will move randomly around the bandwidth. Thus the pitch will be highly focused (very little noise component) but wobble randomly within the band.
- When *timevar* is set to 0, the focus will adopt some position in the band and stay there without moving.
- Intermediate values for *timevar* produce results inbetween these two extremes.

Musical Applications

If you want to get as close as possible to a pure tone, with no warble, set *min-foc* and *max-foc* to 1000.

End of SYNTH SPECTRA

SYNFILT – Noise filtered by time-varying filterbank, with time-variable Q

Usage

synfilt synfilt mode *outfile data srate chans Q hcnt rolloff seed* [**-d**] [**-o**]

Modes

1 Single (varying) pitch
2 Simultaneous pitches

Parameters

outfile – output soundfile. In both modes, the output duration is set by the last entry in *data*.
data – Mode 1: enter filter pitch as time & MIDI-pitch pairs (MIDI range: 0 to 127; 0 produces silence)
Mode 2: VARIBANK-style data for filter bands at successive times, each line containing:
Time MIDIPitch1 Amp1 [MIDIPitch2 Amp2 etc....]
In Mode 2:

- Pitch and amp values must be paired.
- Each line can contain any number of pairs, but must have SAME number of pairs.
- Time values (in secs) must be in ascending order (and >=0.0)
  MAXIMUM TIME must be >0.03 secs (30mS).
- To eliminate a band in any line(s), set its amplitude to 0.0
- Amp values may be numeric, or dB values (e.g. -4.1dB)
- Comment-lines may be used: start these with ';'.

srate – Sample rate of output file (Vals: 96000, 48000, 88200 or 44100 Hz)
chans – Output mono (1) or stereo (2).
Q – tightness of filter. High values produce more clarity of pitch. (Range: 0 to 10000.0)
hcnt – no. of harmonics of each pitch to use. (Range: 1 to 8, default 1)
Care! High harmonics of high pitches may be beyond Nyquist frequency.
(No. of pitches times no. of harmonics determines program speed).
rolloff – drop in level from one harmonic to next. (Range: 0 to -96.0dB)
seed – initialises random-noise generation (Range: 1 to 1000; default 0).
-d – double filtering.
-o – Drop out if filter overflows.

Understanding the SYNFILT Process

SYNFILT filters noise to create a time-variable single pitch (Mode 1) or a time-variable chord (Mode 2). Mode 1 (single pitch) uses a breakpoint daafile of Time and MIDI-pitch values, while Mode 2 (chord) uses the VARIBANK format of Time, Pch1, Amp1, Pch2, Amp2,... The output length is set by the last time value in this datafile.

The filter tightness (Q) determines how noisy the output will be: a high value (in the 100s) focusses mostly on the pitch(es) and less noise is present. Note that this is time-variable. Double-filtering will also make the pitch more prominent.

Because white noise is randomly generated, the output is more naturally varied than some synthesis methods. You also specify the number of harmonics to generate for each pitch: care must be taken that these never exceed the Nyquist frequency for the sample rate (typically 11050Hz for 44100 Sample Rate). *Rolloff* shapes the decrease in amplitude from one harmonic to the next, higher harmonics being normally quieter than lower ones.

There are two ways of introducing silence: one is by specifying MIDI-pitch 0 (more suitable for Mode 1), the other is by setting amplitude in Mode 2 to 0, which can be done for different pitches at different times, as required. As always with CDP, if you don't want a gradual change from one pitch to the next, insert a value just before the change that is the same as the previous value.

Note that the CDP Usage is currently misleading: the parameters are as above, in that order – the datafile is Param.1, Sample Rate is Param. 2, and so on. The supposed *dur* param. does not exist (duration is set by the datafile) and *gain* also does not exist, nor does the **-n** flag.

Musical Applications

The link with VARIBANK (Mode 2) is significant, as you can employ an existing datafile (as used by VARIBANK filtering or TUNEVARY, for instance) and possibly fade from one process into the other using MIX BALANCE. SYNFILT's output can also be rhythmicised using VOCODE, CROSS or similar processes.

End of SYNFILT

SYNSPLINE – Synthesise waveforms by smoothly joining randomly generated points

Usage

synspline synspline *outfile srate dur frq splinecnt interpval seed* [**-s***maxspline*] [**-i***maxinterp*] [**-d***pdrift* **-v***driftrate*] [**-n**]

Parameters

outfile – output soundfile.

srate – sample rate of synthesized sound.
Legal *srates* are 16000, 22050, 32000, 44100, 48000, 88200 and 96000

dur – duration of synthesized sound. (Range: 0.04 to 7200 secs.)

frq – fundamental frequency of output. (Range: 0.001 to 10000 Hz)

splinecnt – number of random values (to smooth between) per half-wavecycle. (Range: 0 to 64)

interpval – number of wavecycles over which one waveshape morphs to the next. (Range: 0 to 4096)

seed – same seed value produces identical output with same parameters. (Range: 0 to 64)

-s*maxspline* – maximum *splinecnt*. Causes random values of *splinecnt* to be generated in the range *splinecnt* to *maxspline*. (Range: 0 to 64; >=*splinecount*; ignored if zero.)

-i*maxinterp* – maximum *interpval*. Causes random values of *interpval* to be generated in the range *interpval* to *maxinterp*. (Range: 0 to 4096; >=*interpval*; ignored if zero.)

-d*pdrift* – semitone (half-)range of random drift in frequency. (Range: 0 - 12)

-v*driftrate* – average time (in mS) to the next drift offset. (True time is 1/2 to 3/2 of this.) (Range: 1-1000 mS)

-n – normalise every cycle. (Default: cycles retain random valued amplitudes.)

frq, splinecnt, interpval, maxspline and *maxinterp* may vary over time.

Understanding the SYNSPLINE Process

SynSpline smoothly connects randomly generated points to create a constantly changing synthetic waveform. SPLINECOUNT sets the number of points to create per half-cycle, while MAXSPLINE sets an upper range for this value - the actual value being between SPLINECOUNT and MAXSPLINE; both of these are also time-variable.

The wavecycles morph (change shape) over a number of cycles. This is set by INTERPOLATE; again, the actual value is between this and MAXINTERP and, again, bth are time-variable.

Optional parameters PITCHDRIFT and DRIFTRATE set an amount of pitch variation and the rate of pitch drift. Minute changes of pitch are also an important factor in synthesising "realistic"-sounding tones. The level of each cycles is also set randomly, with an option to normlalize all cycles evenly.

Musical Applications

...

End of SYNSPLINE

TS OSCIL – Create sound from time-series text data

Usage

ts oscil *indata outsnd downsample* [**-d***maxdur*] [**-c**] [**-f**]

Parameters

indata – text data as a list of numerical values.
outsound – output soundfile.
downsample – downward transposition of data in octaves (0 - 16)
= timestretch (by speed) as a power of two
downsample can vary over time: time values are times in output sound.
maxdur – maximum duration of output sound.(Range 1 - 600 secs)
-c – Interpolate downsampled data using cubic spline (default linear).
-f – Force duration to 'maxdur' (if Ness) by looping input data.
Flag is invalid if *maxdur* is not specified.

Understanding the TS OSCIL Process

Time-series data for a waveform consists of a series of amplitude values – normally in the range -1.0 to +1.0. TS OSCIL takes a textfile list of such numbers and treats it as a soundwave plot, converting them into sound. The numbers are arbitrary (within the range) and can come from any source. The utility **TS CONVERT** can be used to convert arbitrary values into the required range for sound.

A simple test for this program is:

1. take an existing very short sound, apply **SNDINFO PRNTSND** (which outputs two columns),
2. then apply SNDINFO GETCOL to column 2, to extract the list of amplitudes
   NB: at CD sampling rate of 44.1KHz there will be over 41,100 numbers per second!
3. apply TS OSCIL to turn the numbers back into the source sound.
   (The sample rate is not specified but appears to be 441000.)

Musical Applications

TS OSCIL is a means of turning an arbitrary series of numbers into sound. It may have been inspired by Xenakis's UPIC system, in which users could draw a waveform that could be sonified. A table of numbers, representing a single wavecycle, could be copied and pasted a number of times to lengthen it (though a dedicated application would be preferable). Note that unless the numbers are cyclic in some way, the result will be noise of some sort.

It may be possible to utilise CDP's **PACKET** and **WAVEFORM** functions to assist in creating an interesting series of numbers.

The reference to "Ness" in **-f** suggests that this function (part of the SoundLoom Data Pack) was intended for use with Ness Brass (physical modelling).

Related Functions

TS TRACE: Create sound from time-series data treated as a pitch-trace
TSCONVERT: Convert input data to specified range and format

End of TS OSCIL

TS TRACE – Create sound from time-series data treated as a pitch-trace

Usage

ts oscil *indata outsnd harmdata tstr frq hrange* [**-d***maxdur*] [**-c**] [**-f**]

Parameters

indata – text data as a list of numerical values.
outsound – output soundfile.
harmdata – textfile list of number pairs (harmonic-number & amplitude) for each harmonic in waveform.
Fractional values (> 1) for harmonic-number generate inharmonic spectra.
'0' (zero), instead of a filename, outputs a pure sinetone.
tstr – time-stretch of data (Range 1 - 10000).
tstr can vary over time: time values are times in output sound.
frq – frequency of the mean pitch of the output. (Range 16 - 11025 Hz)
hrange – pitch range upwards (or down) from mean. (Range 0 - 48 semitones)
maxdur – maximum duration of output sound.(Range 1 - 600 secs)
-c – Interpolate downsampled data using cubic spline (default linear).
-f – Force duration to 'maxdur' (if Ness) by looping input data.
Flag is invalid if *maxdur* is not specified.

Understanding the TS TRACE Process

A single column of numbers (within frequency range) is treated as the pitch-trace of some defined waveform. For timbre, additive harmonic data in defined in *harmdata*.

To experiment with the program, an actual pitch trace (from a short sound) can be extracted by running **REPITCH GETPITCH Mode 2** (extract pitch as breakpoint file), followed by **SNDINFO GETCOL** for column 2. This gives a raw list of frequencies with no time values.

The relationship between these frequencies and that specified by *frq* is not too clear. In a test, a *frq* value matching roughly that of the extracted pitch-trace produced a very high steady pitch (*hrange*=0) and only the lowest acceptable value for *frq* came close to the original pitch. MAXDUR did not define the output duration, which was extremely short. Some further user experience of this function would therefore be welcome.

Musical Applications

A more useful function along similar lines may be **REPITCH SYNTH**, perhaps preceded by **REPITCH GENERATE** (creates pitchdata from time & midi values) and **BRKTOPI**.

The reference to "Ness" in **-f** suggests that this function (part of the SoundLoom Data Pack) was intended for use with Ness Brass (physical modelling).

Related Functions

TS OSCIL: Create sound from time-series text data
REPITCH SYNTH: Create the spectrum of a sound following the pitch contour in a pitch file
TSCONVERT: Convert input data to specified range and format.

End of TS TRACE

TSCONVERT – Convert input data to specified range and format.

Usage

```
tsconvert indata outdata min max [-cminstep | -r | -q | -Q]
          [-ddur [-ftimes]]  [-mmaxoutdur] [-l]
```

Parameters

indata – Input text data as a list of numerical values.
outdata – Output text data as a list of numerical values or other format.
min – Minimum value in output data values.
max – Maximum value in output data values.
If minimum > maximum, the data contour is inverted in the output.

-c*minstep* – Compact data, so that no absolute value is less than *minstep*, and steps in the same direction are joined into the same step.
-r – Rectify the data around the mean.
Mean value = minimum: other values become deviations from the mean.
-q – Quantise the data to whole-number values only.
-Q – Quantise the data and suppress adjacent equal-values.

-d*dur* – Generate a breakpoint file of *time-value* pairs, and duration *dur*, thus determining the timestep between output values.
-f*times* – Textfile: output (untimed) values at times specified in *times* file (needs **-d** flag).
-m*maxoutdur* – Curtail the total outlength to the maximum of *maxoutdur*.
With breakpoint outputs, values generated beyond *maxoutdur* are ignored.
With the **-c** flag, the absolute size of values are summed.

-l – Output values vary as log of input values.

Understanding the TSCONVERT Process

TSCONVERT is not a synthesis function, but a data-editing utility that converts a simple list of numbers into an output text file that can be used with TS OSCIL, TS TRACE, and other functions as well.

The compuloly parameters ***min*** and ***max*** set limits within which the data will be scaled. For **TS OSCIL**, these would be **-1** and **+1**, for amplitude values. For **TS TRACE**: any reasonable frequency range between 16Hz and 11025 Hz (see its *frq* parameter).

Other options within the parameter set include setting a step value between successive numbers, quantising the data values, and generating a breakpoint data file with either equal timesteps determined by the total outlength or by a an optional list of times.

Musical Applications

The scope of this function is clearly greater than re-scaling values for use with TS OSCIL and TS TRACE.

Related Functions

TS OSCIL: Create sound from time-series text data
TS TRACE: Create sound from time-series data treated as a pitch-trace
COLUMNS: Utilities for data manipulation

End of TSCONVERT

SYNTH WAVE – Generate synthetic waveforms

Usage

synth wave mode *outfile sample_rate channels duration frequency* [**-a***amplitude*] [**-t***tablesize*] [**-f**]

Modes

1 Sine wave
2 Square wave
3 Triangle wave
4 Ramp (sawtooth) wave

Parameters

outfile – soundfile generated
sample_rate – sample rate of *outfile*

Standard sample rates supported:
96000, 88200, 48000, 44100, 32000, 24000, 22050, 16000, 11025, or 8000
rates below 8000 are not supported – a warning is issued for other rates

channels – number of channels in *outfile* (1-16)
duration – duration of *outfile* in seconds
frequency – frequency of *outfile* in Hz. *Frequency* may vary over time
-a*amplitude* – amplitude of *outfile* (0.0 < Range < 1.0 max & default). *Amplitude* may vary over time
-t*tablesize* – size of table storing the waveform (Default: 256). The input value is always rounded to a multiple of 4.
-f – write soundfile as 32-bit floats (Default: 16-bit shorts)

Understanding the SYNTH WAVE Process

A table of the appropriate size is created to store the waveform. It is then used to generate the soundfile with properties as specified by the parameters.

The breakpoint file option gives some additional shaping control, e.g., to create a softer attack and ensure that there will not be clicks.

Musical Applications

Geometric waveforms of this type (square, sine, ramp) do produce upper partials, but are mostly used for test purposes or to illustrate some feature of the software in a clear manner, rather than as source material for musical composition.

End of SYNTH WAVE

Last Updated 27 Apr 2023
Documentation: Archer Endrich, revised R. Fraser