CDP MULTI-CHANNEL Functions

(with Command Line Usage)

Introduction

Unlike most other CDP process groups, there is no group program called 'MULTICHANNEL'. The functions listed below are all separate programs, except for the TANGENT and TRANSIT groups. There is also the **MULTI-CHANNEL TOOLKIT** group of file-handling functions, documented separately. Various other multi-channel functions have also been documented elsewhere and these are listed below under "Other Multi-Channel Processes".

The multi-channel facilities described here generate and manipulate multi-channel **wav** (or **aiff**) soundfiles. The (*N*) channels of these files can be sent to (*N*) loudspeakers arranged in an arc or a ring around the listener. **In the descriptions and explanations given, a clockwise ring arrangement is assumed.**
However, it is not essential. For example:

- The loudspeakers can be arranged in any spatial location or orientation desired.
- The output channels can be assigned to loudspeakers in any way desired.
- The output soundfile can be converted to an Ambisonic or some other format, using the **Multi-channel Toolkit**.

If you do not have a multi-channel soundcard or multiple speakers, there is still much that can be done with these programs. Many processes distibute segments of sound over multi-channel space, but these channels can be split into separate files via **HOUSEKEEP CHANS 2** or **CHANNELX** and re-combined by mixing or interleaving, via **INTERLX** or **SUBMIX INTERLEAVE**.

ALSO SEE: **OVERVIEW** of multi-channel facilities available via CDP.

Index of Multi-Channel functions

All functions are separate stand-alone programs except those in the Tangent and Transit groups.

BROWNIAN
Generate texture of sampled elements following brownian motion in pitch and space

CASCADE
Successive segments are repeat-echoed, and the echosets are superimposed on the source

CRUMBLE
Project segments spatially over progressively smaller groups of channels

CRYSTAL
Generate sound-events based on the position of vertices of a crystal, then rotate the crystal in 3-D space

FLUTTER
Add multi-channel-distributed tremolo to a multi-channel file

FRACTURE
Disperese a mono signal into fragments spread over *N*-channel space

FRAME SHIFT
Reorient or rotate a multi-channel file

MCHANPAN
Pan sounds around a multi-channel space

MCHANREV
Create multi-channel Echoes or Reverb

MCHITER
Iterate the input sound in a fluid manner, scattering around a multi-channel space

MCHSHRED
Sound is cut into random segments which are then reassembled in random order within the original duration

MCHSTEREO
Combine stereo files in a multi-channel output

MCHZIG
Extend by reading back and forth in the soundfile, while panning to a new channel at each 'zig' or 'zag'

MTON
Create a multi-channel equivalent of a mono soundfile

MULTIMIX CREATE
Create a multi-channel mixfile

NEWMIX
Mix from a multi-channel mixfile to give a multi-channel soundfile output

PANORAMA
Distribute *N* source files in a panorama across a specified angle of a sound-surround loudspeaker array

SPIN STEREO
Spin a wide stereo image across stereo / multichannel space, with possible doppler-shift

SPIN QUAD
Spin two wide stereo-images across a 5-channel-wide sound image, with possible doppler-shift

STRANS MULTI
Change the speed or pitch of a multi-channel sound, or add vibrato

TANGENT GROUP:

TANGENT ONEFILE
Make a mixfile to play repeats of a mono soundfile along a tangent path

TANGENT TWOFILES
Make a mixfile to play repeats of two synchronised mono soundfiles along a tangent path

TANGENT SEQUENCE
Make a mixfile to play a sequence of mono soundfiles along a tangent path

TANGENT LIST
Make a mixfile to play a sequence of mono soundfiles as listed in a textfile along a tangent path

TEXMCHAN
Create textures over a multi-channel frame

TRANSIT GROUP:
TRANSIT SIMPLE
Place repetitions of a mono soundfile on a path *into* and *across* an 8-channel array

TRANSIT FILTERED
Place filtered repetitions of a mono soundfile on a path *into* and *across* an 8-channel array

TRANSIT DOPPLER
Place pitch-shifted repetitions of a mono soundfile on a path *into* and *across* an 8-channel array, suggesting a doppler shift

TRANSIT DOPLFILT
Doppler effect on a path *into* and *across* an 8-channel array with filtering, to suggest greater distance

TRANSIT SEQUENCE
Position a sequence of mono sounds (at least 3) on a path *into* and *across* an 8-channel array

TRANSIT LIST
Position a sequence of mono sounds (at least 3), as listed in a textfile, on a path *into* and *across* an 8-channel array

OTHER MULTI-CHANNEL PROCESSES:
MULTI-CHANNEL TOOLKIT
Multi-Channel File Handling Functions
NEWTEX
Generate a texture of grains made from a source sound or sounds
WRAPPAGE
Brassage on one or more sounds, with (moving) multi-channel output

OVERVIEW
An overview of multi-channel facilities available via CDP
Technical Discussion
On setting up a multi-channel composing environment
Appendix 1: M-C Mixfiles
MULTIMIX and multi-channel mixfiles

BROWNIAN – Generate texture of sampled elements following brownian motion in pitch and space

Usage

brownian motion 1 *fi fo chans dur att dec plo phi pstart sstart step sstep tick seed* [**-a***arange*] [**-m***minamp*] [**-s***aslope*] [**-d***dslope*] [**-l**]
OR
brownian motion 2 *fi fo chans dur plo phi pstart sstart step sstep tick seed* [**-a***arange*] [**-m***minamp*] [**-l**]

Modes

1 Source is sampled as a waveform
2 Whole source is transposed for output events

Parameters

fi – (mono) input soundfile to be read at different speeds to generate output events

> Mode **1**: source must start & end at sample value 0.0: sampled as waveform.
> Mode **2**: source can be anything and the whole source is transposed for output events. (In mode **2**, a very long source may take a very long time to finish.)

fo – output file
chans – number of channels in the output file
dur – (max) duration of output file
*att** – rise time of events (Mode **1** only)
*dec** – decay time of events (Mode **1** only)
*plo** – bottom of pitch range (MIDI)
*phi** – top of pitch range (MIDI)
pstart – initial pitch (MIDI)
sstart – initial spatial position (numbering chans 1 - *N*) (ignored if mono output)
*step** – maximum pitch step between events
sstep – max spatial step between events (fraction of distance between channels)
tick* – (average) time between events
seed – seed (initialises random vals; gives reproducible random sequence)
-a*arange** – maximum loudness step between events, in dB (Default: min=0; max=96dB)
-m*minamp** – minimum loudness (Range: >0 to 96dB)

- Default: 0 = No minimum
- (Only comes into play if *arange* is > 0)
- (If min > 0: if amp falls to -min dB, levels 'bounce' off the min value)
- (If min = 0: no min set, and if level falls to -96dB, sound stream halts)

-l – loudspeakers arrayed in a line. (Default: arrayed in a 'circle')
Mode 1 only:
-s*aslope** – attack slope: < 1: rise fast then slows; > 1: rise slow then faster
-d*dslope** – decay slope: < 1: fall slow then faster; > 1: fall fast then slows; *slope* ranges are 0.1 to 10.

> All items marked with "*" can vary through time.

Understanding the BROWNIAN Process

...

End of BROWNIAN

CASCADE – Successive segments are repeat-echoed, and the echosets are superimposed on the source

Usage

cascade cascade 1-5 *inf outf clipsize echos clipmax* [**-e***echosmax*] [**-r***rand*] [**-s***seed*] [**-N***shredno* **-C***shredcnt* **-a**] [**-l**] [**-n**]
OR
cascade cascade 6-10 *inf outf cuts echos* [**-e***echosmax*] [**-r***rand*] [**-s***seed*] [**-N***shredno* **-C***shredcnt* **-a**] [**-l**] [**-n**]

Modes

1,6 N-channels in to N channels out: echo in the same N channels
2,7 Mono (left in output) to stereo: echo-sets pan to right.
3,8 Mono (centre in output) to stereo: echo-sets pan alternately between left and right
4,9 Mono (Ch1 in output) to 8chan: each echo steps right, to next channel.
5,10 Mono (Ch1 in outout) to 8chan: echoes of 1st echo-set step right, next set step left,etc.

Parameters

infile – input soundfile: mono except possibly in modes 1 and 6
outfile – output soundfile: Modes 1 and 6 - channels as input; Modes 2,3,7,8: stereo; Modes 4,5,9,10: 8-channel
clipsize (Modes **1-5**) – duration of segments to echo. Time-variable. (Range: 0.005 to 60 sec)
OR *cuts* (Modes **6-10**) – textfile of (successive) source cut-times, creating segments to echo. 1st cut-time must be > 0.01 secs.
echos – number of echoes, including the source (echo 1) (Range: 1 to 64; Time-variable)
clipmax – max duration of clips (time-variable).

- *clipsize* now read as minimum.
- Actual *clipsize* is a random value between *clipsize* and *clipmax*.
- If *clipmax* is set to zero, it is ignored.

-e*echosmax* – maximum number of echoes (time-variable).

- *echos* now read as minimum.
- The actual number of echoes is a random value between *echos* and *echosmax*.
- If *echosmax* is set to zero, it is ignored.

-r*rand* – randomise timesteps between echoes in the echo-set; Range: 0-1, time-variable.
-s*seed* – with the same non-zero value, randomised values are exactly the same on a new pass
-N*shredno* – in each echo-stream cut the previous echo into *shredno* parts, and random-shuffle the parts to make the next echo. Range 2-16, time-variable.
-C*shredcnt* – the number of shreds to do, to create the next echo element; Range: 1-16, time-variable. Both *shredno* and *shredcnt* must be set for shredding to take place.
-a – also shred the original clip. Only valid if *shredno* and *shredcnt* are set.
-l – echoes decay linearly in level (Default: log decay in which the initial decays are faster).
-n – if the output is low, normalise it. (High output is normalised by default.)

Understanding the CASCADE Process

CASCADE echoes a given segment a number of times and mixes the echoes with the original soundfile (played 'straight'). There are five mode choices, all spatial, plus the ability to select the echo segment by size (*clipsize*) or from a list of cut-times (*cuts*), giving 10 modes overall.

The segment size (Modes 1-5) is set by *clipsize* or randomly between this and *clipmax*. Similarly, the number of repetitions is set by *echos* or randomly between this and *echomax*. *Shredno*, *shredcnt* and the **-a** flag "shred original clip" offer possibilities to garble the segments and echoes.

Musical Applications

The program is an alternative to EXTEND ITERATE, MODIFY REVECHO 3 (stadium echo), and MODIFY REVECHO 1/2 (delay line). CASCADE adds the unusual ability to select the segments and spatialise and/or shred the echoes.

End of CASCADE

CRUMBLE – Project segments spatially over progressively smaller groups of channels

Usage

crumble sound 1 *infile outfile stt dur1 dur2 orient size rand iscat oscat ostrch pscast seed* [**-s***splce*] [**-t***tail*]
crumble sound 2 *infile outfile stt dur1 dur2 dur3 orient size rand iscat oscat ostrch pscast seed* [**-s***splce*] [**-t***tail*]

Modes

1 8-channel output
2 16-channel output (mode faulty at time of writing)

Parameters

infile – input soundfile (MONO)
outfile – output soundfile (MULTI-CHANNEL)
stt – start time of crumbling (Range: 0 to <filelength)
dur1 – duration of section where signal is split into 2 images (secs)
dur2 – duration of section where signal is split into 4 images (secs)
dur3 – (Mode 2) duration of section where signal is split into 8 images (secs)
orient – channel around which input first split (1-8 or 1-16 Mode 2):
input image first splits onto 2 blocks each of ochans/2 adjacent chans
(1) orig. chan. with adjacent clockwise channels
(2) remaining channels
*size** – average duration of cut segments (secs)
*rand** – randomisation of segment size. (Range: 0 to 1)
Value 1 modifies *size* randomly between *size*/2 and (3**size*)/2
*iscat** – scatter start-time (in source) of next segment. (Range: 0 to 1)

- Cut-time in source always advances
- With no scatter, step to next cut-time minus the length of the previous segment cut.
- With maximum *iscat* (i.e. 1), step by random time between 0 and the previous-seglen.

*oscat** – scatter time-placement (in output) in range used. (Range: 0 to 1)

- Time in output always advances.
- With no scatter, the step to the next output-time = the length of the last segment cut.
- With maximum *oscat* (i.e. 1), step by random time between > 0 and the previous-seglen.

*ostrch** – stretch time-placement in output. (Range: 1 to 64)

- The step to the next segment-placement in the output is multiplied by *ostrch*
- *ostrch* > 1 should generate silent gaps in the output.

*pscast** – pitch variation of output segments (Range: 0 to 12 semitones)
seed – same seed value gives identical output on successive process runs.
-s*splce* –length of splices which cut the segments (Range: 2 to 50 mS)
-t*tail** – length of any exponential tail on segments (Range: 0 to 1000 mS;

Parameters marked with '*' can vary over time (i.e., time in the outfile).

Understanding the CRUMBLE Process

CRUMBLE projects mono segments over progressively smaller groups of channels in multi-channel space, with possible timestretch. Specifically, it projects the source onto all channels, then segments it and distribute the segments over smaller and smaller groups of channels.

The choice is limited to 8-channel or 16-channel output, but this can be reduced afterwards by **PAIREX** (extract stereo pairs) or **CHANNELX** (extract single chans) plus optional interleaving (**INTERLX**).

Musical Applications

...

End of CRUMBLE

CRYSTAL – Generate *N* sound-events based on the position of *N* vertices of a crystal, then rotate the crystal in 3-D space and generate another group of *N* events, etc.

Usage

crystal rotate 1-10 *infile1* [*infile2 infile3 ...*] *fo vdat rota rotb twidth tstep dur plo phi* [**-p***pass* **-s***stop*] [**-a***fatt*] [**-P***fpresc*] [**-F***fslope*] [**-S***sslope*]

Modes

1 Mono output
2 Stereo output
3 Two channels of 8-channel output, spaced by a single channel (here, channels 1 and 3)
4, 5, 6 Ditto, channel-pair steps clockwise, anticlockwise or randomly beteween groups-of-events
7, 8, 9 Ditto, but pairs of channels are adjacent (e.g., 1,2 or 5,6)
10 Stereo output: each set-of-vertices are output as a separate soundfile.

Parameters

infile(s) – One MONO infile, multiply-read (with delay), generating out-events
OR
N MONO infiles, generating different events for *N* vertices
outfile(s) – output file(s), see Modes.
fo – generic name for output files
vdat – data file containing:
(1) Triples, being (initial) X,Y,Z coordinates of crystal vertices.
Range: > -1 to < 1. Xsquared + Ysquared +Zsquared < 1 for all vertices
(2) Time-value pairs defining an envelope to be imposed on sound events.
Times start at 0 and increase. Final time = duration of events.
Value range: 0 to 1. First and last values must be zero.
rota, rotb – rotate speed in xy_plane and xz_plane, revolutions per second. (Range: -10.00 to 10.00)
twidth – max time between onsets of first and last event of any *N*-events group
tstep – time-step between each sampling of all *N* vertices of rotating-crystal
dur – total duration of output (must be greater than 'tstep' and 'twidth')
plo, phi – minium and maximum (MIDI) pitch of any event
-p*pass*, **-s***stop* – Pass + stop bands (in Hz) for lopass filter: ***stopfrq - passfrq* >= to Hz**
-a*fatt* – max attenuation produced by filter-stop (dB). (Range: 0 to -96)
-P*fpresc* – gain applid to attenute source before applying filter (Range: 0 to 1)
-F*fslope* – slope curve mixing filtered to unfiltered sound (depth). (Range: 0.10 to 10.00). 'Slope' = 1 for a linear slope.
-S*sslope* – slope curve mixing transposed sound to original (close). (Range: 0.10 to 10.00) 'Slope' = 1 for a linear slope.
outcnt – number of rotated-sets to output. [NB: not shown on commandline and don't know where it should go.]

Understanding the CRYSTAL Process

About the coordinates:

- **X** coord –> time and, if stereo, space-position
- **Y** coord –> pitch
- **Z** coord –> brightness, i.e., **Z**-far sounds are lopass-filtered and mixed to original; **Z**-close sounds move 8va up and are stacked on the original

Musical Applications

...

End of CRYSTAL

FLUTTER – Add multi-channel-distributed tremolo to a multi-channel file

Usage

flutter flutter *inmcfile outmcfile chanseq freq depth gain* [**-r**]

Example command line to create tremolo effects between channels:

```
flutter flutter in4chansnd out4chflutter chanseq.txt 5 7 0.75
```

Parameters

infile – input multi-channel sound file
outfile – output multi-channel sound file
chanseq – On each cycle of the loudness fluctuation, a different set of output channels fluctuates in loudness (rising and falling) while the other channels do not change in level. So the tremolo effect moves rapidly from one set of channels (**chanset**) to another as it proceeds.

- *Chanseq* is a textfile containing a list of these **chansets**.
- Each line of this file has a list of some of the output channels, and constitutes one **chanset**.
- Any number of the output channels can be used on any line.
- Once the end of the list of sets is reached, it begins again at its start, cycling round and round the list – but see the **-r** flag.

freq – frequency (in Hz) of the loudness variation
depth – depth of the loudness variation (Range: 0 to 16)

- As values increase from 0 to 1, the **troughs** of the loudness variation become lower and lower until they reach down to zero level (for *depth* value 1).
- Depth-values greater than 1 increase the **steepness and narrowness** of the loudness peaks (still troughing at zero level).
- Both *freq* and *depth* may vary over time.

gain – overall gain of the loudness variation (Range: 0 to 1)
-r – randomisation of **chanset** order

If this flag is used, once all the **chansets** have been used, their order is randomly permutated before they are used again, and once all have been used a 2nd time, their order is permutated again, and so on.

Understanding the FLUTTER Process

FLUTTER is a special multi-channel version of ENVEL TREMOLO. The latter program causes the loudness of the **entire** soundfile to fluctuate at a particular frequency and depth. FLUTTER, on the other hand, causes these tremulations to pass from one (set of) output channel(s) to another as it proceeds, making the multi-channel file tremble in a specified way.

Musical Applications

FLUTTER belongs to the class of programs like tremolo and vibrato, which can produce subtle or dramatic changes to the quality of a sound, or can be gradually introduced to the tail of a sound to add liveliness to the sonic event.

End of FLUTTER

FRACTURE – Disperse a mono signal into fragments spread over *N*-channel space

Usage

fracture fracture 1 *infile outfile etab chns strms pulse edpth stkint* [**-h***seed*] [**-m***min*] [**-i***max*] [**-r***rnd*] [**-p***prnd*][**-d***disp*][**-v***lrnd*][**-e***ernd*][**-s***rnd*][**-t***trnd*] **-y -l**

fracture fracture 2 *infile outfile etab chns strms pulse edpth stkint cntre frnt depth rolloff* [**-h***seed*] [**-m***min*] [**-i***max*] [**-r***rnd*] [**-p***prnd*] [**-d***disp*] [**-v***lrnd*] [**-e***ernd*] [**-s***rnd*] [**-t***trnd*] [**-a***atten*] [**-z***zpoint*] [**-c***contract*] [] [**-l***lopnt*] [**-f***fmix*] [**-j***ffrq*] [**-k***up*] [**-w***dn*] **-y**

Example command line to create :

```
fracture fracture 1 in.wav out.wav etab.txt 4 4 0.5 1.0 0
```

Modes

1 The output is *N*-channel dispersal in *N*-channel space
2 The output is stereo dispersal (possibly moving) in surround space

Parameters

infile – input soundfile (mono)
outfile – output multi-channel soundfile
etab – a text5file in which each line is a TIME followed by 7 pairs of envelope-data in the form *etime lev*. *etime* is relative to time WITHIN the envelope (Range: 0 to 1), and *lev* is the level at *etime* (Range: 0 to 1). In each envelope data set *lev* must start and end at zero and rise to a maximum value of 1.0.
chns – the number of channels in the output soundfile (Range: 2 to 16). In Mode **2** the number of channels is restricted to multiples of 4.
strms – the number of spatial positions (streams) for resulting fragments (Range: greater than 4)
pulse – the average gap between one set-of-fragments (each in a different stream) and the next set. When *disp* = 1, all fragments are in sync at each (possibly randomised) pulse. When *disp* is > 0, the fragments are time-scattered around the pulse centre (see below).
edpth – envelope depth:

- **1** envelope cuts down to zero
- **0.75** cuts &fraq34; of the way to zero
- **0.1** cuts only one-tenth of the way to zero, and
- **0**
  has no effect on the source.
- Once *edpth* exceeds **1**, fragments begin to stack – but not before: that is, the transposed copies added to fragment, synced at envelope peak.
- *edpth* **2** -> Stack **1**: the first transposed element iss added at full level
- *edpth* **1.5** -> Stack **0.65**: the first transposed element is added at &fraq12; (0.5) level
- *edpth* **2.5** -> Stack **1.5**: the first transposed element is added at frull level, the second at &fraq12; level, etc.

stkint – the interval of (upward) transposition in the stack, in semitones (Range: 0 to 12; the Default, which is 0, is read as octave (12). **NB: no zeros are allowed in stack breakpoint files**.
-h*seed* – If NOT zero, repeating the process with the same seed gives identical output
-m*min* – minimum duration of fragments. If zero, there is no minimum or maximum.
-i*max* – maximum duration of fragments. If zero, there is no minimum or maximum.
-r*rnd* – randomisation of the read-time in the source soundfile (Range: 0 to 1)
-p*prnd* – randomisation of pulse-time in the output (Range: 0 to 1). For both *rnd* and *prnd*, using the maximum range value (1) produces a scatter in range +- the half-duration of the pulse.
-d*disp* – the dispersal (scatter) of output timings between different streams. If *pulse* plus *prnd* gives time "P", then for *disp* = **0** all fragmentes are at "P", and if *disp* = **1** the fragments are scattered within the maximum range (+- half the duration of *pulse*).
-v*lrnd* – randomisation of the levels (i.e., volume) of the fragments (Range: 0 to 1). If *lrnd* is **0**, all fragments are at full level; if *lrnd* is **1**, the fragments will be at random levels between 0 and full volume.
-e*ernd* – randomisation of the envelope used. This is a time range in seconds. An event at "now" reads *etable* at a random time between "now" and now minus *ernd*.
-s*srnd* – randomisation of stack (Range: 0 to 1). Note that the stack value (S) is *depth*-1. If *srnd* is **0**, the stack value is S. If *srnd* is **1**, the stack value is selected at random between 0 and S.
-t*trnd* – random transposition of the fragments (Range: 0 to 1200 cents: i.e., up to 1 octave upwards. The event is pitch-randomised between the existing pitch and plus &#amp;amp; minus *trnd*.
-z – permit stacking of very short events. The Default is 'forbid', i.e., prevent clipping.
-l – for more than two output channels, the loudspeakers are assumed to encircle the listeners, with a single loudspeaker at centre-front. Setting the **-l** flag changes this to a linear array, with leftmost and rightmost speakers.

NB: *stkint*, *max*, *rrnd*, *lrnd*, *srnd* and *trnd* are **inactive** if *depth* is less than 1.

Mode 2 only:
cntre – the channel from which the stereo image spreads out
frnt – the output leading edge:

- **1** = in the central loudspeaker
- **-1** = in the loudspeaker opposite the central one
- **0** = on the bisector of the entire sound-surround space
- **2** = infinitely far away in the direction of the central speaker
- **- (2 + (*depth* * 2))** = infinitely far away in the direction opposite to centre. If the front moves, there is forward movement only, with no reversals of direction, and if it doesn't go to infinity, the event itself fades to 0 from mid-time.

depth – the maximum fraction of all output channels turned on behind the *frnt* (output leading edge)
rolloff – the degree to which the level drops as the signal is spread over several channels (Range: 0 to 1). If *rolloff* = **0** there is no drop in the level, and if *rolloff* = **1** the level is divided evenly over the number of channels in use.
-a*atten* – the level attenuation factor for distance, or sound fade-out (Range: >= 1, where 1 = linear)
-z*zpoint* – the point where the image subtends the zero angle (mono). If *zpoint* = **0** this will be at the edge of the circle, and if *zpoint* = **1** it will be at infinity.
-c*contract* – the contraction factor narrowing the distant image width. (Range: >= 1, where 1 = linear)
-l*lopnt* – the distance at which low-pass filtering is total. If *lopnt* = **0** this will be at the edge of the circle, and if *lopnt* = **1** it will be at infinity.
-f*fmix* – the factor for mixing the low-pass filtered signal into the original, with distance (Range: >=1)
-j*ffrq* – low-pass filter cut-off frequency
-k*up* – If this is NOT zero, it will be the proportion of the overall duration over which the event fades upwards from zero. This is independent of any fadeout-with-approach-to-circle.
-w*dn* – If this is NOT zero, it will be the proportion of the overall duration over which the event fades to zero. This is independent of any fadeout-with-distance-from-circle.
?*gain* – the overall gain (Range: 0 to 1). If *contract* is very much faster than *atten*, there is the rare possibility of overload as the signal is forced to mono.
-z – permit stacking of very short events. The Default is 'forbid', i.e., prevent clipping.

> All the parameters that are the same in both Modes can vary over time EXCEPT *chns*, *strms* and *seed*. However, in Mode **2**, of the additional parameters, only *frnt* can vary in time.

Understanding the FRACTURE Process

FRACTURE cuts the source into enveloped segments of a given duration and scatters them in space.

End of FRACTURE

FRAME SHIFT – Create frame patterns for multi-channel speaker setups

Usage | Modes | Parameters | Understanding | Applications

Usage

frame shift 1 *infile outfile snake rotation* [**-s***smear*]
frame shift 2 *infile outfile snake rotation1 rotation2* [**-s***smear*]
frame shift 3 *infile outfile reorient*
frame shift 4 *infile outfile mirrorplane*
frame shift 5 *infile outfile* [**-b**]
frame shift 6 *infile outfile swapA swapB*
frame shift 7 *infile outfile chaninfo gain*
frame shift 8 *infile outfile* [**-b**]

Example command line :

```
frame shift 1 in4chfile out4chfile snakeroute.txt 0.5 -s0.25
```

Modes

1 & 2 Rotate the entire frame of a multi-channel file
3 Change the channel assignment of a multi-channel file
4 Mirror the channel output around a specified mirrorplane
5 Convert between ring-numbered and bilaterally numbered *outchans*
6 Swap any pair of channels (swapA and SwapB)
7 Allows any channel, or set of channels, to be enveloped independently of the other channels
8 Converts between ring-numbered and BEAST bilateral numbering ('BEAST' = 'Birmingham Electro-Acoustic Sound Theatre', a multi-speaker diffusion system developed by Jonty Harrison et al. at Birmingham University, famous throughout Europe for outstanding performances of electro-acoustic music. Ed.)

Parameters

infile – input multi-channel soundfile
outfile – output multi-channel soundfile
snake – the route the movement through the speakers follows. In a clockwise rotation in, for example, 8 channels, input channel-1 follows the route `1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 1` etc. around the loudspeakers. To produce this (default) rotation pattern, set *snake* to zero. However, channels can 'snake' along a different route, and this route (which could vary over time) can be specified in a data file of *time snake-list* values. Here is an example file with two times and two different 'snake' routes:

```
time      snake-route
0.0       8 6 5 2 7 3 4 1
1.378234  4 1 5 2 6 7 8 3
```

Please note that the first *time* value in the data file must be zero, and subsequent times must increase.

In Mode 2, odd and even channels rotate independently: `1 -> 3 -> 5 -> 7 -> 1` etc. and `2 -> 4 -> 6 -> 8 -> 2` etc. *Snake* data directs motion around a different route.
To illustrate this, the `8 6 5 2 7 3 4 1` route above becomes one snake route around the odd entries: `8 -> 5 -> 7 -> 4 -> 8` etc. and a second snake route around the even entries: `6 -> 2 -> 3 -> 1 -> 6` etc.

rotation – the rotation-rate in cycles per second. These cycles are complete frame-rotations. Mode **2** only works with files with an even number of input channels. It rotates the odd and even channels independently. Mode **2** uses two rotation values *rotation1* and *rotation2*. Note that this is a temporal independence in rotation-rate. Negative values produce anticlockwise rotation.

-s*smear* – the extent to which channel-signals bleed into adjacent channels. (Range: 0 to 0.5. Default: 0)

reorient – list of **all** input channels in the new order required. In doing so, the order of the entries refers to channels 1, 2, 3 ... etc. in order. Thus '4 1 2 3' means:

```
1  2  3  4 channels in order
4  1  2  3 output channel destination send instructions (what you enter)
2  3  4  1 resulting contents of each channel
```

i.e.,

- Send **1st** to channel **4**
- Send **2nd** to channel **1**
- Send **3rd** to channel **2**
- Send **4th** to channel **3**

and so the resulting order is:

```
1  2  3  4 channels in order
2  3  4  1 channel contents
```

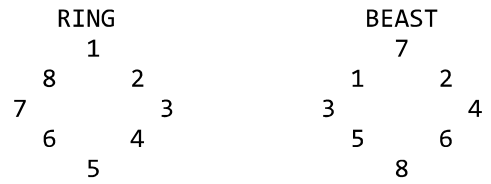
mirrorplane – the line around which the frame is (symmetrically) mirrored. Values can be any (integer) outchannel number OR any half-way position between outchannels, such as 1.5, 2.5 etc. With *N* channels, '*N*.5' lies between the N^{th} and the 1^{st} channel.

-b – (Mode **5**) convert from bilateral to ring. The Default setting is ring to bilateral. The numbering of *outchans* can be ring or bilateral. This can be illustrated with a diagram. For **7** *outchans*:

```
    RING                BILATERAL
      1                     1
   7      2              2     3
 6          3          4         5
   5      4              6     7
```

All multi-channel pan programs assume that the numbering is in the RING pattern. Therefore, use this mode to convert into and out of the bilateral format.

-b – (Mode **8**) convert from ring-numbering to BEAST bilateral numbering (Default: ring to BEAST).
BEAST bilateral numbering for **8** *outchans*:



swapA and *swapB* – the channel numbers of the 2 channels to be swapped.

chaninfo – a single channel number, or a list of channels in a text file (separated by spaces, tabs or newlines).

gain – gain in amplitude level to apply to enveloped channels: increase (> 1) or decrease (< 1). *Gain* may vary over time: define in a *time gain-value* breakpoint file.

Understanding the FRAME SHIFT Process

The **multi-channel frame** is defined by the number of output channels and how these are projected into the space. Usually here we are assuming that each channel goes to a different loudspeaker and that these loudspeakers are positioned in a ring or arc around the listener.

However, the channels may be regarded as abstract or conceptual 'positions' in any layout that the user requires. The output from these processes are simply multi-channel **wav** or **aiff** files, and the channels from this file may be sent to loudspeakers in any desired positional array, or in fact converted to locations in an Ambisonic format defined by the user (see the **Multi-Channel Toolkit**).

FRAME SHIFT is concerned with operations which affect the entire multi-channel frame, such as by *reorienting* it (e.g., by turning it though 90 degrees, **before** we begin to play the sound), or by *rotating* it (e.g., by turning it though 90 degrees, **as** we play the sound).

FRAME SHIFT Mode **1** allows us to rotate the entire **multi-channel** soundfile. Compare this with **MCHANPAN** Mode **1** which allows us to rotate a **mono** file around a multi-channel space (see the example under MCHANPAN).

In the simplest case we simply enter a speed of *rotation* in cycles per second, either positive for clockwise rotation, or negative for anticlockwise rotation. The number we enter is the number of times the entire frame rotates in 1 second. To rotate more slowly, such as 1 rotation in 4 seconds, we would enter a value less than 1 (in this example, 0.25 = ¼).

However, we can also make the channels of the multi-channel sound *snake* around the space in any pattern we choose. In a simple rotation, clockwise, channel 1 moves to 2, while 2 moves to 3, 3 moves to 4 and so on, each channel following the next in a 'snake' defined (for 8 channels) by the sequence 1 ⇒ 2 ⇒ 3 ⇒ 4 ⇒ 5 ⇒ 6 ⇒ 7 ⇒ 8 ⇒ 1 etc. We can write this as: 1 2 3 4 5 6 7 8. This is the *snake-list* part of the *time snake-list* breakpoint textfile submitted to the *snake* parameter.

However, the channels can *snake* around the space in any way you might like. For example, you could do this: 1 goes to 3, 3 goes to 5, 5 goes to 7, 7 goes to 8, 8 goes to 2, 2 goes to 4, and 4 goes to 6. We can write this as: 1 3 5 7 8 2 4 6. This sequence – the sequence in which the channels follow each other around the space – is the *snake* which you can enter as data for this process.

In addition, Mode **6** allows a simple swapping of a pair of channels in the multi-channel file.

Mode ***7*** similarly allows the level in a particular channel to be modified, without altering the level in the others.

Musical Applications

Apart from the obvious musical applications, reorientating the frame or swapping the channels in a multi-channel file may be helpful when mixing together multi-channel files in a situation where the exact spatial location of the outputs is not significant. If in a mix, the output overloads because the sounds in one (or more) of the channels are too loud when combined, reorientating the frame of one of the input files may solve this problem. If this proves impossible, or is musically not desired, altering the level of a particular event in a single channel (mode 7) may be the solution.

ALSO SEE: CHORDER – an alternative to FRAME SHIFT Mode 3 for re-ordering multi-channels.

End of FRAME SHIFT

MCHANPAN – Pan sounds around a multi-channel space

Usage	Modes	Parameters	Understanding	Applications

Usage

mchanpan mchanpan 1 *infile outfile panfile outchans* [**-f***focus*]
mchanpan mchanpan 2 *infile outfile switchdata outchans* [**-f***focus* **-m***minsil*]
mchanpan mchanpan 3 *infile outfile outchans centre spread depth rolloff minsil* [**-s**]
mchanpan mchanpan 4 *infile outfile outchans centre spread depth rolloff*
mchanpan mchanpan 5 *infile outfile antiphon outchans minsil*
mchanpan mchanpan 6 *infile1* [*infile2 ...*] *outfile antiphon outchans eventdur gap splice*
mchanpan mchanpan 7 *infile outfile pandata rolloff*
mchanpan mchanpan 8 Available only via *Sound Loom*
mchanpan mchanpan 9 *infile outfile outchans startchan speed focus* [**-a**]
mchanpan mchanpan 10 *infile outfile outchans* [**-f***focus* **-m***minsil* **-g***grouping*] [**-a**] [**-r**]

Example command line :

```
mchanpan mchanpan 1 inmonofile out8chfile panfile.txt 8 -f0.5
```

Modes

1 Move a mono sound around a multi-channel space (timed move)
2 Switch (silence-separated) mono events in a file from one channel to another
3 Spread (silence-separated) mono events stepwise from one set of channels, to another set
4 Spread source gradually from a centre, across several channels
5 Switch (silence-separated) events antiphonally between 2 specified sets of channels
6 Switch sounds (in 1 or more files) antiphonally between 2 specified sets of channels
7 Pan from one channel configuration to another, passing through surround-mono
8 Rotate a **process** around a multi-channel space (**NB:** available only via the *Sound Loom* GUI)
9 Rotate a mono soundfile around a multi-channel space
10 Switch (silence-separated) mono events randomly from one output channel to another

Parameters

infile – input soundfile: soundfiles should be Mono except for Mode 4 which accepts either Mono OR Stereo soundfiles – **but only these.**

outfile – multi-channel output soundfile

panfile – (Mode **1**) Text file with multi-channel output pan-data in value triples: *time pan-position pantype* (Also see *Files & Formats*)

- **Pan-position** values lie between (channel) 1 and a maximum number-of-channels (>= 3). Positions between 0 and 1 are also possible (see below).

- **Pantype** values can be
  - **0 = direct pan**: pan from 1 to 4 goes directly from output channel 1 to output channel 4. Direct pans must start at a single loudspeaker and position values must be non-zero integers.
  - **1 = clockwise rotation**: pan 1 to 4 goes around output channels `1 >> 2 >> 3 >> 4`
    NB: This is only a clockwise motion if the output loudspeakers are arranged in a ring with numbers ascending in a clockwise direction.
  - **-1 = anticlockwise rotation**: pan 1 to 4 (e.g. with 8 channel output) follows the path:
    `1 >> 8 >> 7 >> 6 >> 5 >> 4`
    Rotations can start/stop anywhere, so position values can be fractional.
    Values between **0** and **1** are positions between the maximum output channel (e.g. 8) and output channel 1

switchdata – (Mode **2**) Textfile containing a listed sequence of output channels

> The sound switches from one output channel to the next. If the end of the list is reached, it returns to the first output channel in the list and proceeds as before.

outchans – (Modes **1-6**) Number of channels in the output file

-f*focus* – (Modes **1-2**) If *focus* = 1.0, position (e.g.) '2' puts all signal in loudspeaker 2. If *focus* = 0.9, position '2' puts 0.9 of signal in loudspeaker 2 and the remainder in the 2 adjacent loudspeakers (1 and 3). If *focus* = 0.5, position '2' puts ½ of signal in loudspeaker 2 and the remainder in the 2 adjacent loudspeakers (1 and 3). This setting (0.5) is recommended for smooth pan movement, e.g., for Mode **1**.

-m*minsil* – (Modes **2-3-5**) For programs operating on 'events' (see below) within a soundfile. The minimum duration (ms) of zero-level signal between events to signify that one event has ended and another has begun.

antiphon – (Modes **5-6**) This parameter consists of two strings of letters, with a hyphen separator ('-'), representing the antiphonal channels.

> For example, **abcd-efgh** = *antiphon* between channels 1,2,3,4 and channels 5,6,7,8. The output channels can be divided up in any way, such as **a-bcde** OR **abc-bcdefg**, etc.

centre – (Modes **3-4**) Start-position of the spreading. This position may vary over time, so the parameter can be a single value or a *time position* breakpoint file.

spread – (Modes **3-4**) Integer value defining the channel spread of the output (far Left to far Right of centre). This parameter may also vary over time. The minimum spread is 1 in Mode **3** and 0 in Mode **4**.

depth – (Modes **3-4**) Integer value defining the maximum number of channels (to Left, and to Right) used, behind leading edges. The signal always reaches the maximum spread, but may have a hole in the middle. So, for an 8-channel file from a *centre* at 1:

- If *spread* is 7 and *depth* >=3.5 we have
  6 7 8 1 2 3 4
- If *spread* is 7 and *depth* = 2 we have
  6 7-------3 4
- If *spread* is 7 and *depth* = 1 we have
  6-----------4

The *depth* parameter cannot be larger than 2 * *spread*. If it is, the program automatically truncates it to the maximum allowed value.

-s – (Mode **3**) Optional parameter to force the output to step wider by 1 channel (to both Left and Right) on every event, as far as the value given for *spread*.

rolloff – (Modes **3-4-7**) The decrease in the amplitude level as the signal spreads over several channels. (Range: 0 to 1).

- **0** = no fall in level.
- **1** = level divided by the number of channels in use.
- Intermediate values (Range 0-1) lie between these extremes.

eventdur – (Mode **6**) The time (duration) in seconds before switching to the next *outchans* set. This parameter can vary over time by using a *time duration* breakpoint file instead of a numerical constant.

gap – (Mode **6**) The time gap (silence) switching between event sets. This parameter can vary over time by using a *time duration* breakpoint file instead of a numerical constant. **NB:** *Gap* cannot be equal to or greater than *eventdur*.

splice – (Mode **6**) The duration (in ms) of the splices used when cutting events.

pandata – (Mode **7**) Textfile with output pan-data in lines, each having *time pan-positions* value pairs.

- *Pan-positions* is a list of **all** the input channels, in any order.
- Each input channel in turn is mapped to each numbered channel in the list.
- Passing from the mapping at one time, to the mapping at the next time, sound in all channels spreads out until it is output equally from all channels, then contracts to the set of outchannel positions in the mapping at the new time.
- To force a maximal-spread at a specific time, enter time and a **list of zeros** (1 for each output channel).

  NB: Some of these processes work only with sounds consisting of **silence-separated** events.

speed – (Mode **9**) The speed of rotation in revolutions per second. This parameter can vary over time, and can be zero. Range: Range 0 - 64

grouping – (Mode **10**) With a grouping of e.g. 3, three events occur at the first (randomly selected) output channel, then three events at the second (randomly selected) output channel, and so on. Range 1 - 100 (Default 1)

-r – With this flag set, the grouping of events at output channels is randomised. For example, with a grouping of 3, either 3, 2 or 1 events may occur at the next (randomly selected) output channel. For the **-r** flag to take effect the *grouping* must be more than 1.

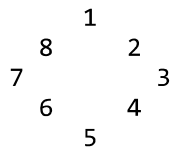
-a – With the this flag set, stepping to an adjacent output channel is avoided. Using this option makes the scattering of events in the multi-channel space more pronounced. Note that this option is only possible with *more than 4* output channels.

Understanding the MCHANPAN Process

We are used to panning sounds within a Left-Right stereo field. MCHANPAN enables us to pan sounds within a much larger speaker array.

Mode 1 is the basic mode of operation of the program. In Mode **1** the path of the multi-channel pan motion is set forth in the *panfile*.

- Panning between adjacent loudspeakers is best achieved with a *focus* of less than 1, so that some of the sound spills into adjacent loudspeakers. This avoids the sound appearing to 'fall into' each loudspeaker as it passes from one loudspeaker to another.
- How might we make a sound circle clockwise around a space? We assume here that channel *N* of the output goes to loudspeaker *N*, and that the loudspeakers are arranged clockwise `1 2 3 4 5 6 7 8` around the listener.



Circling clockwise around this ring could be achieved using the following data file for *centre*, where the 3 columns represent *time*, *position* and *direction*.

```
time    position    direction
0       1           1
3.5     8           1
4       1           1
7.5     6           1
8       4           1
```

- Using this data:
  1. The sound begins (time 0, line 1) in channel 1, moving clockwise (last '1' in line 1) until (at time 3.5, line 2) it reaches loudspeaker 8. This movement has taken 3.5 seconds.
  2. It then proceeds (clockwise, the '1' at the end of line 2) from loudspeaker 8 to loudspeaker 1 (which is the next loudspeaker in the ring of 8) where it arrives at time 4 (line 3) after another half a second, thus completing a full circuit in 4 seconds.
  3. Next it takes another 3.5 seconds to move clockwise from loudspeaker 1 to loudspeaker 6, and then moves all the way around to loudspeaker 4 in the next half a second.
  4. Note how the perceived speed of the pan movement is affected both by the distance to be travelled and the time in which this is to take place.

- Recall the recommendation in the parameters section to use a *focus* of 0.5 to achieve a smooth movement between]the speakers.

There is no specific way to define the centre of the speaker ring as a spatial location. For example, 1.5 will be half-way between speaker 1 and speaker 2 along the ring of speakers. You can, however, move across the centre with, for example, a pan from speaker 1 to speaker 6, whichever ones are physically opposite in your speaker layout. But note the expanding-contracting effect produced by Mode **7**.

Mode 2 handles 'events'. Events are defined as sounds in the input file separated by silences. This implies that the input soundfile needs to be a soundfile containing events actually separated by silence. A switch to a new output channel occurs at the entry of a new event in the input file, meaning that each subsequent sound event just 'turns up' at another speaker location without an actual pan movement between the speakers – the illusion of a pan movement may take place if the time between events is very fast. The list of channels is given in the *switchdata* textfile, such as:

```
1  5  2  6  8  4  3  7  1
```

The channel numbers can be separated by a space, a tab or a newline. The channel hopping finishes when there are no more events, so the number of events in the source soundfile should match the number of speaker locations in the *switchdata* file. If there are more events than speaker positions in the *switchdata* file, as there are likely to be if a granulated sound is used, the output wraps repeatedly around the defined pattern of the speaker positions.

There are a variety of ways to create a soundfile with actual silences between events. This is an important consideration, as a number of the multi-channel effects are dependent on soundfiles of this kind. Mode **3** below is one example.

- **HOUSEKEEP EXTRACT** Mode **1** for the gated extraction of sounds into separate soundfiles.
- **SFEDIT JOIN** to splice together separate soundfiles, placing a short, silent soundfile between each of these soundfiles – see **SYNTH SILENCE** for synthesising a silent soundfile.
- Create a soundfile with an amplitude tremolo (**ENVEL TREMOLO**) and then 'corrugate' the soundfile (**ENVEL WARP**, Mode **11**).
- **MODIFY BRASSAGE** or ***GrainMill*** with (grain) *density* set at less than 1, which will produce intergrain silences. Quite small gaps should still work with MCHANPAN.

Mode 3 spreads events left and right from a defined centre. This mode is a little hard to understand. I think Trevor gave me the key when he said that it does not produce a gradual panning movement from the defined centre to the other speakers. Rather, each new sound finds that it needs to be at such and such speakers. 'Each new sound' means that the input soundfile is made up of a number of sound events. Events are defined as sounds in the input soundfile separated by silences. An increase in the spread only occurs at the entry of a new event in the input file. Note the range of parameters that need to be set in this mode: *centre*, *spread*, *depth*, *rolloff*, and the optional **-s** flag. The *depth* parameter is particularly important in that it can be used to suppress sounds at the defined *centre* (see below).

- The *centre* is defined as the principal location of the sound being panned (i.e., on which channel, or on which loudspeaker, it is centred).
- The *centre* itself can be changing with time, so that, for example, the sound image may circle around the loudspeakers.
- The *spread* defines how wide the sound image is, around that *centre* (i.e. how much subsequent events spill onto loudspeakers adjacent to the defined *centre*).

- The *spread* can also change over time, so that, for example, a sound image clearly focused in a loudspeaker gradually bleeds onto the adjacent loudspeakers.
- Imagine that the image is *centred* in loudspeaker 4, and the *spread* is 5: the sound image will spread out over the five loudspeakers 2,3,4,5 and 6 (centred on loudspeaker 4).
- We can now create a 'hole in the middle' effect, so that the *centre* of the sound image (at loudspeaker 4) is suppressed. We can achieve this effect using the *depth* parameter. If the depth is sufficiently large (in this case, at least 2.5, i.e. 5 ÷ 2), then we will hear the full image we expect over the five loudspeakers. This is because the *depth* of the image on each side of the *centre* is 2.5, so that the image will cover 2.5 + 2.5 = 5 loudspeakers). However, if the *depth* is less than this (say 2), then the image will cover only four (2 + 2) loudspeakers, and there will be no signal on the central loudspeaker (number 4), even though it is the (now empty) centre of the sound image.

Also see FRAME SHIFT, Modes **1** and **2** for another way to define a movement from channel to channel.

Mode 4 makes use of an input source that can be a continuous sound, and the sound spreads gradually over time, interpolating between one spread value and the next. The parameters are almost the same as Mode **3** except that there is no *minsil* parameter and no **-s** option.

Mode 5 makes use of an input source that contains silences. Here the material being antiphonated is events separated by silence within a single soundfile. In this case, the set of outchannels in use changes only with the entry of a new event in the input file. Note that, because silence is involved, the *minsil* parameter is used.

Mode 6 is similar to Mode 5, but the input sounds need not contain silences, and nore than one input sound can be used. The set of *outchans* in use changes after the time specified by *eventdur*.

If there is more than 1 input file, the next file is selected at each such antiphonal switch, and if the end of the input files is reached, the first file in the list is chosen again, and processing continues as before.

Silence can also be inserted prior to an antiphonal switch by using the *gap* parameter.

Mode 7 is the nearest we can get, outside Ambisonics, to making all the sounds (on all channels) move into the (listener) centre of the space, and then back out (to possibly some other orientation of the channels). In reality, on a ring of loudspeakers, all the sounds expand from the channel they are in to fill all the available channels, then contract onto a (possibly new) channel, as specified in the data file.

Suppose we had a *pandata* file containing the following times and pan-positions:

```
time  loudspeaker
 0.0  1
 4.0  4
 8.0  6
12.0  2
```

We start with the sound concentrated on loudspeaker 1. It then expands outwards to *all* channels, but then contracts again so that at *time* 4.0 seconds, it is concentrated on loudspeaker 4. From here it expands outwards again to all speakers, then focusing at *time* 8.0 seconds on speaker 6, etc. coming to rest at *time* 12.0 at speaker 2. The psycho-acoustics of this process are still being researched.

Mode 8 is available only on *Sound Loom* and in *Soundshaper* – i.e., not on the command line. It is a composite function that runs a CDP process within a multi-channel pan context.

Mode 9 In Mode **1**, the pan data file allows you to move the source sound clockwise, anticlockwise, or directly between non-adjacent output channels, and you can change the type or direction of motion at any time, by specifying a change of motion type in the panning data. Mode **9** allows only clockwise or anti-clockwise rotation (but not both), and **the only parameter is the speed of rotation, which may vary over time**. Mode **9** is simpler to use if only (uni-directional) rotation is desired. Use Mode **1** to get to a specific channel at a specified time.

Mode 10 Similar to Mode **2**, but the events are switched *randomly* between output channels. All the output channels are used up before a new random-permutation of the output channels is begun. It is possible to group the events at each channel: for example, 3 events occur at each (randomly selected) output channel. Also see parameter *grouping*.

It is possible to randomise this grouping of events: see the **-r** flag.
It is also possible to avoid stepping to an adjacent output channel; see the **-a** flag.

Musical Applications

The musical results that can be achieved with MCHANPAN can be reviewed by looking through the **list of modes** for the program.

Panning in a multi-channel context is a fundamental procedure. Please refer to Panning and multi-channel Mixfiles below for more information about how to set up a multi-channel *mixfile* while using (already) panned multi-channel soundfiles.

End of MCHANPAN

MCHANREV – Create multi-channel Echoes or Reverb from a mono soundfile

Usage

mchanrev mchanrev *infile outfile gain roll_off size count outchans centre spread*

Example command line:

```
mchanrev mchanrev four fourmchrev 0.5 0.6 0.5 100 4 1.5 2
```

Parameters

infile – input sound file (must be mono)
outfile – output sound file
gain – Gain to apply to input signal before processing (0-1, Default: 0.645654)
roll_off – Rate of loss of level with successive echoes (Range: 0.000031 to 1.0, Default: 1)
size – Multiplies average time between echoes: (Default time 0.1 secs)
count – Number of stadium echoes (1- 1000, Default (max): 1000)
outchans – Number of channels in the output sound
centre – Centre position of reverberated image, amongst the output channels

- With *N*-channel output, range is 1 to *N*
- *centre* can be a fractional value
- Values below 1 lie between outchannel 1 and outchannel N.

spread – Number of output channels over which echoes are spread. (Range: 2 to *N*)

- *spread* can be a fractional value
- For example, with outchans = 8, *centre* = 4, and *spread* = 4.5. the reverb image is spread between positions 1.75 (4 - half of 4.5) and 6.25 (4 + half of 4.5)

  Note that the number of echoes needs to be larger, relative to the same process in a stereo space, to achieve an equally 'dense' result.

Understanding the MCHANREV Process

MCHANREV is the multi-channel equivalent of **MODIFY REVECHO** Mode **3**, i.e., STADIUM ECHO.

The *size* parameter controls the amount of time there is between 'echoes'.

- With *size* of 1 (sec.) or more, individual echoes in the output are likely to be heard, depending on the length of the input soundfile.
- If the value for *size* is longer than the duration of the input soundfile, you will hear the input sound repeating (in the different speaker range defined by *spread*, gradually fading away to nothing. (I have not been able to find an upper limit for *size*.
- With smaller values for *size*, and large values of *count* (the number of echoes), reverberation will be produced.

- Larger values for *size* in connection with larger values for *count* means more overlap, and this will increase the amplitude, which may not be fully compensated for by the program. You may therefore need to apply a *gain* reduction (values increasingly less than 1.)

Note, therefore, the interplay between *size*, *count* and *gain*, which may require some trial and error to get a good signal level in the output without producing distortion.

The Default *roll-off* value is 1.0. This is the maximum decrease in amplitude from one echo to the next. As the value for *roll-off* decreases, there are higher amplitudes in the successive echoes / reverberations, meaning more chance of amplitude overload. This therefore another parameter to handle carefully to get the best signal level without amplitude overload.

The echoes or reverberant output can be centred at a particular channel in the multi-channel space, using the parameters centre, and *spread*. These will determine where the centre of the sound image is located and how wide the reverberant image is: how many adjacent output channels it spreads across.

If the image is spread over all the output channels, the reverberant image will fill the multi-channel space. The wider the output *spread*, the more echoes (larger *count*) and higher echo density (smaller *size*) are needed to produce a similar reverberation result.

Musical Applications

There are a number of other ways to create reverberation in a multi-channel space. For example,

1. The individual channels of the multi-channel file may be extracted HOUSEKEEP CHANS Mode **2**, extracting all the channels of a multi-channel file) to separate (mono) soundfiles.
2. Then standard reverberation MODIFY REVECHO Mode **3**) can be applied to each, and, because Mode **3** produces a stereo *outfile*,
3. the reverberant outputs (which are stereo) are now reduced to mono with HOUSEKEEP CHANS Mode **4** (converts stereo to mono), and finally,
4. these mono files are recombined into a multi-channel file with SUBMIX INTERLEAVE. (NB: The maximum number of channels in the output of SUBMIX INTERLEAVE is 4.) [AE: can two 4-channel soundfiles be joined to make an 8-channel soundfile?]

In this way, the sound on each channel can be provided with its own unique reverberation characteristics.

Alternatively the stereo reverberated files can themselves be remixed to a multi-channel file in such a way that the stereo images are centred on the channels from which they derive, using MCHSTEREO.

On The *Sound Loom* the whole process of channel separation, reverberation, and channel recombination can be accessed in one sweep by applying the **Bulk Processing** procedure to a **single** multi-channel file. Then the channel separation and recombination are carried out automatically.

Achieving distance effects in a multi-channel context requires considerable thought and use of resources. T Wishart: "You can control distance (from the listener) effects in Ambisonics, but the programs here deal only with the plane of the loudspeakers, and distance effects (still) have to be created using loudness, high-frequency roll off, and (to simulate specific types of environment) reverberation effects, as in the stereo case."

ALSO SEE: **FASTCONV**, which will produce dramatic reverberation effects when a sound is convolved with, for example, a spring reverb impulse.

End of MCHANREV

MCHITER – Iterate the input sound in a fluid manner, scattering around a multi-channel space

Usage

mchiter iter 1 *infil outfil outchans outduration* [**-d***delay*] [**-r***rand*] [**-p***pshift*] [**-a***ampcut*] [**-f***fade*] [**-g***gain*] [**-s***seed*]
mchiter iter 2 *infil outfil outchans repetitions* [**-d***delay*] [**-r***rand*] [**-p***pshift*] [**-a***ampcut*] [**-f***fade*] [**-g***gain*] [**-s***seed*]

Example command line to create scattered itertions :

```
mchiter iter 1 inmonosnd outmultichansnd 4 10
```

Modes

1 Specify the duration of the output sound, as *outduration*
2 Specify the number of iterated *repetitions* of the source sound in the output sound

Parameters

infile – input mono (only) soundfile
outmcfile – output multi-channel soundfile
outchans – number of channels in the output file
outduration – duration of the output sound. (Must be greater than that of the input)
repetitions – number of iterations of the input sound in the output
delay – (average) delay between the iterations: must be less than or equal to the duration of the input sound
rand – randomisation of the delay time. Range: 0 to 1 (Default 0)
pshift – maximum of the random pitchshift of each iteration, in semitones. Range: 0 to 12 (Default 0)
ampcut – maximum of the random amplitude fluctuation (reduction) at each iteration. Range: 0 to 1 (Default 0)
fade – (average) amplitude fade from one iteration to the next. Range: 0 to 1 (Default 0)
gain – overall Gain. Range: 0 to 1 (Default 0). Using the special value 0 (default), gives the best guess for no output distortion.
seed – the same seed-number will produce identical output on rerunning the process. The special value 0 produces a different random sequence every time the process is run. (Default 0).

Understanding the MCHITER Process

This is a 'scattering' algorithm, similar to EXTEND ITERATE, but with the addition of multi-channel spatial scattering. Notice that the amplitude of the sound is diminishing with each repetition.

Sound is distributed randomly among the speakers of the multi-channel rig for a given length of time (*outduration*). The various parameter options make it possible to vary the repetitions in a number of ways.

MCHITER ITER extends a sound by iterating it. This can be a simple repetition of (a part of) the source, as in EXTEND LOOP, but more typically, each iteration can be (slightly) transposed or varied in level, to produce a more naturalistic iterated output.

In addition the iterated copies of the sound are distributed at random to the output channels of a multi-channel space. Each channel of the output space is visited before a new random permutation of the output channels is begun.

Musical Applications

MCHITER ITER can be used to distribute a small sound over an entire multi-channel space. See EXTEND ITERATE for more information about the iteration process.

ALSO SEE: **MCHZIG** and **MCHSHRED**. For a more controlled spatial distributon (e.g. over less channels of the space, or in a way which varies through time), see **TEXMCHAN**.

End of MCHITER ITER

MCHSHRED – Sound is cut into random segments which are then reassembled in random order within the original duration

Usage

mchshred shred 1 *infile outmcfile repeats chunklen scatter outchans*
mchshred shred 2 *infile outmcfile repeats chunklen scatter*

Example command line to create randomised reordering in several channels :

```
mchshred shred 1 inmonofile outmcfile 2 0.8 1 8
```

Modes

1 Shreds a *mono* input file to a multi-channel output
2 Shreds a multi-channel input file, reorienting shredded units in the multi-channel space

Parameters

infile – input soundfile, Mono in Mode **1**, Multi-channel in Mode **2**
outmcfile – output multi-channel soundfile
repeats – number of repetitions of the shredding
chunklen – average length of the chunks to cut and permute
scatter – randomisation of the cuts Range: 0 to *K* (Default = 1):

- If *K* = total number of chunks (the input sound duration divided by the average chunk length), then the randomisation can vary from 0 to *K*
- If the *scatter* = 0, the elements will be reordered on each pass, but not further shredded as the cuts will always occur at the same places in the original sound.

outchans – number of channels in the output sound. Mode **1** shreds a mono input into a multi-channel output. Mode **2** shreds a multi-channel input.

Understanding the MCHSHRED SHRED Process

MCHSHRED SHRED shreds a sound in the same manner as **MODIFY RADICAL**, Mode **2**.

In addition:

- In mode **1**, the shredded elements are randomly distributed over a multi-channel space.
- In mode **2**, each of the shredded elements has its frame randomly reoriented in the multi-channel space. For example, input channels 1-2-3-4-5-6-7-8 may end up in output channels 6-7-8-1-2-3-4 respectively.

Musical Applications

See **MODIFY RADICAL**, Mode **2** for more information about the shredding process.

MCHSHRED can be used to distribute the fragments of a sound over an entire multi-channel space. For a more controlled spatial distributon, such as over less channels of the space, or in a way which varies through time, see **TEXMCHAN**.

ALSO SEE: MCHITER and **MCHZIG**.

End of MCHSHRED SHRED

MCHSTEREO – Place several stereo files in a multi-channel output space

Usage

mchstereo mchstereo *infile1* [*infile2 ..*] *outfile ochandata ochans pregain* [**-s**]

Example command line to :

```
mchstereo mchstereo instereosnd1 instereosnd2 outmcsndfile outchanneldata.txt 8 0.9 -s
```

Parameters

infile1 – first stereo input soundfile
infile2 – second stereo input soundfile, etc.
outfile – output multi-channel soundfile
ochandata – centre position of each stereo input in the outchannel frame; example (with two stereo input soundfiles): `2 5`

- This is a list of output channel-numbers.
- There must be one number for each input file.
- For example, with 3 input stereo sounds to an 8-channel output, `1 3 6` means
  - The 1st stereo image is centred at channel 1.
  - The 2nd stereo image is centred at channel 3.
  - The 3rd stereo image is centred at channel 6.
  - Note that some channels may have a mix of e.g., some of the Left of one sound and the Right of another, and *v.vs.* It is probably best to use your ears to decide if you like the mix, rather than try to work out overlaps on paper. Multi-channel soundfile display will also show what is happening.
  - However, note the effect of the **-s** flag (see below).

ochans – number of channels in the output file (Range 2 to 16)
pregain – gain applied to input sounds (Range 0 to 1)
-s – shifts the stereo images half-a-channel to the right. Therefore, in the example above for *ochandata*:

- The 1st stereo image is centred **between** channels 1 and 2.
- The 2nd stereo image is centred **between** channels 3 and 4.
- The 3rd stereo image is centred **between** channels 6 and 7.

Understanding the MCHSTEREO Process

MCHSTEREO allows stereo files to be treated almost like mono files when they are injected into a multi-channel frame. The two channels of stereo remain adjacent, either in adjacent channels of the multi-channel output, or offset from 2 adjacent channels of the multi-channel output. (Note that this may not well-preserve any detailed stereo image, as the angle at which loudspeakers are placed in a multi-channel array may not correspond to the ideal stereo orientation.) It may be used to associate stereo-processed versions of the individual (mono) channels of a multi-channel soundfile with those original channels, in a new multi-channel mix.

Musical Applications

MCHSTEREO allows us to use processes with stereo output (e.g., reverberation) on the individual (mono) channels of a multi-channel soundfile, and then associate those stereo outputs with the original channel sources. To do this we would make a separate multi-channel mix of the stereo outputs, using MCHSTEREO, then mix this with the original multi-channel source, using NEWMIX, ensuring that the orientations of the two multi-channel files correspond with one another.

This might be used, for example, to add a different reverberant space to each of the channels of a multi-channel sound.

End of MCHSTEREO

MCHZIG – Extend by reading back and forth in the soundfile, while panning to a new channel at each 'zig' or 'zag'

Usage

mchzig zag 1 *infile outmcfile start end dur minzig outchans*
mchzig zag 2 *infile outmcfile timefile outchans* [**-s***splicelen*] [**-a**]

Example command line to create randomized zig-zagging between channels:

```
mchzig zag 1 inmonosnd outmcsnd 0.031 4.8 20 0.4 8
```

Modes

1 Random zigzags, output going from the beginning of the input sound to its end, with zigzags between *start* and *end*
2 Zigzag between a list of times supplied by the user

Parameters

infile – input mono soundfile
outmcfile – output multi-channel soundfile
start – time in sec. at which to begin the zigzags (Minimum value is 0.031000 sec.)
end – time in sec. at which to end the zigzags (Maximum value is the length of the input soundfile.)
dur – the (minimum) total duration of the output sound to be generated
minzig – the shortest acceptable 'zig' or 'zag'
minzag – the longest acceptable 'zig' or 'zag'
outchans – number of channels in the output sound
splicelen – length in milliseconds of the splices used to join the segments. (Default: 25 ms)
seed – entering a positive number here will generate a random sequence which can be replicated if the process is run again using the same number. The value 0 generates a different random sequence on every pass. (Default: 0)
timefile – the name of a textfile containing a list of times between which the 'zigs' and 'zags' take place. Each step must be > (3 * *splicelen*)
-a – avoid pans between adjacent channels

Understanding the MCHZIG ZAG Process

Similar to **EXTEND ZIGZAG**, the sound is extended by reading read back and forth within the soundfile, but in addition, during each 'zig' or 'zag', the sound is randomly panned to a new channel of the output space.

Hence the sound is in constant motion over the multi-channel space. The next channel to pan towards is chosen at random, and all the output channels are visited before a new random permutation of those channels is begun.

Musical Applications

See **EXTEND ZIGZAG** for more information about the zigzag process. MCHZIG ZAG can be used to extend a sound and distribute it over an entire multi-channel space. Also compare with **MCHITER ITER** in which amplitude reduction takes place. For a more controlled spatial distributon, such as over less channels of the space, or in a way which varies through time, see **TEXMCHAN**.

End of MCHZIG ZAG

MTON – Convert a mono sound to a multi-channel sound with identical signal in all channels

Usage

mton mton *infile outfile outchans*

Example command line to create a multi-channel soundfile from a mono input:

```
mton mton inmonosnd outmcsnd 4
```

Parameters

infile – input mono soundfile
outfile output multi-channel soundfile
outchans – the number of channels in the output soundfile (Range 2-16)

Understanding the MTON Process

MTON is the multi-channel equivalent of HOUSEKEEP CHANS Mode **5** (convert mono to stereo). The output is a multi-channel file where all channels are identical.

Musical Applications

A mono source can be generated in multi-channel format, then possibly modifed (e.g., with **FLUTTER**) and added to a multi-channel mix, providing an all-encompassing ambience.

End of MTON

MULTIMIX CREATE – Create a multi-channel mixfile

Usage | Modes | Parameters | Understanding | Applications

Usage

multimix create 1 *infile1 infile2* [*infile3..*] *mixoutfile*
multimix create 2 *infile1 infile2* [*infile3..*] *mixoutfile*
multimix create 3 *infile1 infile2* [*infile3..*] *mixoutfile timestep*
multimix create 4 *infile1* [*infile2 infile3..*] *mixoutfile balance*
multimix create 5 *infile1* [*infile2 infile3..*] *mixoutfile stage wide rearwide rear*
multimix create 6 *infile1 infile2* [*infile3..*] *mixoutfile*
multimix create 7 *infile1 infile2* [*infile3..*] *mixoutfile ochans startch* [**-s***skip* **-t***timestep*]
multimix create 8 *infile1 infile2* [*infile3..*] *mixoutfile ochans*

Example command line to create a multi-channel mix file:

```
multimix create 1 horn fourS ramzrota hfrmx.mmx
```

Modes

1 Create a mix where all files start at time zero. (Sound design amalgamation of sounds.)

- Input files can have any number of channels, including a mixture of different numbers of channels. For example the inputs for the example command line above are mono, stereo and 4-channel.
- Outchannel count is set to the maximum channel-count among the input files.

2 Create a mix where each file enters where the last ended. (End-to-end mixing.)

- Input files can have any number of channels.
- Outchannel count is set to the maximum channel-count among the input files.

3 Create a mix where each file enters a given *timestep* after the start of the previous sound (Constant time interval between entries.)

- Input files can have any number of channels.
- Outchannel count is set to the maximum channel-count among the input files.

4 Distributes a mono or stereo input to two stereo pairs, one narrow and one wide, in an 8-channel output frame, with *balance* parameter. (Stereo assignments.)

- Input Stereo-Right (or mono) goes to 1 and 2.
- Input Stereo-Left (or mono) goes to 8 and 7
- All input files are sent to the **same** set of output channels.

5 Distributes mono or stereo input over 8 loudspeakers, with extra level-control parameters. (8-wide distribution.)

- Input Stereo-Right (or mono) goes to 1, 2, 3 & 4.
- Input Stereo-Left (or mono) goes to 8, 7, 6 & 5.
- All input files are sent to the **same** set of output channels.

6 Distributes *N* mono files, in order, to *N* successive output channels. (Distribution in ascending order.)

- Output Channel count = number of input files.

7 Distributes *N* mono files, in order, to *K* successive output channels. (Channels may exceed or be less than inputs.)

- Output Channel count (*ochans*) can be **greater than** the number of input files, in which case the remaining channels are assigned no signal.
- Output Channel count (*ochans*) can be **less than** the number of input files, in which case some signals will be assigned to the same channel
- For example, 6 files into a 4-channel output file gives
  - inmonosnd1-to-1
  - inmonosnd2-to-2
  - inmonosnd3-to-3
  - inmonosnd4-to-4
  - inmonosnd5-to-1
  - inmonosnd6-to-2
- Also note *startch*, *skip* and *timestep* parameters.

8 Each file in the mix starts at zero, leftmost channel to *outchan* 1.

- In Mode **1**, The number of output channels in the mixfile output is determined by the maximum number of channels in any of the input files.
- In Mode **8**, The number of output channels in the mixfile output is explicitly specified.

Parameters

infile1 – name of first mono or stereo soundfile to be mixed
infile2 – name of second mono or stereo soundfile to be mixed
[*infile3..*] – names of any additional mono or stereo soundfiles to be mixed

> All modes accept MONO input soundfiles. Modes **4**, & **5** *also* accept stereo input soundfiles, but not soundfiles with more than 2 channels.

mixoutfile – output multi-channel mixfile, with the extension **.mmx** (Also see *Files & Formats*)

timestep – (Mode **3**) time in seconds between entry of each input file in the mix

balance – (Mode **4**) proportion of the stereo signal assigned to the wider of the 2 stereo pairs

stage – (Mode **5**) level of signal assigned to front loudspeaker pair (8,1)
wide – (Mode **5**) level of signal assigned to front-wide loudspeaker pair (7,2)
rearwide – (Mode **5**) level of signal assigned to rear-wide loudspeaker pair (6,3)
rear – (Mode **5**) level of signal assigned to rear loudspeaker pair (5,4).

ochans – (Modes **7-8**) number of channels in the output soundfile generated by the *mixoutfile*. Range: 2 to 16.

startch – (Mode **7**) In processes where a list of (mono) input sounds is assigned to consecutive channels of a multi-channel output, this parameter defines which of the multi-channel file's channels will get the first sound in the list. Subsequent files are assigned to higher channels, wrapping around to 1 when the highest output channel is reached.

-s*skip* – (Mode **7**) In processes where a list of (mono) input sounds is assigned to consecutive channels of a multi-channel output, this parameter defines the **channel-step** between channel assignment in the output multi-channel soundfile. For example, if the first input soundfile is assigned to output channel 1, and the *skip* is 2, the next output channel to be assigned will be 1 + 2 = channel 3.

-t*timestep* – (Mode **7**) In the mode where a *mixfile* is created with a *timestep* between the entry of each input soundfile, this parameter defines that *timestep*. For example, with a *timestep* of 1.5, the 1st input file will begin at time 0, the 2nd at time 1.5, the 3rd at time 3, etc.

Understanding the MULTIMIX CREATE Process

This process is similar to **SUBMIX DUMMY**, but with additional options for creating a multi-channel mixfil).

The multi-channel mixfile has a slightly different format to standard CDP mixfiles – see ***Files & Formats*** or **NEWMIX MULTICHAN** below, or **M-C Mixfiles** in this document for a fuller explanation.

The mixfile for multi-channel work is the same as standard CDP mixfiles except for the following:

1. There is an extra initial line that states the number of output channels
2. On ensuing lines, input and output channels are numbered 1, 2, 3, etc.
3. Routing of input to output is indicated by **inchan colon outchan**, e.g., `1:4` (no spaces) sends input channel 1 to output channel 4
4. The levels on these channels are in the range -1 to 1
5. An input channel must be routed to an output channel, with a level: e.g., `1:1 1.0 2:4 0.5`, meaning: input channel 1 to output channel 1 with level 1.0, input channel 2 to output channel 4 with level 0.5
6. You can route an input channel to several output channels: e.g., `1:1 0.5  1:2 0.3  1:4 0.7` etc.
7. You must take care with levels, where more than one input goes to the same output. As a rule of thumb, the sum of the levels should not exceed 1.0.

Using MULTIMIX is the easiest way to produce a multi-channel *mixfile* of the correct format. The file can then be edited, either as a textfile or, on the *Sound Loom*, using the **QikEdit** facilties, where many new functions have been added to automatically manipulate the data lines in the *mixfile*.

We have added several examples of multi-channel mixfiles in the **MULTIMIX Mixfiles Appendix** below, along with a discussion about moving on to use the *mixfile* with NEWMIX MULTICHAN, and some notes about facilities available on the *Sound Loom* GUI.

Musical Applications

Generating multi-channel *mixfiles* in the correct format is not always straightforward. Even when MULTIMIX will not generate exactly the file you need it is usually easier to create a dummy file here and subsequently edit it (in a text editor or on the *Sound Loom's* **QikEdit** facility) than to write out a multi-channel *mixfile* from scratch in the correct format.

End of MULTIMIX CREATE

NEWMIX MULTICHAN – Mix from a multi-channel mixfile to give a multi-channel soundfile output

Usage

newmix multichan *mixfile outsndfile* [**-s***start*] [**-e***end*] [**-g***attenuation*]

Example command line to perform a multi-channel mix:

```
newmix multichan mchmixfile.mmx mchoutsndfile
```

Parameters

mixfile – input multi-channel mixfile, with the extension **.mmx**
mchoutsndfile – output multi-channel soundfile
-s*start* – time in seconds at which to start mixing within the *mixfile*. Defaults to 0.0.
-e*end* – time in seconds at which to stop mixing within the *mixfile*. Defaults to the end of the complete mix.

Note that the *start* and *end* parameters are intended for mix testing purposes only, and mixes made with non-default values will switch-on and/or cut-off abruptly. If you want to keep output from such a text mix, you should use ENVEL TOPANTAIL2 to smooth the beginning and the end.

-g*attenuation* – the level of the output mix. Defaults to 1.0 (no attenuation).

Understanding the NEWMIX MULTICHAN Process

Multi-channel mixfiles can also be written as text files, and are similar to the Standard CDP mixfiles. However, it is recommended that these files be generated using MULTIMIX CREATE (and subsequently edited if necessary) to ensure that they are in the correct format. For an overview of the format of **.mmx** files, see ***Files & Formats***. As with standard mixfiles,

- The first entry in a line is the name of an input sound (with a path unless it is in the current directory).
- The second entry in the line is the time in the mix at which it starts
- The third entry in the line is the number of channels in the input sound. The format for the channels for NEWMIX MULTICHAN has been changed to accommodate both a larger number of channels and channel group routing.
- You can examine the output mixfiles of MULTIMIX CREATE to get more examples of multi-channel mixfiles.

However, in the following example of a multi-channel mixfile you will notice some differences:

```
8
[soundfile start chans routing level routing level ...]
sound1.wav 0.0 2 1:1 0.5  2:2 0.5
sound2.wav 0.0 2 1:3 0.5  2:4 0.5
sound3.wav 0.0 4 1:5 0.25 2:6 0.25 3:7 4:8 0.25
sound4.wav 0.0 2 1:1 0.1  1:2 0.03 1:3 0.9 1:4  0.2 1:5 0.2 1:6 0.15 1:7 0.5 1:8 0.32
```

The main differences from standard mixfiles are:

- There is an extra initial line that gives the number of output channels that there will be in the resultant multi-channel soundfile.
- On ensuing lines, input and output channels are numbered 1,2,3 etc. ...16
- Routing of input to output is indicated by 'inchan:outchan' (no spaces) e.g. 1:4 sends input channel 1 to output channel 4
- An input channel must be routed to an output channel, **with a level**: e.g. 2:4 0.5 (input 2 is sent to output 4 with level 0.5)
- The levels on these channel-routings are in the range 0 to 1.
- You can route an input to many outs as with sound4.wav in the mixfile above.
- You must take care with levels where more than 1 input goes to the same output.

Panning and multi-channel Mixfiles: We need to consider how these channel assignments in the *mixfile* relate to a multi-channel input soundfile *that has already been panned* with MCHANPAN.

T Wishart writes: "If a sound has already been panned around the multi-channel space (using MCHANPAN) the panned-sound output consists of a set of discrete signals assigned to the *N* channel outputs of the multi-channel sound. (The movement of the sound is the result of the changing levels in the individual channels of the multi-channel output)."

- **Routing –** To preserve the 'sense' of the panning, when mixing, routing should be ***N*:*N*** i.e., `1:1 2:2 3:3 4:4 5:5` etc. so that the orientation of the signal is not changed. **It is very important to remember this when using panned multi-channel soundfiles in a *mixfile*.**
- **Shifting the whole pattern –** The entire panning pattern could be shifted to a different position. For example, shifted along by *N* loudspeaker positions, by making the assignment **1:1+*N***. Thus, to shift the pattern by 4 loudspeakers, use the assignment `1:5 2:6 3:7 4:8 5:1 6:2 7:3 8:4` (with the output channel values wrapping back to 1 when the maximum output channel is reached). This transformation preserves the original rotation, but starts it in a different place (i.e., at output channel 5).
- **Mirroring –** Alternatively, the panning data can be mirrored (e.g. a clockwise rotation can be made into an anticlockwise rotation) by reversing the output channel order e.g., (for 8 channels) you would put `1:1 2:8 3:7 4:6 5:5 6:4 7:3 8:2`.

  (Note that this transformation will also convert an anticlockwise rotation into a clockwise rotation – this needs a little thought - these channel assignments all take place at the **same time**, before the sound begins to play. We are not 'panning' (actively moving) the sound here, merely reassigning the channels on which the sound will by played).

- **Reorientation –** This reorientation or mirroring can also be achieved directly on the multi-channel sound itself, using the **Reorient** option in **FRAME SHIFT**.

Musical Applications

Note that the program **MULTIMIX CREATE** can be used to automatically generate multi-channel mixfiles of the correct format for use by **NEWMIX**, and this is the recommended approach to creating these files. For more information and various examples of MULTIMIX *mixfiles*, please see the section **MULTIMIX Mixfiles (Appendix 1)**.

It may be useful to summarise the various roles of NEWMIX MULTICHAN, MULTIMIX CREATE and MCCHANPAN:

- NEWMIX MULTICHAN (this function) does the mixing, using a multi-channel mixfile (**.mmx**). This **.mmx** file can be written by hand with a text editor, or your could use the facilities of MULTIMIX CREATE to make it (recommended). For its format, see *Files & Formats*, as well as **M-C Mixfiles** (Appendix 1) in this document. The distinguishing feature of a multi-channel mixfile is that it includes routing information for each channel, expressed in the format IN-CHANNEL:OUT-CHANNEL, as in 2:7, i.e., channel-2 of the input soundfile becomes channel-7 in the output multi-channel mixfile. For example, where the input to MULTIMIX CREATE is a mono file, 'IN-CHANNEL' will be '1' and refer to that particular sound. It can be routed to more than one 'OUT-CHANNEL'. Modes **4** & **5** also accept stereo input, so 'IN-CHANNEL' can be '1' or '2'. *Note that all this is simply a speaker assignment – it does not imply any pan movement.*

- MCHANPAN is the function that produces pan and must be applied to the input MONO soundfile(s) *before* mixing (only Mode **4** & **5** allows a Stereo infile). Because panning is applied before mixing, the level changes that create the illusion of movement are already in place in any panned-soundfile before it is used in the mix. The data in the panfile used to generate the panning motion has *time position direction* information (example) . **When using these already panned soundfiles as inputs to NEWMIX MULTICHAN, *remember to do a straight routing for all channels, i.e., 1:1, 2:2, 3:3 ... 8:8***. This preserves the movement pattern of the already panned input (multi-channel) soundfile.

- It would seem that the crucial aspect of this is keeping track of what sound material is on each channel before starting to mix. NEWMIX MULTICHAN can accept mono, stereo or 2+ channel inputs, and the latter could consist of several mono sounds that have been panned into a multi-channel soundfile. You are therefore advised to find a way to chart out what is where in the multi-channel mixfile.

- Also see Appendix: **M-C MIXFILES**

T. Wishart writes: "MULTIMIX is concerned with the mixing of sounds of any number of channels. Those sounds may themselves be moving around the multi-channel space (see MCHANPAN). This is equivalent to mixing stereo signals containing sound-motion, as in a normal CDP mix. MULTIMIX cannot itself be used to pan sounds (make them move around the multi-channel space – this is done with MCHANPAN) or to change an existing movement. Assigning input channel *N* to output channel *N* will ensure that any motion within an input sound in the mix is preserved. But (as in the examples above) an existing movement can be affected by how the signal is routed - it can be shifted or mirrored or in fact distorted in any desired way, according to how the input channels are assigned to the output channels."

End of NEWMIX MULTICHAN

PANORAMA – Distribute > 1 mono files in a spatial panorama across a specified angle of a sound-surround loudspeaker array

Usage

panorama panorama 1 *insndfile1 insndfile2 [insndfile3...] outmixfile lspk_cnt lspk_aw sounds_aw sounds_ao config* [**-r***rand*] [**-p**] [**-q**]
panorama panorama 2 *insndfile1 insndfile2 [insndfile3...] outmixfile lspk_positions lspk_aw sounds_aw sounds_ao config* [**-r***rand*] [**-p**] [**-q**]

Example command line to create ... :

```
panorama panorama 1 inf1 inf2 panomix.mmx 8 360 360 0 1 -r0
```

Modes

1 Loudspeakers are assumed to be equally spaced
2 Loudspeaker positions are defined in a textfile

Parameters

insndfile1 – first input MONO (only) input soundfile
insndfile2 – second input MONO (only) input soundfile
insndfile3 ... – optional additional input MONO (only) input soundfiles
outmixfile – output multi-channel mixfile for use with the multi-channel mixer **NEWMIX MULTICHAN**. **NB:** The suffix produced is **.mmx** whatever is specified. This distinguishes it from **.mix** files, whose format is very different.

lspk_cnt – the number of loudspeakers. In Mode **1** they are assumed to be equally spaced, but this is not necessarily the case for Mode **2**.
lspk_aw – the angular width of the loudspeaker array (Range: 190° - 360°), front centre at 0. The loudspeaker array is assumed to be symmetrical around a centre-line, running through front-centre of the loudspeaker array and centre of the auditorium. Thus, if front centre is at 0°, the a 190° spread is from -95° to (+)95°. A 360° spread is from -180° to (+)180°.
sounds_aw – the angular width of the output sounds, equal to or less than *lspk_aw*.
sounds_ao – the angular offset of the output sounds. This is only possible if *sounds_aw* is less than *lspk_aw*: the angle between the centre-line of the sounds and the centre-line of the loudspeakers.

config – the distribution of the output sounds within the output angle:

- **config** = 1: sounds are equally spaced
- ***config*** = 2: 2 sounds equally spaced, followed by a gap, etc.
- ***config*** = 3: 3 sounds equally spaced, followed by a gap
- and so on ...
- *config* must be a divisor of the number of input sounds.

-r*rand* – a randomisation of sound positions (Range: 0 to 1)
lspk_positions – In Mode **2** speaker positions are specified in a textfile list of angular positions of (3 - 16) loudspeakers. Positions are specified by positive values between 0° (front-centre) clockwise, so that:

- Values to the **right of centre** lie between 0° (front) and 180° (rear).
- Values to the **left of centre** lie between >180° (rear) and 360° (= 0°) (front).

-p – set this flag to give a pair of loudspeakers at the front. ??NB: if *lspk_aw* is less than 360°, this flag will be ignored ??interpreted differently. Therefore,

- If the angular width of the loudspeakers (*lspk_aw*) < 360°, an **odd number** of loudspeakers gives **one** speaker at centre- front (the Default), and an **even number** of loudspeakers gives **a pair** of loudspeakers centred on the front.
- If *lspk_aw* = 360°, the speaker orientation is ambiguous.

-q – the same logic as for the **-p** flag operates here, but applied to sound positions: ??ignored if *sounds_aw* < 360°.

Understanding the PANORAMA Process

PANORAMA is new in Release 7. It distributes two or more mono soundfiles in a spatial panorama, with a multi-channel mixfile as the output. Loudspeakers are assumed to be effectively surrounding the listening area from the front outwards. The input sounds are distributed in order from the leftmost to the rightmost position: with a 360° spread, 180° is rightmost.

Assuming the listener is in the middle of the auditorium, facing the front, the centre-line passes through the listener in the direction s/he is facing. So in an 8-channel ring, with loudspeaker 1 placed centrally at the front, it passes through loudspeakers 1 and 5.

However, the loudspeakers do not have to encircle the listener. The parameter *ang_width* refers to the total spread angle of the loudspeakers.

- The minimum is 190° (just greater than 180°) with loudspeakers from (just behind) left of the central-listener, through to (just behind) the right of the central listener ... and none in the rest of the space behind the listener.
- The maximum is 360°, with the loudspeakers completely encircling the listener.
- In Mode **1** the loudspeakers are assumed to be equidistant (except for the gap between the leftmost and rightmost speakers in the non-encircling cases).
- In a 190° total spread the leftmost position is at -95° and the rightmost position at +95°.
- In a 360° total spread, the leftmost position is at -180°, and rightmost at +180° (i.e. both in loudspeaker 5 in a centred octagonal arrangement).
- In Mode **2** the loudspeaker positions can be specified.

NB: There is an ambiguity in the usage, as the Mode **2** loudspeaker position information specifies position from 0 (front centre) through to 360, rather than from -180 to +180.

End of PANORAMA

SPIN – Spin one or two stereo images across a multichannel space
A sub-group of two: SPIN STEREO and SPIN QUAD

Usage

spin name mode *infile outfile parameters*, where **name** can be **stereo** or **quad**

SPIN STEREO – Spin a wide stereo image across stereo or multichannel space, with possible doppler-shift

Usage

spin stereo 1 *infile outfile rate dopl xbuf* [**-b**boost] [**-a**att] [**-e**expbuf]
spin stereo 2 *infile outfile rate chns cntr dopl xbuf* [**-b**boost] [**-a**att] [**-k**cmn] [**-c**cmx]
spin stereo 3 *infile outfile rate chns cntr dopl xbuf* [**-b**boost] [**-a**att] [**-k**cmn]

Modes

1 spin with doppler pitch-shift
2, 3 create a 3-channel-wide output, centred on channel "centre", in an "ochans"-channel outfile.

When the spinning image crosses the centre:
Mode 2 uses the outer channels to project a stereo-at-centre image.
Mode 3 uses ONLY the central channel to project a stereo-at-centre image.

Parameters

infile – input soundfile (STEREO).
outfile – output soundfile (STEREO or MULTI-CHANNEL).
rate – spin speed in cycles per second (can vary over time):
Positive values spin clockwise and negative values anticlockwise (as viewed from above).
(Range: -100 to +100 Hz)

Modes 2 & 3 only:
chns (Modes 2 & 3) – number of channels in output file. (Range: 4 to 16)
cntr – central channel of the 3-channel output. (Range: 1 to *chns*)

dopl – maximum doppler pitch-shift (Range: 0 to 12 semitones)
xbuf – multiply size of buffer for Doppler shift (may be necessary for large shift). (Range: 1 to 64)
-b*boost* – optional boost of level, as edges pass through the centre; (Range: 0 to 16)
gradually increases (**boost*) as edge passes "FRONT" centre
and decreases (**1*/*boost*) as edge passes "REAR" centre.
As the two edges pass through the centre simultaneously, one edge gets louder and the other quieter.
-a*att* – overall level attenuation (**att*) as both edges pass through the centre. (Range: 0 to 1)
-e*expbuf*(Mode 1) – [optional parameter not defined]
-k*cmn* (Modes 2,3) – minimum level on the central channel. (Range: 0 to 1)
-c*cmx* (Mode 2 only) – maximum level on the central channel. (Range: 0 to 1, <=*cmn*)

Understanding the SPIN STEREO Process

A stereo sound is rotated within stereo (Mode 1) or multichannel space (Modes 2/3), with optional Doppler Shift. In Mode 1, the main parameters are the *rate* of spin and doppler shift (*dopl*).

Modes 2 and 3 produce a 3-channel-wide rotation within multichannel space, which must be at least 4 channels wide (*chns*).

Musical Applications

...

End of SPIN STEREO

SPIN QUAD – Spin two wide stereo-images across a 5-channel-wide sound image, with possible doppler-shift

Usage

spin quad 1 *infile1 infile2 outfile rate ochns cntr dopl xbuf* [**-b**boost] [**-a**att] [**-k**cmn] [**-c**cmx]
spin quad 2 *infile1 infile2 outfile rate ochns cntr dopl xbuf* [**-b**boost] [**-a**att] [**-k**cmn]

Modes

When the spinning image crosses the centre:
1 use the outer channels to project the stereo-at-centre image.
2 use only the central channel to project the stereo-at-centre image.

Parameters

infile1, infile2 – two input soundfiles (STEREO).
outfile – output soundfile (MULTI-CHANNEL).
rate – spin speed in cycles per second (can vary over time):
Positive values spin clockwise and negative values anticlockwise (as viewed from above).
(Range: -100 to +100 Hz)
ochns – number of channels in the output file. (Range: 5 to 16)
cntr – central channel of the 5-channel output. (Range: 1 to *ochns*)
dopl – maximum doppler pitch-shift (Range: 0 to 12 semitones)
xbuf – multiply size of buffer for Doppler shift (may be necessary for large shift). (Range: 1 to 64)
-b*boost* – optional boost of level, as edges pass through the centre; (Range: 0 to 16)
gradually increases (**boost*) as edge passes "FRONT" centre
and decreases (**1*/*boost*) as edge passes "REAR" centre.
As the two edges pass through the centre simultaneously, one edge gets louder and the other quieter.
-a*att* – overall level attenuation (**att*) as both edges pass through the centre. (Range: 0 to 1)
-k*cmn* – minimum level on the central channel. (Range: 0 to 1)
-c*cmx* (Mode 1 only) – maximum level on the central channel. (Range: 0 to 1, <=*cmn*)

Understanding the SPIN QUAD Process

Two stereo sounds are rotated around multichannel space, with optional Doppler Shift. ("QUAD" refers to 2 x stereo.) The program produces a 5-channel-wide spin within multichannel space, which must be at least 5 channels wide (*ochns*).
The main parameters are the speed of rotation (*rate*) and any Doppler pitch-shift (*dopl*).

SPIN QUAD complements SPIN STEREO which rotates a single stereo sound in stereo or mutichannel space.

Musical Applications

...

End of SPIN QUAD

STRANS MULTI – Change the speed or pitch of a multi-channel sound, or add vibrato

N.B. This function is the multi-channel equivalent of **MODIFY SPEED**.

Usage	Modes	Parameters	Understanding	Musical Applications

Usage

TRANSPOSITION:
(NB! - Transposition is always relative to the **original** speed of the soundfile)
strans multi 1 *inmcfile outmcfile speed* [**-o**]
strans multi 2 *inmcfile outmcfile semitone-transpos* [**-o**]

ACCELERATION:
strans multi 3 *inmcfile outmcfile accel goaltime* [**-s***starttime*]
VIBRATO:
modify speed 4 *inmcfile outmcfile vibfrq vibdepth*

Modes

1 Change the pitch (& speed) of a multi-channel input sound (in a possibly time-varying way) by using a speed-multiplier
2 Change the pitch (& speed) of a multi-channel input sound (in a possibly time-varying way) by using a shift in a (possibly fractional) number of semitones
3 Accelerate (or decelerate) the speed of a multi-channel source file
4 Produce vibrato on a multi-channel sound

Parameters

infile – input multi-channel soundfile to process
outfile – resultant multi-channel soundfile
speed – transposition value (ratio) expressed as a floating point multiplier. With *speed* = 2, for example, the sound is twice as fast and an octave higher. See Chart of Ratios covering up and down 2 octaves. **NB:** use **1.0** for no transposition.
semitone-transpos – transposition value in positive or negative number of semitones; e.g., 12 raises the sound by an octave, and -12 lowers it by an octave. **NB:** use **0** (semitones) for no transposition.

Both *speed* and *semitone-transpos* may vary over time.

accel – multiplication of speed to be reached by *goaltime* – i.e., a transposition ratio
goaltime – time in *outfile* at which the accelerated speed specified by *accel* is to be reached.

- If the *infile* does not end there, it continues to accelerate.
- If the *infile* finishes before *goaltime* is reached, the *outfile* won't reach the specified acceleration value.

starttime – time in *infile / outfile* at which the acceleration begins (Default 0.0)

- **Very** steep deceleration will cause the soundfile to halt, and the process will be terminated with a warning message.
- **Very** steep acceleration will cause the soundfile to be lost off the top of the audible register.

vibfrq – the vibrato frequency in Hz (cyles-per-second) (Range: 0.0 to 120.0)
vibdepth – vibrato depth (pitch shift from centre) in [possibly fractional] semitones (Range: 0.0 to 96.0)

Vibrate and *vibdepth* can vary over time.

-o – breakpoint times are read as times in the *outfile*. The Default is to read them as times in the *infile*

Understanding the STRANS MULTI Process

Speed modification processes change the **duration and the pitch** of the sound together. Thus a faster speed causes a higher pitch, a slower speed a lower pitch.

STRANS MULTI offers a range of functions which affect the speed of the multi-channel soundfile. Perhaps it will be most often used for transposition. Modes **1** and **2** both accept either single values or the names of time-varying breakpoint files with *time transposition* pairs.

The single values act as constants and transpose the whole soundfile up or down by the given amount. In the breakpoint files, transposition can be **almost instantaneous** (almost same time, different transposition value), or **gradual**, creating glissandi (different time, different transposition value). **No transposition** between times is a third possibility (different time, same transposition value). These three possibilities are illustrated in the table below:

Time-varying transposition (using ratios)

time	***speed***	**Comments**
0.0	1.0	No transposition: start at the original pitch
1.0	1.498	Over 1 sec., gliss upwards through a Pefect 5^{th}
3.0	1.498	Hold this new level for 2 sec.
3.0001	0.5	At 3 sec., (almost) instantly drop one 8^{ve} below the original pitch, i.e., to 19 semitones below the previous position
6.0	1.0	Spend the next 3 sec. glissing back to the original pitch

The program will not accept exactly simultaneous values, giving a message to the effect that the times 'are not in increasing order'. To get around this, add a tiny bit to the second time – as in the example – so that the 2^{nd} is nominally later than the 1^{st}, but virtually simultaneous.

IMPORTANT! Note that transposition changes are always **relative to the original speed** of the soundfile, not its current output speed. Thus, in the example above, the soundfile glisses up a perfect 5th (from *speed* 1.0 to *speed* 1.498). With the next ratio, 0.5, the soundfile will drop to half of its **original** speed, and consequently to the octave below its **original** pitch.

Sometimes, you may need certain transpositions to occur at specific times in the *outfile*. The **-o** flag makes this possible. Thus, changing the speed of a file will alter it's duration, and, especially when the speed change itself varies in time (using a breakpoint file), it will be difficult to determine at what time events in the output sound will appear. For example, if you specify a *speed* of 0.5 at *time* 2.0 with the default mode, the speed of the file will reach 0.5 after 2 seconds of the INPUT file have passed – but because the speed of the file has been changed, this will **not** be at 2.0 seconds in the OUTPUT file. **Therefore, if you want the *speed* to reach 0.5 at *time* 2.0 in the OUTPUT file, you should use the -o flag**.

Musical Applications

Transposition which also changes the speed, and therefore the pitch, of the soundfile greatly alters the character of the sound. It is often very interesting to hear what a sound will be like 1, 2 or even 3 octaves below its original pitch. Deep, rich tones can be achieved in this way. These tones can slowly rise or descend if created with a time-varying breakpoint file e.g., moving an octave up or down over the time of the whole sound (airplane takeoff sounds, etc.).

The graphic program ***BRKEDIT*** (PC systems) can create exponential or logarithmic breakpoint data, so glissandi in STRANS MULTI can increase or decrease in speed as well as move in a steady (linear) manner. Alternatively, this can be done with Mode **5**.

Vocal material is very sensitive to pitch changes, so upwards transposition of this type will produce fast, squeaky voices, like the mice trio in *Babe*, and downwards transposition will produce slow, drawn-out ponderous voices.

The vibrato created in Mode **6** is a frequency modulation. Given the very wide ranges allowed, this function is immensely powerful. A slow *vibfrq* with a large *vibdepth* will swing the original sound wildly – increase *vibrate* and it really 'flaps in the breeze' (like a flag in the wind). A fast *vibfrq* with a reasonably tight *vibdepth*, e.g., a minor 3rd, will produce a fluttering effect.

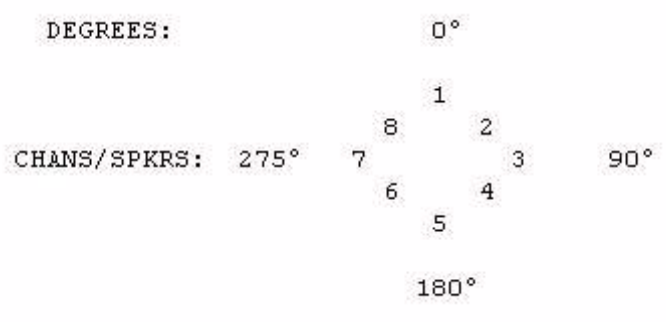
Altogether, a great program to explore and use to push beyond accepted conventions.

ALSO SEE: **MODIFY SPEED**, the mono and stereo version of this function.

End of STRANS MULTI

THE TANGENT GROUP OF MULTI-CHANNEL FUNCTIONS

The Tangent Group, new in Release 7, moves a sound or a sequence of sounds along a tangent to a circle of 8 loudspeakers:



TANGENT ONEFILE – Play repeats of a mono soundfile on a tangent path

Usage

tangent onefile 1 *insndfile outmixfile dur steps maxangle dec* [**-f***foc*] [**-j***jitter*] [**-s***slow*] [**-r**] [**-l**]
tangent onefile 2 *insndfile outmixfile dur steps skew dec* [**-f***foc*] [**-j***jitter*] [**-r**] [**-l**]

Example command line to create a tangent path:

```
tangent onefile 1 in.wav tangent.mmx 10 5 120 0.1
```

Modes

1 When the focus is at 1, the tangent path starts along a line at right angles to loudspeaker 2
2 When the focus is at 1, the tangent path starts along a line formed by loudspeakers 2 and 3

Parameters

insndfile – input MONO soundfile
outmixfile – output multi-channel mixfile (with **.mmx** extension) for use with the multi-channel mixer **NEWMIX MULTICHAN**
dur – the duration of the output soundfile
steps – the count of events in the tangent stream. Note that in all modes, in the part of the stream closest to the focus, the same initial (receding) or final (approaching) event is repeated. These repeated events are additional to this step-count.
maxangle (Mode **1**) – In the motion between three loudspeaker pairs (e.g., 1-2, 2-3 and 3-4), the *maxangle* of the rotation of this motion lies between the third pair (90° ⇒ 135°: loudspeakers 3 & 4 in the example).
skew (Mode **2**) – In the motion between three loudspeaker pairs (e.g., 1-2, 2-3 and 3-4), the *skew* is the ratio of time spent between the last pair and the penultimate pair: loudspeaker pairs 3-4 and 2-3 in the example. (Range: 0 to 1)

dec – the loudness decimation on passing from one event to the next
-f*foc* – the **focal loudspeaker** to or from which the motion takes place (Default: front centre)
-j*jitter* – the randomisation of event timings (Range: 0 to 1, Default: 0, and this parameter may vary over time)
-s*slow* – slow (or speed up) the pan motion **acceleration** (Default: 0). Smaller values slow the arrival of the sound stream at the goal. For approaching motion, **0.5** makes the arrival not too fast. For receding motion, **1.0** makes the initial motion rapid (??and then it slows down).
-r – sounds recede (Default: sounds approach)
-l – motion is to or from **left** of the focal loudspeaker (Default: motion is to or from the **right** focal loudspeaker)

Understanding the TANGENT ONEFILE Process

TANGENT ONEFILE takes a mono soundfile as input and repeats it, moving these repeats along a path tangent to (or from) an 8-channel, front-centred loudspeaker array. That is to say, it is moving from foreground to background (receding) or background to foreground (approaching) along a line adjacent to ('touching'/'tangent') one of the loudspeakers, as defined by *focus*. The loudspeakers are, for the purposes of the program, numbered clockwise, with front-centre as number 1.

End of TANGENT ONEFILE

TANGENT TWOFILES – Play repeats of two synchronised mono soundfiles on a tangent path

Usage

tangent twofiles 1 *insndfile1 insndfile2 outmixfile dur steps maxangle dec bal* [**-f***foc*] [**-j***jitter*] [**-s***slow*] [**-r**] [**-l**]
tangent twofiles 2 *insndfile1 insndfile2 outmixfile dur steps skew dec bal* [**-f***foc*] [**-j***jitter*] [**-r**] [**-l**]

Example command line to create a tangent path:

```
tangent twofiles 1 insndfile1.wav insndfile2.wav tangent.mmx 10 5 120 0.1 0.6
```

Modes

1 When the focus is at 1, the tangent path starts along a line at right angles to loudspeaker 2
2 When the focus is at 1, the tangent path starts along a line formed by loudspeakers 2 and 3

Parameters

insndfile1 – input MONO soundfile
insndfile2 – second input MONO soundfile
outsndfile – output multi-channel mixfile (with **.mmx** extension) for use with the multi-channel mixer **NEWMIX MULTICHAN**
dur – the duration of the output soundfile
steps – the count of events in the tangent stream. Note that in all modes, in the part of the stream closest to the focus, the same initial (receding) or final (approaching) event is repeated. These repeated events are additional to this step-count.
maxangle (Mode **1**) – In the motion between three loudspeaker pairs (e.g., 1-2, 2-3 and 3-4), the *maxangle* of the rotation of this motion lies between the third pair (90° ⇒ 135°: loudspeakers 3 & 4 in the example).
skew (Mode **2**) – In the motion between three loudspeaker pairs (e.g., 1-2, 2-3 and 3-4), the *skew* is the ratio of time spent between the last pair and the penultimate pair: loudspeaker pairs 3-4 and 2-3 in the example. (Range: 0 to 1)
dec – the loudness decimation on passing from one event to the next
bal – the progressive accumulation of the second sound in the mix
-f*foc* – the **focal loudspeaker** to or from which the motion takes place (Default: front centre)
-j*jitter* – the randomisation of event timings (Range: 0 to 1, Default: 0, and this parameter may vary over time)
-s*slow* – slow (or speed up) the pan motion **acceleration** (Default: 0). Smaller values slow the arrival of the sound stream at the goal. For approaching motion, **0.5** makes the arrival not too fast. For receding motion, **1.0** makes the initial motion rapid (??and then it slows down).
-r – sounds recede (Default: sounds approach)

-l – motion is to or from **left** of the focal loudspeaker (Default: motion is to or from the **right** focal loudspeaker)

Understanding the TANGENT TWOFILES Process

TANGENT TWOFILES takes two mono soundfiles as input and repeats them, moving these repeats along a path tangent to (or from) an 8-channel, front-centred loudspeaker array. That is to say, it is moving from foreground to background (receding) or background to foreground (approaching) along a line adjacent to ('touching'/'tangent') one of the loudspeakers, as defined by *focus*. The loudspeakers are, for the purposes of the program, numbered clockwise, with front-centre as number 1.

In this case, the output begins with *insndfile1* and then gradually mixes in *insndfile2*. If one sound is a filtered version of the other, greater distance is suggested. For an approaching sequence (the Default), put the filtered sound as *insndfile1*, but for a receding sequence, put the filtered sound as *insndfile2*. (Note that you must provide the filtered sound.)

End of TANGENT TWOFILES

TANGENT SEQUENCE – Play a sequence of mono soundfiles on a tangent path

Usage

tangent sequence 1 *insndfile1 insndfile2 [insndfile3 ...] outmixfile dur maxangle dec* [**-f***foc*] [**-j***jitter*] [**-s***slow*] [**-r**] [**-l**]
tangent sequence 2 *insndfile1 insndfile2 [insndfile3 ...] outmixfile dur skew dec* [**-f***foc*] [**-j***jitter*] [**-r**] [**-l**]

Example command line to move several sounds along tangent path:

```
tangent sequence 1 insndfile1.wav insndfile2.wav insndfile3.wav tangent.mmx 10 120 0.1
```

Modes

1 When the focus is at 1, the tangent path starts along a line at right angles to loudspeaker 2
2 When the focus is at 1, the tangent path starts along a line formed by loudspeakers 2 and 3

Parameters

insndfile1 – first input MONO soundfile
insndfile2 – second input MONO soundfile
insndfile3... – third and more input MONO soundfiles
outmixfile – output multi-channel mixfile (with **.mmx** extension) for use with the multi-channel mixer **NEWMIX MULTICHAN**
dur – the duration of the output soundfile
maxangle (Mode **1**) – In the motion between three loudspeaker pairs (e.g., 1-2, 2-3 and 3-4), the *maxangle* of the rotation of this motion lies between the third pair (90° ⇒ 135°: loudspeakers 3 & 4 in the example).
skew (Mode **2**) – In the motion between three loudspeaker pairs (e.g., 1-2, 2-3 and 3-4), the *skew* is the ratio of time spent between the last pair and the penultimate pair: loudspeaker pairs 3-4 and 2-3 in the example. (Range: 0 to 1)
dec – the loudness decimation on passing from one event to the next
-f*foc* – the **focal loudspeaker** to or from which the motion takes place (Default: front centre)
-j*jitter* – the randomisation of event timings (Range: 0 to 1, Default: 0, and this parameter may vary over time)
-s*slow* – slow (or speed up) the pan motion **acceleration** (Default: 0). Smaller values slow the arrival of the sound stream at the goal. For approaching motion, **0.5** makes the arrival not too fast. For receding motion, **1.0** makes the initial motion rapid (??and then it slows down).
-r – sounds recede (Default: sounds approach)
-l – motion is to or from **left** of the focal loudspeaker (Default: motion is to or from the **right** focal loudspeaker)

Understanding the TANGENT SEQUENCE Process

TANGENT SEQUENCE takes three or more mono soundfiles as a sequence-input and repeats the sequence, moving these repeats along a path tangent to (or from) an 8-channel, front-centred loudspeaker array. That is to say, it is moving from foreground to background (receding) or background to foreground (approaching) along a line adjacent to ('touching'/'tangent') one of the loudspeakers, as defined by *focus*. The loudspeakers are, for the purposes of the program, numbered clockwise, with front-centre as number 1.

End of TANGENT SEQUENCE

TANGENT LIST – Play a sequence of mono soundfiles as listed in a textfile along a tangent path

Usage

tangent list 1 *intextfile outmixfile dur maxangle dec* [**-f***foc*] [**-j***jitter*] [**-s***slow*] [**-r**] [**-l**]
tangent list 2 *intextfile outmixfile dur skew dec* [**-f***foc*] [**-j***jitter*] [**-r**] [**-l**]

Example command line to create a tangent path:

```
tangent list 1 inlist.txt outmixfile.mmx 10 120 0.1
```

Modes

1 When the focus is at 1, the tangent path starts along a line at right angles to loudspeaker 2
2 When the focus is at 1, the tangent path starts along a line formed by loudspeakers 2 and 3

Parameters

intextfile – input textfile containing a list of MONO soundfiles
outmixfile – output multi-channel mixfile (with **.mmx** extension) for use with the multi-channel mixer **NEWMIX MULTICHAN**
dur – the duration of the output soundfile
maxangle (Mode **1**) – In the motion between three loudspeaker pairs (e.g., 1-2, 2-3 and 3-4), the *maxangle* of the rotation of this motion lies between the third pair (90° ⇒ 135°: loudspeakers 3 & 4 in the example).
skew (Mode **2**) – In the motion between three loudspeaker pairs (e.g., 1-2, 2-3 and 3-4), the *skew* is the ratio of time spent between the last pair and the penultimate pair: loudspeaker pairs 3-4 and 2-3 in the example. (Range: 0 to 1)
dec – the loudness decimation on passing from one event to the next
-f*foc* – the **focal loudspeaker** to or from which the motion takes place (Default: front centre)
-j*jitter* – the randomisation of event timings (Range: 0 to 1, Default: 0, and this parameter may vary over time)
-s*slow* – slow (or speed up) the pan motion **acceleration** (Default: 0). Smaller values slow the arrival of the sound stream at the goal. For approaching motion, **0.5** makes the arrival not too fast. For receding motion, **1.0** makes the initial motion rapid (??and then it slows down).
-r – sounds recede (Default: sounds approach)
-l – motion is to or from **left** of the focal loudspeaker (Default: motion is to or from the **right** focal loudspeaker)

Understanding the TANGENT LIST Process

TANGENT LIST is like TANGENT SEQUENCE except that the soundfiles to be used are listed in a textfile. It takes this list of mono soundfiles as a sequence-input and repeats it, moving these repeats along a path tangent to (or from) an 8-channel, front-centred loudspeaker array. That is to say, it is moving from foreground to background (receding) or background to foreground (approaching) along a line adjacent to ('touching'/'tangent') one of the loudspeakers, as defined by *focus*. The loudspeakers are, for the purposes of the program, numbered clockwise, with front-centre as number 1.

End of TANGENT LIST

TEXMCHAN – Create textures over a multi-channel frame

This process parallels the existing CDP TEXTURE SET processes, but allows the output to be imaged over more than 2 output channels. Only the specifically multi-channel features are documented here.

Usage

texmchan texmchan mode *infile* [*infile2...*] *outfile notedata outdur packing scatter tgrid sndfirst sndlast mingain maxgain mindur maxdur minpich maxpich* ***outchans*** [**-a***atten*] [**-p***position*] [**-s***spread*] [**-r***seed*] [**-w -c -p**]

Note the addition of the *outchans* parameter, and the extended meanings of the *position* and *spread* parameters. The effect on the output soundfile is described below. Otherwise, the documentation for TEXTURE remains the same and can be reviewed in the Reference Manual for **TEXTURE SIMPLE**.

Modes

1 On a given harmonic field
2 On changing harmonic fields
3 On a given harmonic set
4 On changing harmonic sets
5 None (Neutral)

Parameters

outchans – number of channels in the multi-channel output soundfile
position – spatial centre: Range is now 1 to *outchans*
spread – spatial spread: Range is now 1 to *outchans*

Understanding the TEXMCHAN Process

The principal differences from the existing TEXTURE process are

- (1) *Outchans* specifies the number of channels in the ouput file.
- (2) *Position* (the centre of the spatial image of the output) varies between 1 and the number of output channels (*outchans*).
- (3) *Spread* (the spatial spread of the output image) can also vary between 1 and the number of output channels (*outchans*).

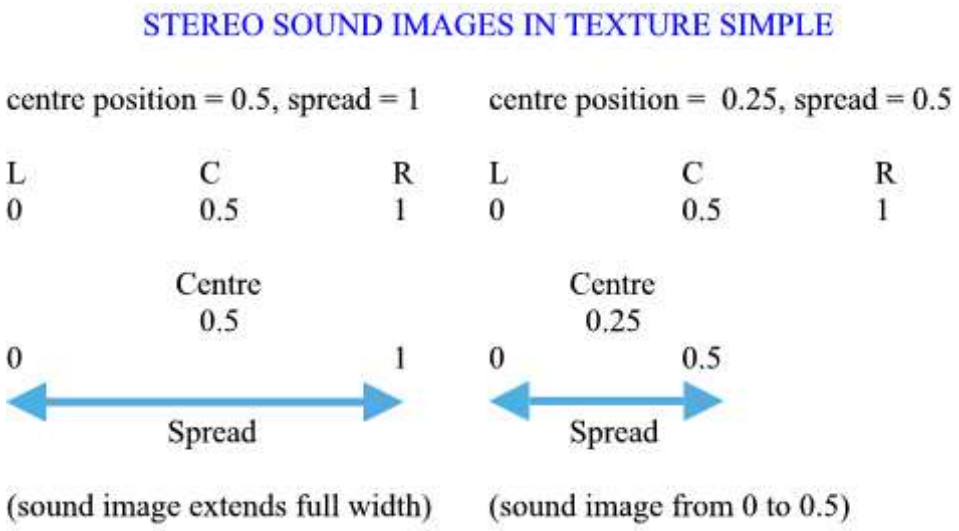
- Both *position* and *spread* can **vary over time** (as can most of the other parameters) so that, for example, the texture can be made to circle, or spread around, the multi-channel output space. (See below.)

How a multi-event texture (with lots of events) gets assigned to channels may not be immediately obvious. It all has to do with the *position* (centre of sound output image) and *spread* (spatial spread of texture events) parameters. These are still there in the multi-channel version of TEXTURE, but their meaning is extended into the multi-channel context. T Wishart has supplied the following explanation.

In the normal TEXTURE programs we have Stereo sound images. If you set the *position* to 0.5 (the centre of the stereo image, halfway between 0 and 1) and set the *spread* to 1 (maximum), events in the texture occur at any place within the full stereo space, which extends from loudspeaker 1 to loudspeaker 2. (Recall that TEXTURE uses a 0 to 1 range, and MODIFY SPACE (for Pan) uses a -1 to +1 range.) The sound image of the entire texture is centred on the point midway between the loudspeakers and extends across the full range between left and right. Note that the *spread* of 0.5 is divided evenly on either side of centre. Using multi-channel numbering, we refer to this centre as 1.5 as it lies halfway between channels 1 and 2.

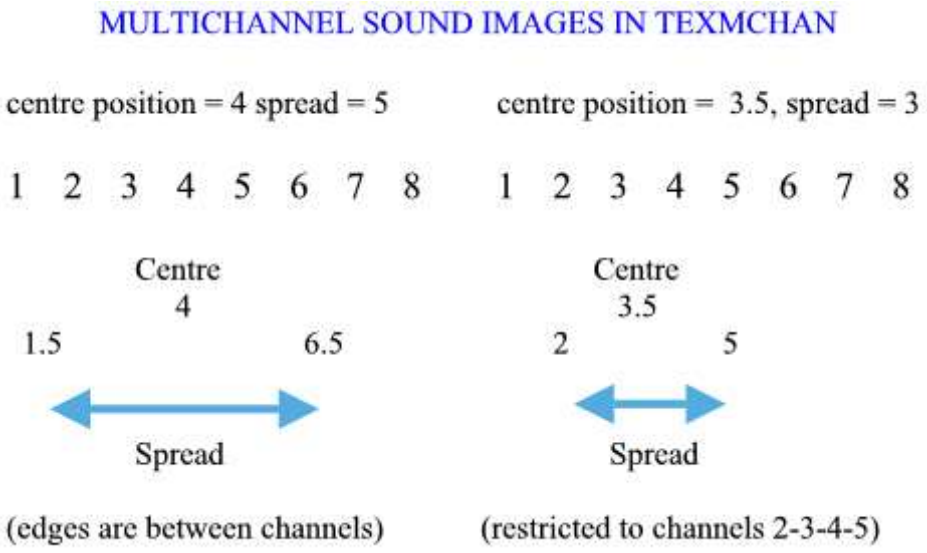
If the *position* (centre) is set at 0.25 and the *spread* is set to 0.5, the sound image will extend from 0 to 0.5, the *spread* of 0.5 being divided evenly on either side of the 0.25 centre-*position*. Using multi-channel numbering, we refer to this centre as 1.25: i.e., a quarter of the way in from the left speaker. The sound image will therefore extend from the left speaker (1) to the mid-point of the stereo space (1.5) between speakers 1 and 2: a *spread* of 0.5.

The diagram below illustrates the result of these *position* and *spread* settings in the existing TEXTURE program.



With the multi-channel TEXMCHAN, the same thinking applies, except that the sound can now be spread over more than 2 loudspeakers, the numbering reflects the multi-channel context, and the centre *position* of the sound image can be anywhere amongst the multi-channel outputs. Therefore, if we set the *position* at 4 and the *spread* to 5, the centre of the entire texture will be located at channel 4 (*position* = centre), and the sounds will spread out equally to either side of this, covering 5 channels in all: 5 ÷ 2 = 2.5 on either side. Thus the left (nominal) edge of the sound image will be at 4 - 2.5 = 1.5 (halfway between channel 1 and 2), and the right (nominal edge of the sound image will be at 4 + 2.5 = 6.5 (halfway between channel 6 and 7). Note that these channels can be set to whichever loudspeaker in the frame. (You might find it helpful to draw a rough diagram of your loudspeaker setup and channel assignments.)

A second diagram illustrates the result of these *position* and *spread* positions in the new TEXMCHAN context.



In addition, the texture can be made to **move around** the multi-channel space by time-varying the value of the *position* parameter, just as a texture made with the TEXTURE program can be made to move over the stereo space.

How might a multi-channel output soundfile from TEXMCHAN relate to MCHANPAN and MULTIMIX and NEWMIX? T Wishart writes:

- A multi-channel output from TEXMCHAN cannot be panned; only mono (and in some cases, stereo) files can be panned with MCHANPAN. However, it can be rotated (etc.) *as a whole*, using FRAME SHIFT.
- A multi-channel output from TEXMCHAN can also become part of a multi-channel mix, using NEWMIX MULTICHAN, to combine it with other sounds. To preserve the original orientation of the multi-channel texture, assign each input channel *N* to output channel *N*, i.e., `1:1 2:2` etc.
- It is possible to automatically generate types of *mixfiles* (for NEWIX MULTICHAN), using MULTIMIX, and these *mixfiles* might use a multi-channel texture files.

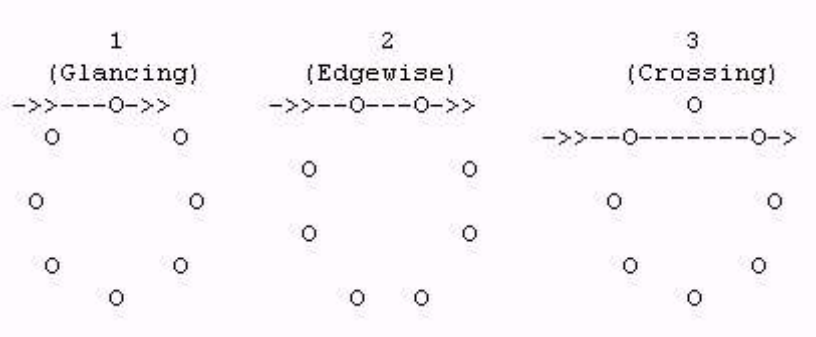
Musical Applications

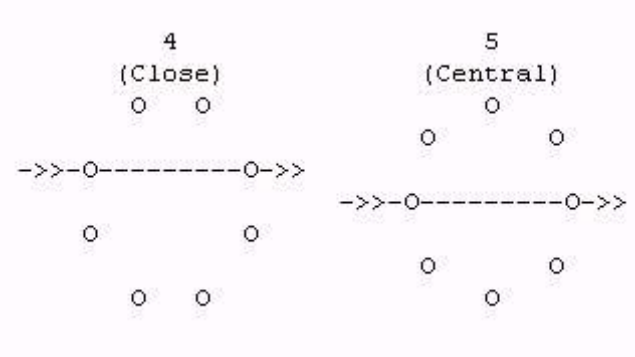
TEXMCHAN may be thought of as a multi-channel extension of the existing **TEXTURE SIMPLE** process with which the texture can be projected into (and made to move around) a multi-channel space.

End of TEXMCHAN TEXMCHAN

THE TRANSIT GROUP OF MULTI-CHANNEL FUNCTIONS

The Transit Group, new in Release 7, moves a sound or a sequence of sounds into and across a circle of 8 loudspeakers. The five modes are:





TRANSIT SIMPLE – Place repetitions of a mono soundfile on a path into and across an 8-channel array

Usage

transit simple 1-5 *insndfile outmixfile focus dur steps max dec* [**-t***thresh*] [**-d***lim*] [**-e***tlim*] [**-m***maxdur*] [**-l**]

Example command line to move a sound into a loudspeaker array:

```
transit simple 1 insnd.wav transit.mmx 1 10 5 45 0.1
```

Modes (see Mode Movement Diagrams above)

1 Glancing
2 Edgewise
3 Crossing
4 Close
5 Central

Parameters

insndfile – input mono soundfile
outmixfile – output multi-channel mixfile for use with the multi-channel mixer **NEWMIX MULTICHAN**, with the file extension **.mmx**
focus – the centre (focal position) of the motion: an **integer** for the odd-numbered modes, and a **floating-point** number (e.g., 1.5, 2.5 etc.) for the even-numbered modes
dur – the duration of the motion from the edge to the centre (**only)**
steps – the count of steps from the edge to the centre (**only**)
maxa (Modes **1-4**) – the maximum angle from the centreline that is reached (<90°)
maxd (Mode **5**) – the maximum distance from the centre reached, where half-radius = 1
dec – the gain decrement on passing from one event to the next (Range: >0 to <1)

NB: To extend the motion further, thereby increasing the duration and the total number of events, the gain decrement can be modified with the following four optional parameters:
-t*thresh* – the threshold overall gain at which the gain decrement starts to increase
-d*lim* – the maximum level of gain decrement after this point (Must be >*dec*)
-e*tlim* – the minimum overal gain at which the event ends (Must be <*thresh*)
-m*maxdur* – the maximum duration of the motion from edge to centre (in case *tlim* is never reached) (Must be >=*dur*)

-l – direct the motion towards the **left** of the focal position (Default: towards the **right**)

Understanding the TRANSIT SIMPLE Process

Repetitions of a mono soundfile move along a path *into* and *across* an 8-channel loudspeaker array. These loudspeakers are presumed to be equidistant in a ring, numbered clockwise starting at '1'. The output is a multi-channel mixfile to be used with NEWMIX.

End of TRANSIT SIMPLE

TRANSIT FILTERED – Place filtered repetitions of a mono soundfile on a path into and across an 8-channel array

Usage

transit filtered 1-5 *insndfile1 insndfile2 outmixfile focus dur steps max dec fdec* [**-t***thresh*] [**-d***lim*] [**-e***tlim*] [**-m***maxdur*] [**-l**]

Example command line to move a sound into a loudspeaker array:

```
transit filtered 1 inf1.wav inf2.wav outmixfilt.mmx 1 10 5 45 0.1
```

Modes (see Mode Movement Diagrams above)

1 Glancing
2 Edgewise
3 Crossing
4 Close
5 Central

Parameters

insndfile1 – first input mono soundfile
insndfile2 – second input mono soundfile
outmixfile – output multi-channel mixfile for use with the multi-channel mixer **NEWMIX MULTICHAN**, with the file extension **.mmx**
focus – the centre (focal position) of the motion: an **integer** for the odd-numbered modes, and a **floating-point** number (e.g., 1.5, 2.5 etc.) for the even-numbered modes
dur – the duration of the motion from the edge to the centre (**only)**
steps – the count of steps from the edge to the centre (**only**)
maxa (Modes **1-4**) – the maximum angle from the centreline that is reached (<90°)
maxd (Mode **5**) – the maximum distance from the centre reached, where half-radius = 1
dec – the gain decrement on passing from one event to the next (Range: >0 to <1)
fdec – progressive mix-in of *insndfile2* (Range: 0 to 1)

NB: To extend the motion further, thereby increasing the duration and the total number of events, the gain decrement can be modified with the following four optional parameters:
-t*thresh* – the threshold overall gain at which the gain decrement starts to increase
-d*lim* – the maximum level of gain decrement after this point (Must be >*dec*)
-e*tlim* – the minimum overal gain at which the event ends (Must be <*thresh*)
-m*maxdur* – the maximum duration of the motion from edge to centre (in case *tlim* is never reached) (Must be >=*dur*)

-l – direct the motion towards the **left** of the focal position (Default: towards the **right**)

Understanding the TRANSIT FILTERED Process

Repetitions of a mono soundfile move along a path *into* and *across* an 8-channel loudspeaker array. These loudspeakers are presumed to be equidistant in a ring, numbered clockwise starting at '1'. The output is a multi-channel mixfile to be used with NEWMIX.

With TRANSIT FILTERED, the second sound should be a filtered version of the first, suggesting greater distance. (Note that you must provide this sound yourself.) Then more of *insndfile2* is gradually mixed in (*fdec*), with greater distance from the centre.

End of TRANSIT FILTERED

TRANSIT DOPPLER – Place pitch-shifted repetitions of a mono soundfile on a path *into* and *across* an 8-channel array, suggesting a doppler shift

Usage

transit doppler 1-5 *insndfile1 insndfile2 [insndfile3 ...] outmixfile focus dur steps max dec* [**-t***thresh*] [**-d***lim*] [**-e***tlim*] [**-m***maxdur*] [**-l**]

Example command line to move a sound into a loudspeaker array:

```
transit doppler 1 inf1.wav inf2.wav inf3.wav outmixdoppl.mmx 1 10 5 45 0.1
```

Modes (see Mode Movement Diagrams above)

1 Glancing
2 Edgewise
3 Crossing
4 Close
5 Central

Parameters

insndfile1 – first input mono soundfile
insndfile2 – second input mono soundfile
insndfile3 ... – third and more input mono soundfiles
outmixfile – output multi-channel mixfile for use with the multi-channel mixer **NEWMIX MULTICHAN**, with the file extension **.mmx**
focus – the centre (focal position) of the motion: an **integer** for the odd-numbered modes, and a **floating-point** number (e.g., 1.5, 2.5 etc.) for the even-numbered modes
dur – the duration of the motion from the edge to the centre (**only**)
steps – the count of steps from the edge to the centre (**only**)
maxa (Modes **1-4**) – the maximum angle from the centreline that is reached (<90°)
maxd (Mode **5**) – the maximum distance from the centre reached, where half-radius = 1
dec – the gain decrement on passing from one event to the next (Range: >0 to <1)

NB: To extend the motion further, thereby increasing the duration and the total number of events, the gain decrement can be modified with the following four optional parameters:
-t*thresh* – the threshold overall gain at which the gain decrement starts to increase
-d*lim* – the maximum level of gain decrement after this point (Must be >*dec*)
-e*tlim* – the minimum overal gain at which the event ends (Must be <*thresh*)
-m*maxdur* – the maximum duration of the motion from edge to centre (in case *tlim* is never reached) (Must be >=*dur*)

-l – direct the motion towards the **left** of the focal position (Default: towards the **right**)

Understanding the TRANSIT DOPPLER Process

Repetitions of a mono soundfile move along a path *to* and *from* an 8-channel loudspeaker array. These loudspeakers are presumed to be equidistant in a ring, numbered clockwise starting at '1'. The output is a multi-channel mixfile to be used with NEWMIX.

The second sound is a pitch-shifted version of the first, suggesting a doppler shift (apparent pitch change). It changes to *insndfile2* after centre is passed.

You could possibly insert more sounds around the motion-centre, which would suggest a gradual doppler change. Note that the order of the sounds will be: first the approaching sound, then the final sound, and then the intermediate sounds, so order your input soundfiles accordingly.

End of TRANSIT DOPPLER

TRANSIT DOPLFILT – Doppler effect on a path *into* and *across* a 8-channel array with filtering, to suggest greater distance

Usage

transit doplfilt 1-5 *insndfile1 insndfile2 insndfile4 [insndfile5 ...] outmixfile focus dur steps max dec fdec* [**-t***thresh*] [**-d***lim*] [**-e***tlim*] [**-m***maxdur*] [**-l**]

Example command line to move a sound into a loudspeaker array:

```
transit doppler 1 inf1.wav inf2.wav inf3.wav inf4.wav outmixdopfilt.mmx 1 10 5 45 0.1
```

Modes (see Mode Movement Diagrams above)

1 Glancing
2 Edgewise
3 Crossing
4 Close
5 Central

Parameters

insndfile1 – first input mono soundfile
insndfile2 – second input mono soundfile
insndfile3 – third input mono soundfile
insndfile4 – fourth input mono soundfile
insndfile5 ... – fifth and more input mono soundfiles
outmixfile – output multi-channel mixfile for use with the multi-channel mixer **NEWMIX MULTICHAN**, with the file extension **.mmx**
focus – the centre (focal position) of the motion: an **integer** for the odd-numbered modes, and a **floating-point** number (e.g., 1.5, 2.5 etc.) for the even-numbered modes
dur – the duration of the motion from the edge to the centre (**only**)
steps – the count of steps from the edge to the centre (**only**)
maxa (Modes **1-4**) – the maximum angle from the centreline that is reached (<90°)
maxd (Mode **5**) – the maximum distance from the centre reached, where half-radius = 1
dec – the gain decrement on passing from one event to the next (Range: >0 to <1)
fdec – ??

NB: To extend the motion further, thereby increasing the duration and the total number of events, the gain decrement can be modified with the following four optional parameters:
-t*thresh* – the threshold overall gain at which the gain decrement starts to increase
-d*lim* – the maximum level of gain decrement after this point (Must be >*dec*)
-e*tlim* – the minimum overal gain at which the event ends (Must be <*thresh*)
-m*maxdur* – the maximum duration of the motion from edge to centre (in case *tlim* is never reached) (Must be >=*dur*)

-l – direct the motion towards the **left** of the focal position (Default: towards the **right**)

Understanding the TRANSIT DOPLFILT Process

Repetitions of several mono soundfiles move along a path *to* and *from* an 8-channel loudspeaker array. These loudspeakers are presumed to be equidistant in a ring, numbered clockwise starting at '1'. The output is a multi-channel mixfile to be used with NEWMIX.

The second sound is filtered version of the first, suggesting greater distance. It gradually mixes in more of *insndfile2*, with greater distance from centre. The third sound is a pitch-shifted version of the first, suggesting a doppler shift. It switches to *insndfile3* after centre is passed. The fourth sound is a filtered version of the third, suggesting greater distance.

You could possibly insert more sounds around the motion-centre, which would suggest a gradual doppler change. Note that the order of the sounds will be: first the approaching sound, then the same sound filtered, then the final sound filtered, followed by any intermediate sounds at the doppler-shift centre.

End of TRANSIT DOPLFILT

TRANSIT SEQUENCE – Position a sequence of mono sounds (at least 3) on a path *into* and *across* an 8-channel array

Usage

transit sequence 1-5 *insndfile1 insndfile2 insndfile3 [insndfile4 ...] outmixfile focus dur max dec* [**-l**]

Example command line to move a sound into a loudspeaker array:

```
transit sequence 1 inf1.wav inf2.wav inf3.wav outmixseq.mmx 1 10 5 45 0.1
```

Modes (see Mode Movement Diagrams above)

1 Glancing
2 Edgewise
3 Crossing
4 Close
5 Central

Parameters

insndfile1 – first input mono soundfile
insndfile2 – second input mono soundfile
insndfile3 – third input mono soundfile
insndfile4 ... – fourth and more input mono soundfiles
outmixfile – output multi-channel mixfile for use with the multi-channel mixer **NEWMIX MULTICHAN**, with the file extension **.mmx**
focus – the centre (focal position) of the motion: an **integer** for the odd-numbered modes, and a **floating-point** number (e.g., 1.5, 2.5 etc.) for the even-numbered modes
dur – the duration of the motion from the edge to the centre (**only)**
maxa (Modes **1-4**) – the maximum angle from the centreline that is reached (<90°)
maxd (Mode **5**) – the maximum distance from the centre reached, where half-radius = 1
dec – the gain decrement on passing from one event to the next (Range: >0 to <1)
-l – direct the motion towards the **left** of the focal position (Default: towards the **right**)

Understanding the TRANSIT SEQUENCE Process

Repetitions of an odd-numbered sequence of soundfiles (at least 3) move along a path *into* and *across* an 8-channel loudspeaker array. These loudspeakers are presumed to be equidistant in a ring, numbered clockwise starting at '1'. The output is a multi-channel mixfile to be used with NEWMIX.

End of TRANSIT SEQUENCE

TRANSIT LIST – Position a sequence of mono sounds (at least 3), as listed in a textfile, on a path *into* and *across* an 8-channel array

Usage **transit list 1-5** *intextfile outmixfile focus dur max dec* [**-l**]

Example command line to move a list of sounds into a loudspeaker array:

```
transit list 1 infile.txt outmixlist.mmx 1 10 45 0.1
```

Modes (see Mode Movement Diagrams above)

1 Glancing
2 Edgewise
3 Crossing
4 Close
5 Central

Parameters

intextfile – input textfile containing a list of mono soundfiles
outmixfile – output multi-channel mixfile for use with the multi-channel mixer **NEWMIX MULTICHAN**, with the file extension **.mmx**
focus – the centre (focal position) of the motion: an **integer** for the odd-numbered modes, and a **floating-point** number (e.g., 1.5, 2.5 etc.) for the even-numbered modes
dur – the duration of the motion from the edge to the centre (**only)**
maxa (Modes **1-4**) – the maximum angle from the centreline that is reached (<90°)
maxd (Mode **5**) – the maximum distance from the centre reached, where half-radius = 1
dec – the gain decrement on passing from one event to the next (Range: >0 to <1)
-l – direct the motion towards the **left** of the focal position (Default: towards the **right**)

Understanding the TRANSIT LIST Process

Repetitions of an odd-numbered sequence of soundfiles (at least 3) as listed in a textfile move along a path *into* and *across* an 8-channel loudspeaker array. These loudspeakers are presumed to be equidistant in a ring, numbered clockwise starting at '1'. The output is a multi-channel mixfile to be used with NEWMIX.

End of TRANSIT LIST

An Overview of Multi-channel facilities available via CDP
~ by T Wishart ~

The CDP environment has been extensively extended to make it possible to work with multi-channel input files and/or to generate multi-channel output files. These processes produce standard format soundfiles, with any number of channels (up to a current limit of 16). In addition, Richard Dobson's ***Multi-channel Toolkit*** allows many diffent formats of multi-channel files (e.g. Ambisonic and WAV_EX formats) to be generated, and facilitates conversions from one format to another.

Note that multi-channel soundfiles now use the CDP program PAPLAY for playback.

There are various ways to create multi-channel files in the CDP environment.

- Existing mono and stereo files can be mixed, in various ways into a multi-channel format, using MULTIMIX CREATE.

- multi-channel soundfiles can also be created from mono files by **panning**, using **MCHANPAN**, **texturing**, using **TEXMCHAN**, **reverberating**, using **MCHANREV**, or **brassaging**, using new multi-channel options in **MODIFY BRASSAGE** or **MODIFY SAUSAGE**, or the new process, **WRAPPAGE**.

The data in multi-channel files can be manipulated

- The data in a multi-channel mixfile, as with a standard mix, can be altered by editing it – on the *Sound Loom* the **QikEditor** gives access to a large variety of functions for doing this.

- The data in multi-channel soundfiles can also be rearranged or edited (channel by channel if required) – see FRAME SHIFT below.

Once made, multi-channel soundfiles can be processed just like mono or stereo files

- Most existing processes that take an *N* channel input (mono or stereo) and give an *N*-channel output, will now work with multi-channel input/output.

- Some programs, which will not work with multi-channel files, have **multi-channel equivalents**
  - **'Tape' transposition, acceleration and vibrato.** The commandline process MODIFY SPEED is mirrored by STRANS MULTI.

- Sometimes we might want to apply the same process to each channel of a multichannel file, then recombine the outputs into a processed multi-channel file, for example, where a process (e.g., DISTORT REPEAT) changes the duration of the inputs unpredictably, or (MODIFY REVECHO) increases the number of output channels to stereo. On the *Sound Loom*, the separation of channels, processing, and recombination of the resulting files can be achieved in a single pass by submitting a **single** multi-channel file to **Bulk Process**, and then calling the desired process (with mono or stereo output) whereupon the channels will be separated and recombined automatically.

In addition, a number of specifically multi-channel processes have been developed.

- **multi-channel pan:** MCHANPAN offers possibilities to
  - Pan a mono source around a multi-channel space
  - Spread a mono or stereo source over a multi-channel space
  - Create antiphonal alternation between different parts of a multi-channel space
  - Step events around a specified sequence of output channels

- **Frame operations:** FRAME SHIFT offers possibilities to
  - Reorient the frame of a multi-channel file
  - Mirror the channel configuration of a multi-channel file about a specified axis
  - Rotate (in a time-varying way) the frame of a multi- channel file
  - Edit some (but not other) channels of a multi-channel file

On the *Sound Loom*, it is also possible to

- **Pan a process:** the process itself moves around the multi-channel space in a way you can specify, leaving the (multi-channel) source where it is.
- **Process all channels independently with a single process:** In some cases,this gives a different result to processing the multi-channel file *as a whole*. Processes giving mono, *or stereo* output on the individual channel data, can be used in this way.
- **Position:** a set of mono or stereo input files on a multi-channel stage (*multi-channel Staging*), using a graphic interface.
- **Collapse:** a multi-channel file into a stereo format of any desired specification (*multi-channel Staging*), using a graphic interface.

On the ***Sound Loom*** these new multi-channel processes will be found on the new **MULTICHAN** menu, on the Process page.

End of OVERVIEW

On setting up a multi-channel composing environment

The main thing to be said here is that your soundcard, mixer and speaker setup must be able to handle multi-channel soundfiles. It is up to you to decide the scope of the facilities you require: i.e., the number of channels to support.

Note, for example, the speaker layout options and diagrams in the **Multi-Channel Toolkit**, particularly those in **FMDCODE**, and consider what will best suit monitoring in your studio space and your composition objectives. A bit of web-surfing regarding multi-channel playback would not go amiss, and there have been numerous relevant articles in the journal *Audio Media* over the years.

Finally, note that although most of the processes in the CDP MULTICHANNEL group assume a clockwise RING layout, you can easily re-order the channel numbers using the functions **CHORDER** or **FRAME SHIFT**, Mode **3**.

End of TECHNICAL

Appendix 1 – MULTIMIX Mixfiles
~ by Trevor Wishart ~

On Generating multi-channel Mixfiles with MULTIMIX: Contexts and Examples

1. Using MULTIMIX to generate a mono 'multi-channel' mixfile from mono input(s)

Let's begin with 4 mono files, *test1.wav*, *test2.wav*, *test3.wav*, and *test4.wav*, and create a mixfile with MULTIMIX, Mode **1**: 'Create a mix where all files start at time zero'.

multimix create 1 *infile1 infile2* [*infile3..*] *mixoutfile*
multimix create 1 *test1.wav test2.wav test3.wav test4.wav mixfile1*

Because in Mode **1** we give no information about the output channel placement, all the mono infiles will be sent to channel 1. The output file becomes:

```
1
test1.wav 0.0000  1  1:1  1.0
test2.wav 0.0000  1  1:1  1.0
test3.wav 0.0000  1  1:1  1.0
test4.wav 0.0000  1  1:1  1.0
```

Comments:

- The first line ("**1**") indicates that this is mono multi-channel mixfile (!), producing a mono output.
- We can see that all the files are routed **1:1**, i.e., from input channel 1 (the mono channel of the input sound) to output channel 1.
- Note that in this example the amplitude levels are all set to full amplitude. You may need to edit this if the inputs are all at a high level and amalgamating them at time zero may cause overload.
- The facilities of MULTIMIX, as with SUBMIX DUMMY, are designed as a time-saving aids, but further editing of the output *mixfile* is usually required.

2. Using MULTIMIX to generate a multi-channel mixfile from mono input(s)

To generate a 4 channel output sound with the same 4 mono sounds, we can use MULTIMIX Mode **6**: 'Distributes *N* mono files, in order, to *N* successive output channels. (Distribution in ascending order.)'

multimix create 6 *infile1 infile2* [*infile3..*] *mixoutfile*
multimix create 6 *test1.wav test2.wav test3.wav test4.wav mixfile2*

This mode sends each (mono) input to a different output channel, and generates the output file:

```
4
test1.wav 0.0000  1  1:1  1.0
test2.wav 0.0000  1  1:2  1.0
test3.wav 0.0000  1  1:3  1.0
test4.wav 0.0000  1  1:4  1.0
```

Comments

- This is a 4 channel output mix: the first line ("**4**") tells us this.
- And we see that
  test1.wav is routed to channel 1 (**1:4**)
  test2.wav is routed to channel 2 (**1:2**)
  test3.wav is routed to channel 3 (**1:3**)
  test4.wav is routed to channel 4 (**1:4**)

To make an 8-channel file, with the same 4 inputs placed in the first 4 channels of the output, we use Mode **7**: 'Distributes *N* mono files, in order, to *K* successive output channels. (Channels may exceed or be less than inputs.)'

multimix create 7 *infile1 infile2* [*infile3..*] *mixoutfile ochans startch* [**-s***skip* **-t***timestep*]
multimix create 7 *test1.wav test2.wav test3.wav test4.wav mixfile3 8 1*

In this example, parameter value '8' is the number of output channels, and parameter value '1' is the first output channel to use. This generates the output *mixfile3*

```
8
test1.wav 0.0000  1  1:1  1.0
test2.wav 0.0000  1  1:2  1.0
test3.wav 0.0000  1  1:3  1.0
test4.wav 0.0000  1  1:4  1.0
```

Comments

- This is identical to the *mixfile2* output except that it will generate an 8-channel output (with the other 4 channels silent).
- If we want our 4 input sounds to go, for example, into the even numbered channels of an 8-channel mix, we can change the *startchannel* parameter to '2', and add the *skip* parameter, setting it to '2'.
- This causes the channel assignment to 'skip' a channel (it advances the channel number by **2**) as it assigns the next input file to the mix. The command line for this is:

multimix create 7 *test1.wav test2.wav test3.wav test4.wav mixfile3 8 2* **-s***2*

- This generates the 8-channel mixfile

```
8
test1.wav 0.0000  1  1:2  1.0
test2.wav 0.0000  1  1:4  1.0
test3.wav 0.0000  1  1:6  1.0
test4.wav 0.0000  1  1:8  1.0
```

- And we see that

test1.wav is routed to channel 2 (**1:2**)
test2.wav is routed to channel 4 (**1:4**)
test3.wav is routed to channel 6 (**1:6**)
test4.wav is routed to channel 8 (**1:8**)

3. Using MULTIMIX to generate a multi-channel *mixfile* from a mixture of mono and multi-channel input(s)

Of course we do not have to use mono files. If we begin with a mono file *test1.wav*, and a 4-channel file *test4ch.wav*, we can go back to Mode **1**. This is because, in this Mode, we give no information about the output channel placement. Therefore all outfiles have their first input channel assigned to output channel 1, i.e., the same channels as in the *infile*.

With the command line

multimix create 1 *test1.wav test4ch.wav mixfile4*

The output file is

```
4
test1.wav   0.0000  1  1:1  1.0
test4ch.wav 0.0000  4  1:1  1.0  2:2  1.0  3:3  1.0  4:4  1.0
```

Comments

- The output file is a 4-channel file, as this is the maximum number of channels amongst all the input files.
- *test1.wav* is routed to channel 1 as before (**1:1**)
- Each channel of *test4ch.wav* is routed to successive output channels, starting at channel 1 (**1:1 2:2 3:3 4:4**) each with standard level **1.0**

Mode **6** and Mode **7** will not work with anything but mono files, so if we want to get our data into a file with more channels, or to place the outputs in different channels, we must edit the mixfile accordingly.

4. Handling the creation of multi-channel mixfiles in *Sound Loom*

In the *Sound Loom*, the **QikEdit** page has a **Reroute** option which obviates the need to directly edit the file.

Global level changes (i.e., changes to the level of all output channels) can also be done rapidly in **QikEdit** just as with standard mixfiles.

For complicated changes of channel assignment or level, however, the full routing information (with levels) must be entered.

The **QikEdit** page, also available with the standard mix program, is accessed from a button on the **Parameters Page** when mixing from the *mixfile*.

- We could begin by changing the number of output channels, by changing the number in the first line of the mixfile. On the **QikEdit** page, we enter the number of required output channels in the **Value** box, and select **More Channels**

- We can change the output routing of the mono file *test1.wav* to output channel 3 by changing the routing code from **1:1** to **1:3**. In **QikEdit**, we highlight this input file on the display, put the value '3' in the Value box, and press **Reroute**

- To send the mono file to channels 5, 6, 7 and 8, we change the routing code from **1:1** to **1:5 1:6 1:7 1:8** with a level for the signal sent to each channel. In the simplest case this will be `test1.wav 0.0000 1 1:5 1.0 1:6 1.0 1:7 1.0 1:8 1.0`. In **QikEdit** we highlight the line, put '5-8' in the **Value** box, and press **Reroute**, and this substitution is done automatically.

- To move the 4 channel file to outputs 5, 6, 7, 8 we change the routing to **1:5 2:6 3:7 4:8**, giving us `test4ch.wav 0.0000 4 1:5 1.0 2:6 1.0 3:7 1.0 4:8 1.0`. Again, on the **QikEdit** page, we just need to highlight the line, put '5-8' in the **Value** box,and press **Reroute**. We can even write '7-2' and the sound will be rerouted **1:7 2:8 3:1 4:2** wrapping around to output channel 1 when it reaches the maximum channel (in this case 8).

- On the **QikEdit** page, a number of other options are possible, like **swapping** the routing between 2 or more input files (where these have the same input channel count), or **rotating** the channels round the output-channel-space (the input channels are reassigned to new output channels ... this is not a *dynamic* rotation - i.e., it will not cause the sound to rotate *through time*.

  To achieve that you need to use the program 1, applying it directly to a multi-channel soundfile. For example, with the 2nd (4-channel) file, we can move it by 5 channels, putting '5' in the **Value** box, and pressing the **Rotate Positions** button. This sends **1:1 2:2 3:3 4:4** into **1:6 2:7 3:8 4:1**, preserving the original level settings. Note that the channels numbers wrap around to 1, once the maximum output channel number (in this case 8) is exceeded.

Using the value '-2' (minus 2) moves the channel numbers downwards, wrapping round to the maximum channel number (in this case channel 8) once the bottom channel is reached. This sends **1:1 2:2 3:3 4:4** into **1:7 2:8 3:1 4:2**, preserving the original level settings.

- On the **QikEdit** page it is also possible to **Mirror** the channel routing. For example, with the channels feeding a ring of loudspeakers numbered from 1 to 8 around the ring, we can exchange the right and left sides of the ring. To do this we need to define the point about which the mirroring takes place. This can be around a specific channel position (use the Channel number), or around the midpoint between two channels. So, to mirror the arrangement around channel 1, we put '1' in the value box, and press **Mirror**. Thus

```
test1.wav   0.0000  1  1:1 1.0
test4ch.wav 0.0000  4  1:1 1.0  2:2 1.0  3:3 1.0  4:4 1.0
```

becomes

```
test1.wav   0.0000  1  1:1 1.0
test4ch.wav 0.0000  4  1:1 1.0  2:8 1.0  3:7 1.0  4:6 1.0
```

Note that channels going to output channel 1 in the original mix (**1:1** and **1:1**) still go to output channel 1, as this channel lies on the mirror plane.

- If we want to exchange low channels with high channels, we can set the mirror at **8.5**, i.e. half way between channel 8 and channel 1. In this case

```
test1.wav   0.0000  1  1:1 1.0
test4ch.wav 0.0000  4  1:1 1.0  2:2 1.0  3:3 1.0  4:4 1.0
```

becomes

```
test1.wav   0.0000  1  1:8 1.0
test4ch.wav 0.0000  4  1:8 1.0  2:7 1.0  3:6 1.0  4:5 1.0
```

i.e., 1 is swapped to 8, 2 to 7, 3 to 6 and so on.

- Note that multi-channel soundfiles can themselves be reoriented, or mirrored, using the program FRAME SHIFT.

5. Making the multi-channel output soundfile

Once a multi-channel *mixfile* has been made, the actual mixing is done with the program NEWMIX MULTICHAN. This is quite similar to the existing program SUBMIX MIX, but produces a multi-channel soundfile as output.

Note that the number of channels in the output soundfile is defined by the *output channel count* on the first line of the multi-channel *mixfile*. (Thus you can also produce stereo or even mono output from this program, if you really want to). If a channel in the output is not assigned any of the input sound (e.g., if no channel of any of the input sounds is sent to output 5), this will be a *silent* channel in the output sound.

The outputs from NEWMIX MULTICHAN are standard "wav" multi-channel soundfiles, identical to your mono or stereo files except that they have more channels of sound. You may want to generate "5.1", or Ambisonic output data, or to use the new WAVEX output file format. Richard Dobson's **multi-channel Toolkit** allows you to change the format of your multi-channel soundfiles (see especially COPYSFX). The **Toolkit** is accessible through the *Sound Loom*, provided that you have the **Toolkit** programs and FRAME SHIFT in your program directory. All executables will of course be in the same directory, but it is particularly important that FRAME SHIFT be present because it must be there for the **multi-channel Toolkit** to appear on the **Multichan** menu.

Please note that if you are using 24-bit multi-channel soundfiles, as far as we know, *Windows Media Player* will not play them unless they are in the WAVEX format. If this is a problem, you could make CDP's PVPLAY or PAPLAY as your default playback program. This is particularly useful in a *Sound Loom* context because they will play both sound and analysis files. Both PVPLAY and PAPLAY come with the CDP software and are part of Richard Dobson's **Multi-Channel Toolkit**.

You can also do Ambisonic panning (in planar sound-surround or periphonically – including height information) using the Toolkit programs.

There are also graphic facilities on the Sound Loom (**multi-channel Staging**) which allow you to

- Assign (several) standard wav files with any number of channels to a multi-channel 'stage'.
- This creates a multi-channel mixfile, mixable with NEWMIX MULTICHAN
- Collapse a standard multi-channel wav file to stereo e.g., for CD release, in any way that you define.

End of MIXFILES APPENDIX